UNIVERSITY OF CALIFORNIA, SAN DIEGO

Variability-aware System-level Design and Analysis

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Computer Engineering)

by

Saumya Chandra

Committee in charge:

Professor Sujit Dey, Chair
Professor Rajesh Gupta
Professor Bill Lin
Professor Tajana Simunic
Professor Kenneth Yun

2009

The dissertation of Saumya Chandra is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2009

DEDICATION

I dedicate this thesis to my family: my parents, Rashmi and Vinod, and my elder brother, Manu.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

A large number of people have provided me with support and guidance through out my doctoral program. I take this opportunity to first extend a heartfelt gratitude towards people who have been instrumental in my research work that culminated in this dissertation. I am extremely thankful towards Prof. Sujit Dey, my advisor, for playing an excellent role in supervising this research. I thank him for being a constant source of encouragement, for giving me flexibility to work on projects of my interest, for introducing me to the best people in the field, for correcting my course at critical junctions, and for always being supportive. I cannot sufficiently express my gratitude towards two people - Dr. Kanishka Lahiri and Prof. Anand Raghunathan - whose guidance has been pivotal in laying the foundation of this dissertation. I would like to express my deepest gratitude, admiration and respect towards Dr. Kanishka Lahiri. I am thankful to him for giving me a direction when it was most required, for introducing me to this topic and for the detailed discussions, insights and comments, especially, for the work presented in Chapters 1 and 2. He has been a wonderful mentor and interacting with him has not only been a great learning experience but also an extremely enjoyable one. It has been a privilege to work with Prof. Anand Raghunathan. His ability to explain complicated concepts clearly and simply, and to maintain a high level of enthusiasm and energy for research at all times is truly remarkable. I am grateful to him for discussing with me, in detail, the contents of Chapters 1, 2 and 3, for giving me invaluable feedback throughout the course of my research, and for being a constant source of inspiration.

I would also like to express my gratitude to my thesis committee members, Prof. Bill Lin, Prof. Kenneth Yun, Prof. Tajana Simunic and Prof. Rajesh Gupta for providing me with valuable comments on this work. I would also like to thank Nikhil Bansal for helping me with the power analysis tool at NEC labs at Princeton.

After coming to UCSD and after looking at similar research groups in several other locations, I feel I have been rather fortunate to have been a part of the MES-DAT group. I would like to thank past and present members of the group, including Xiaoliang Bai, Dongi-Lee, Cathy MacHutchin, Shoubhik Mukhopadhyay, Debashis

Panigrahi, Naomi Ramos, Krishna Sekar, Clark Taylor, Mayank Tiwari, Shaoxuan Wang, Chong Zhao, Yi Zhao, Ann Zhou, and Jun Zhou, for being such an amazing bunch of colleagues. My graduate school experience would have been rather incomplete without the presence of these people. The most awaited coffee breaks provided me with an opportunity to engage in stimulating and fun discussions, and also to share my grad school woes and agonies. Special thanks to Shoubhik for being a patient system administrator and for assisting me in fixing technical problems with my system time and again.

Graduate life taught me the importance of good friends near and far. I owe a special dept of gratitude to all my friends and family, for their encouragement and support through these years. I would especially like to thank Saurabh, Sunhyoung, Dipti, Francesco, my roommates - long term and short term, and all the “Panduz”, for being there for me and for lending a patient ear to all my venting sessions. They listened, they sympathized, they laughed and joked, and somehow still managed to remain my friends.

As a child, I would follow in the steps of my elder brother, Manu. Taking up engineering was probably a natural result of that. I would not have been able to embark on this journey but for his support. I am deeply grateful to him, to my sister-in-law, Jahnavi and to my nephew, Manav, for being a continued source of love, confidence and encouragement.

Last, but not the least, I would like to express my deepest gratitude towards my parents, Rashmi and Vinod. These few lines are not sufficient to capture my heartfelt gratitude towards them. They never stopped believing in me and always managed to bring me back from the moments of deepest despair. I know I have made them proud. They will smile as they see me receive my doctoral degree, and I will feel content to have given them another chance to smile for my achievements.

The text of the following chapters, in part or in full, is based on material that has been published in conference proceedings or journals, or is pending publication in journals, or is in review. Chapter 2 is based on material that has been published in the Proceedings of ACM/IEEE International Symposium On Low Power Electronics and Design, 2006 (S.Chandra, K.Lahiri, A.Raghunathan, and S.Dey, *Considering Process Variations During System-Level Power Analysis*, Proc. International Symposium on Low Power Design, pp. 342 345, Oct 2006) and accepted for publication in the IEEE Transactions on VLSI Systems (S.Chandra, K.Lahiri, A.Raghunathan, and S.Dey, *Variation-aware System-level Power Analysis*, IEEE Trans. VLSI Systems). Chapter 3 is based on material that has been published in the Proceedings of IEEE Design Automation Conference, 2007 (S.Chandra, K.Lahiri, A.Raghunathan, and S.Dey, *System-on-Chip Power Management Considering Leakage Power Variations*, Proc. Design Automation Conf., pp. 877 882, June 2007) and accepted for publication in the IEEE Transactions on VLSI Systems (S.Chandra, K.Lahiri, A.Raghunathan, and S.Dey, *Variation-Tolerant Dynamic Power Management at the System-Level*, IEEE Trans. VLSI Systems). Chapter 4 is based on material that is under review in IEEE Transactions on VLSI Systems. I was the primary researcher and author of each of the above publications, and the coauthors listed in these publications collaborated on, or supervised the research which forms the basis for these chapters.

VITA

2003	B.Tech., Computer Science and Engineering, Indian Institute of Technology, Mumbai
2003	M.Tech., Computer Science and Engineering, Indian Institute of Technology, Mumbai
2005	Summer Research Assistant, NEC Labs. America, Princeton, New Jersey
2006–2008	Teaching Assistant, Dept. of Electrical and Computer Engineering, University of California, San Diego
2003–2008	Research Assistant, Dept. of Electrical and Computer Engineering, University of California, San Diego
2007	Summer Intern, Nvidia Corps. Santa Clara, California
2009	Ph.D., Electrical and Computer Engineering, University of California, San Diego

PUBLICATIONS

S.Chandra, K.Lahiri, A.Raghunathan, and S.Dey, "Variation-aware System-level Power Analysis", *IEEE Trans. VLSI Systems*. (to appear).

S.Chandra, K.Lahiri, A.Raghunathan, and S.Dey, "Variation-Tolerant Dynamic Power Management at the System-Level", *IEEE Trans. VLSI Systems*. (to appear).

N. Banerjee, S.Chandra, S.Ghosh, S.Dey, A.Raghunathan, and K. Roy "Coping with variations", *Proc. VLSI Design*, pp. 581-586, Jan 2009.

S.Chandra, K.Lahiri, A.Raghunathan, and S.Dey, " System-on-Chip Power Management Considering Leakage Power Variations ", *Proc. Design Automation Conf.*, pp. 877–882, June 2007.

S.Chandra, K.Lahiri, A.Raghunathan, and S.Dey, "Considering Process Variations During System-Level Power Analysis ", *Proc. International Symposium on Low Power Design*, pp. 342–345, Oct 2006.

S.Chandra, and S.Dey, "Addressing Computational and Networking Constraints to Enable Video Streaming from Wireless Appliances ", *Proc. Embedded Systems for Real-Time Multimedia (ESTIMedia)*, pp. 27–32, Sept 2005.

ABSTRACT OF THE DISSERTATION

Variability-aware System-level Design and Analysis

by

Saumya Chandra
Doctor of Philosophy in Electrical Engineering (Computer Engineering)

University of California San Diego, 2009

Professor Sujit Dey, Chair

Continued technology scaling has enabled the tremendous growth that semiconductor industry has witnessed in the last half century. However, as the technology scales in into the deep submicron era, variability in device parameters and operating conditions is emerging as a major threat to this growth. In the face of increasing variability, traditional design approaches that use typical or worst-case values can lead to yield loss or increased cost and time to market, causing significant revenue loss in either case. State-of-the-art system-level analysis and design methodologies do not take the impact of variability into account and hence it is natural to question their effectiveness in the presence of variability. We believe that it is imperative that the impact of variability be considered during system level design. Analysis and design techniques that are cognizant of variability can help provide designers valuable feedback about the impact of variations early on in the design cycle and hence, facilitate better design decisions at the system-level while preventing expensive design re-spins.

The specific contributions on this thesis include: (i) system-level variability-aware power analysis methodology while considering the impact of manufacturing-induced variations in effective channel length and operation-induced variations in on-chip temperature, (ii) variation-aware system-level shutdown based power management techniques, (iii) variability-aware voltage level selection to improve the number of chips meeting power and performance targets, and (iv) a methodology for system-level performance analysis under variability, and various architecture level and application level techniques to enable performance recovery in systems affected by variations.

Experiments conducted on various Systems-on-chip designs demonstrate that *variation-aware* design techniques enable significant improvements in overall energy dissipation and performance characteristics. In particular, the resulting distributions are more favorable in terms of reducing the revenue loss due to variations. We believe that the approaches outlined in this thesis are useful with the existing design flow with the current technologies as well as in future systems by facilitating research and development in variation-aware application-level and architecture level techniques.

Chapter 1

Introduction

Integrated circuits (ICs) have evolved remarkably in their capabilities since their inception about half a century ago. IC development started with discoveries illustrating that semiconductors can be used to perform the functionality of vacuum tubes and its application to practice was made possible by technological advances in semiconductor device fabrication. Today integrated circuits have truly become ubiquitous and serve as the brains of almost any imaginable system, from modern household appliances to computing and communication devices, to the most sophisticated transport, automotive, industrial manufacturing, and intelligent security systems. Such tremendous adoption of ICs has been achieved as a result of our ability to consistently incorporate increasing number of functionalities (applications) on a chip while being able to reduce/maintain the cost and the device footprint. This, in large part, can be attributed to the continued shrinking feature size which enabled exponential growth in the number of transistors that could be integrated on a single die (hence, meeting the predictions made by Intel's co-founder Gorden E. Moore in 1965). According to International Technology Roadmap for Semiconductors (ITRS) 2007 edition [2], this exponential growth is expected to continue for atleast another two decades as illustrated in Figure 1.1. However, continuation of this growth is challenged by *variability* that has emerged as an artifact of technology scaling in deep sub-micron technologies. In this thesis, we present analysis and design techniques to address this problem during system design.

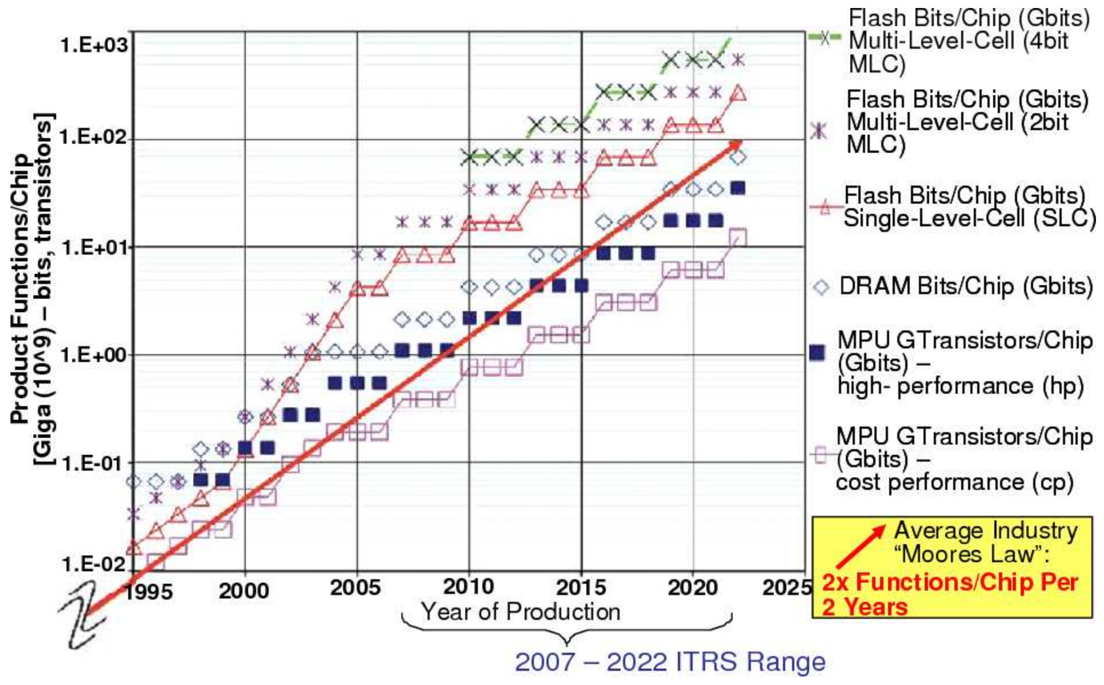


Figure 1.1: 2007 ITRS Product Technology Trends: Product Functions/Chip and Industry Average "Moore's Law" Trends

The process of semiconductor device fabrication has always been susceptible to some variability. However, with scaling of the CMOS technology, precisely controlling the fabrication process is becoming even more difficult and as a result, most of the variability mechanisms are no longer in the "negligible" regime. For example, starting with the 180 nm process node, the ultraviolet (UV) wavelength used for lithography (which is one of the major sources of variability), has become larger than the critical feature size on the die. This gap is steadily increasing with each new process generation, as shown in Figure 1.2 (taken from [3]), further compromising the precision of parameter control. Furthermore, at deep sub-micron technologies, both the impact of the parameter variations as well as some of the variability mechanisms themselves are intensified due to variations in the environmental factors such as circuit's operating temperature.

Variations in process parameters such as channel length and operational parameters such as temperature manifest as variations in the power and performance

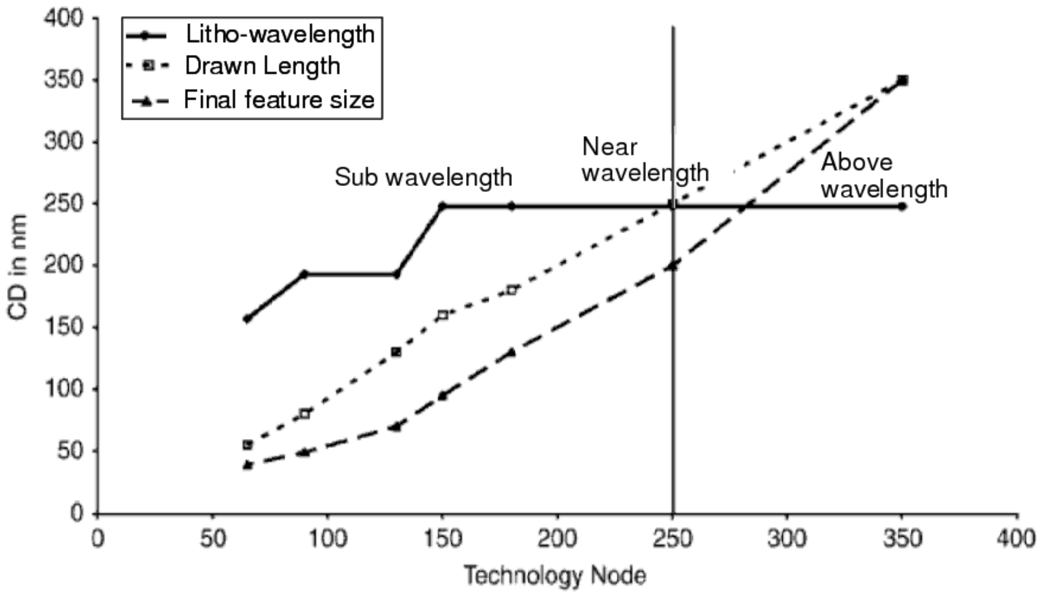


Figure 1.2: Critical dimension (CD) versus lithographic UV wavelength at each technology node

characteristics of gates, register-transfer-level (RTL) components, architectural blocks, and full systems, both spatially (within a die, WID and die-to-die, D2D) and temporally over time. Since manufactured device parameters and operating conditions differ from those considered during design time, systems designed while ignoring the impact of such variations may exhibit suboptimal, and more importantly, unpredictable performance after fabrication. Traditional approaches to design that use typical or worst-case values to model systems might lead to over-design with increased design effort and hence, increased cost and time to market or under-design leading to inability to meet design targets and hence, yield loss; causing significant revenue loss in either case. Moreover, the impact of variations worsens with each successive process generation. It is argued that increasing variations may even cause loss of performance gains of an entire process generation [4].

Recognizing these challenges, various techniques at different stages of the design flow have been developed to aid in variation-aware IC design. Until recently, most of the efforts in this area have focused on addressing the problem at later stages of the design flow, namely at the mask, transistor, and logic levels [5, 6, 7]. While these techniques have shown promise, and some are already being incorporated into commercial IC design flows, they do not completely address the problem, especially in the face of continually increasing variations. It is important to introduce variation-awareness early on in the design flow, *i.e.,* at the system or the architectural level. System-level design for variations can take advantage of information that is not easily available at the lower levels of abstraction. Analysis and design techniques that are cognizant of variability can help provide designers valuable feedback about the impact of variations early on in the design cycle and hence, facilitate better design decisions at the system-level. Therefore, variation-aware system-level analysis and design techniques can not only improve parametric yield but also prevent cost-prohibitive design re-spins.

This thesis focuses on addressing the impact of variations on System-on-chip (SoC) designs. We have developed efficient techniques to incorporate the impact of process variations during system-level power analysis, and during the design of shutdown-based power management policies. We also present techniques for variation-aware selection of supply voltage levels and illustrate that it leads to significant improvements in the number of instances meeting power and performance targets. Furthermore, we have developed techniques to analyze the impact of variations on the overall system performance while accounting for application and architecture interaction and the communication architecture used. We also present architecture level design techniques and show that they can enable significant recovery in performance loss due to variations. In this thesis, we focus on innovative techniques that can address variations without disrupting the existing design flow. As such the techniques developed in this thesis can be incorporated in present day SoCs with little or no overhead.

In this chapter, we first present a background on process variations. We then describe various modeling and optimization techniques that have been developed to analyze and mitigate the impact of variations at different levels of design abstraction. Finally, we conclude this chapter with a brief discussion on the contributions of this thesis and an overview of the remaining chapters.

1.1 Background on variations

In this section, we present a brief background on variations and also develop an understanding of their root causes. We also discuss the impact of variations on power and performance characteristics at various levels of design abstraction.

1.1.1 Sources of variations

There are two broad sources of variations - namely, the manufacturing process causing *manufacturing-induced variations* and the circuit's operation leading to *operation-induced variations*. Variations can be spatial or temporal in nature. Spatial variations in the circuit characteristics occur between different parts of a die, across dies, across wafers, or across lots. Temporal variations, on the other hand, refer to variations in circuit characteristics over time. Spatial and temporal variations often interact with each other, thereby exacerbating the overall impact. It should be noted that like manufacturing-induced variations, operation-induced variations have also become a critical challenge with shrinking feature size, and increasing pattern density and design complexity.

Manufacturing-induced variations arise due to the inability of the fabrication process to precisely control device and interconnect parameters, especially, in the sub-90 nm regime. Manufacturing mechanisms contributing to these variations are lithography, chemical-mechanical polish (CMP), etch, and aberrations introduced by lens steppers. These can be further classified as random and systematic variations. Random variations (also known to arise from *intrinsic* sources of variations) are caused by atomic level differences in the device characteristics. Examples are variations in dopant profiles, film thickness, gate-oxide thickness and line edge roughness (LER). Random dopant fluctuation (RDF) arises from the inability of the manufacturing process to precisely control the number and location of dopant atoms in a channel. The number of dopant atoms in a transistor channel are steadily decreasing with each technology node and is already down to tens in sub-45 nm technologies. This leads to increased statistical fluctuations in the number of dopant atoms, and contributes to variations in the threshold voltage (V_{th}). Line edge roughness is caused due to variations in the incident photon count, the absorption rate, chemical reactivity, and the photoresist composition during lithography. Oxide thickness is down to 1nm range *i.e.,* approximately 5 inter-atomic spacings and is shown to vary randomly by a couple of atomic spacings. It leads to variations in oxide tunneling current. Both LER and oxide thickness variations contribute to variations in the threshold voltage. As the technology scales and the feature size further reduces, such variations will become more pronounced. Unlike random variations, systematic variations (also known to arise from *extrinsic* factors) are caused due to spatial fluctuations in the fabrication process parameters, and hence, exhibit a fairly well understood correlation between the device placement and the extent of variations in the device parameters. Some of the factors leading to systematic variations are temperature gradients across a wafer caused during rapid thermal annealing, change in focus when a mask is stepped across the wafer *etc.* These factors can lead to variations in the device parameters across lots, across wafers, across dies, and also within a die. Examples of systematic variations include variations in effective channel length L_{eff}, chemical-mechanical polishing (CMP) induced variations in metal thickness, and pattern dependent (layout feature density) variations. Variations in effective channel length further contribute to the threshold voltage (V_{th}) variability.

Operation-induced variations arise due to circuit's operation over time and variations in the usage pattern. There are several mechanisms that contribute to the temporal variability, and the associated variation time constants can range from nanoseconds to years [8]. On-chip operating temperature can vary spatially and temporally depending upon the workload characteristics as well as the inherent spatial leakage characteristics of an IC. Negative-bias temperature instability (NBTI) leads to increased threshold voltage of PMOS devices, causing decreased mobility, and thereby, reducing performance. A similar effect is induced by *hot-electron effect* in NMOS devices. Time dependent dielectric breakdown (TDDB) can lead to variations in leakage current or V_{th} due to degradation of conductive paths in the dielectric material. Electromigation (EM) is the gradual transport of metal atoms caused in regions with high current density. It can cause metal to thin down in certain locations leading to increased resistance and eventual failure due to open circuit as the thinning continues. It can also cause short circuits if the displaced metal forms a bridge. Lastly, aging-induced variability due to the wear out factors also reduces circuit performance. Therefore, large variations can sometimes lead to loss of reliability.

Research efforts are being made to develop new transistor structures and materials that can be adopted by future technologies. In these new technologies, some of the current sources of variations might be eliminated. However, these technologies might be susceptible to other sources of variations [8]. For example, in new structures based on fully depleted (FD) channels (or FDFETS), random dopant fluctuations will cease to be a problem. But such devices are expected to be more sensitive to variations in the body thickness. Changes in body thickness can result in changes in the channel potential and hence, V_{th}. Therefore, a careful analysis is required before incorporation of these technological advances in new materials and new structures in order to make sure that inherent advantages are not offset by the added sources of variability.

1.1.2 Impact of variations

In this section, we briefly discuss the impact of manufacturing-induced and operation-induced variations on power and performance characteristics of circuits and systems.

Figure 1.3 illustrates the leakage current variations in N-type and P-type transistors as a function of channel length variations (Source: [1]). It shows that the mean leakage current can be much larger than the nominal leakage current (upto 30% for PMOS). This is because of the exponential dependence of leakage on the channel length. The figure also shows that the PMOS leakage current degrades much faster than the NMOS leakage current. This is because the effect of drain-induced barrier lowering (DIBL) is greater on the PMOS than on the NMOS.

Figure 1.4 illustrates the impact of manufacturing-induced variations on the off-current (leakage) and on-current (drive strength) of a transistors. The data demonstrates that the spread in leakage is two orders of magnitude, while the spread in on-current is a factor of two. The large impact of variations on the leakage current is compounded by the increasing contribution of leakage to total power dissipation.

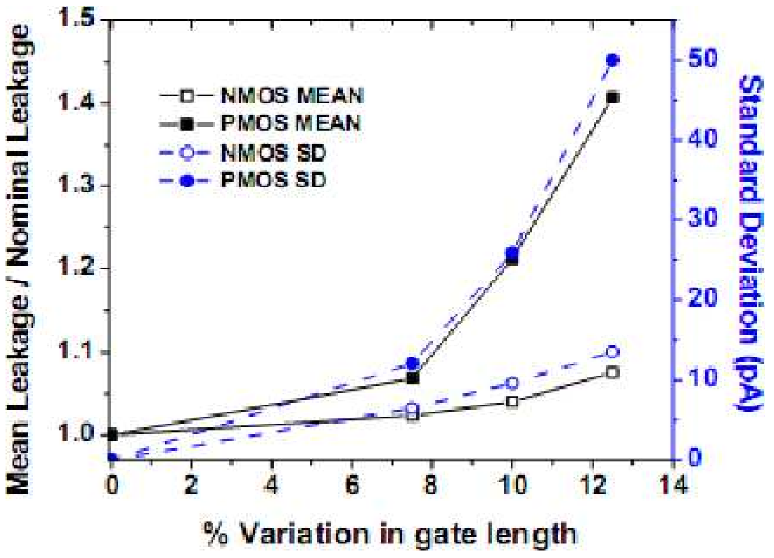


Figure 1.3: Dependence of μ and σ of leakage current on 3σ variation in channel length(Source: [1])

Furthermore, as we show in Chapter 2, the extent by which variations impact different components of a system also strongly depends on their workload profiles. For a component that is mostly *idle*, the leakage power comprises a larger fraction of the total power and hence, the component exhibits larger variations in its total power characteristics.

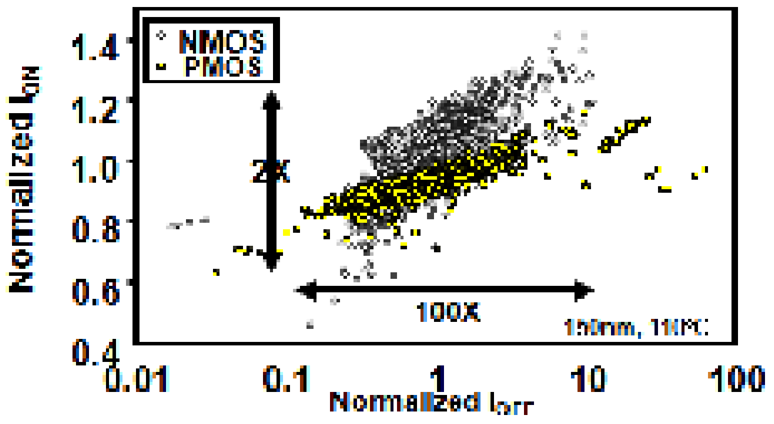


Figure 1.4: Variation in I_{ON} and I_{OFF} of transistors (Source: Intel)

The impact of variations varies with different micro-architectural designs. As the number of critical paths on a die increases (*e.g.,* due to increased parallelism), within-die variations among critical paths cause both the mean (μ) and the standard deviation (σ) of the die frequency to reduce. This is because the probability that atleast one of the paths is slower, increases. Note that, although the spread reduces, the distribution mean also reduces which means that a larger number of chips would exhibit lower performance. This has been shown to be true both with statistical simulations in [4] and testchip measurements in [7]. Similarly, micro-architectures with fewer logic stages experience larger variability since within-die variations do not sufficiently average out. To enable higher clock rates, the number of independent critical paths has been increasing and the logic depth has been decreasing, especially for high performance microprocessors. Such design practices will only worsen the impact of WID process variations [9].

Figure 1.5 shows the frequency and leakage values measured for an Intel microprocessor in 180 nm technology. It shows that process variations can cause upto 20X variation in chip leakage and about 30% variation in chip frequency. The spread increases with further technology scaling. This data also illustrates that a strongly negative correlation exists between power and performance; that is, instances with high performance exhibit poor power characteristics and vice-versa. Therefore, design approaches are required that perform joint optimizations instead of neglecting any one parameter.

In summary, systems manufactured in nanometer process technologies are highly susceptible to process variations. Overall power and performance of such systems exhibit statistical properties. Chip instances with low performance (slow maximum frequency of operation) are either discarded or sold at a much lower price. Chip instances with very high leakage power dissipation are discarded (even though most of the times these chips are capable of high performance) [9]. Therefore, the extent to which variations impact the overall power and performance spread of a system directly impacts yield, and hence, revenue. Worst-case design practices can lead to increased design efforts, time to market, and hence, reduced profit margins and revenue. In the next section, we provide a brief overview of the approaches that are being investigated at various levels of design abstraction to address this issue.

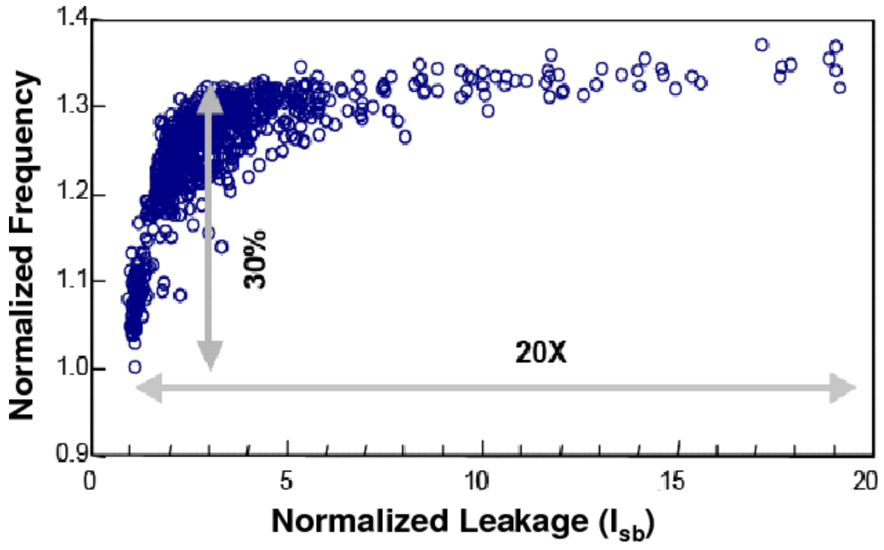


Figure 1.5: Spread in clock frequency and leakage power across 1000 die samples (Source: Intel, DAC'03)

1.2 Related work

There is a tremendous increase in the research efforts aimed towards modeling and addressing the effects of process variations for sub-90 nm technologies at all levels of design abstraction. We first provide an overview of the works that focus on modeling techniques, and then discuss techniques to mitigate the impact of variations.

A number of modeling approaches have been proposed to analyze the impact of process variations on the power and performance characteristics at various levels of abstraction. A number of techniques aimed towards modeling leakage power at the circuit and logic levels have been proposed in [1, 10, 11, 12, 13]. In Chapter 2, we present a methodology to analyze the impact of variations at the system level. In [14], statistical analysis based on principal component analysis is used to decompose process variability using measurements from manufacturing inline benchmarking circuits. A methodology for modeling the impact of within-die interconnect and device parameter variations on circuit performance is presented in [15]. At the logic level, statistical timing analysis, which models gate delays as distributions and uses them to compute the delay distribution of a given circuit, has received great interest [16]. Statistical models to incorporate the impact of variations on microprocessor performance and power consumption are proposed in [17]. Tradeoffs between throughput, power, and area in parallel architectures under process variations are studied in [18]. In [19], a statistical simulator is presented that obtains throughput and maximum clock frequency distributions by performing Monte Carlo simulations on an analytical throughput model.

Design techniques have been proposed at many levels of design abstraction to reduce the impact of variations and improve yield. At the mask level, common techniques to compensate for variations arising due to lithographic effects include, optical proximity correction (OPC) [20] and phase shift masking (PSM). A large body of work focuses on addressing variations at the circuit and logic levels. Some of the proposed techniques include adaptive body bias control and adaptive supply voltage scaling [21, 6], dual threshold voltage based design [22, 23], gate sizing [24], repeater insertion [25], clock tuning [26], dynamic error correction [27], and variation-tolerant dynamic logic [28]. The use of transistor sizing and cell design to address line-width variations due to defocus is proposed in [29]. In the recent years, there has been an increasing interest towards addressing variations at earlier stages of the design cycle, i.e., at the architecture and system levels. Some of the research efforts at these levels include, exploiting parallelism [18], and using optimal number of cores [30], to improve power efficiency. The impact of process variations on embedded SRAM memory architectures are studied in [31]. The authors propose use of configurable buffers together with suitable run-time re-configuration to optimize power and performance. Marculescu *et. al.* have shown that GALS design paradigm can result in significant performance improvements over single island designs [32, 33]. In [34, 35], techniques to determine the optimal body bias for each processing engine of an SoC are developed to meet performance and clock frequency targets. In [36], the authors show that the micro-architectural techniques such as cycle stealing and introducing donor stages in processor pipeline can be utilized to regain performance loss due to variations.

1.3 Thesis overview and contributions

This thesis introduces the need to bring *variation-awareness* early on in the design cycle. Industry trends have shown that process variations are one of the most critical design challenges facing the semiconductor industry today. This thesis establishes the need to consider variability during system-level analysis and optimization. In particular, variability-aware system-level design can be broadly divided into following sub-problems: (a) variability-aware power and performance estimation techniques, and (b) variability-aware power and performance optimization techniques.

An important requirement to develop any optimization technique is the availability of effective methodologies and techniques to analyze the impact of variations on the overall performance and power characteristics. We have developed an efficient and accurate variability-aware system level power analysis methodology based on efficient trace-based Monte Carlo analysis. Our proposed methodology takes into account both manufacturing-induced (D2D and spatially correlated WID) variations and operation-induced on-chip temperature variations. We have also developed a methodology to effectively compute the impact of variations on the SoC performance while taking into account the application and architectural interactions.

At system level, shutdown-based power management is one of the main techniques available to minimize overall energy dissipation. We study the impact of process variations on the effectiveness of shutdown-based power management in the context of an *ideal* Oracle-based power management policy as well as a *realistic* timeout-based power management policy. We show that computing policy parameters without accounting for variations can lead to sub-optimal energy characteristics. We also present two approaches, namely, *chip-specific* approach and *design-specific* approach, to determine optimal policy parameter values in the presence of variations.

The supply voltage, V_{dd}, has a profound impact on the operating clock frequency and energy consumption of an integrated circuit. In particular, systems that can support multiple voltage levels exhibit improved power and performance characteristics. However, we argue that in the presence of variations, determining voltage levels based on nominal or worst-case characteristics can be ineffective. We therefore present a methodology to determine voltage levels at design-time or post-fabrication in a *variation-aware* manner. Our experiments indicate that using variation-aware voltage levels determined using the proposed methodology leads to significant improvements in the number of chip instances meeting power and performance targets as compared to using voltage levels computed using deterministic characteristics.

The remainder of this thesis is organized as follows:

- In Chapter 2, we present a methodology for system-level power analysis while considering the effects of die-to-die variations, spatially correlated within-die variations, and operation-induced temperature variations.

- In Chapter 3, we present techniques to determine variation-aware policy parameters during system-level shutdown based power management (dynamic power management). The techniques are presented in the context of an *ideal* Oracle-based framework and a more realistic timeout-based framework. In this chapter, we also introduce metrics that are more meaningful under variability to quantify the effectiveness of various power-management policies.

- In Chapter 4, we present a methodology for variation-aware voltage selection. We also discuss approaches at design-time and post-fabrication to estimate the frequency-voltage characteristics of a sample set of SoC instance.

- In Chapter 5, we present a system-level performance analysis methodology that takes into account the application characteristics, the architecture characteristics and the specific mapping of the application to the given architecture. We also present design techniques that can be used to recover the performance loss due to variations.

- In Chapter 6, we conclude with a description of future possibilities for variation-aware design.

Chapter 2

Variation-aware System Level Power Analysis

2.1 Introduction

It has been established that the impact of variations on leakage power is very pronounced, more so compared to the impact on maximum frequency of operation. [7, 37, 38]. While this has led a number of researchers to consider variability during power analysis and optimization, most of the efforts are focused on later stages of the design flow, namely at the mask [5], transistor [7, 6] and logic levels [16, 1, 13], as is mentioned in Chapter 1. These methods, although important, are not sufficient to handle the problem because the impact of variations intensifies with each generation of process technology. In this chapter, we propose techniques to systematically analyze the impact of leakage power variations during system-level power analysis. Proposed techniques are based on effective Monte Carlo analysis achieved by combining fast trace analysis with power-state based leakage modeling and thermal modeling. In the past, many techniques have been presented for system-level power estimation [39, 40] and some use instruction-level trace-based methods [41]. However, to our knowledge, none of the previous works propose methods for system-level power analysis under process variations while accounting for manufacturing and operation-induced variations.

2.1.1 Chapter Contributions and Overview

In this chapter, we address the problem of incorporating the impact of manufacturing and operation-induced variations into system-level power analysis. In particular, we focus on the impact the variations in device-level parameters and temperature have on SoC power consumption. We consider both inter-die and spatially correlated intra-die variations in device-level parameters. These variations are static, *i.e,* their values do not change post-fabrication for a given instance, but may vary from one die instance to another and between devices located at different locations on the same die. These device-level parameter variations result in die-to-die and within-die variations in leakage power consumption. However, leakage power also happens to be a strong function of the operating environment, namely, the computational workload and die temperature. Temperature in turn also varies spatially across the die, as well as temporally depending on the workload. Carefully accounting for the dependencies between device-level parameters, leakage power, temperature and workload characteristics is a major challenge during system-level power analysis. Furthermore, for system-level power analysis techniques to be useful in exploring architectural trade offs, they need to be highly efficient, while maintaining acceptable accuracy. The focus of our work lies in developing techniques to meet these goals. In particular, we make the following contributions in this chapter:

- We highlight the need for considering parameter variations during system-level power analysis by analyzing the extent of power variations in an example SoC.
- We demonstrate how simple extensions of conventional approaches to system-level power analysis either fail to provide acceptable analysis efficiency, or fail to properly capture the interdependence between different factors that influence SoC power variations.
- We describe a new analysis methodology that efficiently and accurately generates SoC power distributions under variations. Our main contribution is to properly account for critical dependencies between the various factors mentioned above, while maintaining high analysis efficiency. We describe the different

components of the methodology in detail, including techniques for variation-aware system-level leakage modeling, thermal analysis, and fast Monte-Carlo based trace processing, and demonstrate their integration with state-of-the-art, simulation-based system-level power analysis tools.

The rest of this chapter is organized as follows. In the next section, we present a brief background on process and temperature variations. In Section 2.3, we analyze power variations in an example SoC. We discuss basic extensions of existing analysis techniques in Section 2.4. We describe the proposed methodology in detail in Section 2.5 and Section 2.6. In Section 2.7, we present the experimental results, and conclude in Section 2.8.

2.2 Background - Process and Temperature Variations

As mentioned in Chapter 1, in sub-90 nm technologies variations in process parameters such as channel length, gate oxide thickness and doping concentration are induced during fabrication and manufacturing. This results in process parameters to vary not only from one chip to another (inter-die variations) but also within a chip (intra-die variations). Once fabricated the process parameters of a given chip instance do not change *i.e.*, the process variations are static in nature and cause overall power characteristics of IC circuits to vary inter (or intra) chip. However, the power dissipation is also a strong function of the ICs operating conditions such as workload characteristics and on-chip temperature. The on-chip temperature in turn depends on the power densities and hence, exhibits strong dependence on the local on-chip activity profile and power characteristics. As a result, on-chip temperature, and hence power dissipation characteristics, vary dynamically as well during the life time of a chip instance.

The dynamic nature of temperature leakage inter-dependency complicates the task of accurately and efficiently estimating system-level power under process variations. In the next section, we illustrate the challenges involved and motivate the problem through experimental studies conducted on an example SoC.

2.3 Impact of Variations on SoC Power

In this section, we study the impact of variations on system-level power consumption using an example System-on-Chip. First, we analyze the extent of variations in SoC power consumption arising due to manufacturing-induced variations in device parameters. Next, we evaluate the effect of dynamic variations in die temperature on system-level power consumption. Through these studies we motivate the need for incorporating variation-awareness during system-level power estimation, and develop insights into the key factors that need to be considered by such analysis frameworks.

2.3.1 Impact of Channel Length Variations

In the following example, we focus on inter-die variations in the effective channel length (L_{eff}), since channel length variations have a strong impact on leakage power [1].

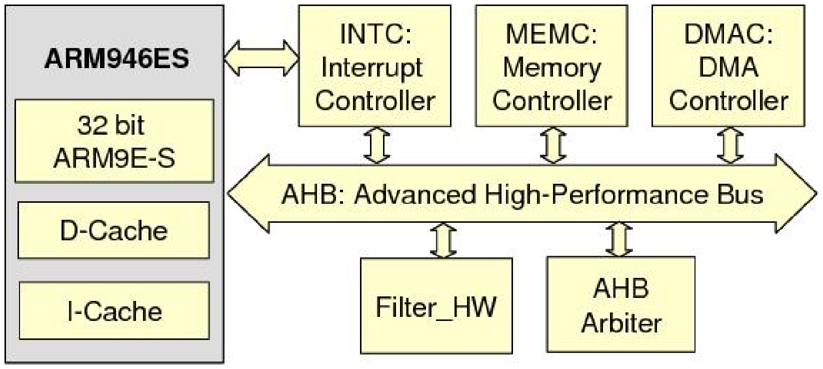


Figure 2.1: Example System-on-Chip design for an image processing application.

Example: Consider the SoC platform illustrated in Figure 2.1. SoC contains an ARM946 processor [42] that executes an image processing application, and a dedicated hardware (Filter_HW) that accelerates image filtering operations. Other SoC components are an on-chip bus (AHB) [43], a memory controller (MEMC), a DMA controller (DMAC) and an interrupt controller (INTC). The SoC is implemented using a commercial, $90\ nm$ standard cell library [44] and is operated at a frequency of $206\ MHz$ and a voltage of $1\ V$. The SoC is modeled using an instruction-set sim-

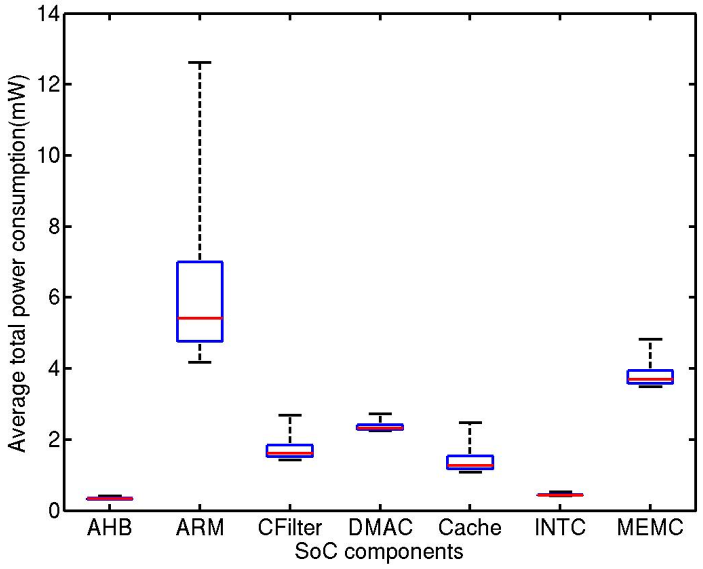


Figure 2.2: Variations in total power consumption of individual components of an example SoC implemented using a 90 nm process.

ulator for the ARM core and SystemC-based cycle-accurate models for rest of the components. An in-house, cycle-accurate, simulation-based system-level power analysis tool [45] is used to calculate the average power consumption (including both dynamic and leakage power) for each component while executing a given test-bench. We used conventional Monte-Carlo techniques (described in Section 2.4) to estimate the impact of chip-to-chip variations in effective channel length (L_{eff}) on the average power consumption, assuming nominal die temperature. For this study, we assumed that L_{eff} follows a normal distribution with $\mu = 90\ nm$ and $3\sigma = 30\%\ \mu$.

The power variations across a sample size of 10,000 chips are captured using the box-and-whisker representation of Figure 2.2. The lower and upper extremities of each box represent the 25^{th} and 75^{th} percentiles of a component's average power consumption. The whiskers denote minimum and maximum values. We notice that the inter-quartile range (the box height) for the ARM core is 25% of its average power, suggesting that variations significantly impact its overall power characteristics. On the other hand, for AHB, the inter-quartile range is only 8% of its average power. The apparent discrepancy is explained as follows. While leakage power is highly sensitive to channel length variations, dynamic power is relatively immune. Hence, components for which leakage accounts for a greater portion of their power consumption display larger power variations. The relative contribution of leakage and dynamic power for a particular component in turn depends on how much time it spends in its different *power-states* (active, idle, sleep, deep-sleep, *etc.*), while processing a given workload. In the presented study, the ARM core spends a significant amount of time in the *idle* power-state, during which its power consumption is almost entirely due to leakage currents. Hence, the average power characteristics of the ARM core are significantly affected by variations. On the other hand, in this study, the AHB is almost always active, serving requests from either the ARM core or other masters. Its average power consumption is largely determined by dynamic power, and hence, is less susceptible to variations.

This example suggests that the extent to which variations affect individual component power characteristics depends critically on a component's *power-state profiles* over time, and the ratio of dynamic and leakage power consumptions. Accounting for this dependence is a key objective of the proposed methodology. We next examine how manufacturing-induced power variations interact with variations in die temperature.

2.3.2 Impact of Temperature Variations

Example: We continue with the SoC described in the previous example. In this experiment, we illustrate the impact of taking thermal variations and spatially correlated intra-die variations into account. We integrated the in-house system-level simulation-based power analysis tool with floor-planning [46] and thermal modeling tools [47], in order to accurately capture the interdependence between die temperature and total power consumption. Using Monte-Carlo simulations (described in Section 2.4), we obtain the average power distributions of the SoC under three scenarios: (a) considering inter-die variations only (D0), (b) considering inter-die variations and thermal variations (D1), and (c) considering inter-die and spatially correlated intra-die variations and thermal variations (D2). We assume that the effective standard deviation

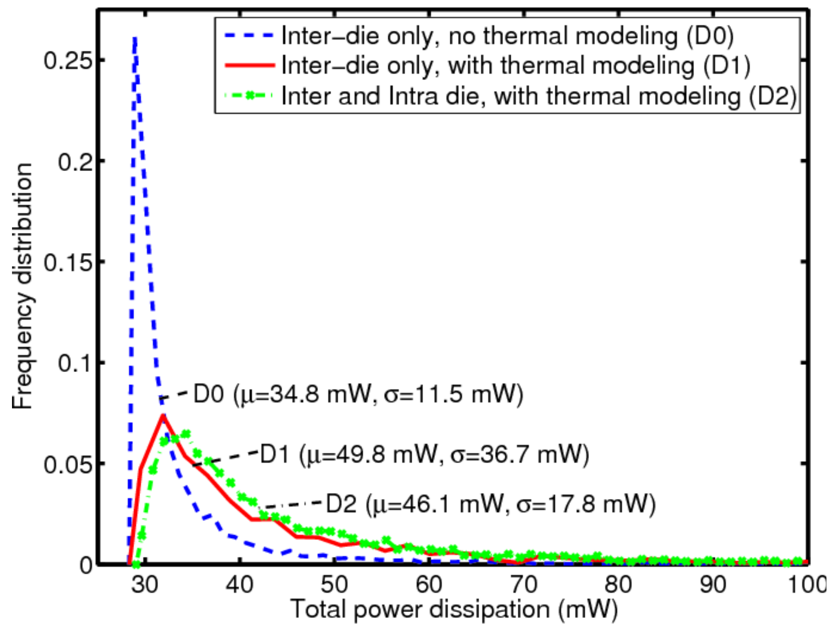


Figure 2.3: Impact of temperature variations and spatial correlations on the average power distribution

($\sqrt(\sigma_{inter} + \sigma_{intra})$ in L$_{eff}$ in case (c) is same as the standard deviation σ_{inter} used in cases (a) and (b). We compare the power distributions for all three scenarios in Figure 2.3. Note that distribution D1 has a high variance due to a long tail, which has been truncated in Figure 2.3 for clear illustration. First, we note that *including thermal effects increases the mean and standard deviation of the power distribution significantly*. Comparing D0 and D2, the estimated mean of the power distribution increases by 32.5% and the standard deviation increases by 55%. The large increase in the standard deviation can be explained as follows. The chip instances with large leakage currents (due to manufacturing-induced variations) generate higher die temperatures that cause further increase in the leakage power dissipation. On the other hand, for chip instances with lower leakage currents, die temperatures are less affected by leakage and hence, the feedback is weaker. This results in the increased spread. Secondly, we note that *ignoring intra-die spatial variations leads to a highly pessimistic estimate of the spread of the power distribution*. In this example, ignoring intra-die spatial variations leads to 43% increase in μ and 219% increase in the σ of the estimated power distribution (D0 Vs D1).

To evaluate the extent to which a design is affected by variations we use the following metric. We define the *power yield* of a design at a level X to be the fraction of die samples that consume on average, no more that X Watts while executing a specific workload. In the above example, we observe that, if die temperature and spatially correlated intra-die variations are ignored (D0), the power yield at a $70\ mW$ level is estimated to be 98%. Including die temperature in the analysis (D1) results in a pessimistic estimate of 88% yield while including both die-temperature and spatially correlated intra variations (D2) lead to an estimate of 90%.

The above studies highlight the need for tools to consider both manufacturing and temperature-induced variations during system-level power analysis. They also demonstrate the importance of properly incorporating the interdependences between manufacturing-induced variations, power-state profiles, leakage power, and die temperatures. For driving the design of variation-aware systems, it is imperative that such tools estimate the system-level power distribution accurately. In addition, at the system level, the design space is large. Hence, in the interest of effective exploration, efficient analysis techniques are required. In the next section, we examine simple extensions of conventional techniques and study their ability to meet the above requirements.

2.4 Techniques

In this section, we consider two approaches for system-level power estimation: (i) simulation-based power analysis, and (ii) spreadsheet-based analysis. We consider simple extensions of these approaches to consider the impact of parameter variations, and evaluate their merits and drawbacks. These methods also serve as the base cases against which we compare our proposed method (described in the next section).

2.4.1 Direct Monte Carlo Approach

A direct Monte Carlo based power estimation method that incorporates both device-level parameter and temperature variations is illustrated in Figure 2.4. In this method, full-system simulation, power estimation and thermal analysis are iteratively performed in order to generate a distribution for average power consumption of the system. For each simulation, *Monte Carlo sampling* is performed to generate a random sample point from a pre-defined distribution of device-level parameters (*e.g.*, transistor channel length). In order to include intra-die variations, spatial correlations and a system-level floorplan are also required while generating a Monte Carlo sample. This is discussed in detail in Section 2.6. Now, given variations in device parameters, corresponding leakage power for each SoC component can be determined using appropriate leakage power models. For each sample point, a complete functional model of the SoC is simulated along with integrated power models to generate the overall power consumption characteristics. In this work, we use an in-house cycle-accurate system simulator [45] with integrated power models to obtain dynamic and leakage power traces for each SoC component. The total power trace hence obtained is analyzed by a thermal analysis tool [47]. The thermal analysis tool also takes the system-level floor-plan as input, and generates component-level temperature traces. Since leakage power is temperature-dependent, power analysis and thermal analysis are iteratively performed until convergence is reached. The inner-loop terminates with the estimate of the average power consumption of the given sample point (chip instance), under the given test-bench with proper inclusion of the effect of temperature variations. The outer loop is executed for each point in the manufacturing parameter sample space to obtain a power distribution for the SoC. The larger the sample size (more outer-loop iterations), the higher the degree of confidence that can be associated with the results regarding the mean and the standard deviation of the final distribution. Given that system simulation, power analysis and thermal simulation are all computationally intensive processes, this method is prohibitively inefficient. It may take up to several days (depending on the length of the simulation trace) to generate the power distribution.

In summary, direct Monte Carlo based techniques while potentially very accurate, are too time consuming for architectural exploration.

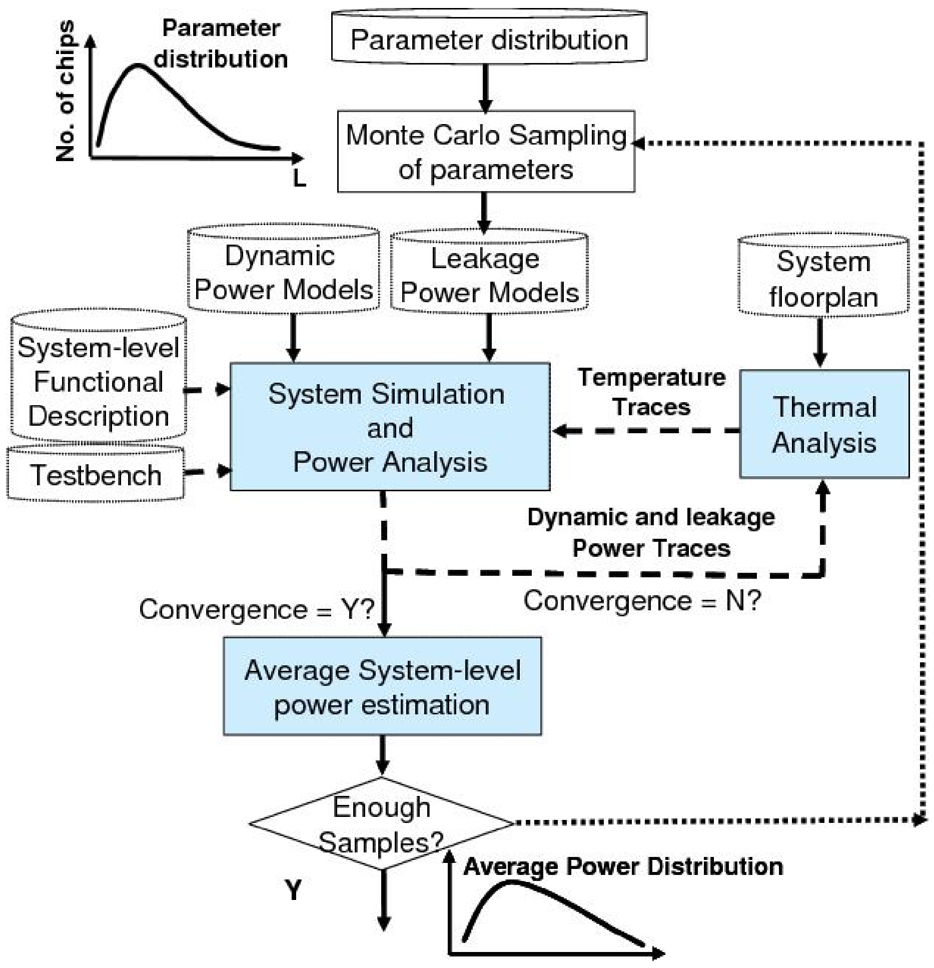


Figure 2.4: Variation-aware power estimation based on direct Monte Carlo system simulation

2.4.2 Spreadsheet-Based Approach

In this method, simple calculations are performed to obtain the average power distribution based on design and process data such as the gate count, activity profile, distribution of process parameters and typical operating temperatures.

Example: Consider the example SoC of Figure 2.1, implemented using a $90\ nm$ process technology. Let us suppose that based on design datasheets, it is known that the active power under a set of benchmarks is $28.5\ mW$ and the total leakage at nominal temperature is $7.4\ mW$. Let us assume that it is known from the process characteristics that for $\sigma_{inter} =\ 8\%\ \mu$ and $\sigma_{intra} = 6\%\ \mu$ variations in channel length, the leakage power variations follow a log-normal distribution with $\sigma =\ 174\%\ \mu$. Based on the above information, we predict the average power distribution at a typical operating temperature of $60 C as follows. Using BSIM3 equations, we estimate that at 60$C, total leakage increases to $24\ mW$. Assuming that the σ/μ ratio for leakage power remains unchanged with temperature, leakage power at 60 follows a log-normal distribution with $\mu\ =\ 24\ mW$, and $\sigma\ =\ 42\ mW$. We therefore conclude that total power also follows a log-normal distribution, with $\mu = 24 + 28.5 = 52.5\ m$W, and $\sigma = 42\ m$W.

Figure 2.5 compares the distribution obtained through such spreadsheet-based analysis to one obtained via direct Monte Carlo simulations. Clearly, there is a notable discrepancy between the two distributions. First, for a specific workload, the spreadsheet-based approach fails to consider the extent to which different system components contribute to power variations. For example, components with long idle periods may power down, and may contribute only slightly to leakage power variations. Also, for components that are mostly active, the larger contribution of dynamic power may overwhelm the variations in leakage power. Second, operating temperatures vary spatially across a chip, according to the relative power density of each component, and dynamically, as the workload varies for a given component. The simplistic assumption of a constant operating temperature leads to inaccurate leakage power estimates. Hence, the spreadsheet-based approach fails to accurately capture the dependence between workload characteristics, component power-states, tempera-

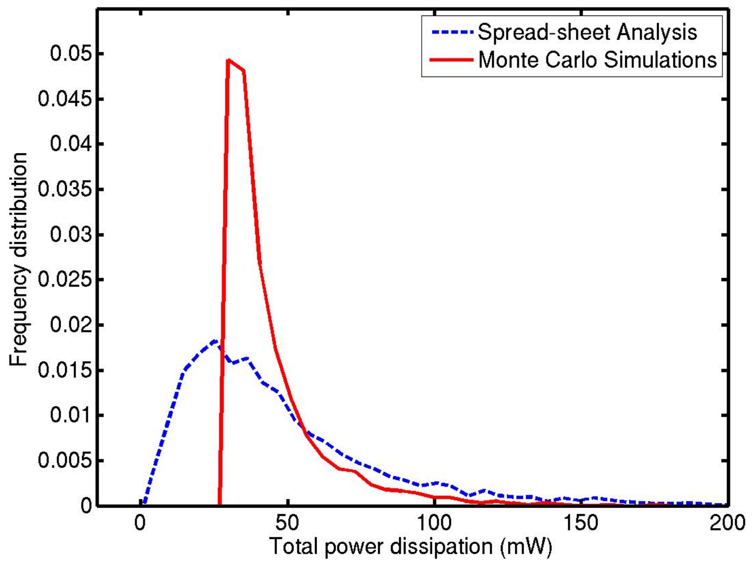


Figure 2.5: Comparing power distributions obtained using direct Monte Carlo simulations and spreadsheet-based analysis

ture, and leakage power. As illustrated earlier, these factors significantly influence the impact of variations on power.

In summary, the above discussion shows that the spreadsheet-based approach while efficient, does not yield results accurate enough to drive architectural exploration. On the other hand, direct Monte Carlo based methods are accurate, but too inefficient. In the next section, we describe our methodology for system-level power estimation that addresses these deficiencies.

2.5 Proposed Analysis Methodology

In this section, we describe a methodology for system-level power analysis that incorporates the impact of both manufacturing-induced inter and intra-die variations and operation-induced variations. We focus on channel length and temperature variations, but the approach is general, and can be extended to consider other variable parameters as well. For the discussion in this section, we assume that a sample set of the SoC instances is available. A sample is a *leakage vector* comprising of leakage power of each component in all of its power states and characterizes a chip instance. The framework and methodology used to obtain the sample set is described in Section 2.6.

The key attribute of our approach is that speedup is achieved by performing both system simulation based dynamic power analysis and thermal analysis *outside the Monte Carlo loop without compromising significantly on accuracy*. To achieve this, we use a three-phase methodology. For clarity of explanation, we first give a brief overview of the methodology as shown in Figure 2.6 and then discuss it in detail.

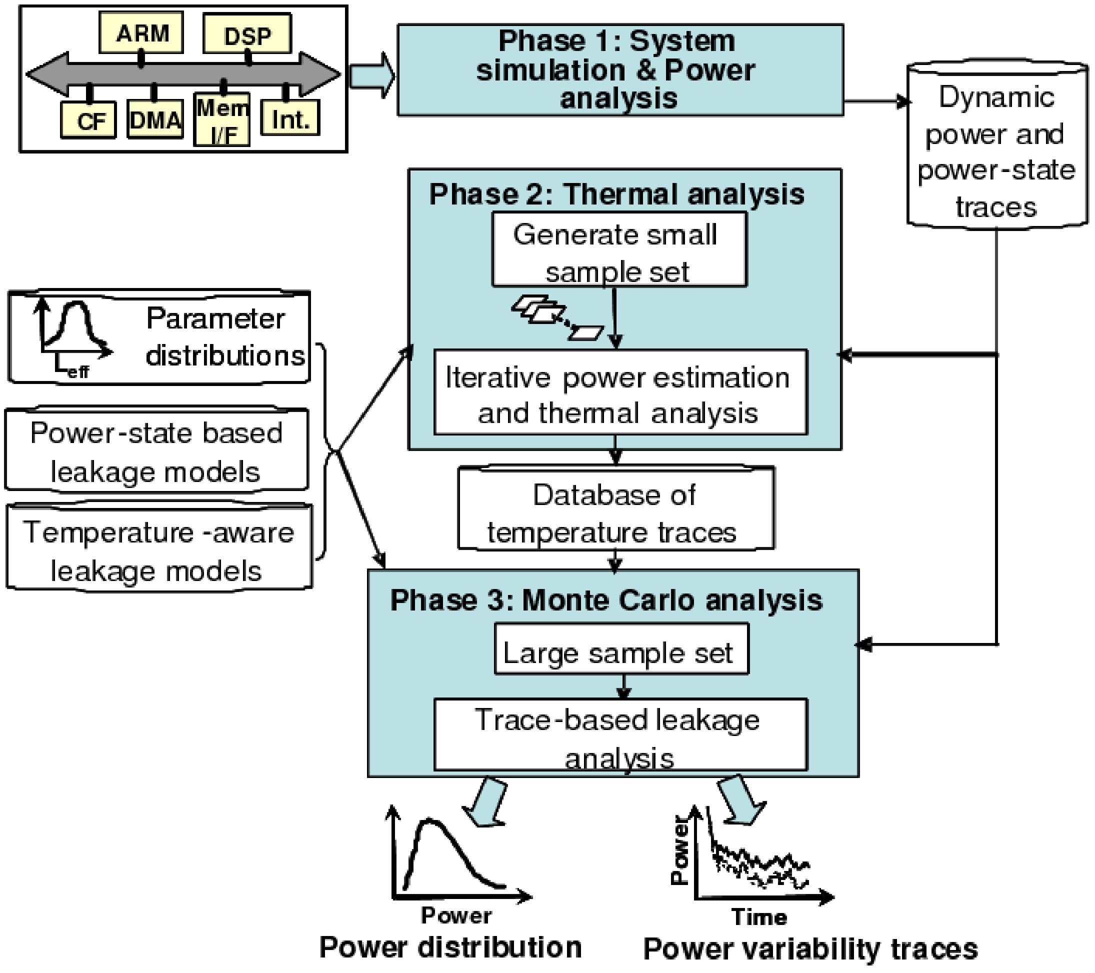

ARM
DSP
CF
DMA
Mem I/F
Int.
Phase 1: System simulation & Power analysis
Dynamic power and power-state traces
Phase 2: Thermal analysis
Generate small sample set
Iterative power estimation and thermal analysis
Parameter distributions
Leff
Power-state based leakage models
Temperature -aware leakage models
Database of temperature traces
Phase 3: Monte Carlo analysis
Large sample set
Trace-based leakage analysis
Power
Power distribution
Power
Time
Power variability traces

Figure 2.6: Methodology for system-level power analysis considering process and temperature variations

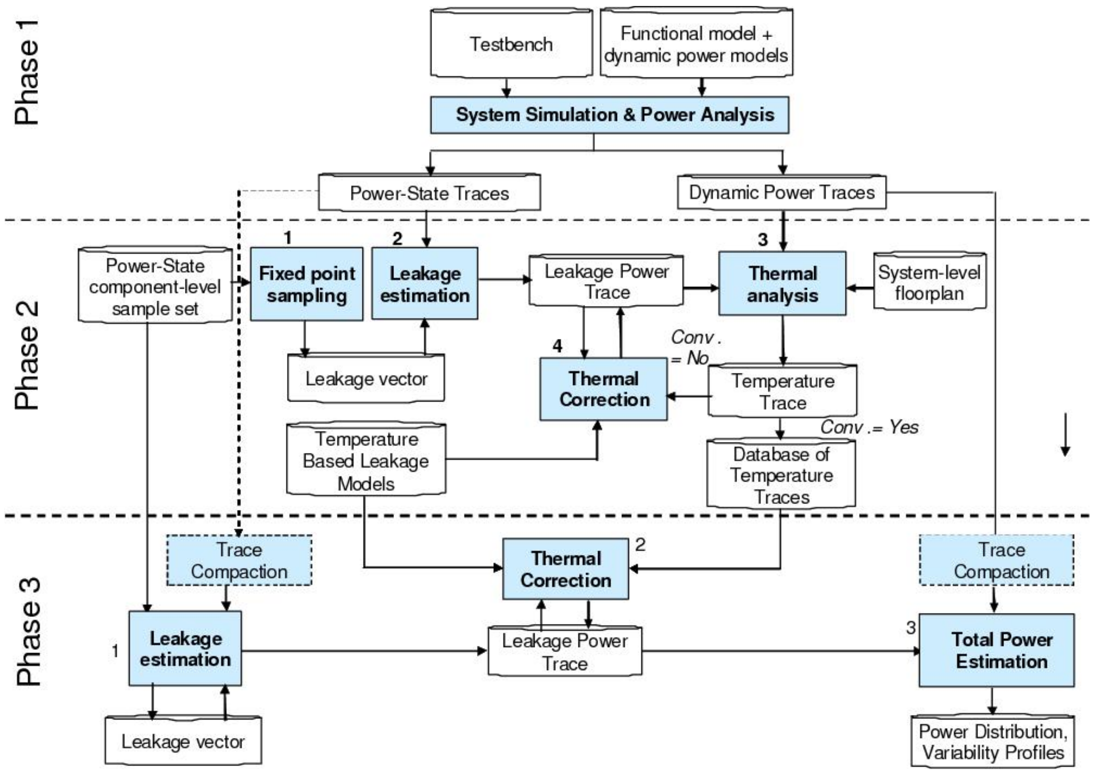


Figure 2.7: Methodology for system-level power analysis considering process and temperature variations

- In *Phase 1*, system simulation is performed just once to capture necessary information in the form of component-level dynamic power and *power-state traces*.
- In *Phase 2*, a small number (N_{calib}) of chip instances are 'selected' and a corresponding database of temperature traces is obtained as follows. For each chip instance, the *power-state traces* are used to compute component-level leakage traces using the *leakage vector* corresponding to the chip instance. These leakage traces are combined with dynamic power traces to perform thermal analysis. Leakage and thermal analysis are performed iteratively until convergence. This compute-intensive iterative analysis is limited to a small number (N_{calib}) of chip instances.
- In *Phase 3*, Monte-Carlo analysis is performed using the entire sample set. For

each sample, we find the ’selected’ chip instance in Phase 2 that is ‘closest’ to it, and its leakage power is obtained using the corresponding pre-computed temperature trace in the database. The rationale behind this approximation is that temperature does not vary significantly for small variations in power and hence, this ‘quantization’ does not impact accuracy of the overall power distribution significantly as is shown quantitatively in Section 2.7.

We now describe all the phases of the methodology in detail. Details of each phase are illustrated in Figure 2.7.

Phase 1 (System simulation and power analysis): The inputs to this phase are a functional model of the target system, a system-level test-bench that models typical operating conditions, and a set of power models that track component-level dynamic power consumption [48, 49, 50]. Cycle-accurate system-level functional simulation is performed with concurrent dynamic power estimation to obtain a set of cycle-level traces for dynamic power consumption over time for each component. In addition, the functional model of each component is instrumented to generate a time trace of its *power-state*. The rationale behind capturing the power-state trace is; (a) a component can have distinctly different leakage characteristics in different states since the number of transistors that are powered on (and hence, leaking) depends on the power-state, and (b) for a given workload the component-level, power-state traces are same for all chip instances irrespective of leakage power variations. These component-level dynamic power and power-state traces contain all the necessary information for subsequent power analysis.

Phase 2 (Temperature Calibration): The inputs to this phase are:

1. Power-state and dynamic power traces from Phase 1.
2. Leakage power sample set [1]: each sample point in the set defines the leakage characteristics of a chip instance. It comprise of a *vector* of N-type and P-type leakage power values for all power-states of each SoC component. N-type

[1]The framework used to obtain the sample set is described in Section 2.6.

and P-type leakage currents vary differently with temperature and hence are computed separately. For simplicity, the N-type and the P-type leakage powers are collectively referred to by the single term, ‘leakage power’, even though the separate power values are combined only at the end when the total power is computed.

3. Temperature-based leakage model: the model is used to determine leakage power at a desired temperature given the leakage power at a nominal temperature. The model is discussed in Section 2.5.2 and the derivation and verification details are provided in *Appendix B*.

4. System-level floor-plan, obtained using fast floor-planning tools.

This phase results in a database of component-level temperature traces corresponding to a pre-defined number, N_{calib}, of ‘selected’ chip instances. There are four main steps to this phase (Figure 2.7).

In **Step 1**, *fixed point sampling*, N_{calib} representative points that are used to determine the database of temperature traces are ’selected’ from the sample set. The points are selected using the leakage power sample of the component having potentially the largest impact on the temperature. Therefore, the component with highest leakage power dissipation, calculated as the weighted-average across power-states, is chosen. N_{calib} equi-spaced points are determined from the leakage power distribution corresponding to this component. Subsequently, the entire sample set is divided into N_{calib} bins according to these *fixed* sample points.

In **Step 2**, *leakage estimation* for a given workload is performed. For each fixed sample point generated in the previous step, leakage power traces are computed using the *leakage vector* of the sample point and the power-state traces obtained in Phase 1. These are component-level leakage power traces at nominal temperature.

In **Step 3**, *thermal analysis* is performed. The thermal analysis tool takes a component-level system floor-plan and component-level power traces (leakage + dynamic) as inputs. The thermal analysis tool uses power values averaged over a specified sampling interval. The output of this step is a set of component-level temperature traces for the given sample point.

In **Step 4**, *thermal correction* is performed. In this step, the leakage power traces are updated to consider the impact of temperature variability. We use the *temperature-based leakage power models* (described in Section 2.5.2) to obtain the leakage power traces corresponding to the temperature traces.

Due to the interdependence between temperature and leakage power, Steps 3 and 4 may have to be iteratively performed until convergence is reached. Steps 2, 3 and 4 are repeated for all the N_{calib} points to generate a coarse-grained *database of temperature traces*. The rational behind generating temperature traces for only a selected set of samples is that temperature variations are not significant for small variations in power. In other words, chips with leakage characteristics close to the leakage characteristics of a selected chip, will exhibit very similar temperature characteristics.

Phase 3 (Monte Carlo trace analysis): The inputs to this phase are power-state and dynamic power traces from Phase 1, the leakage power sample set and temperature-based leakage models, and the database of temperature traces obtained in Phase 2. The output of this phase is a system-level power distribution and a power variability profile over time. Optionally, *trace compaction* can be performed on the power-state, dynamic power and temperature traces before beginning Phase 3 for more efficient analysis at the expense of some accuracy. Trace compaction techniques are described in the next sub-section. Now for each Monte Carlo sample point, following three steps are performed.

In **Step 1**, *leakage estimation* is performed to compute the leakage power traces using the power-state traces and the leakage vector corresponding to the sample point.

In **Step 2**, *thermal correction* is applied to the leakage power traces obtained in the previous step to compute the leakage values at the estimated on-chip temperatures. For a given sample point, temperature trace for the corresponding bin is used for the thermal correction. The bins are determined according to the *fixed sample points* as is already described in Phase 2.

In **Step 3**, *total power estimation* is performed. The leakage power and the dynamic power traces are combined, in a single pass, to compute the total power dissipation for each component. In addition, a running average of the power dissipation for each component and the entire system for the given Monte Carlo sample point is obtained.

The total energy distribution of a System-on-Chip under a given test-bench is obtained by repeating Phase 3 for each sample point. The number of samples to be analyzed depends on the desired error and confidence levels in estimating the system-level power distribution. The larger the sample size, the lower is the sampling error. Note that, unlike the direct Monte Carlo approach described earlier (Section 2.4.1), time consuming functional, power, and thermal simulations are not part of the Monte Carlo analysis loop in the proposed methodology. Therefore, the methodology permits efficient construction of distributions with large sample sizes to obtain a distribution of system-level power, and optionally, profiles of power variability versus time. In the following subsections, we describe details of the trace compaction step and our method for temperature-based leakage modeling.

2.5.1 Trace Compaction

The cycle-level trace generated in Phase 1 contains fields for the cycle number, dynamic power, and the power-state for each component. Analyzing this trace in Phase 3 can provide high cycle-level profiling accuracy. However, analysis efficiency can be substantially improved using simple trace compaction techniques, with little impact on accuracy. Three levels of trace compaction that we considered are described.

Level 1: At the first level[2], consecutive cycles in which a component's power state do not change are collapsed into a single entry in the trace as long as the power state does not span multiple thermal sampling intervals (which is the granularity at which the temperature traces are stored). If a power-state spans a *thermal interval*, as many entries as the number of thermal intervals spanned are created. The resulting trace contains the number of collapsed cycles, average dynamic power for those cycles, the power-state, and the associated temperature. This level preserves temporal accuracy

[2]This is also the base level of power analysis in our methodology

at the granularity of the thermal sampling interval.

Level 2: At the second level, consecutive cycles in which the power-state is constant are collapsed into a single entry, even if they span multiple thermal sampling intervals. Each trace entry is then associated with the average temperature of the *thermal intervals* that it spans. Dynamic power values are also averaged for each entry. Here too, temporal ordering of power-states, dynamic power and temperature values is preserved, albeit at a coarser granularity. Hence using these traces, profiles of power *vs* time and power variability *vs* time can still be generated.

Level 3: At the third level, all temporal information is sacrificed and the trace is reduced to a distribution of power-states (the amount of time spent by each component in each power-state), and the corresponding dynamic power estimates that are averaged over all occurrences of each power-state. Similarly, the database of temperature traces is also reduced to an average temperature estimate for each power-state of each component. Using this compact representation (essentially a lookup-table indexed by the component name and power state), Phase 3 Monte Carlo analysis is extremely fast but is incapable of estimating temporal profiles of power, or power variability.

2.5.2 Temperature Based Leakage Power Model

We now describe the temperature based leakage power model that is used to compute the leakage power during the *thermal correction* step. In majority of the models proposed in the past for leakage current estimation under temperature variations [51] [52], leakage at temperature T is modeled as $I(T) = I(T_0) * f(T, T_0)$, where T_0 is the nominal temperature and f is a non-linear function in T and T_0. However, for our purpose we found these models to be inappropriate since the coefficients describing f typically depend on device parameters such as channel length. In our case we desire a model that depends only on leakage power of a sample at nominal temperature T_0 and target temperature T. Dependence of leakage power on device parameters is taken into account while obtaining the leakage power sample set as described in Section 2.6. We hence propose the following model (*derivation and verification are provided in Appendix B*) to compute leakage at a given temperature:

$$\frac{I(T)}{I_0} = (\frac{T}{T_0})^{\mu_{te}+2} * e^{\frac{A(I_0)*(T-T_0)}{T*T_0}} \tag{2.1}$$

Here, $I(T)$ is the leakage current at the desired temperature T, I_0 is the leakage power at the nominal temperature T_0 and μ_{te} is the mobility temperature coefficient. $A(I_0)$ is a temperature independent term and depends only on the leakage power at the nominal temperature. In the proposed model (details in *Appendix B*), $A(I_0)$ is modeled as a second order polynomial in $ln(I_0)$ as follows,

$$A(I_0) = C1 * (ln(I_0)^2) + C2 * ln(I_0) + C3 \tag{2.2}$$

where, $C1$, $C2$, $C3$ are the fitting coefficients. These are obtained separately for the N and P type devices using BSIM3 simulations, and depend only on the technology node.

2.6 Obtaining sample set under parameter variations

We now describe our approach to obtain the leakage power sample set under inter and intra-die variations. We first present the framework used to model inter-die and spatially correlated intra-die variations in device parameters, and then describe our method to obtain power-state based component-level leakage power for the sample set.

2.6.1 Modeling Parameter Variations

Under inter and intra-die variations, the parameter value (say, effective channel length) of a given device j of a chip instance i is given by,

$$L_{i,j} = L_{NOM} + \delta L_i^{inter} + \delta L_j^{intra}$$

Here, L_{NOM} is the nominal effective channel length. δL_i^{inter} is the contribution of inter-die variations and is assumed to follow a Gaussian distribution $N(0, \sigma_{inter})$. It introduces identical parameter variation in all devices of a given chip instance. δL_j^{intra} is the contribution of intra-die variations. To account for spatial correlations, intra-die variation across devices is assumed to be a correlated multivariate normal distribution $N(0, COV)$. δL_j^{intra} for $j \in \{1, N\}$ is a normal RV with variance σ_{intra}^2 and COV is an N x N matrix that specifies the covariance between different RVs. To reduce the number of spatially correlated random variables, we use a framework similar to [13] in which a chip is divided into N ($N_{rows}\ X\ N_{cols}$) grid cells. Grid size depends on the process technology considered. Devices in the same grid cell are assumed be identically impacted by intra-die variations and represented by one RV, δL_j^{intra}. We assume that the correlation coefficient between RVs in two grid cells is a function of inverse of the normalized distance between the grid cells.

For each sample point (chip instance), L^{inter} is obtained by Monte Carlo sampling of $N(L_{NOM}, \sigma_{inter})$ and is added to δL^{intra}. δL^{intra} is a vector of length N consisting of correlated intra variation in L corresponding to each grid cell of the chip. It is obtained using following equations.

$$\begin{aligned} \delta L^{intra} &= A * \delta L_{UnCor}^{intra},\ where, \\ A &= P * sqrt(D) \end{aligned}$$

Here, δL_{UnCor}^{intra}, is a vector of uncorrelated values obtained by sampling the normal distributions independently. D is a diagonal matrix containing eigenvalues of COV and P is a matrix containing eigenvectors of COV.

We model leakage currents in N-type and P-type transistors, as functions of channel length. These models are used to obtain transistor-level leakage currents, i_n and i_p, corresponding to the sample set obtained above. The models are described in *Appendix A* of this thesis.

2.6.2 Power-state based Component-level Leakage Power Modeling

In this subsection, we describe a procedure for obtaining the power-state based component-level leakage power sample set, used in our methodology. Different steps of the procedure are described below:

- **Step 1a:** For each component, a set of power-states (*e.g.*, active, idle, sleep, deep-sleep, *etc.*) is obtained from a design datasheet. The rationale for identifying power-states is that under different states, component leakage power is distinctly different, depending on the number of transistors that are powered on (and hence, leaking) in a particular power-state.

- **Step 1b:** Mapping of the components to the grid cells is determined using system level floorplan as shown in Figure 2.8. In this step, we determine the device count corresponding to each component $comp$ in each grid j, $N_{cnt}(comp, j)$.

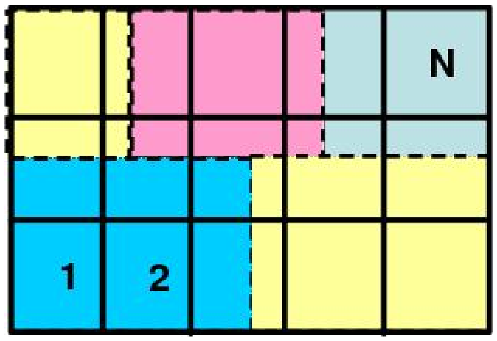


Figure 2.8: Component to grid mapping

- **Step 2:** For each power-state of each component, the total active device width associated with N-type and P-type devices (K_n, K_p respectively) is estimated. Typically at this stage of design, detailed information of the physical implementation is not available. Therefore, we determine K_n and K_p based on two assumptions, (a) for each power state of a component, the active device width is uniformly divided amongst the constituent grids of the component. (b) the circuit can be modeled as sea of identical inverters. We then use approximate gate-counts to estimate K_n and K_p. Similar approaches have been used for high-level leakage estimation [53].

  However, our methodology allows more accurate estimation if the design contains hard macros for which detailed layout information is available. For those components, factors K_n and K_p can be determined using the layout information available. For rest of the components, the approach discussed above can be used.

- **Step 3:** Total N-type and P-type leakage currents corresponding to a given power-state of a given component are computed as follows,

$$\begin{aligned} I_{n,tot} &= \sum_{grid,j=1}^{N} I_{n,tot}(j) = \sum_{grid,j=1}^{N} K_{n,tot}(j) * i_n(j) \\ I_{p,tot} &= \sum_{grid,j=1}^{N} I_{p,tot}(j) = \sum_{grid,j=1}^{N} K_{p,tot}(j) * i_p(j) \end{aligned}$$

  Here, $K_{n,tot}(j)$ and $K_{p,tot}(j)$ (identified in **Step 2**) represent the effective N-type and P-type leakage width associated with a given power-state of a given component in grid j. Similarly, $i_n(j)$ and $i_p(j)$ represent the leakage current of unit width N-type and P-type devices, respectively, in grid j and are obtained using leakage models described in *Appendix A*.

This is done for the entire sample set to obtain component-level leakage power samples, which is used as input to our methodology described in Section 2.5.

2.7 Experimental Results

In this section, we describe our experimental set up, present results that evaluate the accuracy and efficiency of the proposed method, and compare it with the simple extensions of existing techniques described in Section 2.4. Finally, we illustrate the application of the proposed analysis methodology towards developing new, variation-aware strategies for power management.

2.7.1 Experimental Setup

For our experiments, we use the System-on-Chip design described in Section 2.3. The system is modeled using the cycle-accurate models described in SystemC [54] for all the hardware components, an instruction-set simulator for the ARM core, and transaction-level models for the AHB on-chip bus. All the components are enhanced with corresponding power models that track dynamic power consumption over time [45]. Power-state based leakage power distributions for the SoC components are generated using the procedure described in Section 2.6.2. Characterization data for calibrating the statistical leakage power model and the temperature based leakage power model is obtained via HSPICE simulation of the Berkeley Predictive Technology $90\ nm$ BSIM3 model card [55]. Total active device width for different power states is estimated using high-level techniques as described in Section 2.6.2. Thermal analysis is performed using HotSpot [47]. The system-level floor-plan is generated using the U.C. Santa Cruz floor-planning tool [46]. The methodology is integrated using MATLAB [56]. Distribution sampling, leakage estimation, thermal correction, and trace-analysis methods are implemented using MATLAB's C-MEX utility.

2.7.2 Accuracy Comparison

First, we present results that compare the accuracy of our approach with direct Monte Carlo simulation and spreadsheet-based analysis. For this experiment, we consider a workload of length $500\ ms$, with 50% system activity. We use $N_{calib} = 10$ in Phase 2 of the methodology. Figure 2.9 illustrates the average power distribution of the SoC, as predicted using the proposed methodology and direct Monte Carlo simulations (presented in Section 2.4 A). We observe that the distribution obtained using our approach ($\mu = 52.3\ mW$, $\sigma = 35.3\ mW$) matches closely with the one obtained using direct Monte Carlo simulations ($\mu = 52.7\ mW$, $\sigma = 36.4\ mW$), with relative error less than 0.5%. The slight disparity in the distributions arises because in the proposed methodology, the temperature trace corresponding to a sample point is approximated with that of the nearest fixed sample point as opposed to the direct Monte Carlo simulations in which the temperature trace is individually obtained for each Monte Carlo sample point. One of the reasons for the proposed method to result in such close accuracy is that in the calibration phase we create bins based on leakage distribution of the component with highest contribution of leakage power. The error is further reduced if a larger value of N_{calib} is used.

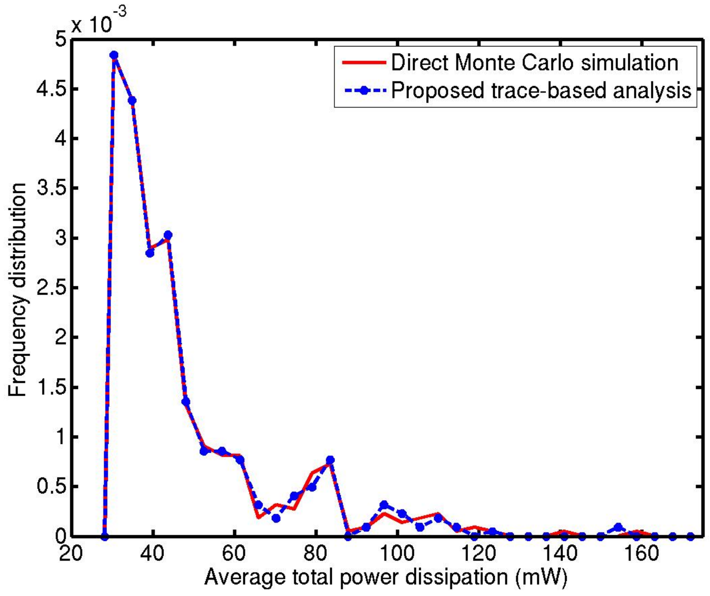


Figure 2.9: Accuracy Comparison with Monte Carlo

Table 2.1 compares the 'power yield', *i.e.,* the fraction of manufactured chips that meet a specified power constraint during normal operation, as predicted by the proposed trace-based analysis with that predicted by direct Monte Carlo simulations and spreadsheet-based analysis. The table contains results for four power constraints. We observe that our approach results in negligible accuracy loss with respect to direct Monte Carlo simulations (rows 3-5). We also note that there is no accuracy loss when trace compaction techniques are employed. This is because temperature variations within power-state phases are not significant enough to introduce error in leakage power estimation due to averaging of temperature values. Rows 6 and 7 present the power yield estimates obtained using spreadsheet-based analysis. In the pessimistic approach (row 6), all the devices are assumed to be leaking all the time. Therefore, the SoC leakage distribution ($\mu = 7.4\ mW$, $\sigma = 12.8\ mW$), is re-estimated at the typical operating temperatures and is directly added to the dynamic power estimate. In the optimistic approach (row 7) the idle components are assumed to be in deep-sleep state, thereby consuming negligible leakage power in those periods. It can be seen that the spreadsheet-based analysis can only provide loose bounds on the actual power yield. The proposed methodology accurately captures not only the times spent by each SoC component in each of its power-states, but also the die temperatures of each component, hence, enabling a more accurate consideration of their contributions to the total system power.

Table 2.1: Accuracy comparison

Analysis Methods \ Power Constraints		50mW (%)	60mW (%)	80mW (%)	100mW (%)
Direct Monte Carlo Analysis		67	78	87	94
Trace Based Analysis	Level 1 compaction	67	78	88	95
	Level 2 compaction				
	Level 3 compaction				
Spreadsheet-based Analysis	Pessimistic	60	70	82	89
	Optimistic	74	84	95	98

Power Variability Profile: Figure 2.10 shows the temporal variation in the system-level power dissipation and its spread due to process variations, as generated by our methodology. The bottom and top waveforms represent the 25^{th} and 75^{th} percentiles of the power distribution, respectively, while the waveform in the middle represents the median. The difference between the top and bottom waveforms represents the inter-quartile range of power dissipation (a measure of spread). This information can be used to identify intervals during which the system's power variations are high, and the corresponding power-state combinations of SoC components. Such information could be used to modify the system architecture, or to design appropriate power management schemes.

2.7.3 Efficiency Comparison

We now present results to evaluate the efficiency of our proposed methodology. It takes $6.4\ hrs$ to generate $500\ ms$ long dynamic power and power-state traces during Phase 1 (system simulation and power analysis). Phase 2 takes $560s$ to process

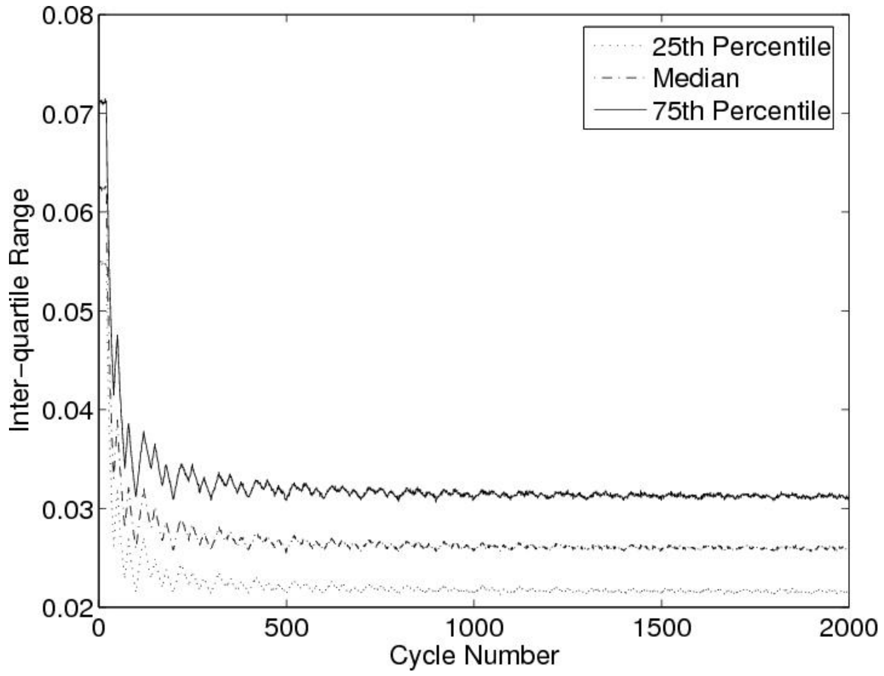


Figure 2.10: Output of the analysis tool: power variability profile over time

this trace, when a sampling interval of $1\ ms$ is employed by the thermal analysis tool for 10 equi-spaced points (N_{calib}=10). Phase 3 execution times for a sample size of 500 points are presented in Table 2.2 (col 3-5). We estimated that direct Monte Carlo simulations will take about 68 days to finish this 500 point simulation. Therefore, in this table, we compare the execution time of Phase 3 of the proposed methodology with an optimized version of direct Monte Carlo simulations[3], in which power-state based leakage modeling and trace analysis is used while performing thermal modeling within the Monte Carlo loop (see Section 2.4.1). We observe that with proposed techniques, efficiency gains of 3-4 orders of magnitude can be achieved over the optimized direct Monte Carlo simulations.

In Figure 2.11, we compare the execution time taken to generate the total power distributions using the optimized direct Monte Carlo simulations and the proposed methodology as trace length increases. It can be seen that the time taken by optimized direct Monte Carlo simulations grows very fast and the method soon be-

[3]We refer to it as ‘optimized direct Monte Carlo simulations’.

Table 2.2: Efficiency comparison

	Optimized direct Monte Carlo Analysis (s)	Trace Based Analysis		
		No trace compaction (s)	Trace compaction with temporal information (s)	Trace compaction without temporal information (s)
Time taken	24479	6	<1	<1
Speed up	1	4080	24479	24479

comes computationally infeasible. On the other hand, our methodology exhibits much slower growth in execution time, enabling analysis of very long traces.

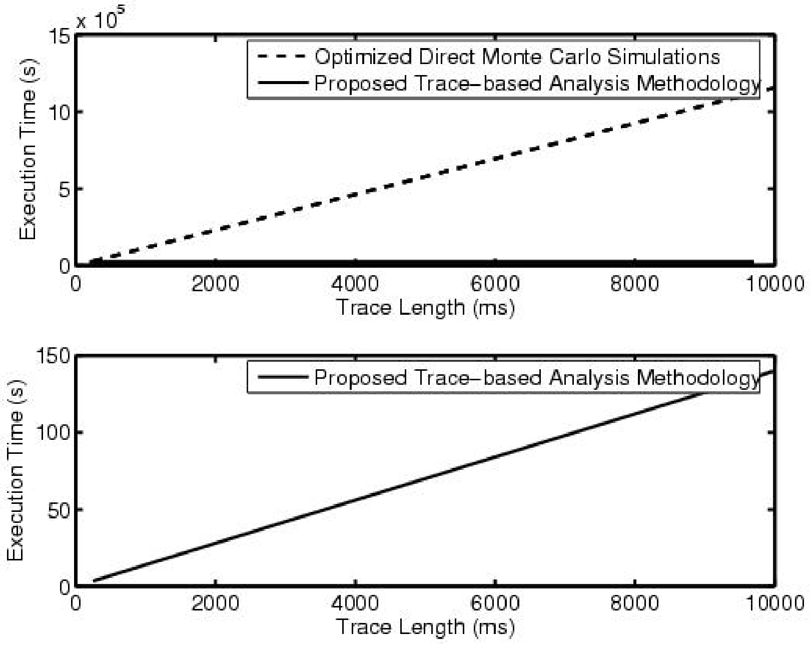


Figure 2.11: Execution time vs. trace length

In Figure 2.12, we present execution times for the two methodologies with increasing number of sample points. It can be seen that, the execution time for optimized direct Monte Carlo simulations increases linearly with a large slope as the number of sample points increase. On the other hand, the execution time for our methodology does not increase as fast. This is because in the proposed methodology, time consuming steps are not part of the Monte Carlo simulation phase (Phase 3). Our methodology enables analysis over a large sample space and hence, a more accurate estimation of the power distribution.

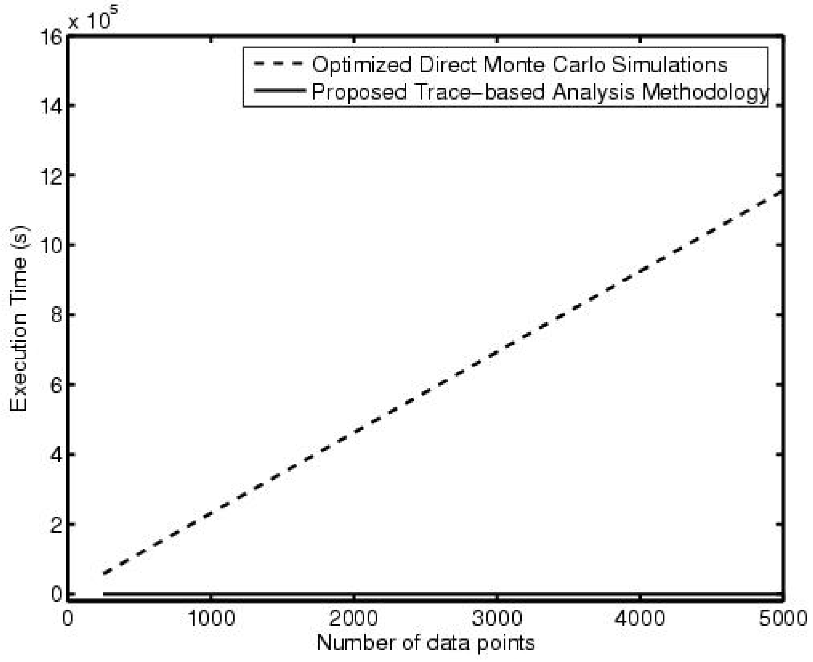


Figure 2.12: Execution time vs. sample size

2.7.4 Impact of N_{calib} on accuracy and efficiency

We now present results to study the impact of number of *fixed sample points*, N_{calib}, 'selected' in Phase 2 of the methodology on accuracy and efficiency. For these experiments, we again consider a workload of 500 ms, with 50% system activity and a sample size of 500 points. N_{calib} is varied between 2 and 20. We plot the accuracy in terms of relative error (compared to direct Monte Carlo simulations) and efficiency in terms of the computational time for Phase 2, in Figure 2.13. Note that N_{calib} does not impact the computational time for Phase 1 and Phase 3 of the methodology. From the figure, we see that high accuracy can be achieved with relatively small number of bins. For example, relative error of less than 1% can be achieved with as low as 6 bins whereas relative error of less than 0.5% can be achieved with 10 bins. For these experiments we have used $\sigma_{inter} = 8\%$ and $\sigma_{intra} = 6\%$ in L_{eff}. For larger variation in process parameters, more number of bins will be required to achieve a given accuracy target. In terms of efficiency, Phase 2 computational time varies almost linearly with N_{calib} but remains with-in an order of few minutes. Therefore, using our methodology N_{calib} can be used as a parameter that can be easily configured according to how quickly the designer aims to generate the power distribution.

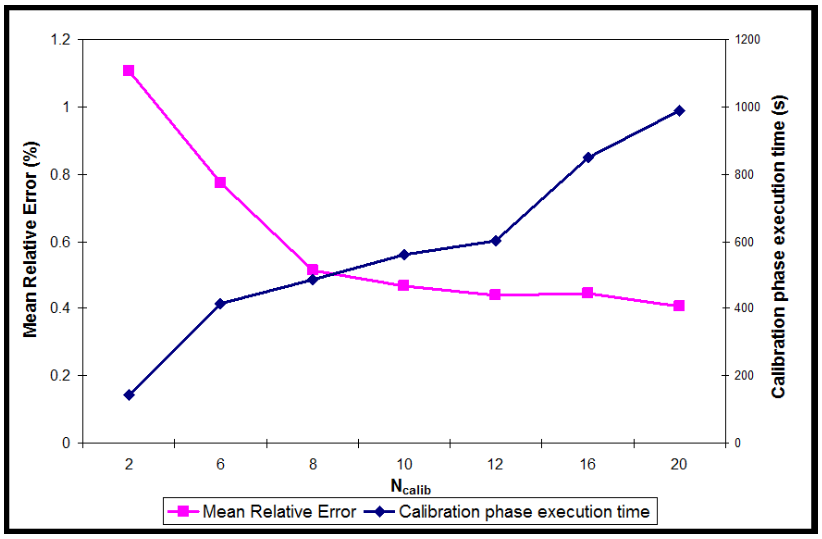


Figure 2.13: Effect of N_{calib} accuracy and *Phase 2* computational time

2.8 Conclusions

In summary, we described techniques for systematically taking manufacturing and operation-induced variations into account during system level power analysis. In our work, we found that a three phase approach, based on fast trace analysis, power-state based leakage power modeling, and off-line temperature calibration, provides an efficient and accurate means of analyzing the impact of parameter variations on power consumption at the system level.

The text of this chapter, in part, is based on material that has been published in the Proceedings of ACM/IEEE International Symposium On Low Power Electronics and Design, 2006 (S. Chandra, K. Lahiri, A. Raghunathan, and S. Dey, *Considering*

Process Variations During System-Level Power Analysis, Proc. International Symposium on Low Power Design, pp. 342 345, Oct 2006) and accepted for publication in the IEEE Transactions on VLSI Systems (S. Chandra, K. Lahiri, A. Raghunathan, and S. Dey, *Variation-aware System-level Power Analysis*, IEEE Trans. VLSI Systems). The dissertation author was the primary researcher and author, and the coauthors listed in these publications collaborated on, or supervised the research that forms the basis for this chapter.

Chapter 3

Variation-Tolerant Dynamic Power Management at the System-Level

3.1 Introduction

As discussed earlier in this thesis, the impact of variations on leakage power is very pronounced (as has been reported widely in the literature [7, 37, 38]), making it imperative to consider variations during power analysis and optimization. In this chapter, we consider the impact of process variations on shutdown-based power management, which is one of the most commonly used power optimization techniques. In shutdown-based power management, a component is put into low power states during periods of inactivity. Examples of shutdown-based power management include timeout-based policies, history-based policies and stochastic policies, [57, 58, 59, 60]. Current approaches for designing power management schemes assume that components have deterministic power characteristics (*e.g.*, power consumption in each power state, transition overheads between states, *etc.*). These assumptions break down in the presence of variations and hence it is natural to question the effectiveness of power management schemes designed on their basis.

3.1.1 Chapter Overview and Contributions

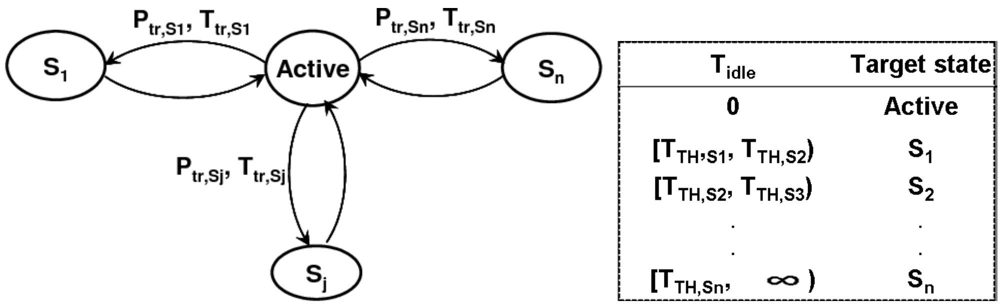
In this chapter, we make the following contributions.

- We study the impact of variations on power management, and show that conventional approaches, which do not consider variations, can result in significant sub-optimality and energy wastage.
- We show that in the presence of variations conventional metrics such as mean energy dissipation are not sufficient to measure the effectiveness of a power management policy. We suggest alternative metrics such as $\mu + \sigma$, or the N^{th} percentile of the overall energy distribution across chips, that may be better from the perspective of variation-aware power optimization.
- We propose two strategies to design effective, variation-aware aware power management schemes, namely, *design-specific* and *chip-specific* approaches. In the design-specific approach, the set of parameter values that is fixed across all fabricated instances of the chip is derived with a goal of optimizing a specific distribution metric whereas in the chip-specific approach, parameters values are uniquely determined for each chip instance based on its individual power characteristics. These strategies are applicable to other known power management policies such as timeout-based, predictive and stochastic, by adapting the appropriate policy parameters in each case. Note that, our objective in this work is not to propose new power management policies; rather, we wish to examine how existing policies can be adapted considering the effects of variations.
- We present these approaches in the context of an ideal oracle-based prediction framework in order to study the impact of variations independent of the problem of idle time prediction. We then perform additional analysis to study the impact of variations on timeout-based policies which are perhaps the most widely used in practice.

We evaluate the proposed approaches to variation-aware power management of SoCs, using a cycle-level power and performance model of an ARM946 processor core for a range of workloads. The proposed approaches result in large improvements in metrics such as $\mu + \sigma$ and Nth percentile over conventional policies. For example, improvements of up to 59% for the oracle-based framework and up to 43% for the timeout-based framework are obtained for metric $\mu + \sigma$.

3.1.2 Related Work

A significant body of research over the past few years has focused on developing techniques to estimate and to reduce the impact of variations on the leakage power. Some of the estimation techniques are presented in [1, 11, 12, 13]. Techniques to reduce the impact of variations on leakage power include adaptive body biasing (ABB), gate sizing, dual threshold voltage-based design and repeater insertion [7, 22, 25, 24, 21, 23, 6]. These techniques are shown to be promising but do not completely address the problem since they introduce variation-awareness at later stages in the design cycle. Moreover, techniques such as ABB to reduce leakage variations are likely to be less effective with technology scaling [61, 62]. Some of the recent research in this area addresses the problem at the architectural and system levels. In [17, 32], analytical models are developed to study the impact of variations on overall performance and power consumption for different micro-architectural choices and single and multiple voltage-frequency systems. The effects of variations on system-level behavior of SRAM memories are studied in [31]. In [63], trace-based analysis techniques are proposed for efficient variation-aware system-level power estimation. Efforts at architectural and system-level to improve power efficiency in the presence of variations include exploiting parallelism [18] and using optimal number of cores [30]. While these works clearly demonstrate a move to consider variations at higher levels of abstraction, the impact on the commonly used technique of shutdown-based power management has not been studied to date.



T_{idle}	Target state
0	Active
$[T_{TH,S1}, T_{TH,S2})$	S_1
$[T_{TH,S2}, T_{TH,S3})$	S_2
.	.
.	.
$[T_{TH,Sn}, \infty)$	S_n

Figure 3.1: Power state machine and power management policy

3.2 Preliminaries

In this section, we describe the conceptual framework around which the discussion in rest of this chapter is based. We model an SoC as a collection of *power manageable components (PMCs)*. During system execution, each component exists in one of several *power states* each with differing power consumption characteristics. A *power management policy* specifies the conditions for the component to transition between power states with an objective of minimizing the total energy consumption. Power management policies are usually designed based on the nature of the component's workload as well as its power characteristics.

3.2.1 Power Manageable Component

Power state machine of a *PMC* under an oracle-based framework is shown in Figure 3.1. The *PMC* can be in the *Active* state or it can be in one of the n low-power states from S_1 to S_n. Examples of low-power states in practice are *Doze*, *Sleep*, *Hibernate*, *Deep-sleep*, *etc.* While in the *Active* state, if the component is not performing any useful operation it is said to be *Idle*. When *Idle*, a component consumes lower dynamic power compared to when it is actively operating. Transition to a low-power state can be made during these idle time periods to avoid unnecessary power dissipation. A component can transition between low power states as well, for example from *Sleep* to *Deep-sleep*. However, under the assumption of oracle prediction of idle periods, the length of the idle period, T_{idle}, is known apriori to the power manager. Hence, it is possible to calculate the optimal low-power state and directly transition to it from the *Active* state at the beginning of the idle period itself, thereby eliminating any possibility of transition between lower power states. In the power state machine of Figure 3.1, only the transitions used under an oracle-based policy are shown.

The power dissipation of the component in the low-power state S_j is denoted by P_{S_j}, consisting of the dynamic power dissipation P_{D,S_j} and the leakage power dissipation, P_{L,S_j}. Transition between *Active* and S_j is associated with a power overhead denoted by P_{tr,S_j} and a delay overhead denoted by T_{tr,S_j}. We use the vector $\bar{P}$ to represent the power dissipation characteristics of the component *i.e.*, $\bar{P} = \{P_A, P_{S_1}, \ldots, P_{S_n}, P_{tr,S_1}, ..., P_{tr,S_n}\}$. For a given T_{idle}, the PMC transitions to a unique low-power state such that the total energy dissipation during T_{idle} is minimized.

3.2.2 Power Management Policy

The corresponding power management policy is completely specified if the idle time ranges for which the component should transition into each of the low-power states from the active state are determined. These ranges can be represented as a set of threshold values, denoted by T_{th,S_j}. If $T_{idle} \in [T_{th,S_j}, T_{th,S_{j+1}})$, the optimal target state is S_j and so on. For each possible value of T_{idle} there is exactly one low-power state which minimizes the energy dissipation. Typically, in-state power consumption P_{S_j} gets smaller with lower power states, from S_1 to S_n, *i.e.,* $P_A \geq P_{S_1} ... \geq P_{S_n}$ and transition overheads increase with lower power states, *i.e,* $P_{tr,S_1} \leq ... \leq P_{tr,S_n}$, and, $T_{tr,S_1} \leq ... \leq T_{tr,S_n}$. Therefore, it is beneficial to transition to lower power states only if the idle period length is long enough *i.e.,* $T_{th,S_1} < T_{th,S_2} < ... < T_{th,S_n}$. In other words, lower power states become optimum with increasing idle time. The oracle-based policy is optimum under the above described power management framework. Energy dissipation is minimized for each T_{idle} by transitioning to the most appropriate state and therefore, total energy consumption under any workload, $\mathcal{W} = \{T_A, \mathbf{T}_{idle}, \mathbf{N}_{idle}\}$ is minimized. Here, T_A is the total active time, $N_{idle} \in \mathbf{N}_{idle}$

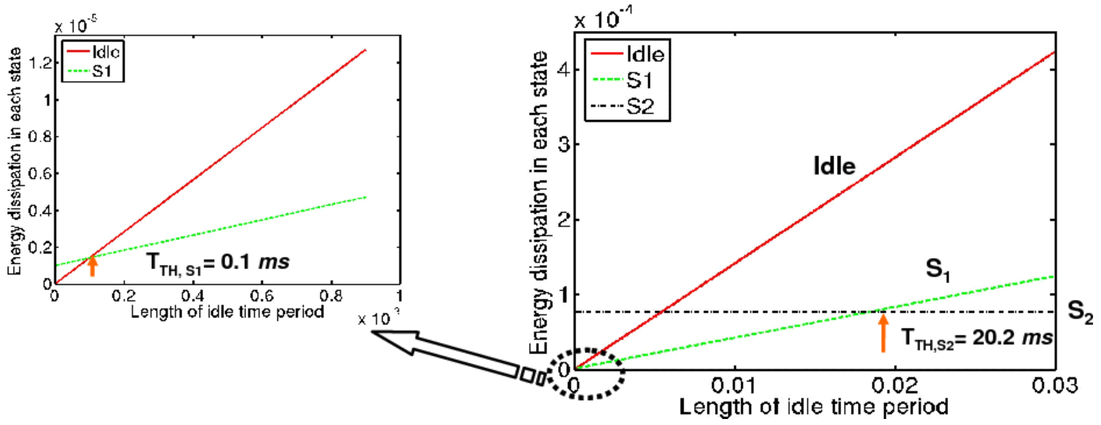


Figure 3.2: Energy dissipation in a given state as a function of T_{idle}

represents the number of times, idle duration of $T_{idle} \in \mathbf{T}_{idle}$ occurs in the workload.

Now we describe the derivation of the optimal parameter values, *i.e.,* $T_{th,S_1} \ldots T_{th,S_n}$. Total energy dissipation of a *PMC*, transitioned to a low-power state S_j during an idle time period $T_{idle}(> T_{tr,S_j})$, consists of energy dissipation during transition and in the low-power state S_j, *i.e.,*:

$$\begin{aligned} E(\bar{P}, S_j, T_{idle}) &= P_{tr,S_j} T_{tr,S_j} + P_{S_j}(T_{idle} - T_{tr,S_j}) \\ &= (P_{tr,S_j} - P_{S_j}) T_{tr,S_j} + P_{S_j} T_{idle} \end{aligned} \tag{3.1}$$

Therefore, for each state S_j, E is a linear function of T_{idle}, as shown in Figure 3.2 for three example states.

Figure 3.2 shows that state S_1 has the least energy dissipation for all idle periods such that $T_{idle} \in [T_{th,S_1}, T_{th,S_2})$. Similarly, state S_2 has least energy dissipation for all $T_{idle} \geq T_{th,S_2}$. In other words, the threshold times $T_{th,S1}$ and $T_{th,S2}$ are determined by the intersection of lines corresponding to *Active* and S_1, and S_1 and S_2, respectively. In general, for a given T_{idle},low-power state S_{j+1} is favorable over low-power state S_j, if $E(\bar{P}, S_{j+1}, T_{idle}) < E(\bar{P}, S_j, T_{idle})$. We compute the threshold $T_{th,S_{j+1}}$ as the break-even point in the trade-off between states S_j and S_{j+1}, *i.e.*, it is the value of T_{idle} that satisfies the equation $E(\bar{P}, S_{j+1}, T_{idle}) = E(\bar{P}, S_j, T_{idle})$. Using Eq. 3.1, the break-even time for a target low-power state S_{j+1} is given by:

$$T_{th,S_{j+1}} = \frac{(P_{tr,S_{j+1}} - P_{S_{j+1}}) T_{tr,S_{j+1}} - (P_{tr,S_j} - P_{S_j}) T_{tr,S_j}}{P_{S_j} - P_{S_{j+1}}} \tag{3.2}$$

In the next section, we examine the impact of leakage power variations in the context of the above framework.

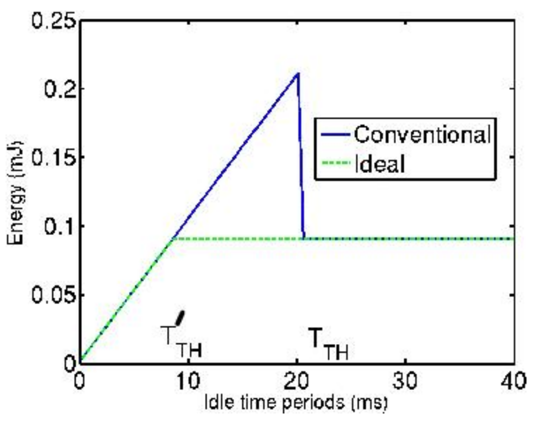
3.3 Impact of Variations

In this section, we examine the impact of leakage power variations on the effectiveness of power management under the oracle-based framework described in the previous section. In the presence of leakage power variations, power consumption in a given state, P_{S_j}, and during transitions, P_{tr,S_j} is a random variable rather than one deterministic value. In such a scenario, one could use either the worst-case or mean power consumption values to calculate the policy parameters, T_{th}. In this work, we assume that using mean of the power distribution to calculate policy parameters is a reasonable baseline case and refer to it as a conventional policy or a mean-based policy. We analyze the effectiveness of such a policy under variations, using ARM946 processor core [42] as the example component. In particular, we consider three of the ARM946 core power states, namely, *Active*, *Sleep* and *Deep-sleep*.

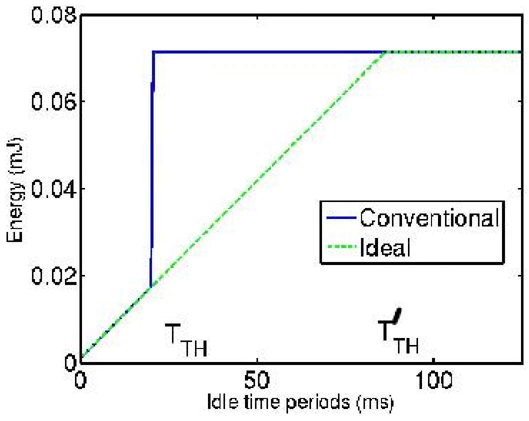
Active: In this state, the component is powered on, and performs useful operations. The dynamic power consumption ($P_{D,A}$) is large due to high switching activity in this state. When the component is not performing any useful operations and not transitioned to a low-power state, it is said to be *Idle* and the $P_{D,A}$ is much smaller. Also, since most of the component is powered on, active leakage ($P_{L,A}$) is a key contributor to the total power consumption, P_A.

Sleep: In this low-power state, clock gating is employed to largely eliminate dynamic power. However, since the component remains powered on, device leakage currents are unaffected. Hence, power consumption in this state (P_S) is dominated by leakage power ($P_{L,S}$). Transition from the *Active* state incurs delay and power costs of $T_{tr,S}$ and $P_{tr,S}$, respectively.

Deep-sleep: In this low-power state, power-supply gating is used to power-down as much of the component as possible (barring circuits required for state restoration). Hence, the power consumed in this state (P_{Ds}) is negligible with significantly re-



(a) Chip $P_L^i > \mu[P_L]$



(b) Chip $P_L^i < \mu[P_L]$

Figure 3.3: Illustrating energy wastage under a PM policy that uses parameters based-on conventional break-even analysis

duced dynamic and leakage components. However, the associated transition overheads $(P_{tr,Ds}, T_{tr,Ds})$ are significantly higher than those associated with the *Sleep* state.

For this study, we assume that the ARM946 core is implemented in a $90nm$ process technology known to exhibit $3\sigma = 30\%\mu$ variations in the channel length. We use the leakage power model presented in [63] to determine the leakage power distribution of the component in each of its power-states. Leakage power in the *Active* and *Sleep* states are random variables represented by $\mathcal{P}_{L,A}$ and $\mathcal{P}_{L,S}$[1], respectively. In this example, we assume that the leakage power is same in the *Active* and *Sleep* states and denote it by random variable $\mathcal{P}_L$. For the example component ARM946 core, $3\sigma = 30\%\mu\ L_{eff}$ variation results in a log-normal distribution for $\mathcal{P}_L$ with parameters $\mu = 4\ mW$, $\sigma = 7.5\ mW$. We use this data to obtain the threshold values associated with the ***Deep-sleep*** and *Sleep* states by using $P_{S_j} = P_{D,S_j} + \mu(P_{L,S_j})$ in Eq. 3.2. We obtained $T_{th,S} = 0.1\ ms$ and $T_{th,Ds} = 20.2\ ms$. Note that $T_{th,S}$ is independent of leakage characteristics, since leakage in the *Active* and *Sleep* states cancels out when using Eq. 3.2. Therefore, in the following discussion, for brevity, we use T_{th} to refer to $T_{th,Ds}$. We now consider two die samples with contrasting leakage characteristics, P_L^1 and P_L^2, and analyze the effectiveness of the policy with threshold

[1]We use calligraphic capital letters such as $\mathcal{P}$ and $\mathcal{E}$ to represent random variables.

value for the *Deep-sleep* state as calculated above.

Die 1 — $P_L^1 > \mu[\mathcal{P}_L]$: This case represents a chip with higher than average leakage. Figure 3.3(a) depicts variation of energy dissipation with the idle period length. The bold line corresponds to the policy based on a T_{th} of $20.2\ ms$. The dotted line corresponds to a policy based on a $T_{th}\prime$ of $8.5\ ms$, obtained by using P_L^1 rather than $\mu[\mathcal{P}_L]$ in Eq. 3.2. Note that, $T_{th}\prime$ is the *ideal* threshold value for the given instance of the chip since is it based on the chip's very own power characteristics. From the figure, we observe that the deviation of leakage from the mean leakage has effectively forced $T_{th}\prime$ towards smaller values. This is because for this sample, higher leakage implies that it is more attractive to pay the transition penalty and exploit the *Deep-sleep* state even for certain idle periods that are smaller than the conventionally chosen value of T_{th}.

Die 2 — $P_L^2 < \mu[\mathcal{P}_L]$: This case represents a chip with lower than average leakage. The results of similar studies for this case are presented in Figure 3.3(b). In this case, the sample is less leaky than expected, which forces $T_{th}\prime$ towards higher values. This is because for certain idle periods that are longer than the conventional T_{th}, it remains preferable to avoid the penalty of state transition to the *Deep-sleep* state.

It can be observed from the above example that for idle periods lying between the conventional and ideal T_{th} values, the conventional policy results in higher energy dissipation. This study highlights the fact that leakage power variations cause power state characteristics, and hence the effectiveness of power management policies, to vary from one chip to another. Therefore, we conclude that, under variations, power management policies can potentially perform quite sub-optimally for certain chip instances and the extent of sub-optimality depends upon the individual power characteristics of a chip and the idle period distribution in the workload.

3.4 Metrics of Optimization

In the presence of process variations, total energy dissipation differs from one chip to another under a given policy for a given workload and hence, is a distribution

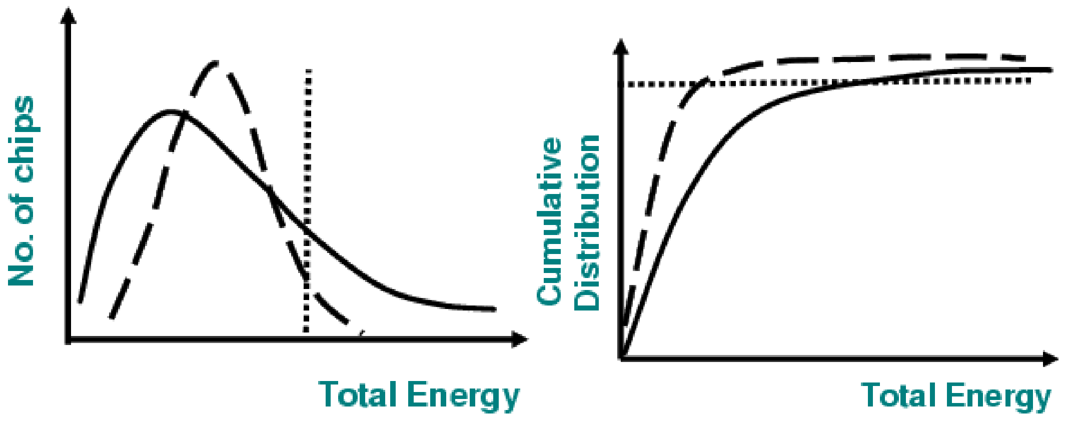


Figure 3.4: Desirable metrics for power management policy evaluation

across chips. Therefore, the effectiveness of power management must be evaluated in the context of an *energy distribution* rather than a deterministic measure of the energy consumption. One possible metric is the mean of the energy distribution. However, other metrics may be more desirable in certain cases. We argue that traditionally used metric i.e., mean of the distribution does not provide a sufficient measure. To see that, consider two distributions as shown in Figure 3.4(a). The distribution with solid envelope has a lower mean compared to the distribution with dashed envelope. However, the dashed distribution might be more desirable since it provides a better guarantee in terms of the number of chips meeting a certain energy constraint. Therefore, higher moments of the energy distribution should be taken into account to quantify it. For example a simple metric can be a function of μ and σ, such as $\mu + \sigma$. Another good metric is the N^{th} percentile of the distribution. Similarly, in Figure 3.4(b), it can be seen that the dashed cumulative distribution (CDF) is more desirable since it has a lower value of the given percentile and hence, corresponds to larger number of chips with better energy characteristics. This is analogous to the frequency binning motivation in which semiconductor engineers want to increase the number of chips in the highest frequency bin since that earns the highest revenue. In the rest of this chapter, we focus on three metrics to compare distributions under different policies, namely, μ, $\mu + \sigma$ and the N^{th} percentile.

We now present two approaches to variation-aware power management, namely the design-specific approach and the chip-specific approach.

3.5 Design-Specific Approach

In the *design-specific* approach, policy parameters are fixed for all fabricated chip instances of a design. These parameters are selected to optimize the overall energy distribution under a given workload with respect to a chosen metric. For this discussion, we denote a conventionally configured policy (based on mean leakage characteristics) by PO^M and the corresponding parameter set by T_{th}^M[2]. The optimized variation-aware design-specific policies are denoted as PO^{VA-DS} and the corresponding parameter set as T_{th}^{VA-DS}. In this section, we first derive the policy parameters that minimize the mean of the energy distribution and then present a general method to derive parameters that optimize other metrics of interest.

3.5.1 Minimizing Mean Energy : Mean-based policy is optimal

We claim that, in order to minimize the mean of the energy distribution, the policy parameters derived assuming mean leakage characteristics are in fact optimal, *i.e.*, $T_{th}^{VA-DS} = T_{th}^M$. Let us denote the policy corresponding to T_{th}^M as PO^M and the one corresponding to T_{th}^{VA-DS} as PO^{VA-DS}. Consider the power-state machine of Figure 3.1 and the example PMC of Section 3.3. As mentioned earlier, threshold time to ***Sleep***, $T_{th,S}$ is independent of leakage characteristics, and hence can be chosen conventionally. Here we focus on how to select $T_{th,Ds}$. Under PO^M, the threshold value associated with the ***Deep-sleep*** state is determined by solving $E(\mu[\bar{\mathcal{P}}], Ds, T_{idle}) = E(\mu[\bar{\mathcal{P}}], S, T_{idle})$. Therefore, for $T_{idle} < T_{th}^M$, ***Sleep*** is more favorable than ***Deep-sleep*** and vice-versa, *i.e.*,

$$E(\mu[\mathcal{P}], S, T_{idle}) < E(\mu[\mathcal{P}], Ds, T_{idle}),\ for\ T_{idle} < T_{th}^M \tag{3.3a}$$

$$E(\mu[\mathcal{P}], Ds, T_{idle}) < E(\mu[\mathcal{P}], S, T_{idle}),\ for\ T_{idle} > T_{th}^M \tag{3.3b}$$

[2]Note that, the conventional policy is same as the μ-based design-specific policy

Now, the two policies behave identically for idle time periods either less or greater than both thresholds. For $T_{idle} < min(T_{th}^{VA-DS}, T_{th}^{M})$, all the chips transition to *Sleep* under both the policies. Similarly, for $T_{idle} > max(T_{th}^{VA-DS}, T_{th}^{M})$, all the chips transition to the *Deep-sleep* state. For remaining idle times, the policies behave differently as described in the following two cases.

Case A: $T_{th}^{DS} \leq T_{idle} < T_{th}^{M}$ — In this case, the chips transition to *Sleep* under PO^{M} and to *Deep-sleep* under PO^{VA-DS}. Let, $\mathcal{E}_{T_{idle}}^{PO}$ represent the energy random variable for T_{idle} under the policy *PO*. Using Eq. 3.1 it can be seen that the mean of the energy random variable is same as the energy dissipation of a sample with mean leakage energy, *i.e.,*

$$\begin{aligned} \mathbf{P}O^{M} &: \mu[\mathcal{E}_{T_{idle}}^{M}] = \mu[\mathcal{E}(\bar{\mathcal{P}}, S, T_{idle})] = E(\mu[\bar{\mathcal{P}}], S, T_{idle}) \\ \mathbf{P}O^{VA-DS} &: \mu[\mathcal{E}_{T_{idle}}^{VA-DS}] = \mu[\mathcal{E}(\bar{\mathcal{P}}, Ds, T_{idle})] = E(\mu[\bar{\mathcal{P}}], Ds, T_{idle}) \end{aligned}$$

From Eq. 3.3a, we know that if $T_{idle}<T_{th}^{M}$, then $E(\mu[\mathcal{P}], s, T_{idle})<E(\mu[\mathcal{P}], Ds, T_{idle})$, therefore $\mu[\mathcal{E}_{T_{idle}}^{M}] \leq \mu[\mathcal{E}_{T_{idle}}^{DS}]$. For the given T_{idle}, mean of the energy distribution under PO^{M} is smaller than the mean of the distribution under PO^{VA-DS}.

Case B: $T_{th}^{M} \leq T_{idle} < T_{th}^{DS}$ — In this case all the chips transition to *Deep-sleep* under PO^{M} and to *Sleep* under PO^{DS}. From analysis similar to that in **case A** and Eq. 3.3b we again get $\mu[\mathcal{E}_{T_{idle}}^{M}] \leq \mu[\mathcal{E}_{T_{idle}}^{DS}]$.

This shows that for any given T_{idle}, mean energy is minimized under PO^{M} *i.e.,* $\mu[E_{T_{idle}}^{M}] \leq \mu[E_{T_{idle}}^{DS}]$. Therefore, for any given workload ($\mathcal{W} = \{T_A, \mathbf{T}_{idle}, \mathbf{N}_{idle}\}$), mean of the total energy, $\mu[E_{TOT}]$, is minimized under PO^{M} as shown below,

$$\mu[\mathcal{E}_{TOT}] = \mu[\mathcal{E}_A + \sum_{T_{idle}} N_{idle} * \mathcal{E}_{T_{idle}}^{M}] = \mu[\mathcal{E}_A] + \sum_{T_{idle}} N_{idle} * \mu[\mathcal{E}_{T_{idle}}^{M}]$$

Therefore, to minimize the mean energy, $T_{th}^{DS} = T_{th}^{M}$. This result holds for a generic system with $n(> 2)$ low-power states.

3.5.2 Optimizing Other Distribution Metrics

In this section, we first show that distribution metrics such as ($\mu + \sigma$) or the percentile values introduced in section 3.4 are not optimized under a μ-based conventional policy (PO^M) . We, then present an analytical method to obtain policy parameters to optimize the desirable distribution metric.

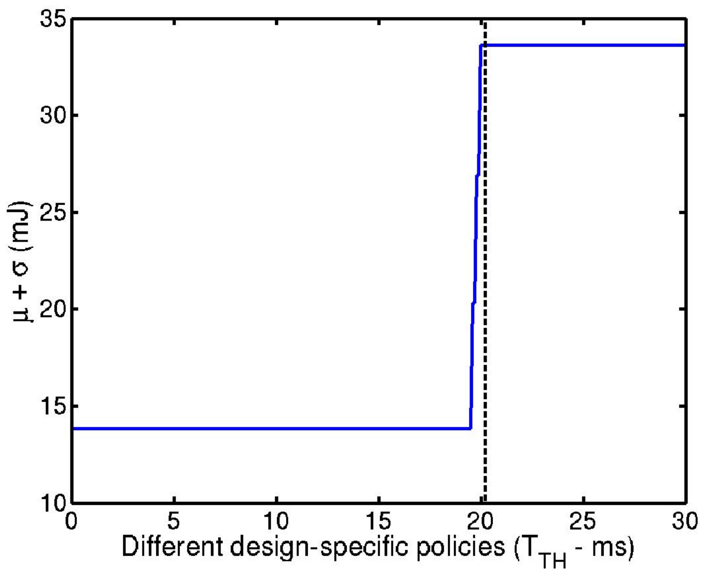
$\mathcal{D}istribution\ metrics\ under\ PO^M$:

In Figure 3.5(a) we plot the variation of ($\mu + \sigma$) of the total energy distribution under a certain workload for different configurations of the policy parameter T_{th}. The dotted vertical line indicates conventional policy with $T_{th} = T_{th}^M$. The figure shows that the metric ($\mu + \sigma$) is highly sensitive to the value of the policy parameter. In particular, policies with $T_{th} < 20\ ms$ are able to achieve 59% reduction in ($\mu + \sigma$), over the conventional mean-based policy. The CDF of the total energy distribution obtained under a policy PO^{DS} ($T_{th}^{DS} = 17\ ms$) is compared with PO^M ($T_{th}^M = 20.2\ ms$) in Figure 3.5(b). It is clearly seen that the variation-aware design-specific policy ($T_{th} = 17\ ms$) provides a superior energy distribution as compared to the mean-based policy. Similar conclusions can be drawn for other metrics, such as percentiles, energy yield, *etc.*

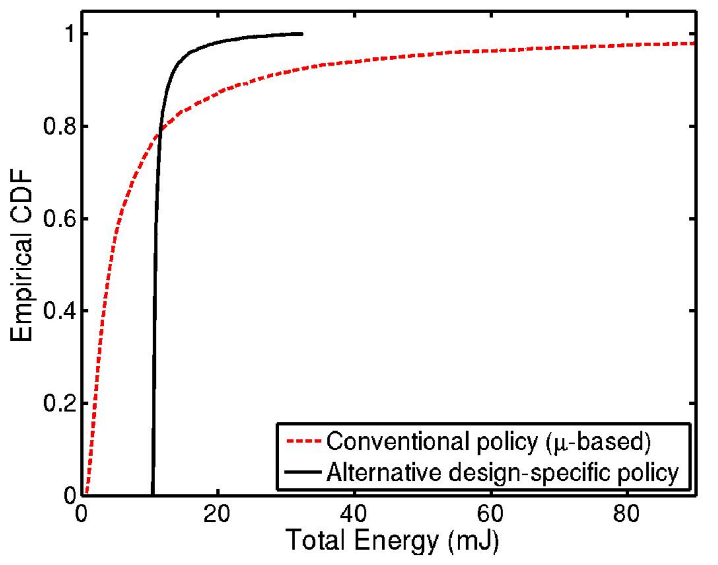
$\mathcal{D}eriving\ T_{th}^{VA-DS}$ *to optimize other distribution metrics* :

Now we present an analytical technique to derive optimized policy parameters for metrics such as ($\mu + \sigma$) and N^{th} percentile. We first show that if a given metric $\mathcal{M}et$ satisfies the linearity property then the optimized parameters can be obtained by simple break-even analysis. Then we show that the property holds for the metrics ($\mu + \sigma$) and the N^{th} percentile.

Consider the power state machine of Figure 3.1. The number of times a low power state occurs in a workload depends on the idle time distribution and the policy parameter $T_{th} = \{T_{th,S_1}, T_{th,S_2}, ..., T_{th,S_n}\}$ that determines the target power state for each idle period length. The total energy dissipation under a workload $\mathcal{W}$ is the sum of the energy dissipation for each idle period length weighted by the frequency of occurrence of the idle period length. From the discussion in Section 3.2, the threshold times to enter a state increase from states S_1 to S_n, *i.e.*, $T_{th,S_1} < T_{th,S_2} < ... < T_{th,S_n}$. Hence,



(a)



(b)

Figure 3.5: (a) Effectiveness of different design-specific policies for the $\mu + \sigma$ metric and, (b) the energy CDFs

total energy dissipation can be decomposed in the constituent states as follows:

$$\begin{aligned}\mathcal{E}_{TOT} =& \mathcal{P}_A * T_A + \sum_{T_{idle}} N_{idle} * \mathcal{E}_{idle} \\ =& \mathcal{P}_A * T_A + \sum_{T_{idle}=T_{th,S_1}}^{T_{th,S_2}} N_{idle} * \mathcal{E}(\mathcal{P}, S_1, T_{idle}) + \\ & ... + \sum_{T_{idle}=T_{th,S_n}}^{\infty} N_{idle} * \mathcal{E}(\mathcal{P}, S_n, T_{idle})\end{aligned} \tag{3.4}$$

We want to calculate the policy parameter $T_{th} = \{T_{th,S_1}, T_{th,S_2}, ..., T_{th,S_n}\}$ such that the desirable metric of total energy distribution $\mathcal{M}et[\mathcal{E}_{TOT}]$ is optimized. We show that if the $\mathcal{M}et$ of the distribution is linear *i.e.,* satisfies **property 1**, then the optimum parameter set can be obtained using a simple break-even analysis.

Property 1: Metric $\mathcal{M}et$ is linear *i.e.,* it holds true for a linear combination of random variables. Using Eq. 4.2,

$$\mathcal{M}et[\mathcal{E}_{TOT}] = \mathcal{M}et[\mathcal{P}_A * T_A] + \sum_{S_j=1}^{n} \sum_{T_{th,S_j}}^{T_{th,S_{j+1}}} N_{idle} * \mathcal{M}et[\mathcal{E}(\mathcal{P}, S_j, T_{idle})] \tag{3.5}$$

Each term in the double summation is $\mathcal{M}et$ of the energy dissipation in state S_j during T_{idle}. Eq. 3.5 shows that $\mathcal{M}et$ of the total energy distribution can be represented as the sum of the $\mathcal{M}et$ of the energy distribution in each idle period weighted by the frequency of occurrence of that idle period. Therefore, if the metric is optimized within each of these terms then the linear combination thereof will be optimized for all workloads $\mathcal{W}$ as $N_{idle} \geq 0$. Therefore, for each T_{idle}, if the states are chosen such that the metric is optimized in each of these terms, then the total energy distribution is optimized with respect to $\mathcal{M}et$. This problem is similar to the original problem of T_{th} calculation to minimize total energy of a given chip instance, presented in Section 3.2 of this chapter. Therefore, a similar break-even analysis but for $\mathcal{M}et[E(\bar{P}, S_j, T_{idle})]$ (instead of $E(\bar{\mathcal{P}}, S_j, T_{idle})$) can be employed to calculate the optimum set of T_{th} values under the specified metric $\mathcal{M}et$. Following this approach we write Eq. 3.2 as,

$$T_{th,S_{j+1}} = \frac{(\mathcal{M}et[P_{tr,S_{j+1}}] - \mathcal{M}et[P_{S_{j+1}}])T_{tr,S_{j+1}} - (\mathcal{M}et[P_{tr,S_j}] - \mathcal{M}et[P_{S_j}])T_{tr,S_j}}{\mathcal{M}et[P_{S_j}] - \mathcal{M}et[P_{S_{j+1}}]}$$

Next, we analyze the above property for a few desirable metrics introduced earlier in this chapter. Before we present analysis for the individual metrics, we make an important observation that allows us to reduce the number of random variables from the number of terms in the *double summation* (Eq. 3.5) to 1. Note that, leakage power in given power state or state transition is a fraction (α) of the total leakage power of the chip depending on the number of leaky transistors in that state or state transition. Let the vector $\bar{\alpha} = \{\alpha_A, \alpha_{S1}, \ldots, \alpha_{Sn}, \alpha_{tr,S1}, ..., \alpha_{tr,Sn}\}$, represent the leakage of the component in each of its power states and state transitions relative to the total component leakage. The dynamic power dissipation and $\bar{\alpha}$ is constant across various chip instances. Therefore, power consumption vector can be written as $\bar{\mathcal{P}} = \bar{P_D} + \bar{\alpha}\,\mathcal{P}_L$. Here, only the $\mathcal{P}_L$ term is a random variable. Therefore, using Eq. 3.1, the energy dissipation of the PMC in state S_j for an idle period of length T_{idle} can be decomposed into a constant dynamic dissipation term and a random leakage dissipation term as shown in the following equation.

$$\mathcal{E}(\bar{\mathcal{P}}, S_j, T_{idle}) = E(\bar{P_D}, S_j, T_{idle}) + \mathcal{P}_L * T_E(\bar{\alpha}, S_j, T_{idle}) \tag{3.6}$$

Here, $E(\bar{P_D}, S_j, T_{idle})$ is the dynamic energy dissipation in state for a given T_{idle} and $T_E(\bar{\alpha}, S_j, T_{idle})$ is the total equivalent time of the leakage contribution of the energy dissipation. From Eq. 3.6 it can be seen that the energy dissipation in a given power state for a given T_{idle} is a random variable that follows same distribution as the leakage power with μ and σ given by,

$$\begin{aligned}\mu[\mathcal{E}(\bar{\mathcal{P}}, S_j, T_{idle})] &= E(\bar{P_D}, S_j, T_{idle}) + \mu[\mathcal{P}_L] * T_E(\bar{\alpha}, S_j, T_{idle})\\ \sigma[\mathcal{E}(\bar{\mathcal{P}}, S_j, T_{idle})] &= \sigma[\mathcal{P}_L] * T_E(\bar{\alpha}, S_j, T_{idle})\end{aligned} \tag{3.7}$$

We use Eq. 3.6 to show that the metrics $(\mu + \sigma)$ and the N^{th} percentile of the total energy distribution satisfy property 1.

1. $\mu + \sigma$**:** Let us decompose Eq. 4.2 into the into dynamic and leakage energy terms using Eq. 3.6 and re-write it as follows,

$$\begin{aligned} E_{TOT} = & P_A * T_A + \sum_{S_j=1}^{n} \sum_{T_{idle}=T_{th,S_j}}^{T_{th,S_{j+1}}} N_{idle} * E(\bar{P_D}, S_j, T_{idle}) + \\ & P_L * \{\alpha_A * T_A + \sum_{S_j=1}^{n} \sum_{T_{idle}=T_{th,S_j}}^{T_{th,S_{j+1}}} N_{idle} * T_E(\bar{\alpha}, S_j, T_{idle})\} \end{aligned} \tag{3.8}$$

Therefore, the standard deviation of E_{TOT} is,

$$\begin{aligned} \sigma[\mathcal{E}_{TOT}] = & \sigma[P_L] * \{\sum_{S_j=1}^{n} \sum_{T_{idle}=T_{th,S_j}}^{T_{th,S_{j+1}}} N_{idle} * T_E(\bar{\alpha}, S_j, T_{idle})\} \\ = & \sum_{S_j=1}^{n} \sum_{T_{idle}=T_{th,S_j}}^{T_{th,S_{j+1}}} N_{idle} * \sigma[\mathcal{P}_L * T_E(\bar{\alpha}, S_j, T_{idle})] \end{aligned} \tag{3.9}$$

Using Eq. 3.9 and the fact that μ is linear, it can be seen that $\mu + \sigma$ satisfies Property 1 and is minimized under a parameter set (T_{th}) obtained through a simple break-even analysis using $\mu + \sigma$ of the energy dissipation $\mathcal{E}(\bar{P}, S_j, T_{idle})$ in Eq. 3.1.

2. N^{th} **Percentile :** We use $\Pi_{\mathcal{N}}$ to denote the N^{th} percentile of a distribution. From Eq. 3.6 we also observe that for given T_{idle} and S_j, the energy dissipation of a chip instance is a non-decreasing linear function of chip leakage. Hence, if N% chips exhibit leakage of P_{L_N} or less, the same N% chips will exhibit energy dissipation of $E(\bar{P_D}, S_j, T_{idle}) + P_{L_N} * T_E(\bar{\alpha}, S_j, T_{idle})$ or less. Similar arguments can be made for E_{TOT} given by Eq. 3.8. The N^{th} percentile of E_{TOT} is same as the N^{th} percentile of P_L multiplied by the term in the braces and incremented by first term. Since the underlying random variable is same, the following holds (take P_L inside the double summation),

$$\begin{aligned} \Pi_{\mathcal{N}}[\mathcal{E}_{TOT}] = & \Pi_{\mathcal{N}}[\mathcal{E}_{TOT(\bar{\mathcal{P}},A,T_A)})] + \\ & \sum_{S_j=1}^{n} \sum_{T_{idle}=T_{th,S_j}}^{T_{th,S_{j+1}}} N_{idle} * \Pi_{\mathcal{N}}[\mathcal{E}(\bar{\mathcal{P}}, S_j, T_{idle})] \end{aligned}$$

Therefore, Property 1 holds and T_{th} to optimize a given N is obtained by using $\Pi_{\mathcal{N}}[\mathcal{E}(\bar{P}, S_j, T_{idle})]$ in Eq. 3.1.

3.6 Chip-Specific Approach

In the chip-specific approach, the power management policy parameters of each fabricated SoC instance are configured according to its specific leakage characteristics. Since we are considering each chip individually, the metric of optimization can be local, *i.e.*, to minimize total energy of the specific chip under the given workload. If the total energy consumption of each chip is minimized, desired properties of the energy distribution across chips (such as the ones discussed in the previous section) are automatically optimized. To appropriately configure policy parameters, each chip needs to be calibrated, to determine its leakage characteristics. Calibration requires the use of appropriate measurement equipment, and may also require suitable design of the on-chip power supply network and the use of on-chip measurement circuitry [64], the details of which are beyond the scope of this work. In this section, we discuss three calibration methods, which vary in the calibration effort and consequently, optimality of the power management policy parameters derived.

3.6.1 Full Calibration

In this method, the calibration phase explicitly measures the power consumption of each power manageable component in the SoC for each of its power states. At the end of the calibration phase, chip-specific parameters (T_{th} values) for each component are computed and programmed into the power management control unit, resulting in a SoC-level policy that minimizes the energy consumption of the chip. The advantage of this approach can be explained as follows. In any design-specific solution, under a given workload, all the chips exhibit the identical sequence of power states, as defined by the fixed set of policy parameters. Since the leakage characteristics of each chip is different, it is likely that for many chips, transitions are executed that are sub-optimal for their individual power characteristics. In the chip-specific solution, each chip makes state transition decisions independent of others, preventing sub-optimal transitions and power wastage. However, the disadvantage is that full calibration requires substantial hardware support and calibration effort that may not be realizable for many cost-sensitive SoCs.

3.6.2 Approximate Calibration

For this method, we exploit the observations that (i) performance and leakage are positively correlated, and (ii) speed-binning is a common post-fabrication step used to partition manufactured samples into bins with distinct performance characteristics. In this method, we assume that each chip instance is aware of the precise frequency bin to which it has been allocated. Based on the frequency bin, the chip uses frequency-leakage correlation data for the target process to calculate a corresponding leakage bin. In the calibration phase, each chip samples the leakage power bin to "guess" its leakage characteristics. Values of leakage power thus obtained are then used to compute the policy parameters using break-even analysis, and configure the power management policies, as in the full calibration method. Knowledge of leakage correlation between different components of the SoC can be utilized to determine individual component leakage from the estimated chip leakage. More advanced approximate methods could be conceived, where performance and coarse-grained current measurements are used in conjunction to estimate leakage characteristics.

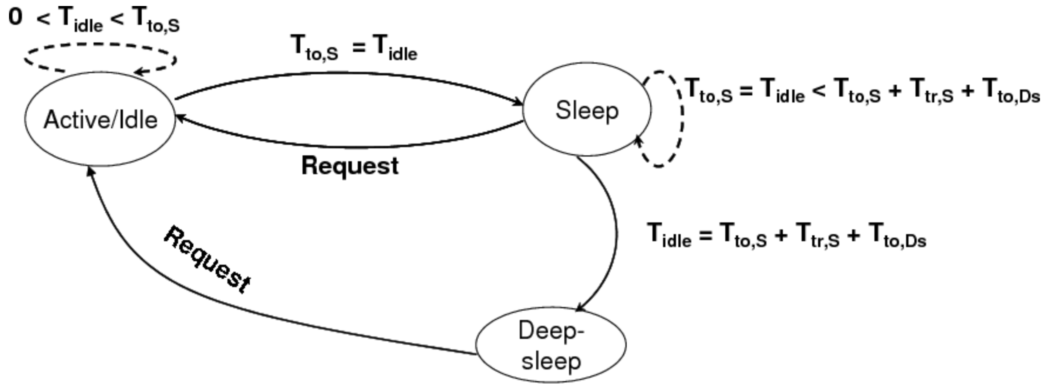
3.6.3 Random Calibration

In this method, the distribution of the break-even times, T_{th}, is determined from the leakage power distribution of the target SoC. In the calibration phase, each chip randomly picks a value from this break-even time distribution and uses it to calibrate the power management policies. The leakage power distribution can be obtained pre-fabrication using variation-aware leakage analysis tools, or sampling a subset of manufactured instances for leakage measurement.

In the next section, we study the impact of leakage variations on the effectiveness of a widely used power management policy namely, the timeout policy.

3.7 Optimizing Timeout-Based Policies for Variations

By assuming Oracle prediction of idle periods during analysis and optimization of power management policies in the presence of leakage variations, we have


$0 < T_{idle} < T_{to,S}$
$T_{to,S} = T_{idle}$
Active/Idle
Sleep
Request
$T_{to,S} = T_{idle} < T_{to,S} + T_{tr,S} + T_{to,Ds}$
$T_{idle} = T_{to,S} + T_{tr,S} + T_{to,Ds}$
Request
Deep-sleep

Power States	[Eg. ARM]	
• **Active** : Pd + Pl	(34.8mW,	3.7mW)
Idle : Pd (idle) + Pl	(10mW * ,	3.7mW)
• **Sleep** : Pl	(0,	3.7mW)
• **Deep-sleep** : 0	(0,	0)

State Transitions times:

- Active to sleep = $T_{tr,S}$ = 0.1ms
- Sleep to deep-sleep = $T_{tr,Ds}$ ~ 20ms

Figure 3.6: Timeout policy: Power state machine and power management policy

eliminated the effect of workload variations. However, in practice power management policies are challenged by workload variations and uncertainty in idle period prediction. Therefore, it is important that we study the applicability of the techniques presented in this chapter in the context of practical power management policies. In particular, we use timeout-based policies that have widespread application today due to ease of implementation and substantial energy savings. In timeout-based power management, the system waits for a pre-specified timeout (T_{to}) to expire before it transitions into a low-power state. In this section, we discuss the timeout-based power management framework, study the impact of leakage variations and then propose methods to derive variation-aware timeout parameter (T_{to}) values to optimize the specified metrics of the overall energy distributions.

3.7.1 Preliminaries

Figure 3.6 shows the power state machine of our example *PMC*, ARM946 core, under timeout-based power management framework. When the *PMC* is *Idle*, it first waits in the *Active* state for time $T_{to,S}$ before transitioning to the *Sleep* state where it remains until a second timeout ($T_{to,Ds}$) expires before transitioning to the *Deep-sleep* state. It transitions back to the *Active* state as soon as a service request is encountered. In this work, we assume that transition overheads are primarily due to state restoration to resume activity in the *Active* state and hence are associated with transitions *from* a lower power state to the *Active* state. Transition *to* a lower power state is assumed to be instantaneous with negligible energy dissipation. Note that, this is only a convenient model that can be used to represent the actual transition overheads that may be associated with transition to and from low power states. Even if transition overheads to a lower power state are significant, that can be lumped with the overheads of transitioning back to the *Active* state. Therefore, energy dissipation of the*PMC*, ARM core, with power characteristics $\bar{P} = \{P_A,\ P_{idle},\ P_S,\ P_{Ds},\ P_{tr,S},\ P_{tr,Ds}\}$, for a given T_{idle} is given by Eq. 3.10. Note that, if the component transitions to a *Deep-sleep* state then it incurs only the transition penalty of transitioning back from the *Deep-sleep* to the *Active* state. In this work we are only concerned with the energy overheads of state transitions and not the latency introduced due to state transition.

$$E(\bar{P}, T_{idle}) = \begin{cases} P_{idle}T_{idle}, & \text{for } T_{idle} \in \{0, T_{to,S}\} \\ P_{idle}T_{to,S} & +P_S(T_{idle} - T_{to,S}) + P_{tr,S}T_{tr,S}, \\ & \text{for } T_{idle} \in \{T_{to,S}, T_{to,S} + T_{to,Ds}\} \\ P_{idle}T_{to,Ds} & +P_S T_{to,Ds} + P_{Ds}(T_{idle} - T_{to,Ds} - T_{to,Ds}) \\ +P_{tr,Ds}T_{tr,Ds}, & \text{for } T_{idle} \in \{T_{to,S} + T_{to,Ds}, \infty\} \end{cases} \tag{3.10}$$

The graphical representation of the energy dissipation *vs.* T_{idle} as given by Eq. 3.10 is shown in Figure 3.7. The steep jump in the energy dissipation characteristics at times $T_{to,S}$ and $T_{to,S} + T_{to,Ds}$, is due to overhead incurred in transition to a low-power state as the timeout expires. The energy dissipation during a given idle period

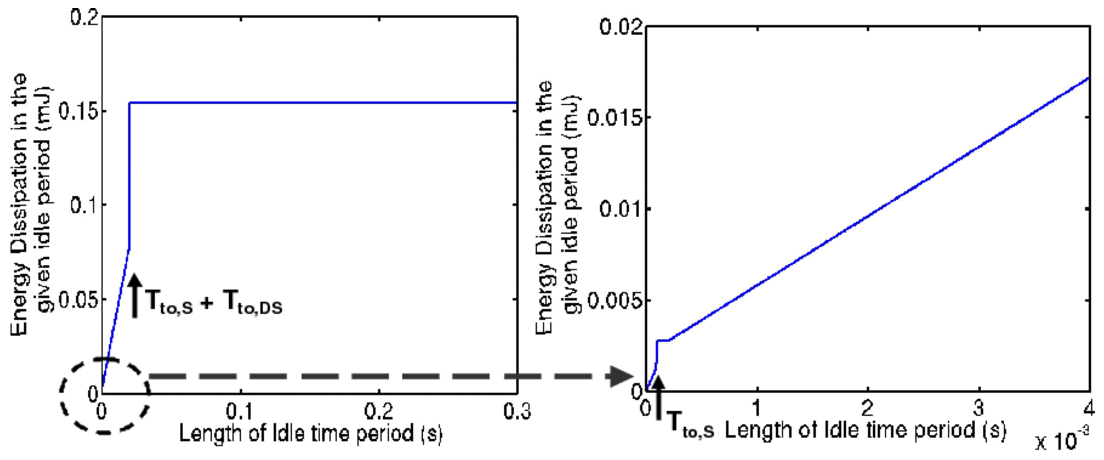


Figure 3.7: Variation of energy dissipation for a given idle time period

depends strongly on the choice of the timeout parameter. The timeout parameter, $T_{to} = (T_{to,S}, T_{to,Ds})$ characterizes a timeout policy.

Now, the total energy dissipation under a workload $\mathcal{W} = \{T_A, \mathbf{T}_{idle}, \mathbf{N}_{idle}\}$ is obtained by the sum of the energy dissipation in each idle period weighted by the frequency of the idle period as shown by Eq. 3.11. Note that, we have dropped the term involving P_{Ds} because power dissipation in the *Deep-sleep* state is negligible.

$$\begin{aligned} E_{TOT} = & E_A + \sum_{T_{idle}=0}^{\infty} E(T_{idle}) N_{idle} \\ = & P_A T_A + \sum_{T_{idle}=0}^{T_{to,S}} P_{idle} T_{idle} N_{idle} + \\ & \sum_{T_{idle}=T_{to,S}}^{T_{to,S}+T_{to,Ds}} (P_{idle} T_{to,S} + P_{tr,S} T_{tr,S} + P_S (T_{idle} - T_{to,S})) N_{idle} + \\ & \sum_{T_{idle}=T_{to,S}+T_{to,Ds}}^{\infty} (P_{idle} T_{to,S} + P_S T_{to,Ds} + P_{tr,Ds} T_{tr,Ds}) N_{idle} \end{aligned} \tag{3.11}$$

The choice of the timeout parameters is crucial to the performance of the timeout policy and depends closely on both the workload characteristics and the power characteristics of the given PMC. Ideally, the T_{to} should be set such that E_{TOT} is minimized under any given workload. However, in general, E_{TOT} as expressed by Eq. 3.11 cannot be reduced to a simple (or even convex) function of T_{to} and a timeout value that is optimum under a general workload cannot be obtained using analytical optimization techniques. Researchers have proposed various methods including stochastic timeout power management to obtain optimum T_{to} for workloads with certain inter-arrival time distributions [60]. However, such methods remain hard to implement. In practice, simple timeout values are used. One of the most common methods is to set the T_{to} equal to the threshold time T_{th}. This policy works well in general and in the worst-case when all the idle time periods are of length equal to the timeout values, it consumes at-most twice the energy consumed by using the oracle-based policy [59]. For this work, we focus on this simple timeout policy and observe its behavior under variations.

3.7.2 Impact of process variations

To study the performance of a conventionally designed timeout policy as described above in the presence of leakage variations, the timeout value is set equal to the threshold times calculated using mean leakage characteristics. Using calculations of Section 3.3, we get $T_{to,S} = 0.1\ ms$ and $T_{to,Ds} = 20.1\ ms$. We compare the total energy distribution obtained under this conventional timeout policy with timeout policies that are obtained by systematically varying $T_{to,Ds}$ values [3]. We find that there exists a value $T'_{to,Ds}$ = 5.9 ms, that results in 27% lower $\mu + \sigma$ compared to the conventional policy ($T_{to,Ds}$ = 20.1 ms). Since $T'_{to,Ds}$ is obtained by exhaustive search, the improvement could be a manifestation of taking workload variations into account. To ensure it is indeed a result of taking leakage variations into account we obtain best offline timeout value by another exhaustive search assuming there are no variations and all chip instances consume mean leakage power. The timeout value hence obtained is $T_{to,Ds} = 20.9ms$. We compare total energy distributions under this best offline timeout policy and variation-aware timeout policy. The variation-aware timeout policy out-performs the best offline policy by 28% in terms of the $\mu + \sigma$ metric and by 32% in terms of the 95th percentile. This clearly indicates that by designing variation-aware timeout policies significant improvement in overall energy distribution can be

[3]As shown earlier, $T_{th,S}$ and hence $T_{to,S}$ is independent of the leakage power characteristics.

obtained.

3.7.3 Methodologies to obtain variation-aware timeout parameter

We now present three methods to derive variation-aware timeout parameters with varying degree of efficiency.

To re-iterate the problem statement - we are interested in finding out the timeout parameter value T_{to} such that the metric of interest of the total energy distribution $\mathcal{E}_{TOT}$ is minimized for the workload $\mathcal{W}$. The total energy dissipation of one instance of the PMC with power characteristics $\bar{P}$ under a timeout policy for a given workload is given by Eq. 3.11. Under variations each term becomes a random variable with a certain distribution. As mentioned earlier it is not possible to express E_{TOT} or $\mathcal{E}_{TOT}$ (under variations) as a simple function of T_{to} and solve it using analytical optimization techniques. Similar to the case of oracle-based power management, we can employ design-specific and chip-specific approaches to determine the optimum parameter values.

In the ***design-specific*** approach, policy parameters are fixed for a design across all fabricated chip instances and are selected such that system-level energy distribution under a given workload is optimized with respect to a specified metric. We present three search based methods to obtain timeout parameter values optimum under the specified metric.

Method 1:
A straight-forward method to obtain variation-aware timeout that optimizes the specified metric is outlined below. This method is based on exhaustive search and Monte Carlo sampling.

1. Define a search space for the value of the timeout parameter. Also define the granularity of search.

2. For each possible value of the timeout parameter

    - Monte Carlo sampling: Sample the chip population (leakage power distribution) to obtain the power characteristic of the given chip instance

- Compute the total energy dissipation of this chip instance for the current timeout parameter value using Eq. 3.11
- Perform the above two steps for a large sample to obtain the total energy distribution under the current timeout parameter value
- Compute the metrics of interest such as μ, σ, μ+σ, Nth percentile

3. Select the values of the timeout parameters that optimizes the metric of interest

Combination of exhaustive search in the timeout parameter space and Monte Carlo sampling for each possible parameter value makes this method highly computationally intensive and hence impractical except with very coarse granularity.

Method 2:
We present analysis similar to the one presented in the case of the oracle policy to separate out the dynamic and leakage power terms in order to eliminate the iterative Monte Carlo sampling step in the above method.

Let α represent the fraction of the chip that is subject to leakage in each state and state transition. Let $\bar{\alpha} = \{\alpha_A, \alpha_{Idle}, \alpha_{S1}, \ldots, \alpha_{Sn}, \alpha_{tr,S1}, ..., \alpha_{tr,Sn}\}$, then $\bar{\mathcal{P}^i} = \bar{\mathcal{P}_D} + \bar{\alpha}\,\bar{\mathcal{P}^i_L}$. Note that, the dynamic power dissipation and $\bar{\alpha}$ are roughly constant across various chip instances. Using this in Eq. 3.11 and then separating out terms corresponding to dynamic energy and leakage we get,

$$E_{TOT}(\bar{P}, \mathcal{W}, T_{to}) {=} E(\bar{P_D}, \mathcal{W}, T_{to}) + \mathcal{P}_L * T_E(\bar{\alpha}, \mathcal{W}, T_{to})$$

Here, T_E represents an equivalent time for which the chip can be considered to be leaking in order to calculate the energy dissipation. Therefore, the parameters of the total energy distribution can be obtained as follows,

$$\mu[E_{TOT}(\bar{\mathbf{P}}, \mathcal{W}, T_{to})] {=} E(\bar{\mathbf{P}_D}, \mathcal{W}, T_{to}) + \mu[P_L]\; T_E(\bar{\alpha}, \mathcal{W}, T_{to})$$

$$\sigma[E_{TOT}(\bar{\mathbf{P}}, \mathcal{W}, T_{to})] {=} \sigma[P_L]\; T_E(\bar{\alpha}, \mathcal{W}, T_{to})$$

$$\Pi_{\mathcal{N}}[E_{TOT}(\bar{\mathbf{P}}, \mathcal{W}, T_{to})] {=} E(\bar{\mathbf{P}}_d, \mathcal{W}, T_{to}) + \Pi_{\mathcal{N}}[P_L]\; T_E(\bar{\alpha}, \mathcal{W}, T_{to})$$

Therefore, in method 2 the inner iterative loop of method 1 is efficiently replaced by the above analysis to calculate distribution characteristics for each possible timeout candidate.

Method 3:
The exhaustive search presented in the above two methods is limited by the specified granularity of search both in terms of efficiency and accuracy. We propose use of the *Nelder Mead direct search* algorithm which is a widely used method for non-linear unconstrained optimization [65] and therefore, is well suited to the problem at hand. The algorithm essentially evaluates the function value with k variables at k+1 points forming the vertices of a k-dimensional simplex. The simplex is updated in each iteration by discarding the point that results in the largest value and replacing it with a point having a lower function value.

In the ***chip-specific*** approach, the parameter values are obtained such that total energy dissipation is optimized for each individual chip instance. If the power characteristic of each chip is known either an exhaustive search or Nelder-mead search, as outlined above, can be employed to determine the timeout value for each chip instance. Power characteristics of each chip can be obtained by using one of the calibration methods as outlined in Section 3.6 for the oracle-based power management policy framework.

3.8 Experimental Results

In this section, we describe our experimental methodology, and present results that compare the effectiveness of the proposed variation-aware power management policies.

3.8.1 Methodology

For our experiments, we use an ARM946 processor core [42] described in Section 3.3 as the PMC. Dynamic power estimates are obtained using an in-house simulation-based power estimation framework [45] that uses instruction-based power models. Leakage power estimates for different power states are obtained using approximate gate-counts and high-level techniques similar to [53]. We use statistical leakage power models described in [63] to obtain leakage power distributions for all

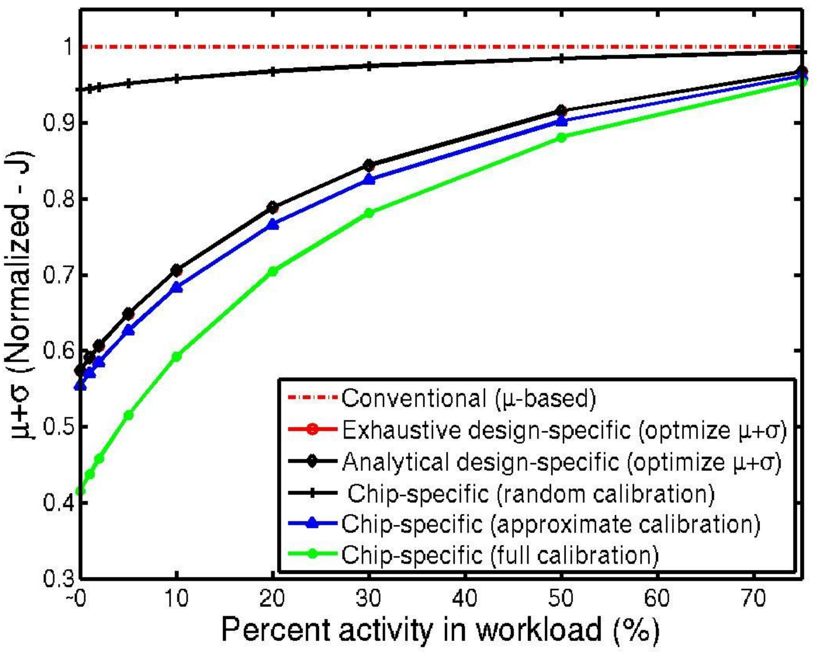


Figure 3.8: Oracle-based power management framework, variation in $\mu + \sigma$ with increasing system activity

the power-states, assuming inter-die variations in channel length with $3\sigma = 30\% \ \mu$. Characterization data for calibrating the statistical leakage power models is obtained via HSPICE simulation of the Berkeley Predictive Technology Model's $90\ nm$ BSIM3 model card [55]. We implemented a 5-stage ring-oscillator in HSPICE to generate frequency-leakage correlation data used for the approximate leakage power calibration of chips. The methods for obtaining and evaluating optimum design-specific and chip-specific parameters are implemented in MATLAB [56]. In the following subsections, we first present results to compare the proposed methods under the ideal oracle-based power management framework and then we present results to compare different methods under the timeout-based framework.

3.8.2 Results – oracle-based power management

We compare the proposed approaches in terms of their effectiveness in optimizing μ+σ and 95^{th} percentile of the energy distribution in Figure 3.8 and Figure 3.9, respectively. The x-axis represents the percentage of active time in a given workload.

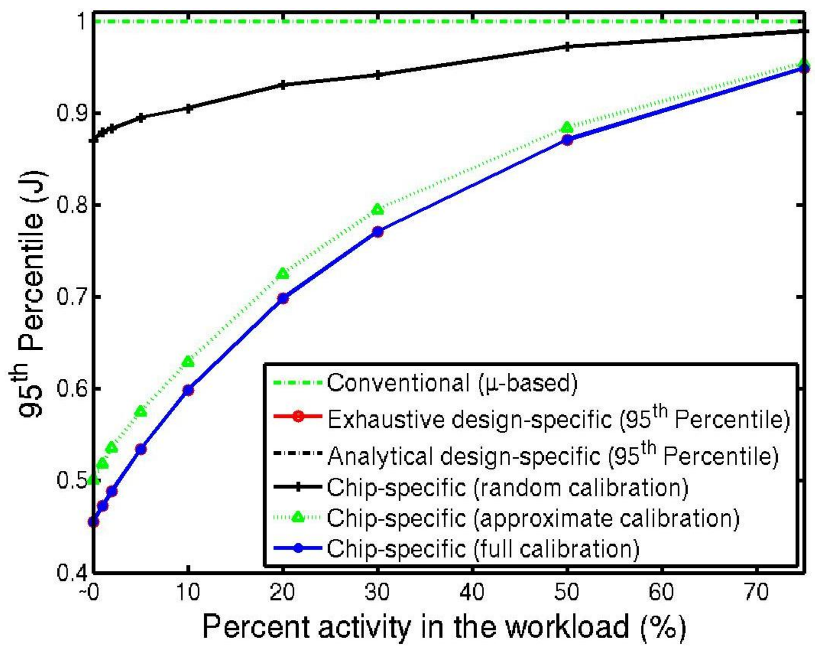


Figure 3.9: Oracle-based power management framework, variation in 95^{th} percentile with increasing system activity

The y-axis represents $\mu + \sigma$ (or 95^{th} percentile) of the total energy distribution normalized with respect to the conventional μ-based policy. From Figure 3.8, we observe that the chip-specific policy with full calibration outperforms all the other policies in terms of minimizing $\mu + \sigma$. As system activity increases from $\sim$ 0%[4] to 75%, the chip-specific policy achieves 59% to 5% gains over the conventional policy. The chip-specific policy with approximate calibration has the next best performance with 44% to 4% gains. The design-specific policy optimized for $\mu + \sigma$ is able to achieve an improvement of 42% to 5%.We also implemented an exhaustive search of the parameter space to obtain the optimum design-specific parameter set to compare with our analytical derivation. As expected, both the analytical and exhaustive design-specific policies exhibit identical behavior.

From Figure 3.9, we observe that for the given workloads, both the chip-specific policy with full calibration and the design-specific policy optimized for the

[4]Note that if there is no activity in the workload *i.e.,* 0% activty, then all the power management policies will behave identically, and will transition the component to the *Deep-sleep* state. A near ($\sim$)0% activity means that the idle time periods are interspersed with some activity which is negligible compared to the total idle time in the trace.

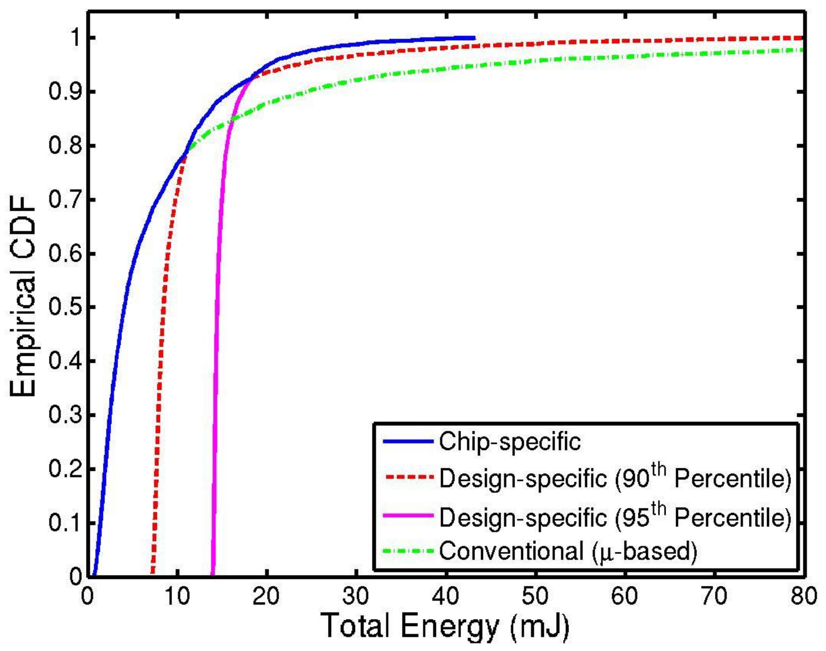


Figure 3.10: Oracle-based power management framework, CDFs obtained under various proposed policies

specified percentile value results in 55% to 5% improvement over the conventional policy. The chip-specific policy with approximate calibration is able to achieve similar gains, while the one with random calibration is only able to achieve 12% to 2% improvement. Note that, even though the design-specific policy matches the chip-specific policy for the percentile metric, in general, the chip-specific policy will result in a superior energy distribution, as illustrated by the CDFs presented in Figure 3.10.

3.8.3 Results – timeout-based power management

To compare the effectiveness of the proposed approaches under the timeout-based framework, we conducted experiments for three different idle time distributions (workloads 1, 2 and 3). For workload 1, the idle time distribution is randomly generated and for workload 2 it is obtained experimentally through system simulations. For workload 3, we use a family of Pareto distributed idle times. For the design-specific approach we have implemented the three methods outlined in Section 3.7, namely, exhaustive search with Monte Carlo simulations (Method I), efficient exhaustive search (Method II) and Nelder-Mead direct search (Method III). For the chip-specific approach, we have implemented the exhaustive search (Method I), an adaptive search (Method II) and the Nelder-Mead search (Method III) for each chip instance, assuming full calibration for each method. In the adaptive search, first a course-grained search is performed and the granularity of search subsequently gets finer in the regions of interest. We have also implemented exhaustive search of time-out values with approximate calibration.

Workload 1 - Randomly generated

The idle time distribution is shown in Figure 3.11 and is generated by assigning frequency of each T_{idle} (at the granularity of $1\ ms$) as a product of two uniformly distributed random variables, rounded to the nearest integer. These random variables are weighted heavily for $T_{idle} \leq 10\ ms$ and lightly for $T_{idle} \geq 30\ ms$. For $T_{idle} \geq 100\ ms$, we assign zero weight. Separate weighing in different regions is to make sure that best timeout values for this distribution do not tend towards zero.

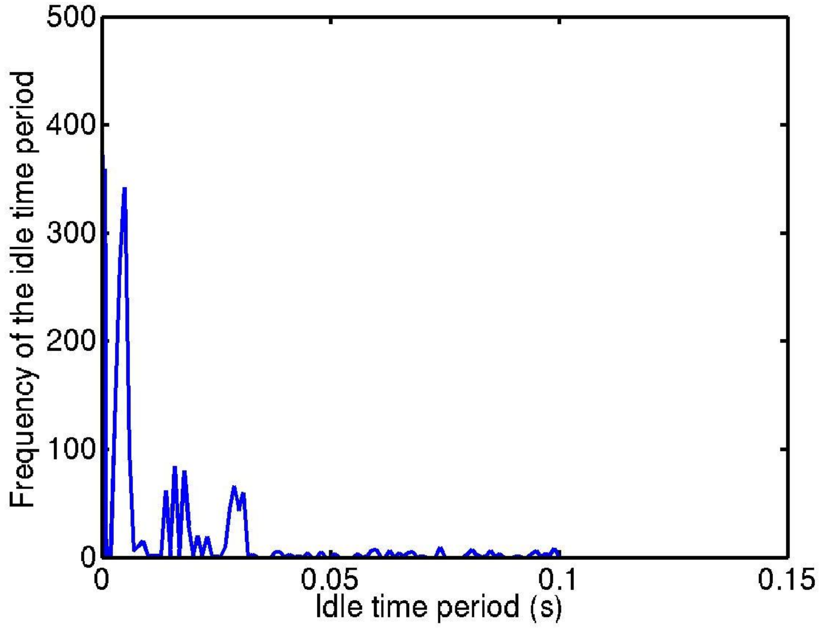


Figure 3.11: Idle time distribution for workload 1

In Figure 3.12, we plot the normalized $\mu + \sigma$ metric obtained under different timeout policies as the percent activity in the workload varies from $\sim$0% to 75%. We observe that the design-specific timeout policies with exhaustive search result in 27% to 2% improvements. The design-specific policy with Nelder-Mead search performs marginally better with improvements ranging from 27.5% to 2%. Chip-specific policies with either exhaustive or adaptive search results in 43% to 3% improvements. The chip-specific policy with Nelder-Mead search yields 41% to 2% improvements, whereas exhaustive search with approximated calibration yields improvements from 29% to 2%. Similar results are obtained for the 95^{th} percentile metric as well. We find that, the design-specific policies with exhaustive search and the chip-specific policies with exhaustive and adaptive searches yield up to 36% to 2% improvements. The chip-specific policy with Nelder-Mead search results in 35% to 3% gains. The design-specific policy with Nelder-Mead search and the chip-specific policy with exhaustive search assuming approximate calibration yield 32% to 2% improvements.

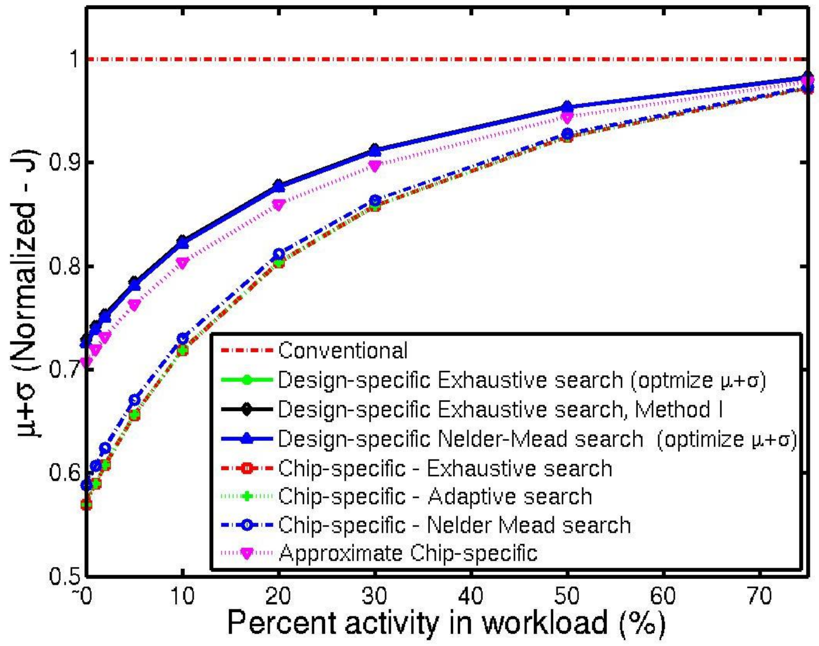


Figure 3.12: $\mu + \sigma$ variation with increasing system activity, for a timeout-based PM

Table 3.1: Comparing various policies in terms of $\mu + \sigma$, and different percentile values, under various different workloads

(a) Design-specific

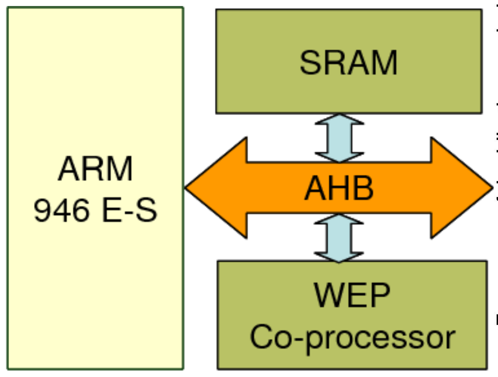
Metric	Mean + Sigma (%)			90th Percentile (%)			95th Percentile (%)			98th Percentile (%)			99th Percentile (%)		
Design-Specific search Method	I	II	III	I	II	III	I	II	III	I	II	III	I	II	III
Workload 1 (~0 % activity)	27.2	27.2	27.5	23.4	23.4	23.7	36	36	31.8	50.6	50.6	37.9	55.3	55.3	--
Workload 1 (2 % activity)	24.7	24.7	25	21	21	21.2	33.2	33.2	29.4	47.8	47.8	35.8	52.7	52.7	--
Workload 2 (~0 % activity)	6.5	6.5	6.5	5.9	5.9	5.9	7.1	7.1	7.1	7.9	7.9	7.9	8.1	8.1	8.1
Workload 2 (2 % activity)	6.0	6.0	6.0	5.4	5.4	5.4	6.7	6.7	6.7	7.6	7.6	7.6	7.8	7.8	7.8

(b) Chip-specific

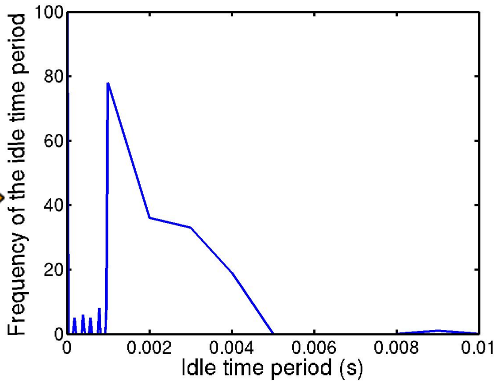
Metric	Mean + Sigma(%)			90th Percentile (%)			95th Percentile (%)			98th Percentile (%)			99th Percentile (%)		
Chip-Specific search Method	I	II	III	I	II	III	I	II	III	I	II	III	I	II	III
Workload 1 (~0 % activity)	43	43	41	23.4	23.4	13.6	36	36	35.1	50.6	50.6	49.6	55.3	55.3	54.3
Workload 1 (2 % activity)	39	39	38	21	21	12.2	33.2	33.2	32.4	47.8	47.8	47	52.7	52.7	51.7
Workload 2 (~0 % activity)	7.9	7.9	3.7	5.9	5.9	5.9	7.1	7.1	7.1	7.9	7.9	7.9	8.1	8.1	-2.7
Workload 2 (2 % activity)	7.4	7.4	3.4	5.4	5.4	5.4	6.7	6.7	6.7	7.6	7.6	7.6	7.8	7.8	-2.6

Workload 2 - Derived from system simulation

We obtain the idle time characteristics for workload 2 using the SoC shown in Figure 3.13(a). It implements 802.11b wireless LAN MAC layer protocol. For each MAC frame, first the cyclic redundancy checksum (CRC) and the encrypted frame is computed, and then the CRC of the encrypted frame with the first CRC appended is computed. The CRC computation is done in software and is mapped to the ARM946 processor. The compute-intensive WEP encryption is implemented using custom hardware that interrupts the processor after encrypting each frame. We process 200 frames of varying sizes and note the resulting idle times. The corresponding idle time distribution is shown in Figure 3.13(b). We obtain improvements ranging from 7% to 1% for both the metrics under various timeout policies as workload activity varies from ∼0% to 75%.



(a)



(b)

Figure 3.13: (a) The wep-encryption SoC, (b) Idle time distribution during processing of 200 frames with varying sizes

We compare the percent improvement in different distribution metrics for workloads 1 and 2 under design-specific and chip-specific policies in Table 3.1(a) and Table 3.1(b), respectively. As observed before, we see that for various metrics the chip-specific policies in general yields larger gains as compared to the design-specific policies and the extent of improvement reduces with increased workload activity. Moreover, we note that improvements for workload 1 are significantly higher than those for workload 2. This shows that the improvement strongly depends not only on the idle time contribution but also on the idle time distribution characteristics of the workload. We also note that larger improvements are seen for the higher percentile metrics. For example, for workload 1, both the design-specific and chip-specific policies result in improvements ranging from 23% to 55% as the metric changes from 90^{th} percentile to 99^{th} percentile. Note that the Nelder-Mead search, although more efficient than exhaustive search methods described in Section 3.7, does not guarantee better results or results consistent with other methods. This is because in this case the Nelder-Mead search is performed over a 1-dimensional simplex that is modified in a heuristic manner and hence, can sometimes converge to a local minima.

Workload 3 - Pareto Distributed

For our final experiment, we assume that the idle times follow Pareto distribution. The probability density function of a Pareto distribution is given by $f(x; k, X_m) = \frac{k\ X_m^k}{x^{k+1}}$, for $x \geq\ X_m$, where k and X_m are distribution parameters. We systematically vary X_m and k to obtain a family of idle time distributions as shown in Figure 3.14. For a given X_m, peak of the distribution increases with k. The distributions shift to the right as X_m increases. We obtain the total energy distributions for workloads with T_{idle} distributions of Figure 3.14, each with varying workload activity. For these experiments, we use exhaustive design-specific and exhaustive chip-specific policies since we want to show the effect of systematically varying T_{idle} distributions on the extent of energy savings and not compare policies. Furthermore, previous two experiments show that although improvements achieved from design-specific and chip-specific policies differ significantly, there is no significant difference in the improvements obtained under their respective variants.

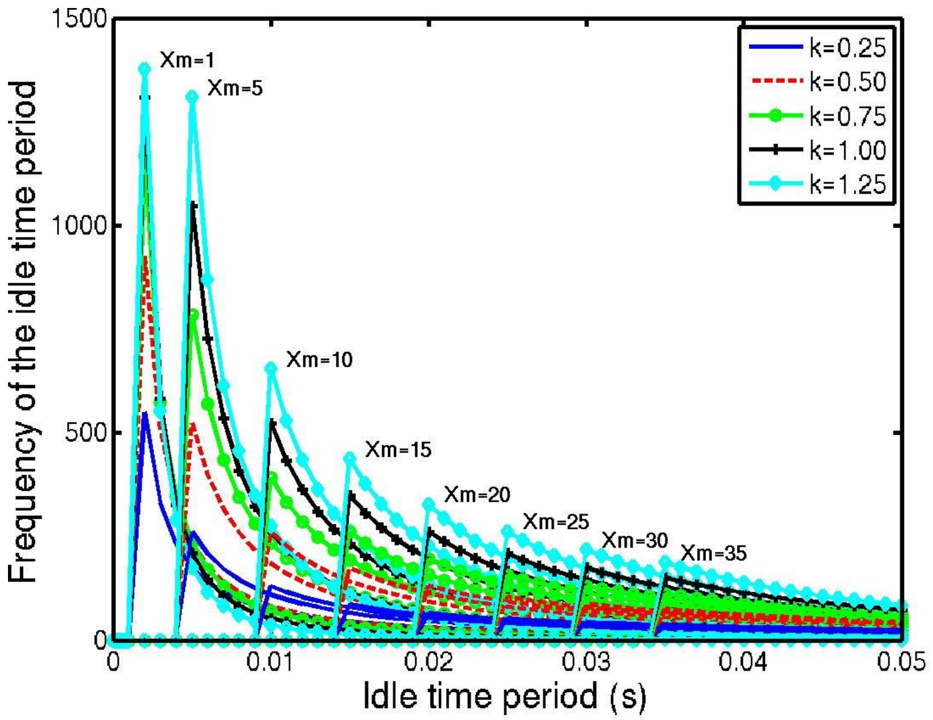
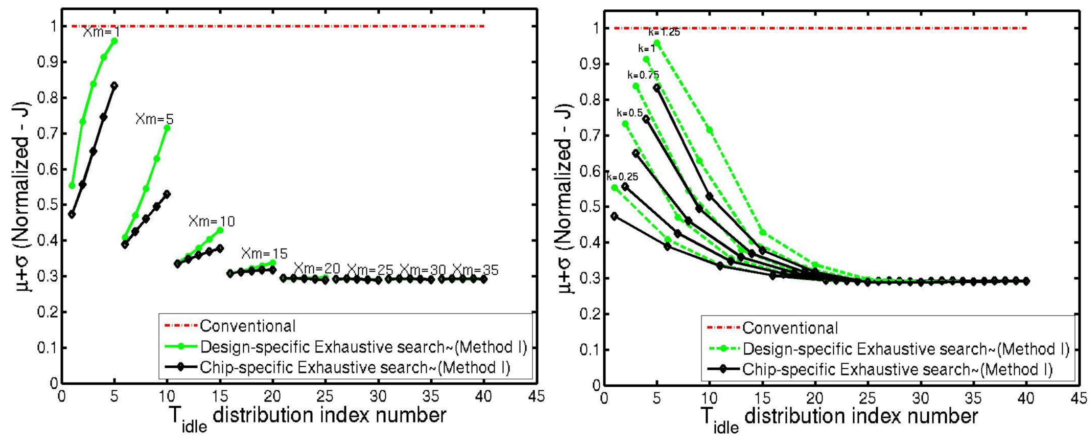
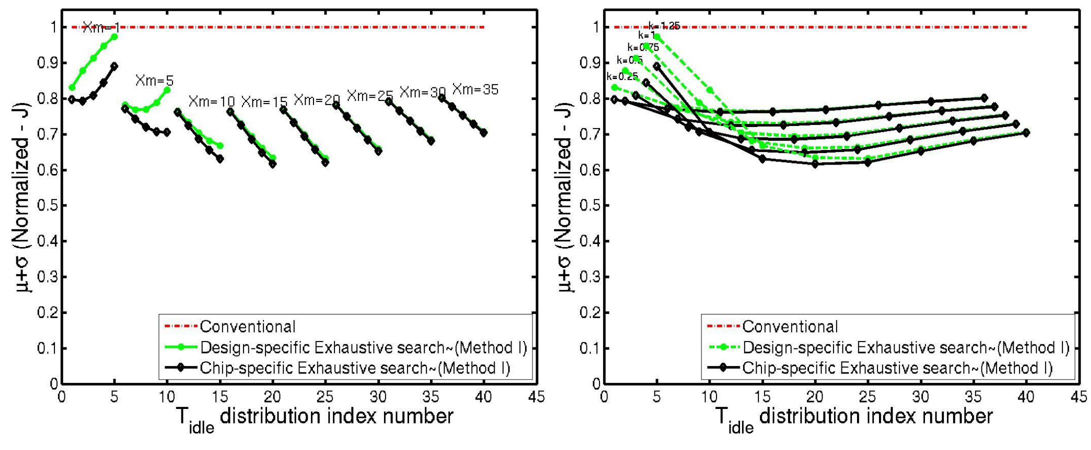


Figure 3.14: Family of idle time distributions following Pareto PDFs for various k and X_m values



(a) (b)

Figure 3.15: $\mu + \sigma$ variation with varying T_{idle} distributions for 0% workload activity under timeout-based PM (a) variation with k (b) variation with X_m



(a) (b)

Figure 3.16: $\mu + \sigma$ variation with varying T_{idle} distributions for 10% workload activity under timeout-based PM (a) variation with k (b) variation with X_m

In Figure 3.15, we plot the $\mu+\sigma$ metric obtained for various T_{idle} distributions[5] for workloads with near zero ($\sim$0%) activity, under different timeout policies. The X-axis represents the T_{idle} distribution index number. In Figure 3.15 (a), variation in $\mu + \sigma$ is shown as k increases for various values of X_m, while in Figure 3.15 (b) $\mu + \sigma$ variation is shown as X_m increases for various values of k. We observe that for $\sim$0% activity in workloads, $\mu + \sigma$ increases as k increases for a given X_m. For example, for $X_m = 5$, improvement in $\mu + \sigma$ ranges from 59% to 28% for design-specific and from 61% to 47% for chip-specific policies, as k varies from 0.25 to 1.25. However, $\mu + \sigma$ decreases as X_m increases for a given k (Figure 3.15 (b)). For $k = 0.25$, improvements ranging from 45% to 71% for the design-specific and from 53% to 71% for the chip-specific policies are obtained. In Figure 3.16, we show the $\mu + \sigma$ variation with varying T_{idle} distributions for workloads having 10% activity. We observe that, unlike Figure 3.15, $\mu + \sigma$ decreases as k increases for $X_m \geq 5$ for design-specific policy and $X_m \geq 10$ for chip-specific policies. We believe that this reversal is because of reduction in variance of the resulting distributions due to increased contribution of dynamic power. Total improvements ranging from 37% to 3% are obtained in this case. We found that, as workload activity increases from $\sim$0% to 75% the improvement in $\mu + \sigma$ metric averaged across all the T_{idle} distributions range from 63% to 2% for the chip-specific policy and from 60% to 2% for the design-specific policy.

3.9 Conclusions

In this chapter, we studied the impact of process variations on power management, and showed that conventional approaches lead to sub-optimality and energy wastage. We presented two approaches to design optimum power management policies under process variations, namely design-specific and chip-specific approaches. We also study the effectiveness of these approaches under oracle-based and timeout-based power management frameworks. Experimental results show that the power

[5]The distributions (Figure 3.14) are indexed from $1 - 40$, with first order variation in X_m. For example, for distributions indexed from $1 - 5$, $X_m = 1$ and k varies from $0.25 - 1.25$ and so on.

management policies with parameters optimized for the presence of variations can significantly out-perform policies designed without considering the effect of variations.

The text of this chapter, in part, is based on material that has been published in the Proceedings of IEEE Design Automation Conference, 2007 (S. Chandra, K. Lahiri, A. Raghunathan, and S. Dey, *System-on-Chip Power Management Considering Leakage Power Variations*, Proc. Design Automation Conf., pp. 877 882, June 2007) and accepted for publication in the IEEE Transactions on VLSI Systems (S. Chandra, K. Lahiri, A. Raghunathan, and S. Dey, *Variation-Tolerant Dynamic Power Management at the System-Level*, IEEE Trans. VLSI Systems) The dissertation author was the primary researcher and author, and the coauthors listed in these publications collaborated on, or supervised the research that forms the basis for this chapter.

Chapter 4

Variation-aware Voltage level Selection

4.1 Introduction

In the last chapter, we showed that *variation-aware* determination of shutdown-based power management policy parameters can result in significant improvements in the overall energy distribution. In this chapter, we present a methodology for *variation-aware* voltage level selection aimed at maximizing the number of SoC instances with improved power and performance characteristics.

The supply voltage, V_{dd}, has a profound impact on the operating clock frequency and energy consumption of an integrated circuit. Typically, logic delay increases with reduction in the supply voltage, and both switching and leakage components of the power consumption have a superlinear relationship with the supply voltage. In the context of addressing variations, inherently slow parts can be operated at a higher voltage to meet frequency targets while inherently fast parts can be operated at a lower voltage to enable energy savings. However, it is extremely hard to design systems such that each instance can be operated at its *ideal* voltage, and in general, all instances need to use a pre-determined set of voltage levels. In this chapter, we show that determining the set of voltage levels in variation-unaware manner can lead to highly unfavorable power/performance characteristics, and present a methodology to determine a set of variation-aware voltage levels.

Traditionally, different supply voltages have been used for different workload scenarios in conjunction with frequency scaling to enable power savings in systems (popularly known as dynamic voltage and frequency scaling or DVFS). More recently, multi-frequency, multi-voltage SoC design style has caught attention of many researchers [66]. These systems allow different parts of a chip, sometimes known as voltage islands (VIs) or voltage domains, to be operated at different voltage and frequency settings. Originally developed as an effective architecture to meet power and performance goals, the multi-frequency, multi-voltage systems are also inherently robust to variations, as is shown in [67]. In most of the prior work involving supply voltage based techniques to meet power or performance targets, the set of voltage values is either assumed to be uniformly spaced or is chosen apriori based on the deterministic (nominal or worst-case) frequency-voltage characteristics of the SoC. However, in the presence of variations, frequency-voltage characteristics of an IC vary from one chip instance to another, making it imperative to re-evaluate the effectiveness of conventional voltage selection schemes. In this chapter, we demonstrate that determining suitable variation-aware voltage levels is critical in systems that are impacted by process variations. Moreover, techniques presented in this chapter are applicable to single or multiple voltage island based designs (both application-specific and general purpose processing SoCs).

4.1.1 Chapter overview and contributions

In this chapter, we make the following contributions.

- We motivate the need for selecting voltage levels in a variation-aware manner. We use an ARM processor core as an example, and (a) *quantitatively* illustrate that different instances (or voltage islands in multi-voltage island systems) have different frequency-voltage characteristics, each having an optimal operating point that meets performance targets with minimal energy dissipation, (b) using voltage levels that are chosen without consideration for variations result in a suboptimal energy distribution.

- We present a methodology to determine voltage levels in a variation-aware manner. In the proposed methodology, an *ideal* voltage distribution corresponding to each voltage island is generated based on the frequency-voltage characteristics of instances in a sample set. The problem of voltage level selection is then formulated as quantization of the *ideal* voltage distribution.

- We present approaches at design-time and post-fabrication for estimating the frequency-voltage characteristics of a sample set of SoC instances. To drive the design-time approach, we have developed high-level voltage-delay and leakage-voltage models under channel length variations. We use these models, along with Monte Carlo sampling of process parameters to design and evaluate various voltage selection schemes.

- To evaluate the concepts presented in this chapter, we use an SoC implementation of the 802.11 MAC protocol tasks and an ARM processor core model. We show that with variation-aware voltage level selection, the number of chip instances with improved power-performance characteristics is significantly higher as compared to voltage level selection based on either the worst-case or the nominal characteristics.

This chapter is organized as follows. In the next subsection, we present the related work and distinguish our work. In Section 4.2, we introduce the framework used in this work. In Section 4.3, we illustrate quantitatively how customizing supply voltage for each SoC instance can be used to meet performance targets and minimize energy dissipation under process variations, and discuss the implications on practical systems with only a few available levels. In Section 4.4, we present a methodology to determine a set of variation-aware voltage levels. We present experimental results in Section 4.5, and conclude in Section 4.6. The leakage and delay models under channel length and voltage variations are presented in Appendix B of the thesis.

4.1.2 Related work

Recognizing the profound impact that supply voltage has on the overall power and performance characteristics of circuits and systems, a large number of researchers have investigated and developed voltage scaling based techniques to meet power and performance targets. In this section, we will first present an overview of the works that employ voltage scaling for energy reduction without considering the impact of variations. Then, we present works that use supply voltage based techniques specifically targeted towards mitigating the impact of process variations.

The concept of varying supply voltage originally emerged to enable energy savings during times when a system is not required to operate at the maximum frequency. A number of researchers exploited this concept, and a number of design time and runtime policies have been proposed to enable reducing energy consumption, an exhaustive description of which is far beyond the scope of this work [68, 69, 70, 71]. In [68], a survey of power-aware and battery-aware scheduling algorithms for DVFS in uniprocessor and distributed systems is presented. Researchers have proposed DVFS based techniques assuming the availability of both continuous and discrete voltage levels. While the assumption of continuous voltage and frequency levels is useful in determining the upper bound on total energy savings, it is not practical to implement. In [72], the authors show that if only a few discrete levels are available, at most two voltage levels are required to minimize the dynamic energy consumption. Some of the recent works such as [73] and [74] account for leakage power during DVFS, and show that in nanoscale technologies, there exists a lower limit on the supply voltage beyond which voltage scaling might result in increased energy consumption. In [75], the authors propose a combination of dynamic voltage scaling (DVS) and adaptive body biasing (ABB) to maximize the energy savings. Recently, researchers have proposed temperature-aware DVFS algorithms for energy optimization [76].

A number of researchers have investigated the effectiveness of using supply voltage to mitigate the power and performance impact of process variations at various levels of abstraction. In [21], the authors use test chip measurements to show that using adaptive voltage scaling (AVS) in conjunction with adaptive-body bias (ABB) leads to increase in the number of dies meeting power and performance targets. In [77], the authors present an approach to select V_{dd}, V_{th} and T_{ox} values for devices in a given path in order to minimize total power dissipation. In [27, 78], on-chip circuitry for error detection and correction is presented that is aimed at allowing supply voltages to scale down until the point of functional breakdown. ARM's Intelligent Energy Manager (IEM) [79] makes use of National's PowerWise technology to implement adaptive voltage scaling to control voltage using on-chip regulators in a closed-loop manner [80, 81]. Techniques using on-chip voltage regulators are also investigated in [82]. These works facilitate fine grained control of the supply voltage levels in a closed loop manner using on-chip voltage regulators. Designing cheap on-chip voltage regulators that can reliability power large circuits is still a challenge in digital CMOS technologies [83]. Therefore, implementation of such techniques is associated with high design overhead and cost, and hence, these techniques are not readily accepted in the main stream. The techniques presented in our work do not require the presence of on-chip voltage regulators.

Multiple voltage island (VI) design style inherently allows improved power-performance characteristics by isolating the impact of variations within an island as is shown by [67, 84, 85]. In [86], the authors propose variability-aware DVFS algorithms in multi-core systems while accounting for core-to-core variations. In [87], algorithms for variation-aware application scheduling are presented in the context of chip multiprocessors. The authors also present algorithms to select voltage and frequency levels for each core such that average throughput is maximized. However, in most of the above mentioned works, it is assumed that the discrete voltage levels are computed based on deterministic characteristics. Some of the works assume that discrete voltage levels are present in a range separated by uniform values. In most of the commercial designs, these levels are determined based on the worst-case frequency-voltage characteristics (slow-slow corner). In our work, we show that choosing the voltage levels while accounting for variations can lead to significant improvements in the design objectives, especially, when the system can support only a small number of voltage levels. Unlike most of the existing work in the literature, we do not focus on determining optimal task and voltage schedules, but focus on the orthogonal problem of determining the set of voltage levels such that the voltage scaling schemes are more effective in the presence of process variations. As such, the systematic methodology presented in this chapter is generic and complements the already existing work. Moreover, the proposed methodology incurs no hardware overheads and requires no changes in the design flow unlike [79, 27, 82].

4.2 Preliminaries and assumptions

In this section, we introduce the SoC design framework used in this chapter and state the associated assumptions. In this work, we assume that an SoC can be partitioned into various frequency-voltage islands and is capable of operating in a number of *performance* states.

Multiple voltage island design style is inherently robust to variations since it enables isolating the impact of variations on different parts of the SoC. However, overheads introduced by designing interfaces between islands operating at different frequency and voltage levels puts a limit to the number of frequency-voltage islands that can be supported in a system. In this work, we assume that the SoC has already been partitioned into appropriate frequency-voltage islands while taking into account such overheads. Each voltage domain is associated with an independent voltage regulator and a clock generator. Voltage regulators are assumed to be adjustable, and can be on-chip or off-chip. Most of the commercial systems today utilize off-chip regulators since designing cheap and reliable on-chip voltage regulators is still a challenge. Now, each island j, can be designed to operate at multiple frequency levels, $\mathbf{F^j} = \{F_1^j, \ldots, F_m^j\}$ and multiple voltage levels $\mathbf{V^j} = \{V_1^j, .., V_n^j\}$. Note that, while m is determined by the possible workloads in each island, n is limited by the capabilities of the voltage regulator. Therefore, in general, systems can have different number of frequency and voltage levels (*i.e.,* $m \neq n$ should be possible). An example SoC implemented in a multi-voltage island design style is shown in Fig. 4.9. The figure shows our implementation of a 802.11 MAC protocol processing system. It is divided into two islands, and is described in detail in Section 4.5.

Most power efficient systems today support a number of performance states (P-states in Intel terminology) [88]. A performance state can be defined by the minimum performance that is guaranteed in that state. For example, the MAC layer processing SoC (Fig. 4.9) may be guaranteed to generate processed frames at a data rate of 1.5 Mbps in the highest performance state, and at the data rates of 1.2Mbps, 1Mbps and 0.5Mbps in the subsequent states. The frequency setting for each state is assumed to be determined during design time analysis based on the performance targets guaranteed for the applications in each state. *Conventionally*, each performance state is associated with a frequency/voltage pair (*i.e.,* $m = n$). At runtime, systems can dynamically switch between different states depending upon the workload. Frequency and voltage settings remain constant in a given state. In the next section, we will show that in the presence of variations it is beneficial to decouple this frequency-voltage association and customize the voltage setting for each chip instance according to its individual frequency-voltage characteristics.

In most of the SoCs employing multiple frequency/voltage settings (or performance states) for each island, only a small number of voltage levels are supported. This is because of the high cost and complexity involved in designing systems with multiple voltage levels. In the past, voltage levels used were typically limited to ones for which standard cell libraries were characterized, and the designers strove to achieve timing closure for the targeted frequencies at each of these voltage levels. However, such timing closure for multiple voltage levels is extremely difficult, especially, in the presence of variations. In the absence of mature tools for integrated statistical design flow, many ASIC design manufacturers still rely heavily on worst-case design in which the libraries are characterized at the slow-slow process corner. This causes increased design effort, and often leads to power inefficient designs. For sub-90 nm technologies additional post-fabrication characterization is required. Because of the high costs associated with the post-fabrication infrastructure, chip instances can be characterized for only a small number of voltage levels. The primary contribution of our work is to illustrate that by determining *the set* of voltage levels in a *variation-aware* manner, significant improvements are possible in the number of instances meeting power-performance targets, while working under the limitations of the current design flow.

4.3 Variation-aware voltage selection

In this section, we first illustrate quantitatively how customizing supply voltage for each SoC instance can be used to reduce energy dissipation and meet performance targets in the presence of variations. To introduce the concept we assume that continuous voltage setting is possible. Although impractical to implement, a continuous voltage scheme serves as a lower bound to compare the effectiveness of various voltage selection schemes. We then discuss how the concept of customizing voltage levels for each instance can be applied in practical systems that can support only a small number of discrete voltage levels, and contrast its effectiveness with 'conventional' schemes that use fixed frequency-voltage pairing for each instance.

Example study: For this study, we use an ARM processor core as an example uniprocessor system that is implemented in a 90nm process technology known to exhibit $\sigma = 10\%\mu$ variations in the channel length. It is capable of operating at nominal a frequency of F_{nom} = 206 MHz, at a nominal voltage of V_{nom}=1V. Suppose, for a certain workload this system is required to operate at a minimum frequency of $F_{req} = 133.3\ Mhz$ in order to meet the required performance. For the ARM946 processor, we computed that $F_{req} = 133MHz$ can be achieved at 0.82 V for a nominal instance that is not affected by the process variations. However, in the presence of process variations different chip instances have different frequency-voltage (F-V) characteristics and hence, can operate at F_{req} at different voltage levels. In Figure 4.1, we show the F-V characteristics of three instances under variations. Here, $instance0$ corresponds to the nominal instance that is not affected by variations. The figure shows that $instance1$ requires a supply voltage of 0.96 V to meet the performance target of $F_{req} = 133MHz$ whereas $instance2$ can achieve this target even at 0.72 V. This difference arises because $instance1$ (with $L_{eff} > L_{nom}$) is inherently slower while $instance2$ (with $L_{eff} < L_{nom}$) is inherently faster. Therefore, setting the voltage to 0.82 V for all the instances can be very ineffective. It can result in $instance1$ not meeting the setup time constraint for the target F_{req}, and can cause $instance2$ to dissipate more energy than required to perform the given operation. On the other hand, if the worst-case characteristics are used to select the voltage levels most of

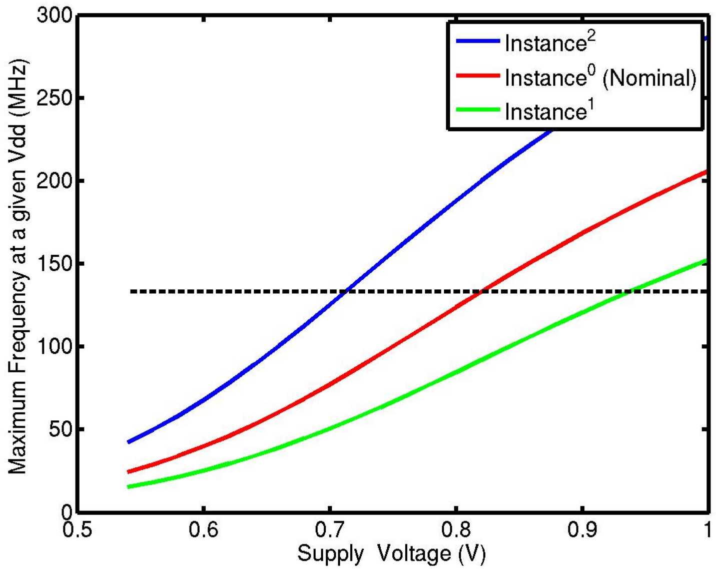


Figure 4.1: Impact of variations

the instances will dissipate significantly higher energy than required. To illustrate the effectiveness of customizing operating voltage according to individual chip instances, we present the overall energy distributions while using (i) chip-specific voltage setting that is aware of the F-V characteristics of the individual chip instances, and (ii) deterministic voltage setting based on the nominal characteristics, in Figure 4.2. The figure shows that using chip-specific voltage setting results in a far superior energy distribution compared to using deterministic voltage setting. It leads to more than 74% improvement in terms of the 90^{th} percentile of the energy distribution. Moreover, we estimated that the percentage of instances meeting the performance target in the variation-aware chip-specific case is 99% compared to 53% for the deterministic nominal case. This shows that voltage scaling when customized to each chip instance can enable power savings in inherently leaky dies and help meet required performance targets for naturally slow dies.

As mentioned earlier, continuous frequency and voltage levels are not practical to implement because of the complications involved in system design including designing the voltage regulator circuitry and the phase-locked loop (PLL) circuitry. In

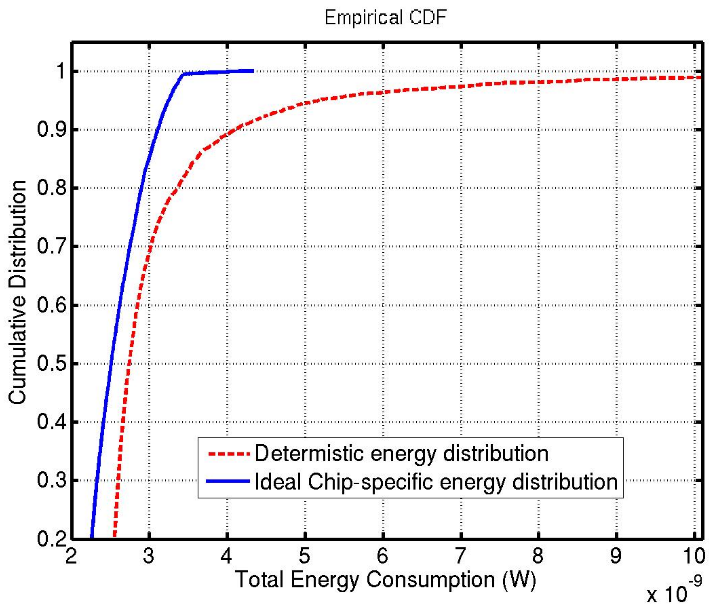


Figure 4.2: Cumulative energy distributions obtained using deterministic and *ideal* chip-specific approaches (continuous voltage levels)

practice, most of the systems can support only a small number of discrete levels (that remain same for all the instances). In the conventional systems, for the ease of implementation, discrete frequency-voltage pairs for different *performance states* are fixed during design time. Therefore, each performance state k has an associated operating point fixed at $\{F_k^j, V_k^j\}$, for each island j [1]. We refer to this scheme as 'conventional'. However, in the presence of variations, frequency-voltage characteristics vary across chip instances (and voltage domains), and hence, such fixed pairing can be extremely inefficient. In our work, we assume that frequency and voltage levels can be decoupled. Therefore, it is possible that different instances operating at same frequency have different voltage settings. Moreover, as mentioned in the previous section, the number of frequency (m) and voltage (n) levels can differ. This allows different instances of systems with even single specified performance state to operate at different voltages. We call this scheme as the 'variation-aware' (VA) scheme. Therefore, in the variation-aware scheme, for island j of chip instance i, the operating voltage is

[1]Note that all these values need not be unique, *i.e.,*, some of the islands may have same {F,V} operating point in more than one performance state of the *system*.

selected as follows,

$$V_k^{i,j} = min[\mathbf{V^j} > V_k^{i,j}(req)] \tag{4.1}$$

Here, $V_k^{i,j}(req)$ is the *ideal* voltage required by island j of instance i to operate at F_k^j, and is obtained using its individual frequency-voltage characteristics. $\mathbf{V^j}$ is the set of available voltage levels for island j. We now compare the energy distributions under the 'conventional' and the 'variation-aware' discrete voltage schemes. For both the schemes, the set of available voltage levels is same and is generated using the nominal characteristics. The found that the variation-aware scheme can lead to significant improvements in the overall energy distribution. In terms of the 99^{th} percentile, an improvement of up to 65% is observed. Additionally, the percentage of instances that can operate at the target frequencies is 81% in the variation-aware case as compared to 50% in the 'conventional' case in which the voltage levels are determined using the nominal characteristics. This shows that *utilizing* the voltage levels in the variation-aware manner, by customizing it for each instance can lead to significant improvements in the overall power and performance characteristics, even if only a small number of levels can be supported by the system.

In the next section, we describe the proposed methodology to determine a set of *variation-aware* voltage levels such that the performance yield is maximized without incurring severe penalty in the energy consumption.

4.4 Proposed methodology

In this section, we first describe the intuitive concept behind selecting the set of voltage levels in a *variation-aware* manner, and then, present our methodology for the same. We also present design-time and post-fabrication techniques to obtain the F-V characteristics of various instances, which are used by the proposed methodology.

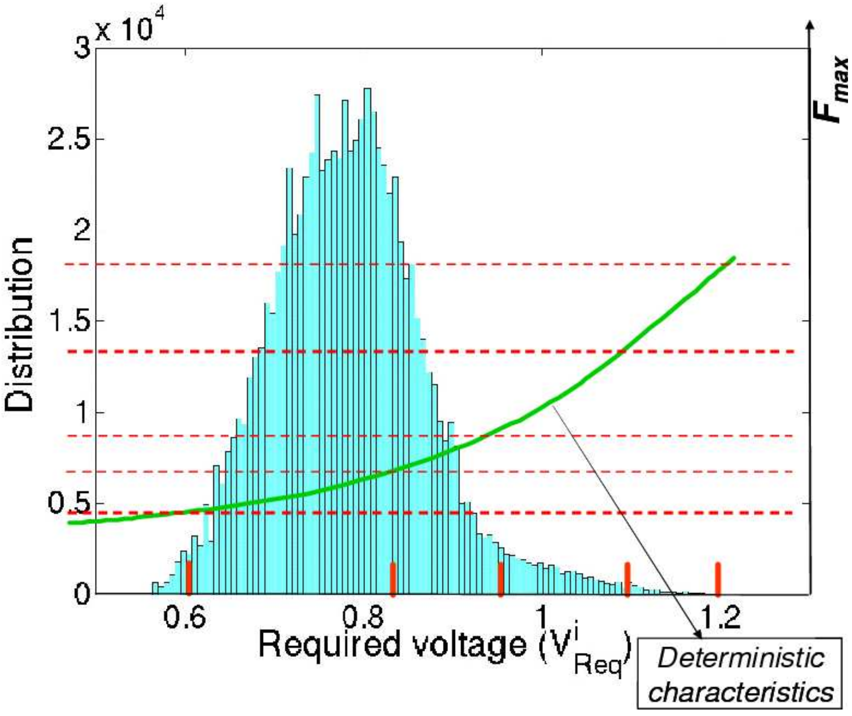
4.4.1 Motivation

As illustrated in Section 4.3, for a given F_{req}, each $instance^i$ has a different *ideal* voltage of operation, say, V^i_{req}. This *ideal* voltage taken across all the performance states and all the SoC instances results in a *required voltage* distribution ($\mathcal{V}_{req}$), for each island of the SoC, as shown in Figure 4.3. The resulting $\mathcal{V}_{req}$ distribution depends on the process, circuit, as well as the workload characteristics (*e.g.,* the weights of different performance states). However, since only a few discrete levels can be supported, an instance is often operated at the minimum available voltage level that is greater than V^i_{req}. This leads to dissipation of more energy than required. To obtain the optimal power and performance characteristics, each instance should be set to a voltage value as close as possible to the *ideal* voltage (V^i_{req}). This cannot be guaranteed if the voltage levels are determined on the basis of deterministic frequency-voltage characteristics (either the nominal or the worst-case) or chosen uniformly. In Figure 4.3 (a), we show the voltage levels selected based on deterministic worst-case characteristics for an island supporting five frequency levels. It shows that the voltage levels thus selected, are not conducive in the context of the given *required voltage* distribution, since for most of the instances the operating voltage *used* will be much higher than *required*. On the other hand, Figure 4.3 (b), shows a case in which the selected voltage levels *are* conducive for the given distribution since more levels are present in the voltage range with higher distribution density. This is the main intuition behind why determining the set of voltage levels based on the *ideal* voltage distribution can lead to improvements in the overall power-performance characteristics.

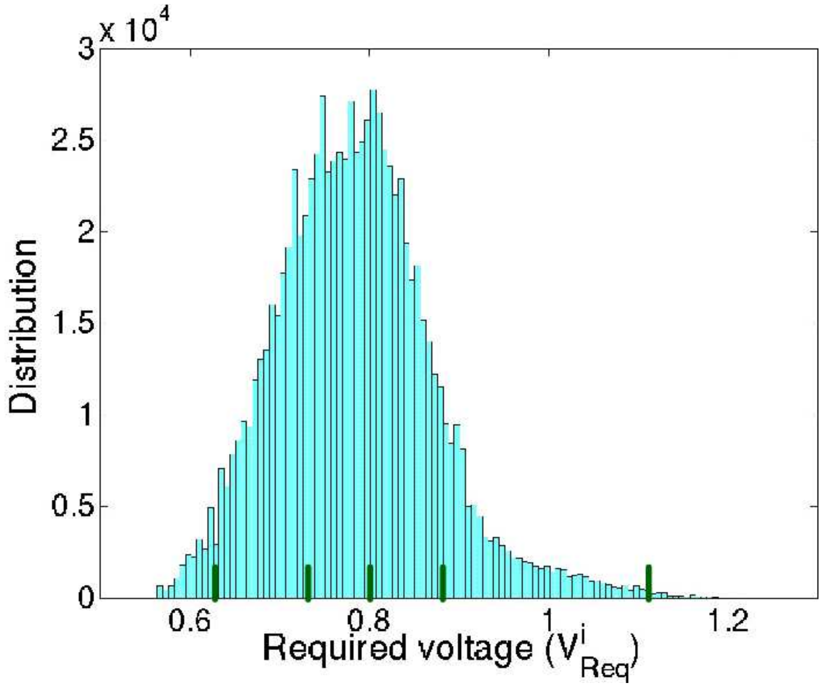
4.4.2 Workload model

We assume that the frequency levels associated with each performance state k, $k \in \{1, ..., K\}$, have already been determined for all the islands using extensive design time analysis. Furthermore, we assume that the percentage of time the SoC is likely to spend in each state has also been determined [2]. This workload model allows decoupling the impact of workload variations (and hence, the specific voltage

[2] Note that in each state different islands can operate at different frequencies.



(a)



(b)

Figure 4.3: Variation-aware voltage levels

scheduling scheme employed) from the effectiveness of choosing the set of voltage levels. We assume that there is a maximum voltage allowed, V_{max} that is also specified based on the design considerations. In this work, we evaluate the effectiveness of the proposed methodology under two scenarios. In the first scenario, the mix of the performance states is maintained for each chip instance and the goal is to optimize the overall energy distribution. Therefore, all the chip instances are required to exhibit same performance characteristics. Instances that cannot meet the required performance mix are discarded. In the second scenario, the goal is to improve the performance distribution. Therefore, in this case, instances that are unable to meet the required frequency targets even at V_{max} are allowed to execute at highest available frequency level possible at V_{max}. In both the scenarios, we show that the number of chip instances meeting specified power and performance targets significantly improve using variation-aware voltage selection.

4.4.3 Objective

The objective is to find a set of voltage levels for each SoC island such that the distribution of the total energy consumption is optimized with respect to a certain distribution metric in the presence of variations. In other words, find (**V**) such that metric $\mathcal{M}et$ of the total energy distribution $\mathcal{E}_{Tot}$ is optimized. $\mathcal{M}et$ can be any property of interest such as mean, mean+sigma, yield for a given energy constraint or $\mathcal{N}^{th}$ percentile of the total energy distribution. The total energy random variable is given by $\mathcal{E}_{Tot} = E^{Tot}_{dyn} + E^{Tot}_{leak}$. Here, $\mathcal{E}^{Tot}_{dyn}$ and $\mathcal{E}^{Tot}_{leak}$ represent the total dynamic and leakage energy dissipation, respectively. Now, the total energy dissipation of island j of a chip instance i is given by,

$$E^{i,j}_{Tot} = \sum_{k=1}^{N_{states}} E^{i,j}_{dyn,k} + E^{i,j}_{leak,k} \tag{4.2}$$

where,

$$E^{i,j}_{dyn,k} = E_{dyn,nom} * (\frac{V^{i,j}_k}{V_{nom}})^2 * nCC_{j,k} \tag{4.3}$$

$$E^{i,j}_{leak,k} = I^{i,j}_{leak} * (V^{i,j}_k) * V^{i,j}_k * t^{i,j}_k \tag{4.4}$$

In equation 4.2, the summation is over all the possible performance states of the given SoC. $V_k^{i,j} = min(\mathbf{V^j} > V_{req,k}^{i,j})$ is the minimum of the available voltage levels that is greater than the *ideal* voltage required by the given island j within the given chip instance i. $nCC_{j,k}$ is the total number of cycles spent in the k^{th} state by island j and $t_k^{i,j}$ is the total time spent in state k.

4.4.4 Overall methodology

Each term of $\mathcal{E}_{Tot}$ is a random variable. In general, $\mathcal{M}et(\mathcal{E}_{Tot})$ (or even E_{Tot}^i) cannot be expressed as an analytical (or closed form) function of the set (**V**), and hence, the optimal (**V**) cannot be obtained by analytical optimization techniques. In this work, we present a three step methodology to determine a set of variation-aware voltage levels, as shown in Figure 4.4. We now discuss each of the steps in detail.

Step 1: Obtain a sample set of chip instances by Monte Carlo sampling of the process parameter space. Each instance in the sample set is associated with the frequency-voltage (F-V) characteristics and the leakage-voltage characteristics corresponding to each SoC island. We generate the sample set while accounting for inter-die variations and spatially correlated intra-die variations in effective channel length using the framework presented in Section 2.6. Leakage power for each island is obtained using the high-level modeling approach presented in [63]. We use ring oscillators of equivalent logic depth for each island in order to obtain the delay (frequency) characteristics of the island. We discuss the delay and leakage models that we have developed to compute the leakage-voltage and delay-voltage characteristics under variations in Appendix B of the thesis.

Step 2: For each island j of instance i in the sample set, compute the minimum voltage, $V_{req,k}^{i,j}$ required to meet the target frequency of operation in each state. If the required voltage is greater than the maximum allowed voltage V_{max}, then either discard the instance (*scenario 1*) or determine the maximum available frequency level that can be supported at V_{max} (*scenario 2*).

Step 3: Quantize the *ideal* voltage distribution obtained in Step 2 such that the quantization error is minimized.

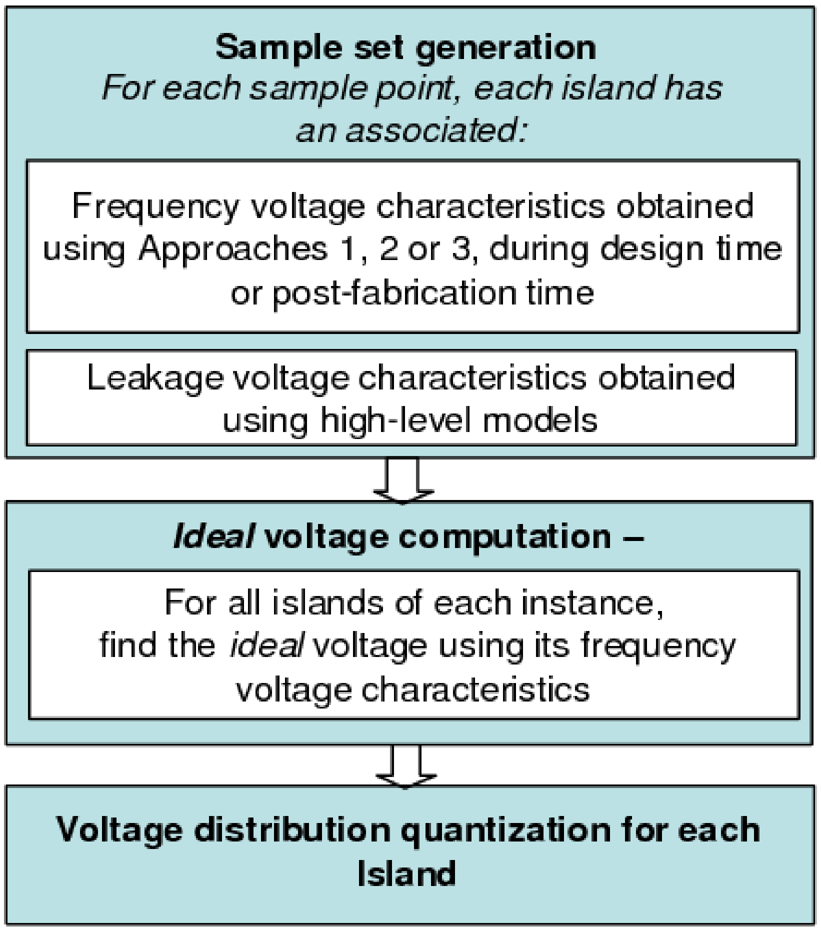

Sample set generation
For each sample point, each island has an associated:
Frequency voltage characteristics obtained using Approaches 1, 2 or 3, during design time or post-fabrication time
Leakage voltage characteristics obtained using high-level models
Ideal voltage computation –
For all islands of each instance, find the ideal voltage using its frequency voltage characteristics
Voltage distribution quantization for each Island

Figure 4.4: Methodology for variation-aware voltage level selection

In the rest of this section, we discuss the generation of the sample set, the quantization of the *ideal* voltage distribution, and the proposed approaches for estimating frequency-voltage characteristics of the instances.

4.4.5 Sample set generation

As previously mentioned, we use the framework presented in Section 2.6 to generate a sample set of instances. Here, we briefly discuss how it is used in the context of the work presented in this chapter. In this work, we obtain delay and leakage characteristics at the island level. We use a system-level floorplan to determine the mapping of the components, (and hence, the voltage islands) to the grids, as shown in Fig. 4.5. This mapping determines the device count corresponding to each component/island in each grid. This is used by the high-level leakage models presented in [63], that we use to compute the leakage dissipation of the SoC components. These models use the transistor (N-type and P-type) level leakage currents as inputs. The transistor level leakage current models are developed as a function of channel length and voltage variations (Appendix B), based on the data obtained using SPICE simulations. To obtain the delay characteristics, we assume that the delay of each island can be correlated with the ring oscillator (RO) of the same logic depth as the island. We assume that for each instance, the transistors of such an RO can be characterized with the mean channel length of the devices in that island. Considering that at the granularity of an island, impact of intra-die variations is more likely to average out, we believe this assumption is quite reasonable in the context of our work.

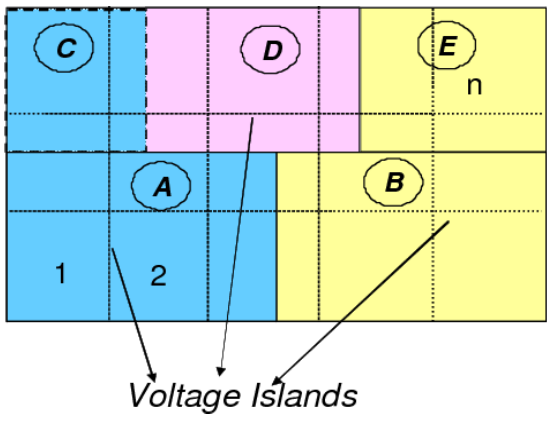


Figure 4.5: Component to grid mapping

4.4.6 Voltage level quantization

Given the V_{req}^{j} distribution for each island j, we want to select a set of n_j voltage levels ($\mathbf{V^j}$) such that the overall error resulting from mapping the *ideal* voltage for each instance ($V_{req,k}^{i,j}$) to the next available level in the set is minimized. This is similar to the scalar quantization problem. In this work, we propose to solve it using *Lloyds-Max* algorithm [89]. In the *Lloyds's-Max* algorithm, for a given data set X, a k-level partition $D = (d_0, d_1, \ldots, d_k)$ and a reference value for each partition $R =$

$(r_1, r_2....r_k)$ is obtained such that the mean square error (MSE) arising from mapping $x_i \in X$ to r_j if $x_i \in [d_{j-1}, d_j)$ is minimized. The algorithm iteratively computes r_j (as the centroid of the j^{th} partition) and d_i as the mean of adjacent reference points until the MSE is within a specified tolerance. We use the *required* voltage distribution obtained in step 2 of the methodology as the data set, and obtain the discrete voltage levels as the reference points [3]. This method ensures improvement in the overall energy distribution, but does not ensure optimality.

4.4.7 Determining the frequency-voltage characteristics

As mentioned earlier in this section, to generate the *ideal* voltage distribution we need to be able to compute the frequency-voltage characteristics of SoC islands for each sample instance. Since process variations is a post-fabrication phenomenon, post-fabrication characterization at island-level granularity is likely to generate the most accurate frequency-voltage characteristics, and hence, the most accurate voltage distribution. In this approach, the frequency-voltage characteristics of each island of all the instances can be determined by finding the maximum frequency that can be supported at each voltage level. This requires sweeping through both the voltage levels and the frequency levels at a pre-determined granularity, testing the island for correct operation at each step. However, this method is prohibitively time consuming, and expensive considering the costs associated with the currently available post-fabrication testing equipment. We present three alternative approaches that can be used instead to determine the frequency-voltage characteristics, and hence, generate the distribution of ideal voltage levels:

- **Approach 1:** In this approach, we generate a sample set of SoC instances and determine the frequency-voltage characteristics of the sample set using design-time analysis. Most of the time, information about process variations can be obtained from the foundries. Therefore, it is possible to generate a sample set based on the available information and component-level layout of the SoC. We

[3]Note that there is additional error introduced here since the algorithm computes r_i as the centroid of the partition but we want the voltage value to round up for each partition.

describe generation of the sample set in Section 2.6 of the thesis. We have developed high-level SPICE models to determine the leakage-voltage and the frequency-voltage characteristics as described in Appendix B. It is generally not possible to simulate the SPICE netlist of the entire SoC. Therefore, we use a ring oscillator (RO) of the equivalent logic depth as each of the islands to simulate the delay-voltage (and hence, the frequency-voltage) characteristics. Using these models, we can compute the ideal voltage required by an instance for a given delay (frequency) using the delay information of the instance at a known voltage. As shown in Figure 4.6, our models provides an accurate fit for the experimental data with mean relative error less than 1%. Details of these models are presented in Appendix B. The advantage of this approach is that the F-V characteristics and hence, the set of voltage levels can be determined at the design time. This would help in generating the initial power-performance specifications of the SoC at the design time and thereby, can efficiently drive design iterations.

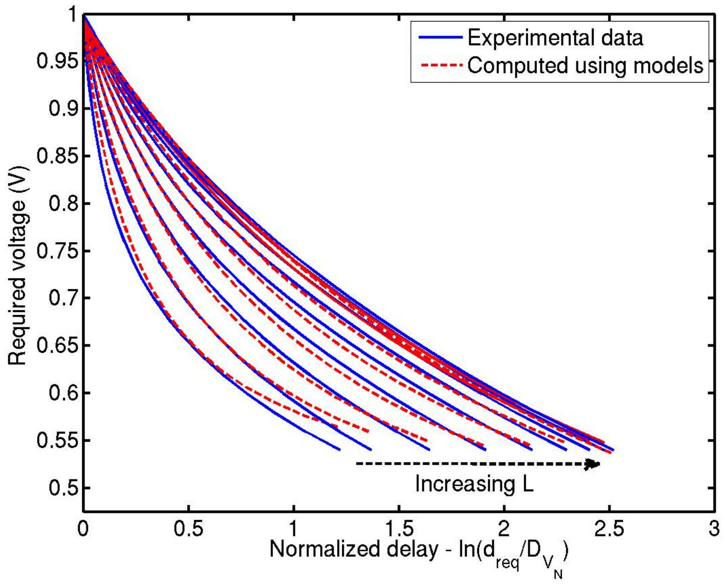


Figure 4.6: Voltage as a function of circuit delay for different chip instances

- **Approach 2:** In the second approach, we propose using design time analysis as well as post-fabrication on-chip measurements to obtained the frequency-voltage characteristics for a given population of instances. We assume that most circuits follow delay-voltage model of the form presented in Appendix B of this chapter. We used ring-oscillators of different sizes and a 16x16 multiplier circuit to verify that the delay-voltage models follow the same form (Appendix B). For the multiplier circuit, we found that the fitted model closely followed the experimental data obtained through simulations using NANOSIM [90] and SPICE, as is shown in Figure 4.7. The mean relative error is less than 2%.

  Now, assuming most of the circuits follow the same form of the delay-voltage models, we propose that the models are calibrated post-fabrication using on-chip measurements. The primary advantage of this approach is that only a few measurements are required to completely calibrate the models since the model form is known apriori. For example, the delay model presented in this chapter can be calibrated using a total of 12 measurements across 3 instances. This approach will provide more accurate estimate of the *ideal* voltage distribution. It can either be used by itself or along with Approach 1 to refine the voltage values after fabrication.

- **Approach 3:** In this approach, we propose using only the data obtained using post-fabrication on-chip measurements. We propose that using post-fabrication measurements delay-voltage characteristics of a small sample set of chip instances can be obtained. These characteristics can then be used to develop the delay-voltage best-fit models which may differ from one circuit to another (or can be of the same form as proposed in this chapter, Appendix B). Now, for a given instance, the delay information measured at a known voltage (usually, V_{max}) can be used to determine its the frequency-voltage characteristics. This approach is more accurate, and is likely to be useful in cases where process information is not available apriori, such as first spin of a particular chip in a new process. However, with this approach it is not possible to provide any design time estimates. To verify the effectiveness of this approach we use the follow-

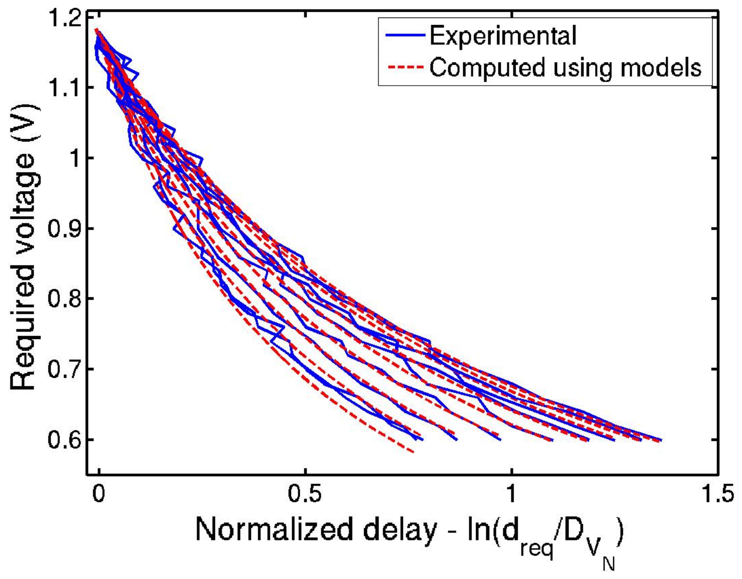


Figure 4.7: Delay-voltage characteristics for a 16x16 multiplier circuit

ing method. We developed the delay-voltage models using a small number of instances, and then, used the hence derived models to compute the required voltage levels for various delay values for a large number of instances of the 16x16 multiplier circuit. In Figure 4.8, we plot the accuracy of the model obtained as a function of number of sample points used to create the model. We see that accuracy of almost 2% can be obtained by characterizing as few as 500 sample points.

At runtime: As is shown in Appendix B, the proposed delay-voltage models use the delay information of a given instance at a known voltage (V_{max}) to compute the *ideal* voltage ($V_k^{i,j}(req)$) required by the instance to meet the target frequency ($F_k^j(req)$). In the current design flow, this delay (at V_{max}) is determined during post-fabrication testing and binning process. Therefore, for each island of each instance a look-up table can be determined in which maximum frequency level possible for the given instance can be computed and stored for the selected set of voltage levels. These frequency values can be computed using the models proposed in this work. Alternatively, a more accurate look up table can be computed by measuring the delay

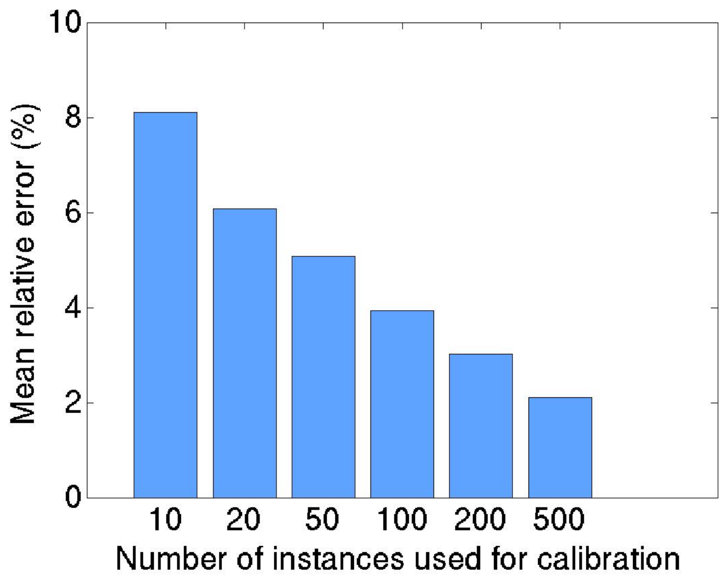


Figure 4.8: Model accuracy in as a function of number of sample points used to create the model

only at the selected voltage levels. This look up table can be easily used at runtime to determine the frequency-voltage setting for each performance state, customized to the F-V characteristics of the given instance.

As mentioned previously, though *Approach 1* is likely to introduce some errors it will be crucial in making design iterations and providing initial estimates. Then, based on the design, approach 2 or 3 can be used at post-fabrication time to compute more accurate voltage levels. In the next section, we present the experimental results and show that the proposed methodology although not claimed to be optimal, leads to significant improvements in the overall power-performance characteristics.

4.5 Experimental results

In this section, we first describe our experimental methodology, and then present results to establish the effectiveness of variation-aware voltage level selection.

4.5.1 Methodology

For our experiments, we consider an ARM946 processor [42] based system which is described in the next sub-section. An in-house simulation-based power estimation framework [45] that uses instruction-based power models is used for dynamic power estimation. Leakage power estimates at the nominal frequency and voltage levels are obtained using approximate gate-counts and high-level techniques similar to [53]. For our studies, we assume inter-die variations of $\sigma = 8\% \ \mu$ and spatially correlated intra-die variations of $\sigma = 6\% \ \mu$ in effective channel length. Data for calibrating the leakage power models is obtained via transistor-level HSPICE simulations. We use the Berkeley Predictive Technology $90\ nm$ BSIM3 model card [55] for the simulations. We implemented an 11-stage ring-oscillator in HSPICE to generate the frequency-voltage characteristics of the system under consideration. The methods for obtaining and evaluating various voltage sets are implemented in MATLAB [56].

4.5.2 System Description

We consider an SoC implementation of the 802.11 protocol's Media Access Control (MAC). Various tasks of the 802.11 MAC sub-system are shown in Figure 4.9(a). First, the LLC task (or link layer control) writes incoming frame bits to the MAC memory. Then, Integrity Checksum Value (ICV) of the frame is computed in parallel with Wired Equivalent Privacy (WEP) encryption of the same frame. WEP encryption consists of two main tasks, namely, WEP_INIT and WEP_ENCRYPT. In WEP_INIT, a 256 byte S-box is initialized, while in WEP_ENCRYPT, the incoming frame is encrypted and the ciphertext is generated. Task HDR generates the frame header, and assembles it with the ciphertext and the ICV. Then, Frame Check Sequence (FCS) is computed over the assembled frame. The entire frame (with header, ICV and FCS) is then sent to the physical layer (PLI) for transmission. MAC CTRL implements the channel access algorithm (CSMA/CA). Figure 4.9(b) illustrates a multi-voltage SoC implementation of the MAC layer operations discussed above. LLC is the hardware implementation of the link layer control. It receives frames from the link layer, writes them in the memory MEM_1 and enqueues frame addresses in Queue1. These addresses are read from the queue by (i) WEP, a HW co-processor implementing the entire Wired Equivalent Privacy (WEP) task, and (ii) ARM_1, which computes the ICV for each frame. The ICV and WEP tasks proceed in parallel. The encrypted frames are written in MEM_2, and the frame addresses are written in Queue2. The embedded CPU, ARM_2 implements the HDR and the MAC_CTRL tasks. After processing is done, a HW implementation of the Physical Layer Interface (PLI) dequeues the frame addresses from Queue3, retrieves the encrypted frames from MEM_2, and passes them to the physical layer hardware. Communication between various components is implemented using two AMBA bus segments, AHB_1 and AHB_2 that communicate though the BRIDGE component.

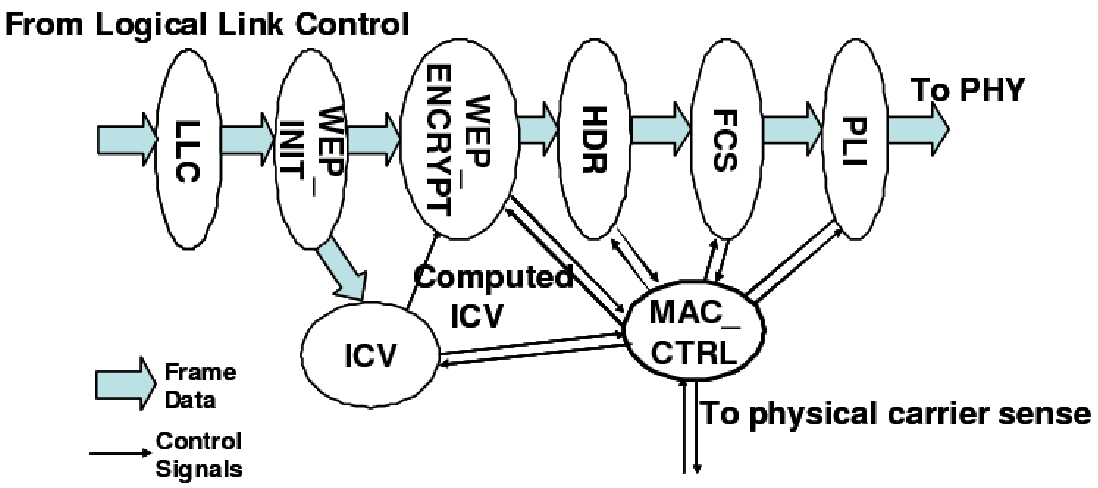
Table 4.1: Different performance states of the system

Performance States	Data Rate	Frequency of island 1	Frequency of island 2	(Probability of occurrence)
State-1	1.5 Mbps	350 MHz	300 Mhz	0.05
State-2	1 Mbps	250 Mhz	200 Mhz	0.65
State-3	750 Kbps	150 Mhz	200 Mhz	0.2
State-4	500 Kbps	100 Mhz	200 Mhz	0.1

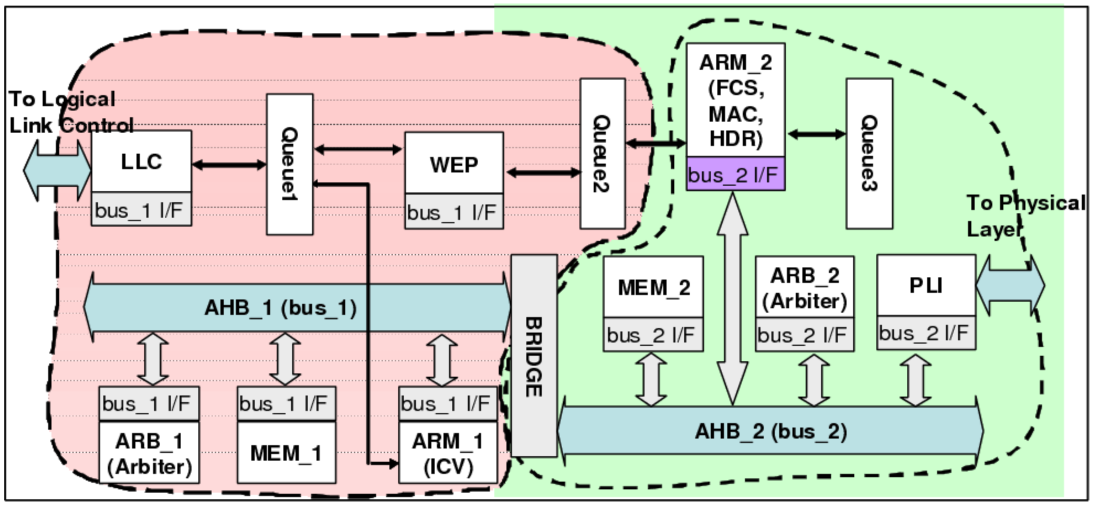
4.5.3 Results

Experiment 1: For the first set of experiments we assume that the SoC is capable of operating in 4 performance states, as shown in Table 4.1. The table also lists their probability of occurrence that is chosen arbitrarily for these experiments. Also, we assume that chip instances, for which it is not possible to meet the target frequency settings corresponding to each of the performance states, are discarded. In Figure 4.10, we illustrate the energy-performance characteristics for various voltage selection schemes described in this chapter. The x-axis represents the energy dissipation whereas the y-axis represents the percentage of chip instances that can meet performance targets for different values of energy dissipation. Therefore, the area under the curve corresponds to the number of chips that satisfy performance targets.

We see that using the worst-case characteristics, a large number of instances can meet the performance targets but at a very high energy cost. With the proposed



(a)



(b)

Figure 4.9: (a) Tasks involved in the IEEE 802.11 MAC protocol (b) 802.11 MAC processor architecture with two voltage/frequency islands

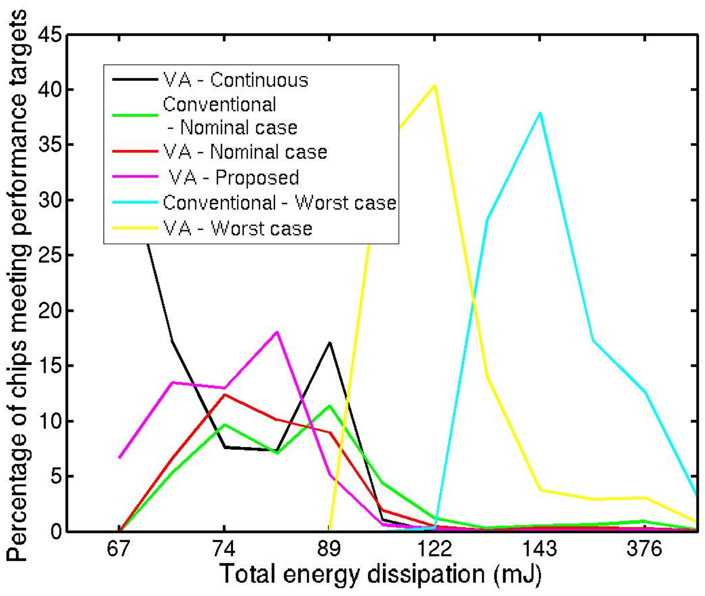


Figure 4.10: Energy distribution of SoC instances

variation-aware voltage selection methodology, number of chips meeting performance requirements with energy dissipation of less than 90mJ, is 68% higher as compared to the conventional scheme with nominal voltage selection and 48% higher as compared to the variation-aware scheme with nominal voltage selection. We present the energy scatter plot for different voltage selection schemes in Figure 4.11. It can be seen that with the proposed voltage-selection scheme, energy dissipation of a given chip instance is closer to the *Ideal* (Continuous) case.

Experiment 2: The second set of experiments is conducted under *scenario2*. For these experiments we assume that the required data rate is 1.2Mbps. For the given SoC it can be achieved by setting the operating frequency of Island 1 to 300Mhz and that of Island 2 to 200 Mhz. We present the data rates for 10000 SoC instances in Figure 4.12. For comparison purposes we plot the data rates obtained using the conventional scheme in which only a single level is supported for each frequency, as well the *ideal* case in which continuous levels can be supported. Our experiments indicate that with the proposed techniques, significant improvement in the number of instances meeting performance targets can be obtained. In particular, 77% of the instances meet

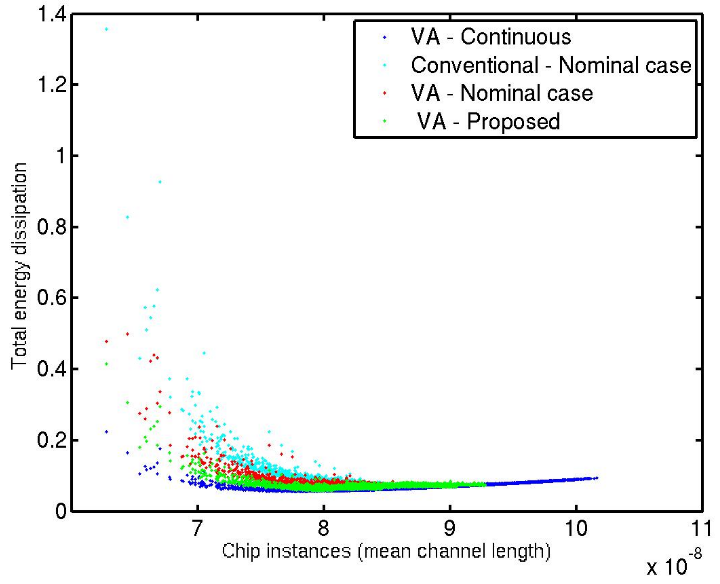


Figure 4.11: Energy dissipation for 10000 instances of the 802.11 MAC processor

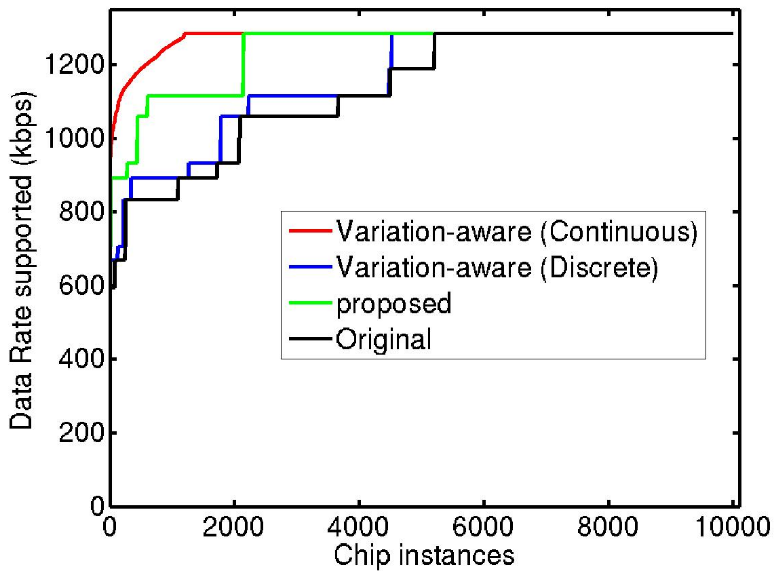


Figure 4.12: Data rates for chip instances

targets as opposed to 48% with the conventional or 54.8% with deterministically determined levels.

4.6 Conclusion

In this chapter, we studied the impact of process variations on the effectiveness of deterministic voltage selection schemes for SoCs using multiple-voltage levels to enable energy reduction. We show that conventional approaches that assume deterministic frequency-voltage characteristics can cause significant sub-optimality and energy wastage. We present techniques to enable selection of variation-aware discrete voltage levels at either design time or post-fabrication. We also discuss how at run-time, voltage levels can be chosen in a chip-specific manner. Experimental results show that with voltage-levels selected using the proposed variation-aware methodology, significant improvements in the number of instances that meet given power and performance targets can be achieved as compared to schemes using voltage levels selected based on the nominal or the worst case characteristics.

The text of this chapter, in part, is based on material that is under review in the IEEE Transactions on VLSI Systems (S. Chandra, A. Raghunathan, and S. Dey, *Variation-aware Voltage Level Selection*, IEEE Trans. VLSI Systems). The dissertation author was the primary researcher and author, and the coauthors listed in these publications collaborated on, or supervised the research that forms the basis for this chapter.

Chapter 5

Variation-aware Analysis and Design Techniques at the Architecture-level

5.1 Introduction

Until now we focused on *system-level* techniques for considering variations during power analysis and optimization. In this chapter, we present techniques to analyze the impact of architecture and application task graph interaction on the extent of impact of variations on the overall system performance. As mentioned earlier, variations in the device parameters such as transistor lengths and interconnect widths cause delay characteristics of devices and interconnects to deviate from the nominal values. At system level, this is manifested as variations in the maximum operating frequencies (FMAX) of components such as processors, DSPs and memories [4], and as variations in the latencies of global SoC paths. In such conditions, multi-frequency, multi-voltage systems that are based on globally asynchronous locally synchronous (GALS) design paradigm are inherently more robust to variations as is mentioned earlier in this thesis. In this chapter, we assume that the communication architecture interfaces are capable of supporting asynchronous communication and discuss such an interface in detail later in the chapter. To decouple the impact of supply voltage on performance, we assume that all components operate at the nominal supply voltage.

5.1.1 Overview of progress and future directions

In particular, the contributions of this work are as follows:

- We show how the extent of impact of process variations on the overall system performance depends on the (a) application characteristics such as taskgraph structure and input rate, (b) architectural characteristics such as communication architecture and protocols such as maximum burst size and component priorities, and (c) mapping of the tasks to the SoC components.

- We present an analysis methodology that captures these effects during system level performance estimation while considering process variations. It allows analysis of SoCs with components having independent operating frequencies and can output various statistics such as overall performance distribution, critical path distribution and distribution of output rates of different tasks for a system affected by inter- and intra-die variations. There are some examples of variability analysis tools in the literature. In [19], a statistical simulator is presented that obtains throughput and maximum clock frequency distributions by performing Monte Carlo simulations on an analytical throughput model. However, same operating frequency for all the cores is assumed. Marculescu et. al. have shown that GALS design paradigm can result in significant performance improvements over single island designs [32, 33]. However, their analysis assumes point-to-point communication links and does not take into account the effect of the underlying architecture on performance variability.

- We propose an application level technique and two architecture level techniques that can be employed to recover the performance loss due to variations to a significant extent. The architectural level techniques are based on selectively changing communication protocols such as maximum burst size and component priorities on the shared bus. The application level technique is based on selecting appropriate task mapping.

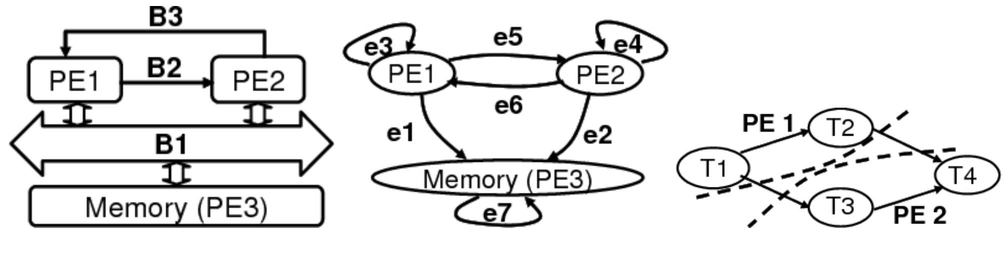
- We illustrate the applicability of the proposed design techniques by presenting the performance distributions for several systems, obtained using our analysis methodology. Our results show that improvement of up to 78% can be obtained in the parametric yield of performance distributions by static design-time application of these techniques.

The rest of this chapter is organized as follows. In Section 5.2, we introduce terms and notations used in this chapter. In Section 5.3, we illustrate the effect of the various application and architectural factors on the overall system performance in the presence of process variations. In Section 5.4 we briefly describe the proposed analysis methodology. In Section 5.5, we present design techniques to recover the performance loss due to variations. We present experimental results in Section 5.6, and conclude in Section 5.7.

5.2 Preliminaries

Consider the system shown in Figure 5.1(a). We call this system SYS1. It consists of 2 processing elements, namely PE1 and PE2. The processing elements[1] can be master elements such as general purpose processors, DSPs and custom hardware accelerators or slave elements such as memories. In SYS1, the memory (PE3) is shared by the two PEs and is accessed using a shared bus B1. In addition, the two PEs can send synchronization (SYNC) signals to each other using dedicated channels indicated by B2 and B3. Figure 5.1(b) shows the event set of SYS1. Events *e3*, *e4* and *e7* represent the computation events on components PE1, PE2 and PE3, respectively. Events *e1* and *e2* represent the communication events between the memory and PE1 and PE2, respectively. Events *e5* and *e6* represent the unidirectional synchronization events (SYNC) from PE1 to PE2 and vice-versa. In Figure 5.1(c), an application taskgraph, referred to as Taskgraph 1, is shown. It consists of 4 tasks namely, T1, T2, T3 and T4. The edges represent the precedence constraints. The figure also shows an associated task mapping wherein, tasks T1 and T2 are mapped to PE1 while tasks T3 and T4 are mapped to PE2. In Figure 5.1(d), events associated with each task of Taskgraph 1 under Mapping 1 followed by their corresponding *weights* are shown.

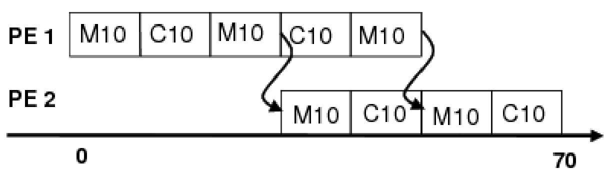
[1]Here, we will use the terms processing elements and components interchangeably.



(a) (b) (c)

T1	*e1*:10-> *e3*:10-> *e1*:10-> *e5*:1
T2	*e1*:10-> *e3*:10-> *e5*:1
T3	*e2*:10-> *e4*:10
T4	*e2*:10-> *e4*:10

(d)



(e)

Figure 5.1: (a) System 1 , (b) Event set, (c) Taskgraph 1, with Mapping 1, (d) Task to event mapping, (e) Execution traces

For computation events, weight is the number of clock cycles for a given computation and for communication events weight is the number of data words being transferred. For example, events associated with T1 are: 10 words read from memory, 10 computational cycles, 10 words write to memory and SYNC to PE2. The execution traces for PE1 and PE2 are shown in Figure 5.1(e). *C* denotes a computation event whereas *M* denotes a communication event. For simplicity of representation, timed execution of handshake and SYNC events is not shown in these traces, but SYNC events are indicated using arrows. Under process variations, all instances have same weight for a given event but the execution time depends on their respective clock frequencies. However, for the ease of calculations, we differentiate in terms of the event weights represented in terms of *time units (tu)*. For example, if PE1′ is an instance of PE1 that is 20% slow due to variations, then it takes 12.5 tu to execute an event of original weight 10 tu.

5.3 Factors affecting impact of process variations

In this section, we illustrate how the impact of process variations on the overall system performance is affected by factors such as application, architecture and mapping characteristics. For simplicity of discussion, here we limit the effect of variation to a single component. However, analysis techniques presented in Section 5.4 applies to all components being randomly affected by variations.

5.3.1 Effect of application characteristics

We first illustrate the effect of taskgraph structure, and then we illustrate the effect of arrival rate of inputs.

Effect of application taskgraph

Consider the taskgraph shown in Figure 5.2(a), executing on SYS1. Task mapping and the associated events are shown in the figure. Task T1 is mapped to PE2 and T2 is mapped to PE1. First set of traces in Figure 5.2(b) illustrates the activity of the two components when no variations are present. PE2 performs 10 tu of computation followed by 10 tu of communication and then SYNC to PE1. PE1 performs 20 tu of computation after receiving the *SYNC*. The system takes 40 tu to processes each request and can process inputs arriving once every 20 tu. The second set of traces in Figure 5.2(b) illustrate the system execution when PE2 slows down by 20% due to variations. It can be seen that the performance of PE1 reduces to one computation every 25 tu as opposed to once every 20 tu, due to the synchronization waits. This example shows that performance of not only the task mapped to the PE affected by variations is impacted but also of a task mapped to an unaffected PE is impacted because of the dependency introduced by the taskgraph. Due to the dependence PE affected more by variations introduces slack in execution of a task on PE affected less.

Effect of input arrival rate

Now consider the operation of Taskgraph 1 (Figure 5.1(c)) under two scenarios, (a) new input arriving every 75 tu and (b) new input arriving every 60 tu. Figure 5.3(a), (b) shows the execution traces for cases (a) and (b), respectively. In both cases, first set of traces are for an instance with no variations and second set of traces for an instance with PE1 20% slow due to variations. The input request is in the form of packets here. We see that in case (a), variations cause the processing of each packet to delay by the same amount i.e., 12.5 tu. However, in case (b) processing of packet 1 gets delayed by 25, packet 2 by 27.5, packet 3 by 30 tu and so on[2]. In this case, the delay introduced for one packet cascades for other packets, because of the limited slack presented by the application, and hence, overall performance gets severely affected compared to case (a). This shows that applications that inherently present lower slack due to tighter deadlines are more likely to be impacted by variations with performance deterioration increasing with application running time.

These example illustrate that the extent of performance degradation due to process variations depends strongly on the properties of the application running over the affected hardware.

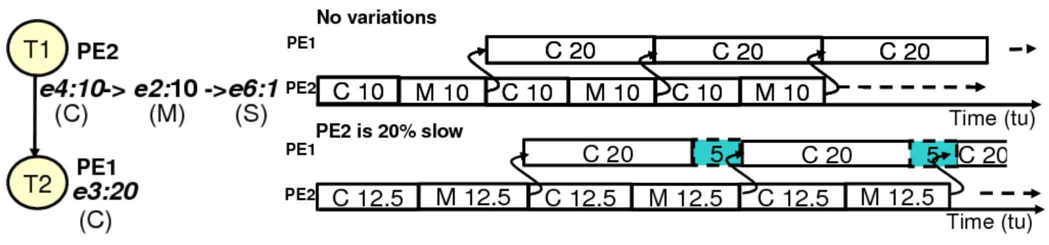
5.3.2 Effect of architectural characteristics

We first illustrate the effect of the underlying communication architecture and then the effect of communication protocols on the extent of performance degradation due to variations.

Effect of the communication architecture

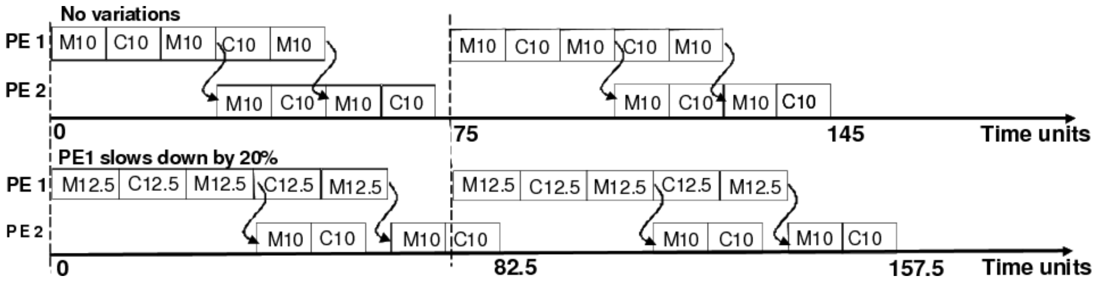
Figure 5.4(a) shows Taskgraph 3 that has two independent tasks. The figure also shows the task mapping (T1 $->$ PE1, T2 $->$ PE2) and events associated with

[2]Here PE1 has higher priority on the bus than PE2

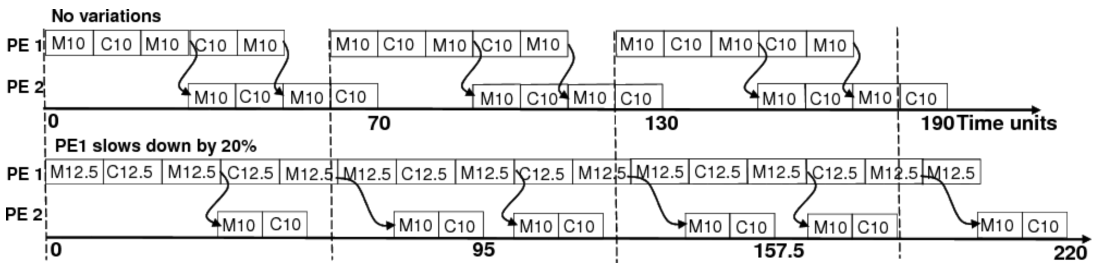


(a) Taskgraph 2 (b) Execution traces

Figure 5.2: Impact of variations depends on the taskgraph



(a) New packet arrives every 75 tu



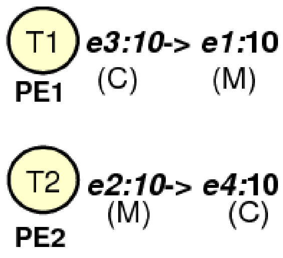
(b) New packet arrives every 60 tu

Figure 5.3: Impact of the input arrival rate

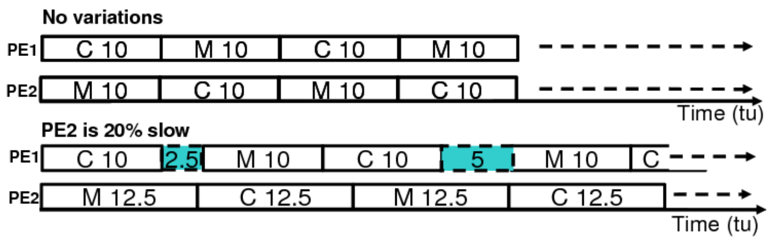
each of the tasks. For T1, events associated are 10 tu of computation and 10 tu of communication. For T2, events associated are 10 tu of communication and 10 tu of computation. Execution traces for no variation case and for PE2 affected by variations are shown in Figure 5.4(b). The shaded blocks in the second set of traces indicate bus wait times that are introduced due to slowing of PE2. In this case, the time to process 10 packets of T1 increases from 220 tu[3] to 252 tu and from 219 tu to 262.5 tu for processing 10 packets of T2. On the other hand, consider a system with an architecture similar to SYS1 but with a dual port memory that allows simultaneous access to both the PEs. For such a system architecture, the degradation is limited to T2 only.

This shows that process variations can lead to performance degradation of not only the component directly affected but but also of an component with independent tasks, due to dependency introduced by shared resources. Accurate performance analysis of SoCs impacted with variations should take such indirect effects into account.

[3]The calculations incorporate time for handshake events.

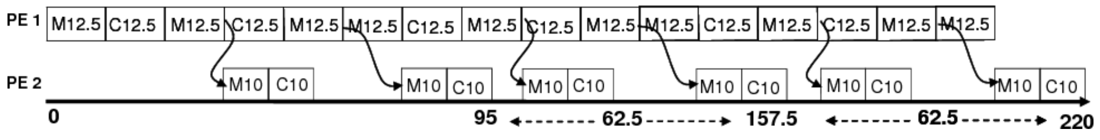


(a) Taskgraph 3

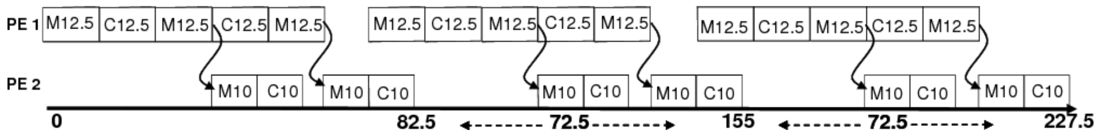


(b) Execution traces

Figure 5.4: Impact of variations depends on the communication architecture



(a) PE1>PE2



(b) PE1<PE2

Figure 5.5: Impact of variation depends upon the bus protocol

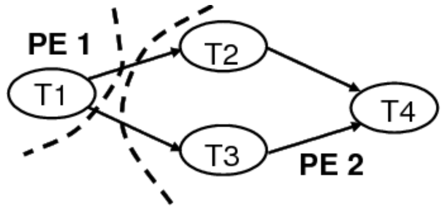
Effect of the communication protocols

Lets again consider Taskgraph 1 (Figure 5.1(c)) mapped to SYS1 and assume input arrival of once every 60 tu. We show the operation of the system when PE1 slows down by 20% due to variations, for two scenarios namely, (a) PE1 has higher priority on the bus and (b) PE2 has higher priority on the bus. The two cases are illustrated in Figure 5.5(a) and (b) respectively. In case (a) processed packets are generated at every 62.5 tu as opposed to case (b) in which packets are generated at every 72.5 tu but with slightly higher initial latency 95 tu as opposed to 82.5 tu in case(b). Hence, for large number of packets ($\geq$ 3) case(a) performs significantly better than case (b).

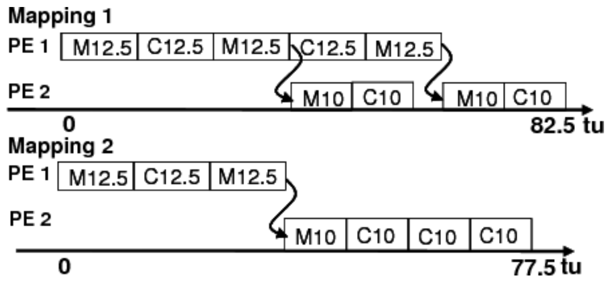
The above two examples illustrate that the extent of performance degradation due to process variations depends strongly on the underlying communication architecture and protocols.

5.3.3 Effect of task mapping

Task mapping determines how the tasks of an application interact when are mapped to the given SoC architecture. Therefore, different task mappings introduce different kind of dependencies and behave differently under process variations. To show this, we again consider Taskgraph 1 under Mapping 1 as shown in Figure 5.1(c). It takes 70 tu to process each packet. Now consider, an alternative mapping of the same taskgraph on SYS1 as shown in Figure 5.6(a). Figure 5.6(b) shows the execution traces when PE 1 slows down by 20% under the two mappings of the task graph on SYS1. It can be seen that Mapping 2 is more robust to variations since PE1 and PE2 are interrelated by one memory transfer as opposed to Mapping 1 wherein PE2's operation depends on two memory transfers for each packet. This example shows that the extent of performance degradation is highly dependent on the task mapping since that determines some of the operational dependencies.



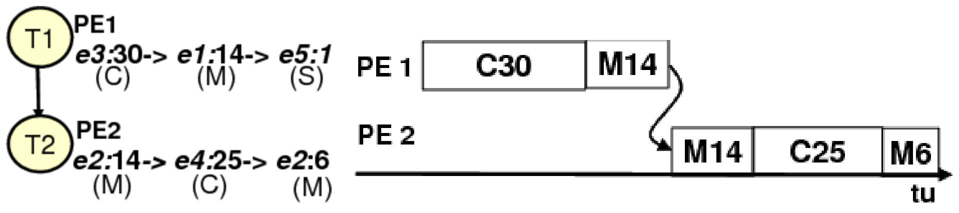
(a) Mapping 2



(b) Execution traces

Figure 5.6: Impact of variation depends task mapping

These cases provide insight into how the performance degradation of a system affected by process variations depends on the properties of the application, the architecture and the mapping. These cases are presented in a simplified manner to effectively illustrate propagation of various effects which otherwise is not possible for more involved examples. System-level performance analysis considering variations should take into account direct and indirect dependencies hence introduced. Next, we show the effect of variations on the critical path.



(a) Taskgraph 4 (b) Execution traces

Figure 5.7: Taskgraph to study impact of variations on critical path

5.3.4 Impact on the critical path

Consider Taskgraph 4, mapped to SYS1 as shown in Figure 5.7(a). Events associated with each task are also shown in the figure and the resulting execution traces for a single packet are shown in Figure 5.7(b). For inputs arriving at every 44 tu, it takes 527 tu to process 10 such requests. The critical path is dominated by PE1 and accounts for 91% of the total critical path timing. However, if PE2 slows down by 20%, total processing time increases to 607 tu.In the critical path timing, 84% of the contribution now comes from PE2. This shows that not only critical path timing increases as expected but also composition of the critical path completely changes. A component having only 9% contribution in critical path contributes to 84% of the total critcal path timing under variations. This illustrates that the critical path for a given system depends on the variation characteristics of the particular instance and in general cannot be determined from the properties of the application, the architecture and the mapping.

5.4 Performance Variability Analysis

In this section, we present our methodology to analyze the system-level performance in the presence of variations while accounting for the application-architecture interaction as mentioned earlier. We first present an overview of the proposed methodology and then briefly discuss the critical phases.

The overall methodology is shown in Figure 5.8. It has three main phases. In the first phase, a representative sample set of SoC instances is generated using the frequency and path latency distribution information for the given process, wherein each instance is associated with a set of *event-delay* characteristics. In the second phase, necessary and sufficient information about the fully synchronous execution of system components is obtained in the form of an symbolic representation called a *Symbolic Execution Graph* (SEG). In the third phase, system-level performance estimation is performed iteratively to obtain performance statistics for each SoC instance obtained in Phase 1. We have modified the system-level performance estimator of [91] to account for variable frequencies of the SoC components and asynchronous communication between components. We now briefly discuss these phases.

5.4.1 Sample set generation

In Section 5.2, we associated a SoC with an event set (Figure 5.1(a),(b)). All operations of the SoC components can be captured by using these events. Note that, events are independent of the specific communication architecture. To take into account varying operating frequencies and path latencies, we augment each event with a delay value. For example, events *e3*, *e4* and *e7* are associated with the time periods of the operating clock frequencies of the corresponding components. Similarly, communication events are characterized by the latencies of their respective SoC-level communication paths[4]. In this manner, each SoC instance is characterized by its own set of event-delay characteristics. Component-level frequency distributions and path latency distributions to generate the sample set can be obtained using statistical models [4, 15] that use variability information of process parameters or using data obtained from process characterization. We leverage on the sample set generation framework that can capture the inter-die and spatially correlated intra-die variations based on system-level floorplan, presented earlier in this thesis.

[4]Note that the latencies depend upon various factors, including interconnect length and driver strength. These values are typically not known at the system-level. However, for the purpose of high-level analysis we obtain interconnect latencies based on the available process data. For a 90nm process, latency of 300ns/mm [92] is used assuming nominal length 10mm for all SoC-level nets.

5.4.2 Symbolic Execution Graph (SEG)

The SEG captures the activity of various system components and their communications over time. It consists of the computation and communication events (data transfer and SYNC events) executed by the SoC components during system simulation. Each event in SEG is characterized by its event ID *e.g.,* events *e1* to *e7* for SYS1, starting time *i.e.,* timestamp (TS) and its weight (Wt). The vertices of the SEG represent these events while the edges between vertices represent the precedence constraints between the vertices. This can be extracted out from the detailed traces obtained from HW/SW co-simulation or synthesized based on the application task graph and input rate. The SEG captures all the temporal dependencies as it is unrolled in time but is independent of the communication architecture.

5.4.3 System-level performance analysis

For system level performance analysis under process variations, we modified the communication architecture performance estimator of [91, 93] to account for variable component frequencies and asynchronous communication between components. The inputs to the original tool are (a) communication architecture of the system, specified by its topology, channel parameters such as bus width, channel access protocols and mapping of the communication events to channels, and (b) system execution traces in form of SEG (Phase 2). The performance estimator generates the transformed SEG that accounts for the effects of communication architecture such as arbitration, split transfers, bus waits and synchronization waits. Implementation and algorithmic details are given in [91].

The modified version of the estimator has a third input that is the event-delay characteristics (Phase 1) of the given SoC instance. While traversing and manipulating the SEG, timestamp of each event is computed. When execution of an event i is over, the timestamps of each of the events j that has event i as their predecessor are updated. In the modified version, timestamp of a computation event is updated as follows:

$$TS_j = TS_i + Wt_j * lTP/dTP \tag{5.1}$$

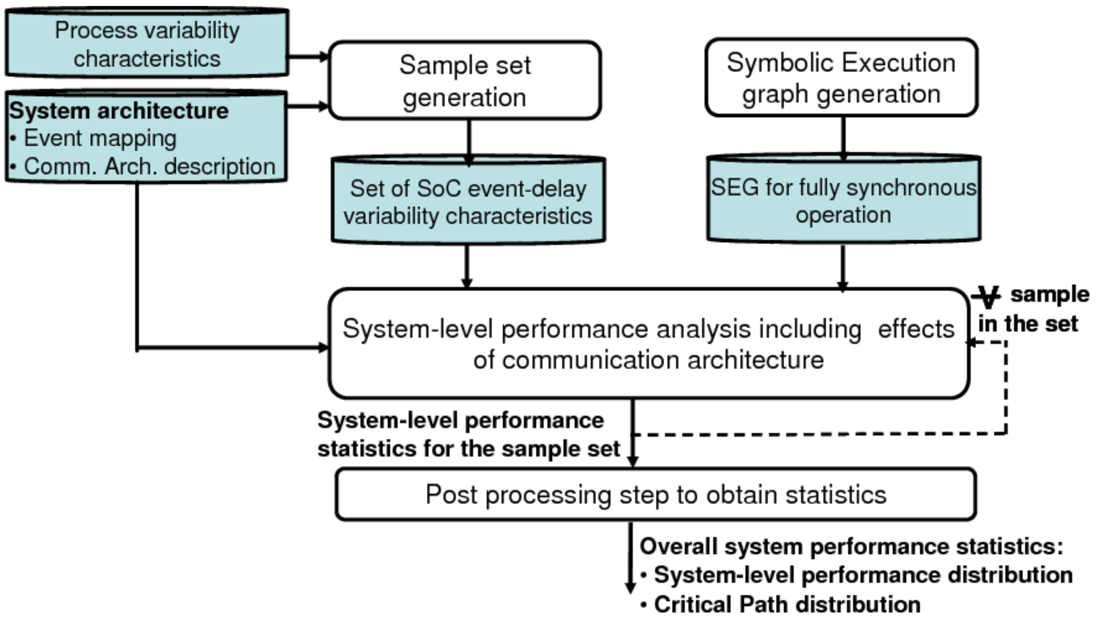

Process variability characteristics
System architecture
• Event mapping
• Comm. Arch. description
Sample set generation
Symbolic Execution graph generation
Set of SoC event-delay variability characteristics
SEG for fully synchronous operation
System-level performance analysis including effects of communication architecture
∀ sample in the set
System-level performance statistics for the sample set
Post processing step to obtain statistics
Overall system performance statistics:
• System-level performance distribution
• Critical Path distribution

Figure 5.8: Overall performance analysis methodology

where, TS_j and Wt_j denotes TS and Wt of event j. lTP represents the time period of components local clock and dTP is the time period corresponding to original design frequency. The ensures that all the calculations are performed in *timeunits* of the original design frequency. The time stamp update of the communication events is is more complicated and is discussed as follows.

Asynchronous communication between modules We now discuss how the communication events between components operating at unrelated frequencies can be handled in our performance analysis methodology. Our analysis is based on the asynchronous multi-point communication architectures proposed in [94, 95]. The multi-point architectures for GALS systems are based on self-timed wrappers for each locally synchronous (LS) component [96]. The wrapper implements clock pausing to prevent metastability and enables asynchronous communication with other LS modules using using four phase bundled data handshake protocol. We show a typical transfer between two LS modules in Figure 5.9. To transfer a new data item, request (Req+) is generated by the sender (Tx). On receiving Req+ the receiver (Rx) stops the local clock, and sends acknowledge (Ack+) to the sender. Data is latched at Ack+. On receiving Ack+, Tx deasserts the request (Req-), receiving which the Rx restarts the local clock and deasserts Ack. Tx can initiate next data transfer on receiving Ack-. Therefore, it takes 4 transfers to finish transfer of one data item. Considering that local control in the wrappers is fairly simple and path lengths much smaller than their corresponding LS component, time for single data transfer $(timeSingle)$ is computed as:

$$timeSingle = \lfloor (4 * PathLatency)/max(sTP, rTP) \rfloor + 1 \tag{5.2}$$

where, sTP and rTP denote the time period of the sender and receiver clocks, respectively. Therefore, for a communication event j TS is updated as follows,

$$TS_j = TS_i + Wt_j * timeSingle * max(sTP, rTP)/dTP \tag{5.3}$$

where TS_i is the TS of the predecessor event.

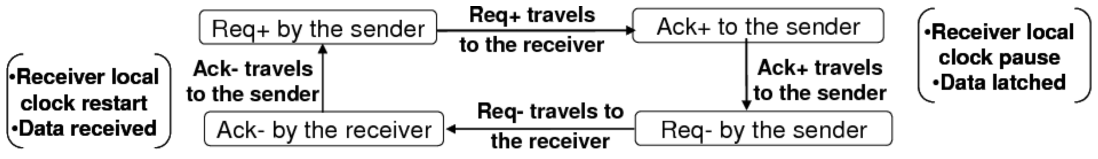

•Receiver local clock restart
•Data received
Req+ by the sender
Req+ travels to the receiver
Ack+ to the sender
•Receiver local clock pause
•Data latched
Ack- travels to the sender
Ack+ travels to the sender
Ack- by the receiver
Req- travels to the receiver
Req- by the sender

Figure 5.9: Data transfer between asynchronous blocks

5.5 Performance enhancement techniques

In this section, we present architecture-level and application level techniques to recover the performance lost due to variations. These techniques exploit the effect of the dependencies between the application task graph, architecture characteristics as well as mapping of tasks to the SoC components.

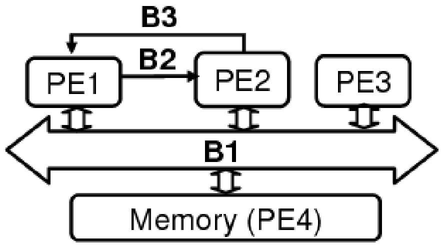
5.5.1 Architectural design techniques

These techniques are based on modifying the communication architecture protocols, namely component priorities on the shared bus and maximum burst size, to mitigate the impact of process variations on the performance.

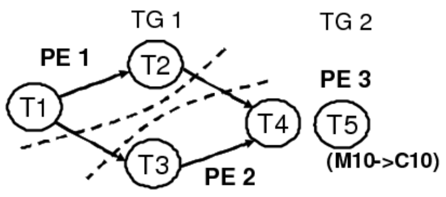
Technique 1: Selecting component priorities To illustrate the effectiveness of modifying priorities on the shared bus we consider system (SYS2) as shown in Figure 5.10(a). Taskgraphs TG1 and TG2, shown in Figure 5.10(b) execute on this system. TG1 maps to PE1 and PE2 and has an input request arriving every 60 tu. The associated events are same as in Figure 5.1(d). TG2 consists of single task T5 that maps to PE3, with a new input every 50 tu. The events associated are communication event of weight 10 followed by a computation event of weight 10 (Figure 5.10(b)).

For SYS2, there are 6 possible configurations for setting relative priorities of PE1, PE2 and PE3 on the shared bus and are enumerated in Figure 5.10(c). We simulate the system in all the configurations for four hypothetical chip instances, (1) not affected by variations, (2) PE1 is 20% slow, (3) PE2 is 20% slow, and (4) PE3 is 20% slow due to variations. Total execution time for each instance under all configurations is shown in Figure 5.10(d). For chip instance 1 (not affected by variations), CONFIG 1 performs the best in terms of total execution time but for chip instance 3 (PE2 slows down) CONFIG 3 performs results in minimum total execution time and has 11% performance gain over CONFIG 1. Therefore, for chip instance 3, changing the configuration from CONFIG 1 to CONFIG 3 enables performance recovery. For chip instances 2 and 4, CONFIG 1 or 3 performs the best.

Technique 2: Maximum burst size Consider two independent tasks running on SYS1 with associated events as shown in Figure 5.11(a). For a system with no variations and maximum burst size of 50 it takes about 203 tu to execute 1 packet. If PE2 slows down by 20%, execution time increases to 246 tu, a degradation of 21%, as shown in first set of traces in Figure 5.11(b). The dotted line represents the critical path. The second set of traces in the figure illustrate the resulting system activity if the first memory transfer on PE2 is allowed a burst of 100 bus words. The total execution time in this case is 213 tu, *i.e.,* performance loss of 5%. Here, 77% of the performance loss is recovered. Note that, increasing the burst size indiscriminately will not work, for instance, increasing the burst size to 75 from 50 will actually lead to further performance degradation. Here we showed that selectively modifying maximum burst size leads to significant performance recovery.



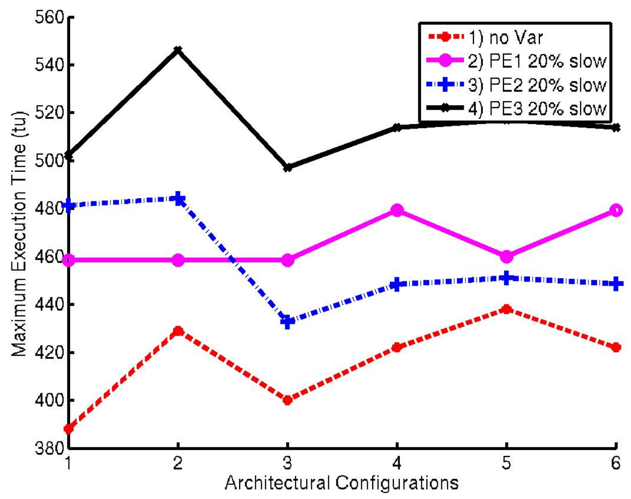
(a) SYS2



(b) Taskgraph 5

CONFIG 1	PE1 > PE2 > PE3
CONFIG 2	PE1 > PE3 > PE2
CONFIG 3	PE2 > PE1 > PE3
CONFIG 4	PE2 > PE3 > PE1
CONFIG 5	PE3 > PE1 > PE2
CONFIG 6	PE3 > PE2 > PE1

(c) CONFIG table

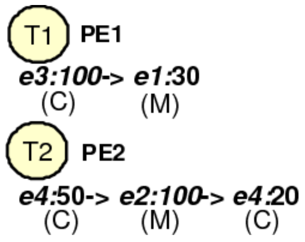


(d) Execution time for each instance under different CONFIGs

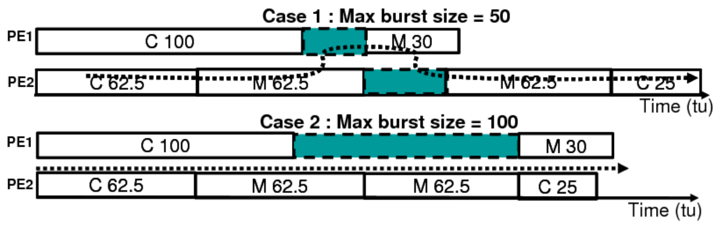
Figure 5.10: Selecting component priorities to enable performance recovery

5.5.2 Task Mapping

To show the effectiveness of modifying task mapping in presence of variations, consider the taskgraph shown in Figure 5.12(a). It can be mapped to SYS1 in 8 possible ways, shown in Table 5.1. Table in Figure 5.12(b) shows the amount of time taken by each task on each PE of SYS1. We simulate the system for 6 packets arriving in an interval of 700 tu, for three chip instances (1) not affected by variations, (2) PE1 is 20% slow, and (3) PE2 is 20% slow due to variations. Total execution time under different task mappings is shown in Figure 5.12(c). It can be seen that for chips with different variation characteristics different mappings should be selected to achieve the optimum performance. Mapping 3 performs the best for instances (1) and (3). But for



(a) Taskgraph 6



(b) Execution traces

Figure 5.11: Selecting maximum burst size to enable performance recovery

instance (2) (PE1 slow) selecting Mapping 2 over Mapping 3 leads to 76% recovery in performance lost due to variations.

Table 5.1: Possible task mappings of Taskgraph 7

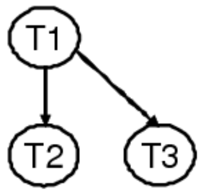
Mappings / PEs	1	2	3	4	5	6	7	8
PE1	T1,T2,T3	T1	T1,T3	T1,T2	T2,T3	T2	T3	-
PE2	-	T2,T3	T2	T3	T1	T1,T3	T1,T2	T1,T2,T3

5.6 Experimental results

In this section, we first describe our experimental methodology and then present results that illustrate the applicability of the proposed analysis and design techniques.

5.6.1 Methodology

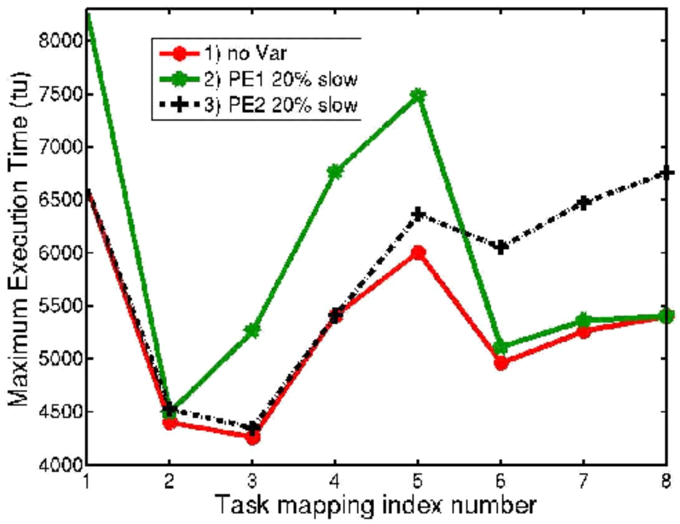
For our experiments, we considered shared bus SoCs, SYS1 and SYS2 presented earlier in this chapter. The applications are modeled using taskgraphs as explained in Section 5.2. For each case, the initial *SEG* that is independent of the effect of the communication architecture and variations, is generated using a trace generator implemented in MATLAB [56]. The performance distributions are obtained using the proposed variation-aware analysis methodology described in Section 5.4. For the sample set generation, we assume random inter-die and intra-die variations in operating frequency of SoC components and global path latencies with $3\sigma = 30\% \mu$.



(a) Taskgraph 7

Tasks \ PEs	PE1	PE2
T1	200	400
T2	500	250
T3	300	150

(b) Time taken by tasks on each PE



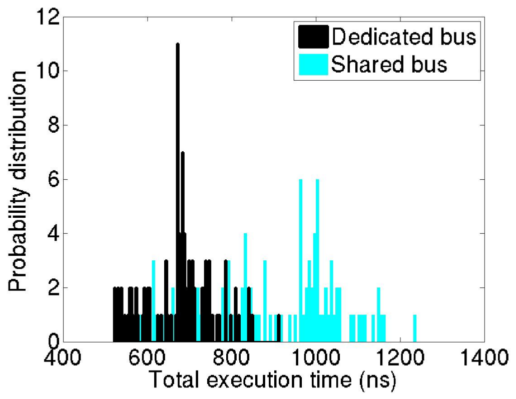
(c) Total execution time for different chips under different mappings

Figure 5.12: Task mapping selection to enable performance recovery

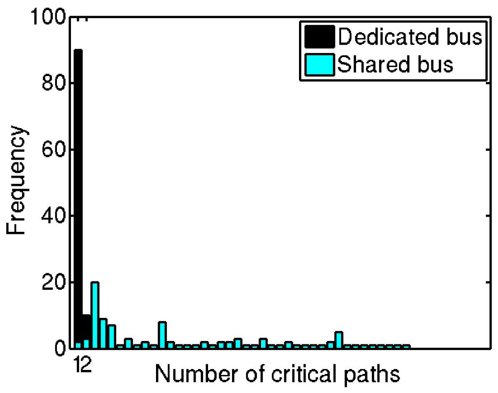
We use nominal frequency of $1\ Gz$ for $90\ nm$ process. We assume that operating frequency of components F is given by the minimum of their respective FMAX and the nominal design time frequency as it may not be possible to run a component at a higher frequency than that designed due to power constraints. For latencies, nominal path length of $10\ mm$ and nominal latency of $175\ ns\ /mm$ [92] for $90\ nm$ node is used.

5.6.2 Performance variability

First, we use the proposed analysis methodology to compare performance under different communication architectures. We simulate the performance of 10 inputs of Taskgraph 4 (Figure 5.7)), for 2 architectures, (1) shared bus architecture (SYS 1) and (2) dedicated bus architecture with a dual port memory. The performance distributions under variations for these 2 cases are shown in Figure 5.13(a) while the corresponding critical path distributions are shown in Figure 5.13(b). The performance distribution of the dedicated bus architecture achieves μ/σ that is 30% better compared to that of the shared bus architecture. It is not only more robust to variations as expected but also exhibits better critical path distribution with only 2 critical paths as opposed to 40 for the shared bus architecture. These results demonstrate the ability of the proposed variation aware performance analysis technique to evaluate and compare performance variability and critical path variability of different SoC architectures.



(a) Performance distribution



(b) Critical Path distribution

Figure 5.13: Evaluating architectures

5.6.3 Effectiveness of the proposed design techniques

In this section, we evaluate the resulting performance distributions when proposed design techniques are statically applied.

Architectural techniques We obtain the cumulative distributions for SYS2 with Taskgraph 5, (Figure 5.10) for all the 6 possible architectural configurations. Our results indicate that no configuration significantly outperforms the other and total execution time distributions are quite close to each other. However, when we look at average time between generating consecutive processed packets there is a clear distinction in the CDFs (Figure 5.14). This average time characterizes the request processing rate and sometimes is a more important metric than total execution time, especially for stream processing applications. The figure shows that CONFIG 2 has the best overall performance distribution and results in parametric yield of 98% for a constraint of 85 tu (shown as vertical line in the figure) on the average time between output packets. This is 78% better than the yield obtained under CONFIG 4 that is the best

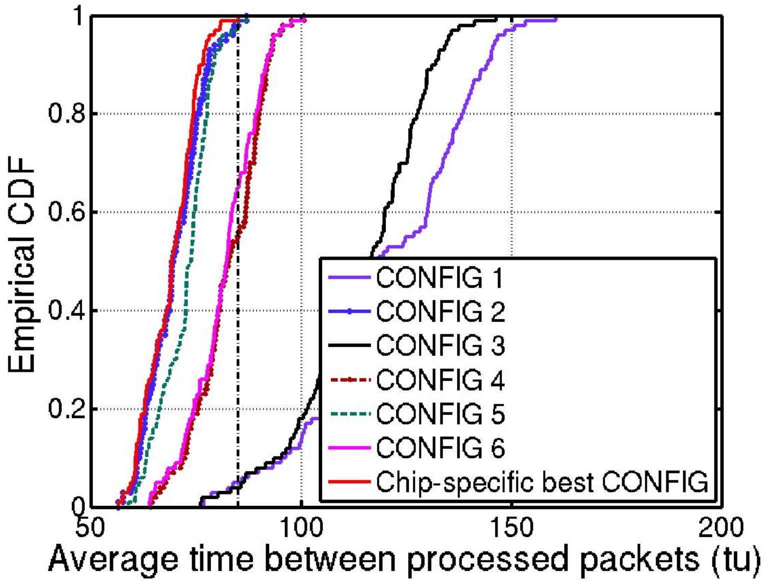


Figure 5.14: CDFs under architecture configurations with different relative priorities

CONFIG for no variation case (See Table 5.2). For this workload, chip-specific configuration results in output rate distribution (leftmost CDF in the figure) with marginal improvement over CONFIG 2.

CONFIG	1	2	3	4	5	6
Average time between O/P packets (timeunits)	69.71	62.81	67.61	60.80	62.40	60.80

Table 5.2: Average time between generating processed packets for different configurations of SYS2 under no variations case

Similarly, significant performance gains are achieved by applying the technique of modifying the maximum burst sizes. For processing 5 requests arriving at every 150 tu of T1 and 200 tu of T2 of Taskgraph 6 (refer to Figure 5.11), we obtained that burst size of 100 leads to the best CDF under variations. It achieves 93% parametric yield for a constraint of 3250tu on the total execution time. For burst size of 50 that is the best for no variations case, the corresponding yield under variations is only 81%.

Task mapping We show the cumulative distributions of total execution time for Task-

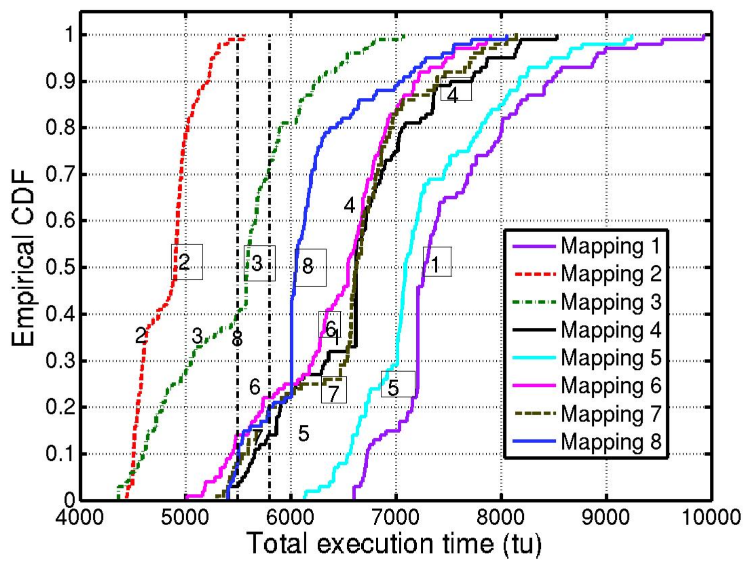


Figure 5.15: CDFs under different task mappings

graph 7 (Figure 5.12) for each of the 8 mappings in Figure 5.15. The number to the right of the curve shows the mapping number. It can be seen from the figure that Mapping 2 is most robust to variations. In terms of 99^{th} percentile Mapping 2 performs 20% better than Mapping 3 that is the best mapping for the no variations case (refer to Figure 5.12(c)). For an execution time constraint of 5500 tu, Mapping 2 achieves 99% parametric yield which is 147% better than 40% yield achieved by Mapping 3. For a constraint of 5800 tu, the improvement is 39%.

5.7 Conclusion

In summary, we illustrated that the degree of impact of process variations on system level performance depends on the application, architecture and mapping characteristics. We presented an effective analysis framework that is used to develop a simulation based methodology to capture these effects. Furthermore, we proposed application level and architecture level design techniques to recover the performance loss due to variations and illustrated that these techniques can lead to significant performance gains. We believe that development of efficient algorithms to exploit these techniques will be crucial to recover any performance loss to variability.

Chapter 6

Future Work

In this final chapter, we conclude by presenting our vision of the *variation-tolerant* systems of the future and the implications of this research on such systems. It is emphasized in this thesis that variability is a growing design challenge and its impact intensifies with each technology generation. Use of new materials or new transistor structures are likely to replace some of the current sources of variability with some other sources. The net effect is variability in power and performance characteristics of the overall system, as is shown in this thesis. We believe that in future systems, *variation-awareness* needs to be built in at all stages of the design cycle and especially, at higher levels of design abstraction. This is because (a) changes made in the early design stages can have a larger impact on the variation-tolerance of the overall system, and (b) high-level power-performance characteristics directly impacts the end-user experience.

Towards that goal, variation-aware SoC architecture design and variation-aware application design emerge as two natural areas that can benefit from the techniques presented in this thesis. We believe that the variation-aware analysis and design techniques presented in Chapter 5, integrated with power analysis and optimization techniques presented in Chapters 2, 3 and 4, will go a long way to design Systems-on-chips that are robust to the impact of variability. On the other hand, we believe that variation-aware application design will be inevitable in future systems as it is unlikely that the impact of variations can be completely contained by architectural approaches.

In the rest of this chapter, we elaborate on the concept of developing applications in a variation-aware manner and present preliminary results of our findings.

6.1 Variation-aware application design

In the face of increasing variations, techniques at the architectural and system levels will reduce but cannot completely eliminate the effect of variations. The net residual effect of variations, in the form of unexpected behavior in the execution time and energy consumption of various parts of the SoC, will have to be seen and absorbed by the applications that execute on the SoC.

Let us consider real-time video encoding application as an example. A video encoding application encodes a raw video stream of a given *resolution* at a specified *frame rate* (FR) to produce an encoded bitstream of specified *bitrate* (say, BR_0) with a *video quality* (say, VQ_0). When the underlying hardware is impacted by process variations, the time to encode a video frame (T) may increase and the encoder may not be able to sustain the specified frame rate, leading to inferior or unacceptable quality of the resulting encoded stream. In the case of encoding of real-time video streams, such as Live TV/events, inability of the encoder to process frames at the specified rate will lead to the overflow of the encoder buffer, causing frames to be dropped, and thereby, adversely impacting the smoothness of the encoded video stream.

To illustrate the impact of variations on video quality, we consider a specific example. Depending upon the extent of spatial variations in a given chip instance and the video encoder architecture, time to encode an I-frame (T_I) and time to encode a P-frame (T_P) may be impacted differently. Consider a case in which T_I increases by 5 % and T_P increases by 7 % due to variations. We estimated that in this case, video quality will be degraded to 28 dB due to random frame dropping. On the other hand, in a *variation-aware video encoder*, parameters such as *group of pictures (GOP)* [1] and *quantization level* can be adapted to reduce the computational requirements at the expense of the compression efficiency. Such an adaptation can enable reduction in

[1] An encoded video stream typically consists of a repeating sequence of an I-frame followed by a certain number of P-frames. GOP represents the size of this sequence.

the number of frames dropped while maximizing the video quality. For T_I variation of 5 % and T_P variation of 7 %, we estimated that a *variation-aware video encoder* can result in a final video quality of 31 dB. In Figure 6.1, we show the estimated video quality obtained with the *variation-aware video encoder* and the *original video encoder* assuming that variations can cause up to 15 % deviation in T_I and T_P. Figure 6.1(a) shows the PSNR under different degrees of variation in T_I and T_P. Let us consider a scenario in which the user specifies a limit of 29 dB for acceptable video quality. Figure 6.1(b) shows the combinations of variation in T_I and T_P for which the *variation-aware video encoder* and the *original video encoder* are able to meet the user requirement (represented as regions that are shaded green). The results clearly demonstrate the potential of the variation-aware encoding technique.

We believe that a large class of applications such as audio/image/video encoding/decoding/transcoding, graphics processing, channel coding, and data compression can benefit from variation-aware techniques similar to the one discussed above. This is because of the following two properties shared by this class of applications. In these applications (1) there exists a correlation between the amount of computation needed, the bitrate (e.g., bitrate of the encoded video stream, bitrate of channel encoding), size of the result (e.g., compressed data), and the quality of the result (e.g., visual quality of the encoded video, error resilience of channel coded data), and (2) there is an inherent tolerance to limited variation in both the desired bitrate/size of the result, as well as the quality of the result. For example, in the case of audio/image/video coding and graphics rendering, the resulting quality is often subjective, with some audio/visual artifacts less perceivable than others. In the case of data compression, there is some tolerance in the size of the result. These properties can be exploited to design applications such that, if variations leads to performance degradation in the underlying hardware, the application can adapt to execute without causing any perceivable degradation in user experience.

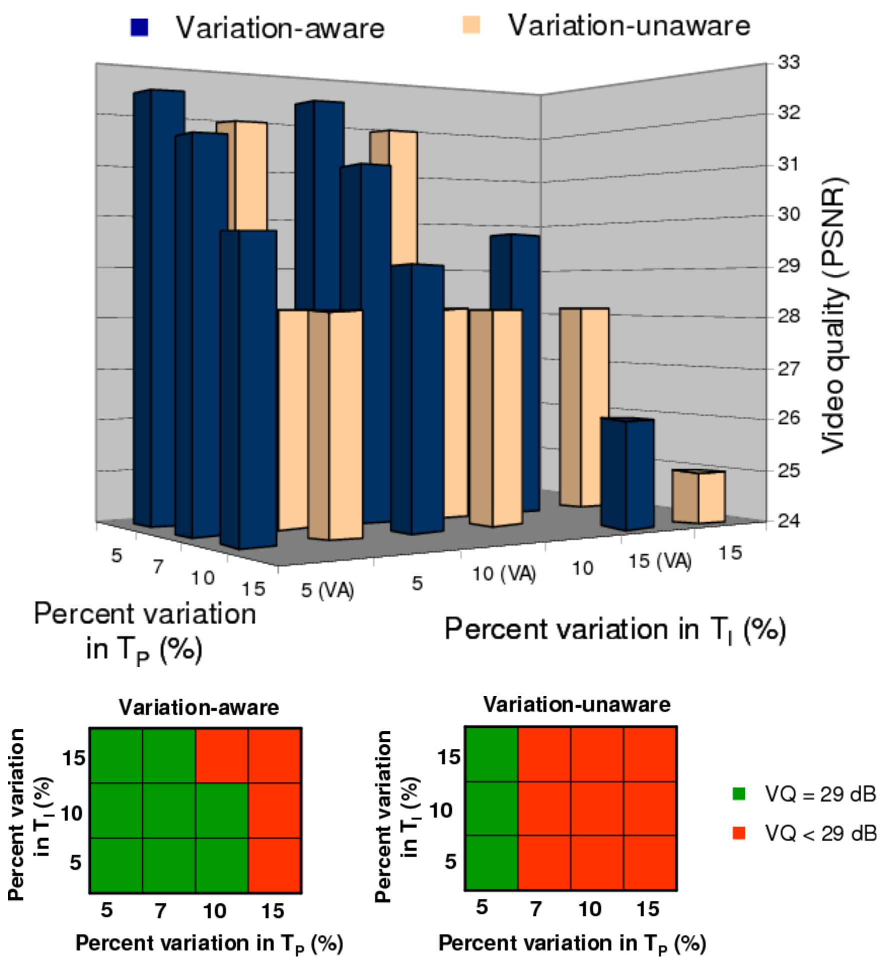


Figure 6.1: Video quality for variation aware and unaware video encoding

6.2 Summary

In this thesis, we illustrated that variation-aware system-level design can lead to significant improvements in the power and performance characteristics of SoCs manufactured in sub-90 nm technologies. We presented (i) techniques for system-level power and performance analysis under variations, (ii) variability-aware techniques to determine optimal policy parameters for shutdown-based power management policies, (iii) a methodology to determine variation-aware voltage levels, and (iv) architecture and application level techniques to recover performance loss due to variations. The techniques presented in this thesis focus on addressing variability without disrupting the existing design flow and hence, can be easily incorporated in current technologies. We believe that in the future, variation-aware applications in conjunction with variation-aware system architectures, will play a crucial role in alleviating the impact of variability as experienced by the end user. The development of integrated application and architectural techniques to address variations poses many new challenges, making it a *rich* area for research and development. We believe that the analysis and design techniques presented in this thesis are an important step towards addressing many of these challenges.

Appendix A

A.0.1 Modeling Leakage Power

Here we describe the models that we have developed to determine leakage currents of unit width N-type and P-type transistors for a given channel length. Many empirical models of the form $I_{leakage} = \alpha e^{f(L,T_{ox})}$ have been proposed [10, 1, 11, 13] that analyze the leakage current of a single gate. Here, α depends on the gate characteristics and is directly proportional to the device width. f is a polynomial function of device parameters and hence, is a random variable under manufacturing variations. Assuming that device parameters are normally distributed, the leakage current of a gate is log-normally distributed and total circuit leakage distribution is given by the sum of correlated log-normal distributions of its constituent gates, $I_{ckt} = \sum_{gate_i} I_i = \sum_{gate_i} \alpha_i e^{Y_i}$. Researchers have proposed various methods to accurately and efficiently calculate this total leakage distribution.

However, in our methodology presented in Sections 2.5 and 2.6 of the thesis, the total leakage of a component is calculated as a function of its power-state and its thermal characteristics. Power-state determines the effective leaking width whereas N-type and P-type leakages are effected differently by the thermal characteristics (that depends on the system-level floorplan). We calculate the total leakage of a component in a given power-state under a simplifying assumption (similar to the one in [53]) that the component can be modeled as a sea of identical gates, since leakage characteristics of all gates is not readily available during system-level design. Therefore, our methodology only requires that a model to accurately and efficiently calculate leakage currents of unit width N-type and P-type transistors for a given channel length parameter be developed. Our experiments indicate that for effective channel length following a Gaussian distribution at 90 nm technology node, transistor leakage can be accurately modeled using models of the form $I_{leakage} = \alpha * C_1 * L^{C_2}$. Here, C_1 and C_2 are model parameters. We have used data obtained from HSPICE simulation of the Berkeley Predictive Technology $90\ nm$ BSIM3 model card [55] to calibrate these models.

We compare our model to the leakage values obtained experimentally in Figure A.1. The figure shows that our model accurately tracks N-type and P-type leakage currents. We obtain mean relative error of 3% for leakage of a N-type device and 15% for leakage of a P-type device.

Note that, statistical leakage power modeling is not the main focus of this work. Therefore, we have developed our models under simple assumptions as described above. If detailed circuit-level information (such as precise estimates of active device widths under different power states) are available, they can easily be integrated within our methodology. Also, if the detail netlist is available, effects such as the stacking effect can be incorporated.

A.0.2 Temperature-aware Leakage Power Model: Derivation and Validation

We derive our model based on the BSIM3 model equations [97]. The subthreshold current is expressed as:

$$I(T) = \mu(T)\, C_{ox}\, \frac{W}{L}\, e^{1.8}\, (V_T)^2\, e^{\frac{(V_{gs}-V_{th}-Voff)}{nV_T}} \tag{A.1}$$

where $\mu(T)$ is the carrier mobility, V_T is the thermal voltage and V_{th} is the threshold voltage of the device. The temperature dependency equations are given by:

$$\mu(T) = \mu(T_0)(\frac{T}{T_0})^{\mu_{te}} \tag{A.2a}$$

$$V_T = \frac{K_B\, T}{q} \tag{A.2b}$$

$$V_{th}(T) = V_{th}(T_0) + (K_{T1} + K_{T2}V_{bseff} + K_{T11}/L)(\frac{T}{T_0} - 1) \tag{A.2c}$$

Here, μ_{te} is the mobility temperature coefficient and K_{T1}, K_{T2}, K_{T11}, represent the temperature coefficient for threshold voltage, the body-bias coefficient of V_{th} temperature effect and the channel length dependence of the temperature coefficient for threshold voltage, respectively. Substituting parameters from Equation A.2 in Equation A.1, we get,

$$I(T) = (Constant) * (\frac{T}{T_0})^{\mu_{te}} * T^2 * e^{-\frac{A(L)}{T} + \frac{B(L)}{T_0}} \tag{A.3}$$

where,

$$A(L) = \frac{V_{th}(T_0) - (K_{T1} + K_{T2}V_{bseff} + K_{T11}/L)}{K_B/q} \tag{A.4a}$$

$$B(L) = \frac{(K_{T1} + K_{T2}V_{bseff} + K_{T11}/L)}{K_B/q} \tag{A.4b}$$

Dividing Equation by leakage current at nominal temperature, we get,

$$\frac{I(T)}{I_0} = (\frac{T}{T_0})^{\mu_{te}+2} * e^{\frac{A(L)*(T-T_0)}{T*T_0}} \tag{A.5}$$

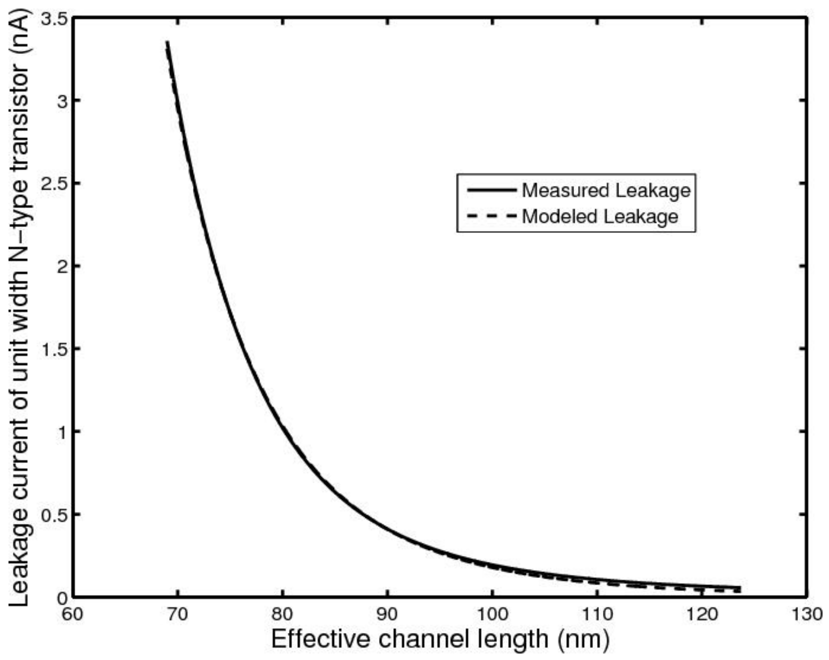
Note that, $A(L)$ is independent of temperature variations and depends only on the process parameters. We model A as a function of I_0, rather than L, since I_0 of a component is easily determined from the *power-state based leakage power models*. We model A as a second order polynomial in $ln(I_0)$ and express it as:

$$A(I_0) = C1 * (ln(I_0)^2) + C2 * ln(I_0) + C3 \tag{A.6}$$

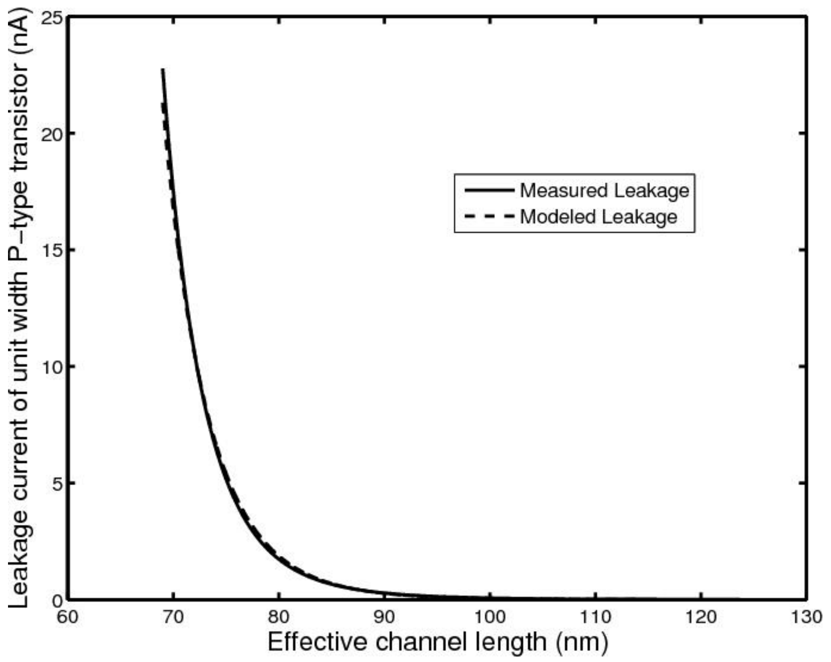
where, $C1$, $C2$, $C3$ are the fitting coefficients. These are calculated separately for the N and P type devices and depend only on the technology node. We are able to achieve less than 0.01% average error for N-type devices and about 1.29% average error for P-type devices using this model.

Model Calibration and Verification To calibrate our leakage current model we use data collected from HSPICE simulation of the Berkeley Predictive Technology $90\ nm$ BSIM3 model card [55]. The relative error curves $(I_{predicted} - I_{experimental}/I_{experimental})$

for various channel lengths are plotted against temperature in Figure A.2. The channel lengths range from 60 nm to 120nm. The error is higher at very high temperatures but still within 5% for the NMOS and 7% for the PMOS device. For typical IC operating temperatures in the range of $300\ K$ to $375\ K$, we observe that our model is very accurate.

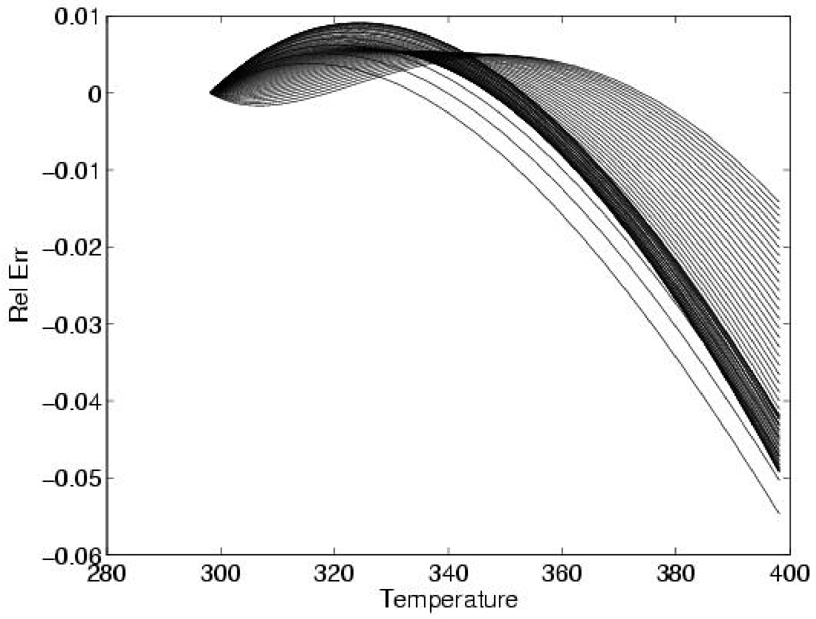


(a) N-Type

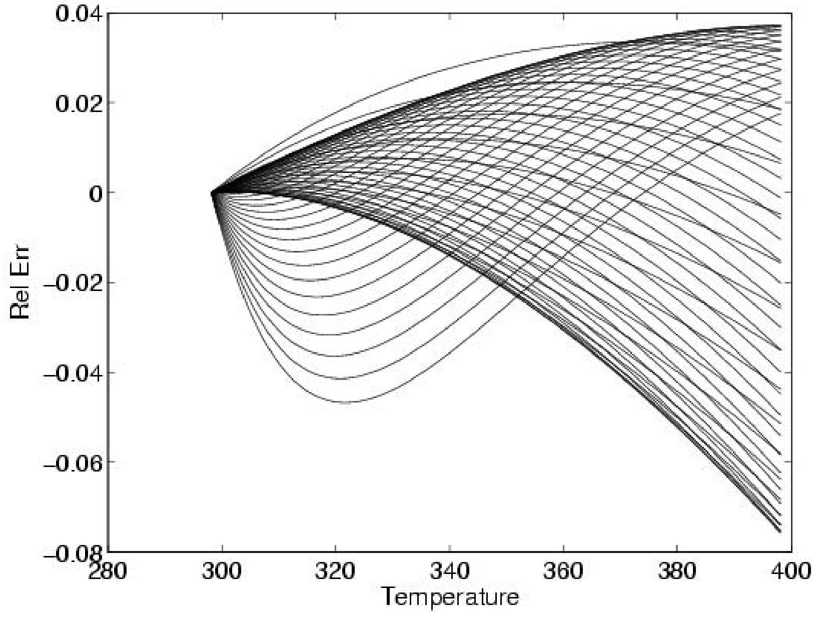


(b) P-Type

Figure A.1: Modeling N-type and P-type leakage currents



(a) N-Type device



(b) P-Type device

Figure A.2: Accuracy of temperature based leakage model for N-type and P-type devices

Appendix B

B.0.3 Voltage and leakage models

In this section, we present models that are used to estimate the required voltage and leakage power of each chip instance. The voltage model is used to compute the minimum required voltage of a chip instance, given the required desired frequency of operation. The leakage model is used to compute the leakage of a given chip instance at the selected voltage of operation. These models are used along with Monte Carlo sampling to design the variation-aware set of voltages. Various approaches in which these models can be calibrated are presented in Chapter 4 of the thesis.

Voltage-delay characteristics

To take into account process variations, we compute the required voltage for the required delay of a given chip instance using the delay information of the instance at a known voltage. Assuming that the required critical delay is td^i_{req}, then, V^i_{req} is given modeled using:

$$V^i_{req}(x) = \frac{vC^i_1 * x + vC^i_2}{x + vC^i_3} \tag{B.1}$$

Here, $x = log(td^i_{req}/D^i_{V_N})$, where $D^i_{V_N}$ is the critical delay of the i^{th} instance at maximum operating voltage.vC_1, vC_2 and vC_3 are chip specific coefficients that are modeled as a function of maximum delay of the chips using the following relations,

$$\begin{aligned} vC_1^i &= K_{11} * (D_{V_N}^i)_{12}^K + K_{13} \\ vC_2^i &= K_{21} * (D_{V_N}^i)^2 + K_{22} * (D_{V_N}^i) + K_{23} \\ vC_3^i &= K_{31} * (D_{V_N}^i)^2 + K_{32} * (D_{V_N}^i) + K_{33} \end{aligned} \tag{B.2}$$

Here, K11, K12, K13, K21, K22, K23, K31, K32 and K33 are constants and depend on process technology and the circuit only. We calibrated our models using HSPICE simulation. Figure 4.6 compares the voltage obtained from our models and experimental data from HSPICE. We observe that the above developed model closely follows the experimental data. We compute that the mean relative error is less than 1%.

Leakage-voltage characteristics

We have developed models to compute the leakage current of a chip instance i at a given voltage v given the leakage current of the instance at a known voltage V_N. In the following equations, we present models to compute transistor level N-type and P-type leakage currents.

$$\begin{aligned} In^i(v) &= In_{V_N}^i * e^{(nCoef_1^i*(v-1)+nCoef_2^i)} \\ Ip^i(v) &= Ip_{V_N}^i * e^{(pCoef_1^i*(v-1)+pCoef_2^i)} \end{aligned}$$

Here, $In_{V_N}^i$ and $Ip_{V_N}^i$, represent the leakage current values at V_N. $nCoef_1$, $nCoef_2$, $pCoef_1$ and $pCoef_2$ are the model constants and are different for each chip instance. These coefficients are computed using the following equations,

$$\begin{aligned} nCoef_1^i &= KIn_{11} * \left(\frac{In_{V_N}^i}{In_{nom}}\right)^{KIn_{12}} + KIn_{13} \\ nCoef_2^i &= KIn_{21} * \left(\frac{In_{V_N}^i}{In_{nom}}\right)^{KIn_{22}} + KIn_{23} \end{aligned}$$

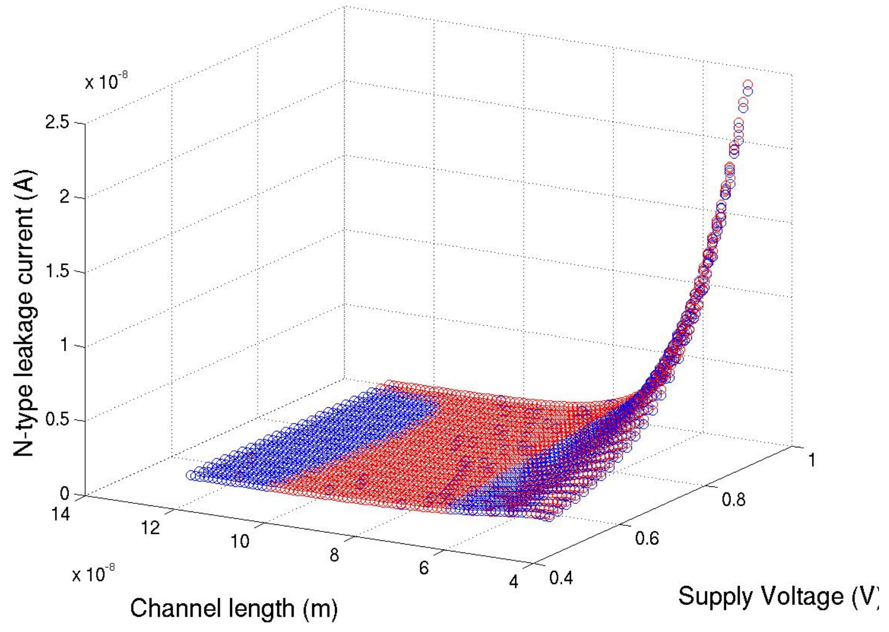
We have similar equations to model the P-type current. The constants KIns and KIps depend on the process technology only.

To obtain circuit level leakage power we use the concepts similar to the ones presented in [53] and model the circuit as a sea of inverters. Then, the total leakage current is computed as

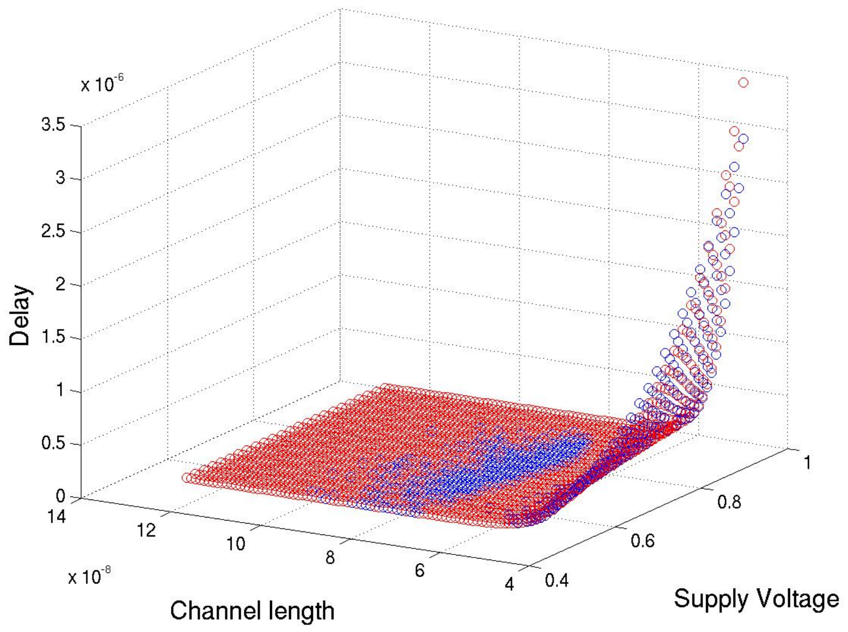
$$I^i_{leak}(V) = KN * In^i(V) + KP * Ip^i(V) \tag{B.3}$$

Here KN and KP are the active leakage widths of the circuit.

We compare the N-type leakage current obtained from our model to the ones obtained using SPICE simulations in Figure B.1. The figure shows that our models closely match the experimental data. The relative mean error is 2% for N-type leakage current and less then 3% for P-type leakage current.



(a) N-type



(b) P-type

Figure B.1: In and Ip leakage currents as a function of voltage for different chip instances

Bibliography

[1] R. Rao, A. Srivastava, D. Blaauw, and D. Sylvester, “Statistical Estimation of Leakage Current Considering Inter- and Intra-die Process Variation,” in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 84–89, Aug. 2003.

[2] “International Technology Roadmap for Semiconductors, 2007 Edition.” http://www.itrs.net/Links/2007ITRS/Home2007.htm.

[3] B. P. Wong, A. Mittal, Y. Cao, and G. W. Starr, *Nano-CMOS Circuit and Physical Design*. WILEY-INTERSCIENCE, JOHN WILEY & SONS, 2005.

[4] K. Bowman, S. Duvall, and J. Meindl, “Impact of Die-to-Die and Within-Die Parameter Fluctuations on the Maximum Clock Frequency Distribution for Gigascale Integration,” in *IEEE J. Solid-State Circuits*, pp. 183–190, Feb. 2002.

[5] L. W. Liebmann, “Layout Impact of Resolution Enhancement Techniques: Impediment or Opportunity?,” in *Proc. ISPD*, pp. 110–117, Apr. 2003.

[6] J. W. Tschanz et. al., “Adaptive Body Bias for Reducing Impacts of Die-to-Die and Within-Die Parameter Variations on Microprocessor Frequency and Leakage,” *IEEE JSSC*, vol. 37, pp. 1396–1402, Nov. 2002.

[7] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, “Parameter Variations and Impact on Circuits and Microarchitecture,” in *Proc. Design Automation Conf.*, pp. 338–342, June 2003.

[8] K. Bernstein et. al., “High-performance CMOS Variability in the 65-nm Regime and Beyond,” *IBM J RES & DEV*, vol. 50, no. 4/5, pp. 433–449, 2006.

[9] J. Tschanz, K. Bowman, and V. De, “Variation-tolerant Circuits: Circuit Solutions and Techniques,” in *Proc. DAC*, June 2005.

[10] S. Narendra, V. De, S. Borkar, D. Antoniadis, and A. Chandrakasan, “Full-chip Sub-threshold Leakage Power Prediction Model for Sub-0.18 um CMOS,” in *Proc. ISLPED*, pp. 19–23, Aug. 2002.

[11] S. Mukhopadhyay and K. Roy, “Modeling and Estimation of Total Leakage Current in Nano-scaled CMOS Devices Considering the Effect of Parameter Variation,” in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 172–175, Aug. 2003.

[12] R. R. Rao, A. Devgan, D. Blaauw, and D. Sylvester, “Parametric Yield Estimation Considering Leakage Variability,” in *Proc. Design Automation Conf.*, pp. 442–447, June 2004.

[13] H. Chang and S. Sapatnekar, “Full Chip Analysis of Leakage Power Under Process Variations, Including Spatial Correlations,” in *Proc. Design Automation Conf.*, pp. 523–528, June 2005.

[14] C. Cho, D. D. Kim, J. Kim, J. O. Plouchart, D. Lim, S. Cho, and R. Trzcinski, “Decomposition and Analysis of Process Variability Using Constrained Principal Component Analysis ,” *IEEE Trans. Semiconductor Manufacturing*, vol. 21, pp. 55–62, Feb. 2008.

[15] V. Mehrotra et. al., “A methodology for Modeling the Effects of Systematic Within-die Interconnect and Device Variation on Circuit Performance,” in *Proc. DAC*, pp. 172–175, 2000.

[16] J. A. G. Jess, K. Kalafala, S. R. Naidu, R. H. J. M. Otten, and C. Visweswariah, “Statistical Timing for Parametric Yield Prediction of Digital Integrated Circuits,” in *Proc. Design Automation Conf.*, pp. 932–937, 2003.

[17] D. Marculescu and E. Talpes, “Energy Awareness and Uncertainty in Microarchitecture-Level Design,” *IEEE Micro*, vol. 25, pp. 64–76, Sept. 2005.

[18] N. Azizi, M. M. Khellah, V. De, and F. N. Najm, “Variations-aware Low-power Design with Voltage Scaling,” in *Proc. Design Automation Conf.*, pp. 529–534, June 2005.

[19] K. Bowman, A. Alameldeen, S. Srinivasan, and C. Wilkerson, “Impact of Die-to-Die and Within-Die Parameter Variations on the Throughput Distribution of Multi-core Processors,” in *Proc. ISLPED*, pp. 50–55, Aug. 2007.

[20] L. D. Huang and M. D. F. Wong, “Optical Proximity Correction (OPC): Friendly Maze Routing,” in *Proc. DAC*, pp. 186–191, 2004.

[21] J. W. Tschanz et. al., “Effectiveness of Adaptive Supply Voltage and Body bias for Reducing Impact of Parameter Variations in Low Power and High Performance Microprocessors,” *IEEE JSSC*, vol. 38, pp. 826–829, May 2003.

[22] A. Srivastava, D. Sylvester, and D. Blaauw, “Statistical Optimization of Leakage Power Considering Process Variations Using Dual-Vth and Sizing,” in *Proc. Design Automation Conf.*, pp. 773–778, June 2004.

[23] A. Davoodi and A. Srivastava, "Probabilistic Dual-Vth Leakage Optimization Under Variability," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 143–148, Aug. 2005.

[24] S. Bhardwaj and S. B. K. Vrudhula, "Leakage Minimization of Nano-scale Circuits in the Presence of Systematic and Random Variations," in *Proc. Design Automation Conf.*, pp. 541–546, June 2005.

[25] V. Wason and K. Banerjee, "A Probabilistic Framework for Power-Optimal Repeater Insertion in Global Interconnects Under Parameter Variations," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 131–136, Aug. 2005.

[26] P. Mahoney and E. Fetzer and B. Doyle and S. Naffziger, "Clock Distribution on a Dual-Core, Multi-Threaded Itanium-Family Processor," in *Proc. Int. Solid-State Circuits Conf.*, pp. 192–193, Feb. 2005.

[27] D. Ernst et. al., "Razor: Circuit-Level Correction of Timing Errors for Low-Power Operation," in *IEEE Micro*, pp. 7–18, 2004.

[28] C. H. Kim and K. Roy and S. Hsu, and R. Krishnamurthy and S. Borkar, " A Process Variation Compensating Technique With an On-die Leakage Current Sensor for Nanometer Scale Dynamic Circuits," *IEEE Trans. VLSI Systems*, vol. 14, no. 6, pp. 646–649, 2006.

[29] P. Gupta, A. B. Kahng, Y. Kim, and D. Sylvester, "Self-compensating design for focus variation," in *Proc. DAC*, pp. 365–368, 2005.

[30] J. Donald and M. Martonosi, "Power Efficiency for Variation-Tolerant Multicore Processors," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 304–309, Oct. 2006.

[31] H. Wang, M. Miranda, A. Papanikolaou, F. Catthoor, and W. Dehaene, "Variable Tapered Pareto Buffer Design and Implementation Allowing Run-time Configuration for Low-power Embedded SRAMs," *IEEE Trans. VLSI Systems*, vol. 13, pp. 1127–1135, Oct. 2005.

[32] D. Marculescu and S. Garg, "System-Level Process-Driven Variability Analysis for Single and Multiple Voltage-Frequency Island Systems," in *Proc. Int. Conf. Computer-Aided Design*, pp. 541–546, Nov. 2006.

[33] D. Marculescu and S. Garg, "System-Level Process Variation Driven Throughput Analysis for Single and Multiple Voltage-Frequency Island Designs," in *Proc. Design Automation & Test Europe (DATE) Conf.*, Apr. 2007.

[34] S. Garg and D. Marculescu, "System-Level Mitigation of WID Leakage Power Variability Using Body-Bias Islands," in *Proc. Int. Symp. on HW/SW Codesign*, Nov. 2008.

[35] S. Kulkarni, D. Sylvester, and D. Blaauw, "A Statistical Framework for Post-Silicon Tuning Through Body Bias Clustering," in *Proc. Int. Conf. Computer-Aided Design*, 2006.

[36] A. Tiwari, S. Sarangi, and J. Torrellas, "ReCycle: Pipeline adaptation to tolerate process variation," in *Proc. Int. Symp. Computer Architecture*, pp. 358–368, June 2007.

[37] A. Devgan and S. Nassif, "Power Variability and Its Impact on Design," in *Proc. Int. Conf. VLSI Design*, pp. 679–682, Jan. 2005.

[38] S. Narendra, D. Blaauw, A. Devgan, and F. Najm, "Tutorial 2: Leakage Issues In IC Design: Trends, Estimation, and Avoidance," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2003.

[39] T. Simunic, L. Benini, and G. D. Micheli, "Cycle-Accurate Simulation of Energy Consumption in Embedded Systems," in *Proc. DAC*, June 1999.

[40] M. Lajolo et. al., "Efficient Power Estimation Techniques for HW/SW Systems," in *IEEE VOLTA*, 1999.

[41] T. D. Givargis, F. Vahid, and J. Henkel, "Trace-driven System-level Power Evaluation of System-on-a-chip Peripheral Cores," in *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2001.

[42] http://www.arm.com/products/CPUs/ARM946ES.html.

[43] "AMBA 2.0 Specification." http://www.arm.com/armtech/AMBA.

[44] "Cell Based IC CB-90 L/M/H Type Features/Basic Specifications, NEC Electronics." http://www.necel.com/cbic/en/cb90/cb90.html.

[45] N. Bansal, K. Lahiri, A. Raghunathan, and S. T. Chakradhar, "Power Monitors: A Framework for System-Level Power Estimation using Heterogeneous Power Models," in *Proc. Int. Conf. VLSI Design*, pp. 579–585, Jan. 2005.

[46] "UC Santa Cruz Floor-planning Tool." http://www.cse.ucsc.edu/research/surf/GSRC/progress.html.

[47] "HotSpot 3.0 Temperature Modeling Tool." http://lava.cs.virginia.edu/HotSpot.

[48] A. Sinha and A. P. Chandrakasan, "JouleTrack - A Web Based Tool for Software Energy Profiling," in *Proc. DAC*, pp. 220–225, June 2001.

[49] M. B. Kamble and K. Ghose, "Analytical Models for Energy Dissipation in Low Power Caches," in *Proc. ISLPED*, pp. 143–148, Aug. 1997.

[50] P. P. Sotiriadis and A. P. Chandrakasan, “A Bus Energy Model for Deep Submicron Technology,” *IEEE Trans. VLSI Systems*, vol. 10, pp. 341–350, June 2002.

[51] H. Su, F. Liu, A. Devgan, E. Acar, and S. Nassif, “Full Chip Leakage Estimation Considering Power Supply and Temperature Variations,” in *Proc. ISLPED*, pp. 78–83, Aug. 2003.

[52] W.Liao, L.He, and K.Lepak, “Temperature and Supply Voltage Aware Performance and Power Modeling at Microarchitechture Level,” *IEEE Trans. Computer-Aided Design*, vol. 24, pp. 1042–1053, July 2005.

[53] “BACPAC - Berkeley Advanced Chip Performance Calculator.” http://www.eecs.umich.edu/~dennis/bacpac.

[54] “The Open SystemC initiative.” http://www.systemc.org.

[55] “BPTM - Berkeley Predictive Technology Models.” http://www-device.eecs.berkeley.edu/~ptm.

[56] “MATLAB - High-Level Technical Computing Environment.” http://www.mathworks.com/products/matlab.

[57] L. Benini, A. Bogliolo, and G. D. Micheli, “A Survey of Design Techniques for System-Level Dynamic Power Management,” *IEEE Trans. VLSI Systems*, vol. 8, no. 3, pp. 299–316, 2000.

[58] R. Golding, P. Bosch, C. Staelin, T. Sullivan, and J. Wilkes, “Idleness is Not Sloth,” in *TCON: Proc. of the USENIX Technical Conference*, 1995.

[59] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. Owicki, “Competitive Randomized Algorithms for Non-Uniform Problems,” in *SODA: Proc. of the first annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 301–309, 1990.

[60] T. Simunic, G. D. Micheli, and L. Benini, “Event-Driven Power Management of Portable Systems,” in *Proc. Int. Symp. System Level Synthesis*, pp. 18–23, 1999.

[61] S. Borkar, “Circuit Techniques for Subthreshold Leakage Avoidance, Control and Tolerance,” in *IEDM*, Dec. 2004.

[62] A. Keshavarzi, S. Ma, S. Narendra, B. Bloechel, K. Mistry, T. Ghani, S. Borkar, and V. De, “Effectiveness of Reverse Body Bias for Leakage Control in Scaled Dual Vt CMOS ICs,” in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 207–212, 2001.

[63] S. Chandra, K. Lahiri, A. Raghunathan, and S. Dey, “Considering Process Variations During System-Level Power Analysis,” in *Proc. ISLPED*, pp. 342–345, Oct. 2006.

[64] C. Kim, K. Roy, S. Shu, K. R. Krishnamurthy, and S. Borkar, “On-Die CMOS Leakage Current Sensor for Measuring Process Variation in Sub-90nm Generations,” in *Symp. on VLSI Circuits*, pp. 250–251, June 2004.

[65] M. H. Wright, “Direct Search Methods: Once Scorned, Now Respectable,” in *Numerical Analysis (D. F. Griffiths and G. A. Watson, eds.), Pitman Research Notes in Mathematics*, pp. 191 – 208, 1995.

[66] D. E. L. *et. al.*, “Managing Power and Performance for System-on-Chip Designs Using Voltage Islands,” in *Proc. Int. Conf. Computer-Aided Design*, 2002.

[67] D. Marculescu and S. Garg, “Process-Driven Variability Analysis for Single and Multiple Voltage-Frequency Island Latency-Constrained Systems,” *IEEE Trans. Computer-Aided Design*, vol. 27, pp. 893–905, May 2008.

[68] N. K. Jha, “Low Power System Scheduling and Synthesis,” in *Proc. Int. Conf. Computer-Aided Design*, pp. 259–263, 2001.

[69] J. Luo and N. Jha, “Power-conscious Joint Scheduling of Periodic Task Graphs and Aperiodic Tasks in Distributed Real-time Embedded Systems,” in *Proc. Int. Conf. Computer-Aided Design*, pp. 357–364, 2000.

[70] F. Xie and M. Martonosi and S. Malik, “Efficient Behavior-driven Runtime Dynamic Voltage Scaling Policies,” in *Proc. Int. Symp. on HW/SW Codesign*, pp. 105–110, 2005.

[71] F. Yao and A. Demers and S. Shenker, “A Scheduling Model for Reduced CPU Energy,” in *Proc. Foundations of Computer Science*, 1995.

[72] T. Ishihara and H. Yasuura, “Voltage Scheduling Problem for Dynamically Variable Voltage Processors,” pp. 197–202, 1998.

[73] R. Jejurikar and C. Pereira and R. Gupta, “Leakage Aware Dynamic Voltage Scaling for Real-time Embedded Systems,” in *Proc. DAC*, pp. 275–280, 2004.

[74] B. Zhai and D. Blaauw and D. Sylvester and K. Flautner, “Theoretical and practical limits of dynamic voltage scaling,” in *Proc. DAC*, pp. 868–873, 2004.

[75] A. Andrei and M. Schmitz and P. Eles and Z. Peng and B. M. Al-Hashimi, “Overhead-Conscious Voltage Selection for Dynamic and Leakage Energy Reduction of Time-Constrained Systems,” in *Proc. Design Automation & Test Europe (DATE) Conf.*, 2004.

[76] M. Bao and A. Andrei and P. Eles and Z. Peng, “Temperature-aware voltage selection for energy optimization,” in *Proc. Design Automation & Test Europe (DATE) Conf.*, pp. 1083–1086, 2008.

[77] A. Srivastava, T. Kachru, and D. Sylvester, “Low-Power-Design Space Exploration Considering Process Variation Using Robust Optimization,” *IEEE Trans. Computer-Aided Design*, vol. 26, pp. 67–79, Jan. 2007.

[78] D. Ernst et. al., “Voltage Scheduling Problem for Dynamically Variable Voltage Processors,” in *Proc. Int. Symp. Microarchitecture*, pp. 7–18, 2003.

[79] K. Flautner and D. Flynn and D. Roberts and D. Patel, “IEM926: An Energy Efficient SoC with Dynamic Voltage Scaling,” in *Proc. Design Automation & Test Europe (DATE) Conf.*, 2004.

[80] “Optimizing the Power for Multiple Voltage Domains.” www.national.com/powerwise/files/SPF_2006_National.pdf.

[81] “PowerWise Adaptive Voltage Scaling (AVS).” http://www.national.com/analog/powerwise/avs_overview.

[82] M .Eireiner and S. Henzler and G. Georgakos and J. Berthold and D.Landsiedel, “ In-Situ Delay Characterization and Local Supply Voltage Adjustment for Compensation of Local Parametric Variations,” vol. 42, pp. 1583–1592, July 2007.

[83] “Frequency and Voltage Scaling Design, Low Power Methodology Manual.” www.springerlink.com/index/r6u34r4n53417290.pdf.

[84] D. Marculescu and S. Garg, “System-Level Throughput Analysis Process Variation Aware Multiple Voltage-Frequency Island Designs,” *ACM Trans. Design Automation Electronic Systems*, vol. 13, Sept. 2008.

[85] K. Niyogi and D. Marculescu, “Speed and Voltage Selection for GALS Systems Based on Voltage/Frequency Islands,” in *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 292–297, 2005.

[86] S. Herbert and D. Marculescu, “Variation-Aware Dynamic Voltage/Frequency Scaling,” in *Proc. Int. Symp. High-Performance Computer Architecture*, Feb. 2009.

[87] R. Teodorescu and J. Torrellas, “Variation-Aware Application Scheduling and Power Management for Chip Multiprocessors,” *SIGARCH Comput. Archit. News*, vol. 36, no. 3, pp. 363–374, 2008.

[88] “Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor.” ftp://download.intel.com/design/network/papers/30117401.pdf.

[89] “ Vector Quantization and Signal Compression .” 1992.

[90] "NANOSIM - High Performance Full-Chip simulation." http://www.synopsys.com/Tools/Verification/AMSVerification/CircuitSimulation/Pages/NanoSim.aspx.

[91] K.Lahiri, A.Raghunathan, and S.Dey, "System-Level Performance Analysis for Designing On-Chip Communication Architectures," *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 768–783, June 2001.

[92] C. Grecu et. al., "A Scalable Communication-Centric SoC Interconnect Architecture," in *Proc. Int. Symp. on Quality Electronic Design*, pp. 343–348, 2004.

[93] K. Lahiri, A. Raghunathan, and S. Dey, "Fast Performance Analysis of Bus-based System-on-Chip Communication Architectures," in *Proc. Int. Conf. Computer-Aided Design*, pp. 566–572, Nov. 1999.

[94] T. V. et. al., "Multi-point Interconnect for Globally-Asynchronous Locally Synchronous Systems," in *Handouts of the Second Asynchronous Circuit Design Workshop*, 2002.

[95] T. V. et. al., "Self-Timed Ring for Globally-Asynchronous Locally-Synchronous Systems," in *Proc. Int. Symp. Advanced Research in Asynchronous Circuits and Systems*, 2003.

[96] J. Muttersbach, T. Villiger, and W. Fichtner, "Practical Design of Globally-Asynchronous Locally-Synchronous Systems," 2000.

[97] "Industry Standard Deep Submicron SPICE MOS Device Model for Circuit Designs." http://www-device.eecs.berkeley.edu/~bsim3.