The Dissertation Committee for Madhu Sarava Govindan
certifies that this is the approved version of the following dissertation:

E^3: Energy-Efficient EDGE Architectures

Committee:

Stephen W. Keckler, Supervisor

Douglas C. Burger

Derek Chiou

Kathryn S. McKinley

Warren A. Hunt Jr

David Brooks

E^3: Energy-Efficient EDGE Architectures

by

Madhu Sarava Govindan, B.E., M.S.C.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2010

To my loving parents, Govindan and Vasanthi,

and my loving wife, Vimala,

and Baba!

Acknowledgments

First and foremost, I am extremely grateful to my advisor, Steve Keckler, for his guidance, and patience throughout my stay in graduate school. This dissertation would not be possible without his constant support, guidance, and research vision. I am deeply indebted to him for teaching me the basics of systems research, moulding, and grooming my skills as a researcher, and for instilling a sense of scientific rigor in me. His research vision, clarity of thought, work ethic, and his breadth of knowledge in Computer Architecture, and in Computer Science, continues to amaze me till date. I am also grateful to Steve for funding my research, and my several conference trips during graduate school. I am very fortunate to have had the opportunity to work with Steve, and I could not have asked for a better advisor. I hope that I can emulate him in my future career.

Next, I would like to thank my de-facto co-advisor, Doug Burger. I am forever indebted to Doug because he was instrumental in bringing me to the University of Texas at Austin, and all of my Ph.D. experience would not be possible without him. As the co-leader of the TRIPS project, and as my advisor through the first two years in graduate school, he has had a major impact on my research style and problem solving. His quick thinking, encyclopaedic knowledge of prior literature in many sub-fields of computer architecture, and his research vision are truly amazing, and I have learned quite a lot from him. Both Doug and Steve have always emphasized the importance of oral and written communication, and have been great role models

for better communication. For that, I am very thankful to both.

I am very grateful to all my committee members, Kathryn McKinley, Derek Chiou, Warren A. Hunt, Jr., and David Brooks. My committee members, despite having busy research schedules, took pains and the time to read through my dissertation, and provided me with comments for making this dissertation better. It was a great learning experience to see how each committee member, who have different research backgrounds, comes up with interesting solutions to different research problems.

I would like to specially thank Professor David Brooks of Harvard University for agreeing to serve in my committee, invaluable advice on the power modeling and validation work, his patience and support with the Turandot/PowerTimer modeling infrastructure, and his deep collaboration with me on the fine-grained DVFS work. I am also grateful to Professor Gu-Yeon Wei, and Wonyoung Kim, and Meeta Gupta of Harvard for their help in refining the idea of fine-grained DVFS policies, and on-die voltage regulator module design. Similarly, our power comparison work was made possible by the two simulators made available to me. I thank Pradip Bose of IBM for providing us access to the Turandot simulator, and Gilberto Contreras for shipping us the XTREM simulator for XScale.

My graduate school experience has been truly amazing, and my involvement with the TRIPS prototype design has made it better. I thoroughly enjoyed my time working on bringing up TRIPS motherboards and chips. It was an amazing experience in an academic setting, and I fully gauged its value during my interview process. My special thanks to the entire hardware team—Raj Desikan, Saurabh Drolia, Divya Gulati, Paul Gratz, Heather Hanson, Changkyu Kim, Haiming Liu, Nitya Ranganathan, Karu Sankaralingam, Simha Sethumadhavan, and Premkishore Shivakumar—and the entire software team—Jim Burrill, Katie Coons, Mark Gebhart, Madhavi Krishnan, Sundeep Kushwaha, Bert Maher, Nick Nether-

cote, Behnam Robatmili, Sadia Sharif, Aaron Smith, and Bill Yoder— for providing a warm and cordial work atmosphere, and for their excellent efforts in making the entire TRIPS system work. My early power modeling work, power validation experiments, and the rest of my dissertation hinge on vast amounts of infrastructure built by all of them, and for that I am grateful to all. Steve Crago, Chen Chen, and Karandeep Singh of Information Sciences Institute (ISI East) require a special mention. My interaction with them for the TRIPS motherboard and chip bring-up was amazing, and I thank them for making my stay at ISI enjoyable, but more importantly, for getting the TRIPS system design right.

I thank all the current and old students of the CART lab, and their family members for making my graduate school experience better, and for the technical camaraderie they brought to the CART meetings and otherwise. I would like to thank Vikas Agarwal, Raj Desikan, and Karthik Agaram for providing support and help during various junctures. M. S. Hrishikesh deserves a special thanks for being a great friend and mentor, and for helping me make important career decisions. I would like to thank his relentless pursuit in bringing me to India to start my career. I really hope that I would be able to get back to India in the near future.

I thank Premkishore, Karu, Ramdas, Simha, Heather, and CK for being wonderful cubicle-neighbors, and for being great role models, and for making life in the ACES building more enjoyable with debates and chats about everything under the sun. Ramdas and Simha have been good personal friends, who have guided, advised, and mentored me at various critical junctures. Special thanks to Ramdas, Premkishore, CK, Nitya, Haiming, and Sai Santosh of Intel for helping with my job search and advising me on career opportunities. Also, I would like to thank Haiming, Jaehyuh, and Paul Gratz for their advice and support at various junctures. I would like to specially thank Divya Gulati, who joined UT Austin in 2003 with me, and shared many of his experiences and struggles with me. The various cricket and

tennis sessions with Divya were unforgettable. Behnam, Dong, and Hadi deserve a special place in this dissertation because many of the TFlex experiments and infrastructure would not be possible without their help. Special thanks to Bert, Katie, Boris, Renee, Jeff, Joel, and Mark of the CART lab for helping me polish the ideas in this dissertation, for proof-reading and helping with paper drafts, and for attending my practice talks and helping me face tough questions. I am especially indebted to Boris for proof-reading my dissertation, and suggesting ideas for its improvement, and for instilling confidence to face my defense. Nitya Ranganathan and her husband, Ram Rangan, have been extremely valuable friends in my life. Nitya extended a great amount of help in the branch confidence work, and Ram helped by extending a patient ear to my frustrations, and for suggesting new ideas for my experiments. Their move to India actually saddened me, and I hope I can follow suit some day.

I am forever grateful to all the folks in the Computer Sciences (CS) department at UT and other departments, who helped me with the required official work. First, I would like to thank the folks at the International Students Office who made my early days at Austin very comfortable. Second, Gloria Ramirez, Katherine Utz, and Lydia Griffith of the CS department helped me wade through the various rules regarding course work, and Ph.D. proposal, and defense–I convey my sincere thanks to all of them for their patience in answering my questions. I would like to specially thank Gem Naivar, who was very kind, patient, and extremely helpful with all the travel arrangements, and other logistics. She also ensured that I received my salary in an orderly fashion every month. Next, the departmental technical support folks, "gripe" , were extremely helpful and patient with me when I filled up the disk volumes several times due to faulting experiments. Special thanks to David Kotz for his help and guidance with the condor system, and for providing me the necessary disk space for conducting my experiments. Dave extended special help during times

when I attempted to compile the TFlex simulator for the standard universe.

I have been fortunate to interact with several experienced researchers during my internships at IBM Research and AMD Graphics group. My interactions with Sani Nassif at IBM Research during my initial TRIPS power modeling work were eye opening in many respects, especially with respect to research in the industry. My IBM research mentor, Charles Lefurgy, has impacted my way of research a lot. He strongly encouraged me to qualitatively evaluate new ideas before actually implementing them to study their effects. My experience at AMD graphics group was amazing, thanks to the support and help of my mentor, Karthik Ramani, and my manager, John Brothers.

I would like to acknowledge Steve and Doug for providing me with funding during my stay at UT. Specifically, I would like to acknowledge NSF award CCF-0916745, Defense Advanced Research Projects Agency under contract F33615-01-C-4106 and NSF CISE Research Infrastructure grant EIA-0303609, which directly funded the research in this dissertation. I also would like to extend my gratitude to the CS department for providing me with teaching assistantships during various semesters of my Ph.D.

My roommates during these years, Piyush Agarwal, Ramtilak Vemu, and Suriya Subramanian, have been an important part of my Austin life and have supported me through tough times, and have shared my joyous moments as theirs. They have become my friends for life, and I hope to be in touch with them. A special mention to the entire Far West gang, Sankar, Sriram, Sundar, Vijaykiran, and Kalyan who made an incredible company during the late-night cricket sessions, movie nights, late-night visits to Ken's Donuts for their samosas, and many other fun-filled activities. Many of my friday evenings were spent at the Intramural Fields playing cricket with many of my friends from UT, especially Sreenivasan, and others. Thanks gang, my life would not have been sane without you all!

My set of friends from my undergraduate institution, Anna University, played a key role in making my life enjoyable, and helped me withstand the struggles of graduate school. The weekend chats with Arumugam, Laks, Pradeep, Sam, Sai, and Vijayaraghavan were all unforgettable memories, and I thank each and every one of them for motivating me during low times. Special thanks to my undergraduate friend, Venkatesan, for providing me with advice with Ph.D. and job search, and for hosting me during my AMD internship. Mani Sridhar deserves a special thanks for motivating me to graduate soon whenever I spoke with him.

My stay at UT Austin was made all the more enjoyable by the students, who joined UT with me in Fall 2003. Their help and support to navigate the new town and UT system have been extremely useful–for that, I thank you all. My experience with the Tamil Cultural Association (TCA), a student organization at UT that caters to the needs of Tamil students from India, has been amazing. By serving as its President for two years, I had great lessons in leadership and team organization, which will be extremely handy in my career. Thanks to Dr. Sata Sathasivan, faculty advisor of TCA, for his constant support and encouragement for TCA activities and my academic research.

This dissertation would not be possible without the constant support, love, and blessings of Shiva Shankar Baba. He has been a friend, philosopher, guide, and an incredible mentor to me during my formative years, and even now. I am ever grateful to him for his love and blessings.

I am forever grateful to my loving wife, Vimala, who has been very understanding and patient with me during my stay in graduate school. She was extremely helpful during my late-night research and writing work, and whenever I fell sick during my dissertation writing. She has been a great source of inspiration during low times, and whenever I was anxious before interviews and just before my defense. I cannot wait to start my life with you, and I look forward to everything life has in

store for us, Vimala!

Last but not the least, I am eternally grateful to my loving parents without whose support this dissertation would not be possible at all. Right from motivating me to go to graduate school to listening to my endless rants filled with frustration to motivating me to complete my Ph.d., they have been with me every step of the way–I am very fortunate to have such loving and understanding parents. They laughed with me when I was happy and achieved in graduate school, and they cried with me when my papers got rejected, and they motivated me to get up every time I fell down. Being their only son, it was one of the toughest decisions of my life to be away from them for an extended period of time at Austin, and to pursue my Ph.d. Looking back, it was one of the best decisions of my life, and it has made a better person out of me. Without the support of my loving parents, and my wife, I would not have travelled this far. To my parents, my wife, and Baba, I dedicate this dissertation.

Madhu Sarava Govindan

The University of Texas at Austin
August 2010

E^3: Energy-Efficient EDGE Architectures

Publication No. _____

Madhu Sarava Govindan, Ph.D.
The University of Texas at Austin, 2010

Supervisor: Stephen W. Keckler

Increasing power dissipation is one of the most serious challenges facing designers in the microprocessor industry. Power dissipation, increasing wire delays, and increasing design complexity have forced industry to embrace multi-core architectures or chip multiprocessors (CMPs). While CMPs mitigate wire delays and design complexity, they do not directly address single-threaded performance. Additionally, programs must be parallelized, either manually or automatically, to fully exploit the performance of CMPs. Researchers have recently proposed an architecture called Explicit Data Graph Execution (EDGE) as an alternative to conventional CMPs. EDGE architectures are designed to be technology-scalable and to provide good single-threaded performance as well as exploit other types of parallelism in-

cluding data-level and thread-level parallelism. In this dissertation, we examine the energy efficiency of a specific EDGE architecture called TRIPS Instruction Set Architecture (ISA) and two microarchitectures called TRIPS and TFlex that implement the TRIPS ISA. TRIPS microarchitecture is a first-generation design that proves the feasibility of the TRIPS ISA and distributed tiled microarchitectures. The second-generation TFlex microarchitecture addresses key inefficiencies of the TRIPS microarchitecture by matching the resource needs of applications to a composable hardware substrate.

First, we perform a thorough power analysis of the TRIPS microarchitecture. We describe how we develop architectural power models for TRIPS. We then improve power-modeling accuracy using hardware power measurements on the TRIPS prototype combined with detailed Register Transfer Level (RTL) power models from the TRIPS design. Using these refined architectural power models and normalized power modeling methodologies, we perform a detailed performance and power comparison of the TRIPS microarchitecture with two different processors: 1) a low-end processor designed for power efficiency (ARM/XScale) and 2) a high-end superscalar processor designed for high performance (a variant of Power4). This detailed power analysis provides key insights into the advantages and disadvantages of the TRIPS ISA and microarchitecture compared to processors on either end of the performance-power spectrum. Our results indicate that the TRIPS microarchitecture achieves 11.7 times better energy efficiency compared to ARM, and approximately 12% better energy efficiency than Power4, in terms of the Energy-Delay-Squared (ED^2) metric.

Second, we evaluate the energy efficiency of the TFlex microarchitecture in comparison to TRIPS, ARM, and Power4. TFlex belongs to a class of microarchitectures called Composable Lightweight Processors (CLPs). CLPs are distributed microarchitectures designed with simple cores and are highly configurable at runtime to adapt to resource needs of applications. We develop power models for the

TFlex microarchitecture based on the validated TRIPS power models. Our quantitative results indicate that by better matching execution resources to the needs of applications, the composable TFlex system can operate in both regimes of low power (similar to ARM) and high performance (similar to Power4). We also show that the composability feature of TFlex achieves a signification improvement (2 times) in the ED^2 metric compared to TRIPS.

Third, using TFlex as our experimental platform, we examine the efficacy of processor composability as a potential performance-power trade-off mechanism. Most modern processors support a form of dynamic voltage and frequency scaling (DVFS) as a performance-power trade-off mechanism. Since the rate of voltage scaling has slowed significantly in recent process technologies, processor designers are in dire need of alternatives to DVFS. In this dissertation, we explore processor composability as an architectural alternative to DVFS. Through experimental results we show that processor composability achieves almost as good performance-power trade-offs as pure frequency scaling (no changes in supply voltages), and a much better performance-power trade-off compared to voltage and frequency scaling (both supply voltage and frequency change).

Next, we explore the effects of additional performance-improving techniques for the TFlex system on its energy efficiency. Researchers have proposed a variety of techniques for improving the performance of the TFlex system. These include: (1) block mapping techniques to trade off intra-block concurrency with communication across the operand network; (2) predicate prediction and (3) operand multicast/broadcast mechanism. We examine each of these mechanisms in terms of its effect on the energy efficiency of TFlex, and our experimental results demonstrate the effects of operand communication, and speculation on the energy efficiency of TFlex.

Finally, this dissertation evaluates a set of fine-grained power management

(FGPM) policies for TFlex: instruction criticality and controlled speculation. These policies rely on a temporally and spatially fine-grained dynamic voltage and frequency scaling (DVFS) mechanism for improving power efficiency. The instruction criticality policy seeks to improve power efficiency by mapping critical computation in a program to higher performance-power levels, and by mapping non-critical computation to lower performance-power levels. Controlled speculation policy, on the other hand, maps blocks that are highly likely to be on correct execution path in a program to higher performance levels, and the other blocks to lower performance levels. Our experimental results indicate that idealized instruction criticality and controlled speculation policies improve the operating range and flexibility of the TFlex system. However, when the actual overheads of fine-grained DVFS, especially energy conversion losses of voltage regulator modules (VRMs), are considered the power efficiency advantages of these idealized policies quickly diminish. Our results also indicate that the current conversion efficiencies of on-chip VRMs need to improve to as high as 95% for the realistic policies to be feasible.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Increasing power dissipation is one of the most serious challenges facing the designers in the microprocessor industry. Current and projected power trends point to a world where all transistors on a microprocessor chip cannot be operated at the maximum possible frequency, and worse, they may not all be switched on simultaneously. Historically, power dissipation and energy efficiency have been main concerns only in the domain of embedded or mobile computing. However, increasing power consumption of processors has started to affect the high-performance computing domain as well, which includes servers, workstations, and supercomputers. Depending on the computing domain, power dissipation affects system costs differently. While power affects overall battery life in the embedded domain, it increases the operating temperature, system cooling, and packaging costs in high-end systems [114].

Industrial data published by researchers show that power consumption of processors has increased steeply over successive generations [3, 40, 74]. Researchers have predicted a super-linear increase in system cooling costs for every additional watt beyond 40 watts consumed by a chip [114]. Similarly, system-level power dissipation affects the overall operating cost of large-scale data centers and supercomputers. Such large-scale systems commonly consume more than 2 Megawatts,

which corresponds to an operational electricity cost of over $1M per year, even at the bargain price of five cents per kilowatt-hour. A recent study shows that the peak load on the US power grid from data centers and servers is about 7 Gigawatts; if current trends continue, this peak load will grow to 12 Gigawatts in 2011 [25]. On the other hand, the Consumer Electronics Association estimates that consumer electronics (cell phones, corded phones, TVs, etc.) consume about 11% and 4% of US residential and total electricity consumption respectively. The expected growth in the number of data centers and the volume of consumer electronics is likely to worsen the problem [90]. Hence, there is an ever-increasing demand for energy-efficient systems in all computing domains.

1.1 EDGE Architectures

Technology trends like increasing processor power dissipation, increasing wire delays [4], and design complexity have forced the processor industry to switch from conventional superscalar processors to multi-core architectures or chip multi-processors (CMPs). While CMPs mitigate the increasing wire delays and the design complexity, they do not directly address single-threaded performance. Also, programs must be parallelized (automatically or manually) to fully exploit the performance of CMPs. Despite the presence of a plethora of parallel language frameworks and APIs (Application Programming Interfaces) [9,17,22,35,80,83,86,105,112,113,115], there is no clear winner yet in terms of universal adoption among programmers. The transition from single-threaded code to pervasive parallel code is steep, and is unlikely to happen soon.

Additionally, the shift to CMPs is likely to worsen the power problem. CMPs must operate within a strict chip-level power budget, (this is true for uniprocessor chips as well) which varies by the market segment targeted by the chip. With the number of cores per chip increasing steadily, operating all the cores in a CMP at

the maximum possible frequency is challenging at best and impossible in the worst case. To stay within the power budget, system designers will be forced to run certain cores at a slower frequency, thereby incurring performance losses in the workloads running on such cores.

As an alternative to conventional CMPs, researchers have recently proposed Explicit Data Graph Execution (EDGE) architectures [14] which enable technology-scalable microarchitectures that can potentially provide good single-threaded performance while exploiting other types of parallelism including data-level and thread-level parallelism [91]. In this dissertation, we examine the energy efficiency of EDGE architectures, specifically one architecture called TRIPS Instruction Set Architecture (ISA) and two microarchitectures – TRIPS and TFlex – that implement the TRIPS ISA. The TRIPS ISA is a novel architecture characterized by two key features: (1) block-atomic execution and (2) dataflow-style execution within blocks [14]. Microarchitectures that implement the ISA fetch, execute, and commit instructions in a group (called a block). This block-atomic execution minimizes the per-instruction book-keeping overheads found in conventional architectures. Additionally, within a block, producer instructions send their results directly to their consumers with sufficient support from the ISA. This dataflow-style execution within a block relieves the hardware from rediscovering the dependence among instructions at runtime, a task that conventional superscalar processors devote significant resources to, and greatly simplifies the underlying microarchitectures.

The TRIPS microarchitecture is a first-generation implementation of the TRIPS ISA [92]. The microarchitecture is partitioned into various small tiles communicating with each other using micronetworks to mitigate worsening on-chip wire delays. The partitioned TRIPS microarchitecture avoids large, centralized hardware structures, typically found in superscalar processors, to reduce the impact of wire delays. The absence of such centralized structures has the potential to decrease the

power consumption of the TRIPS microarchitecture vis-a-vis superscalar processors. On the flip side, the novel ISA and the distributed microarchitecture have their energy overheads, which need to be assessed. Hence, we examine the energy efficiency advantages and overheads of the TRIPS microarchitecture in comparison to other conventional platforms in this dissertation.

The TFlex microarchitecture is a second-generation implementation of the TRIPS ISA, and further improves the energy efficiency of EDGE architectures by a concept called composability. TFlex belongs to a class of microarchitectures called Composable Lightweight Processors (CLPs) – such microarchitectures are distributed and are designed with simple cores. Additionally, CLPs can dynamically aggregate multiple physical cores into a logical processor, and hence, are highly configurable at runtime. By aggregating or disaggregating cores, the TFlex microarchitecture can be adapted to provide logical processors of varying sizes at runtime. This composable nature of TFlex further improves energy efficiency of EDGE architectures by matching resource demands of applications to the right amount of hardware resources. As a key contribution of this dissertation, we examine the energy efficiency of the TFlex microarchitecture in comparison to TRIPS and other hardware platforms as well.

1.2 Dissertation Contributions

This dissertation explores the energy efficiency of EDGE architectures, specifically one architecture called the TRIPS ISA, and that of two microarchitectures: TRIPS and TFlex. This section lists the key contributions of this dissertation.

1.2.1 TRIPS: A Detailed Power Analysis

First, we perform a thorough power analysis of the TRIPS microarchitecture. Building accurate architectural power models is a challenging exercise for several rea-

sons [59], and is especially challenging for novel architectures such as TRIPS. As one of the key contributions of this dissertation, we describe our methodology for building initial architectural power models for TRIPS, and our methodology for validating those models. We also identify common pitfalls in architectural power modeling, and suggest a few recommendations from our validation experience to avoid such pitfalls [36].

Power Model Validation: In this part of the dissertation, we first describe how we develop architectural power models for TRIPS. We then improve the accuracy of the architectural power models using hardware power measurements on the TRIPS prototype system combined with detailed Register Transfer Level (RTL) power models from the TRIPS design. Using these refined architectural power models, we perform a detailed performance and power comparison of the TRIPS microarchitecture with two different processors: 1) a low-end processor designed for energy efficiency (ARM/XScale [20][1]) and 2) a high-end superscalar processor designed for high performance (a variant of Power4 [54]).

Such an exercise in performance and power comparison is very challenging mainly because these processors have been designed using different process technologies, and design methodologies. We normalize the performance and power models of all platforms to the best extent possible to ensure a very fair comparison of the underlying architectures and microarchitectures. This detailed power analysis provides key insights into the advantages and disadvantages of the TRIPS ISA and the TRIPS microarchitecture compared to other processors like XScale and Power4. Our results indicate that the TRIPS microarchitecture achieves achieves 11 times better energy efficiency compared to ARM, and 12% better energy efficiency than Power4, in terms of the Energy-Delay-Squared (ED^2) metric.

[1]We use the terms ARM/XScale interchangeably in this dissertation

1.2.2 TFlex Power Analysis and Comparison

Second, we analyze the power efficiency of the TFlex microarchitecture [57]. As TFlex belongs to a family of Composable Lightweight Processors (CLPs), it can be configured to match the resource needs of a variety of applications. If an application has abundant parallelism, TFlex aggregates multiple physical cores[2] into a logical processor, and improves system performance or throughput. On the other hand, if the application has poor parallelism TFlex devotes fewer logical cores to the application, thereby optimizing for energy efficiency. We evaluate the energy efficiency of the TFlex microarchitecture by developing its power models based on the validated TRIPS power models. Our quantitative results indicate that by better matching execution resources to the needs of applications the composable TFlex system can operate in both regimes of low power (similar to ARM) and high performance (similar to Power4). We also show that composability of TFlex helps achieve a signification improvement (2x) in energy-efficiency compared to TRIPS, in terms of the ED^2 metric.

1.2.3 Composability versus DVFS

Third, we examine the efficacy of processor composability – the ability of dynamically composing physical cores into a logical processor – as a potential performance-power trade-off mechanism using the TFlex microarchitecture. A technique called Dynamic Voltage and Frequency Scaling (DVFS) has served as the mainstay performance-power trade-off mechanism in processors. To explain how DVFS serves as a performance-power mechanism we look at the different components of processor power dissipation, and how voltage and frequency affect the power dissipation.

[2]The exact number depends on various factors, and can be decided by the compiler or the operating system or some combination of the two

Processor power consumption is governed by the equation

$$P = CV^2 f + VI_{\text{off}} \tag{1.1}$$

. In Equation 1.1, $P, V, f, C, I_{\text{off}}$ denote power dissipated, supply voltage, clock frequency, active capacitance (capacitance that is charged and discharged), and leakage current of the processor respectively. The first term is dynamic power due to charging and discharging of circuit capacitance as the processor operates. The second term is static power due to leakage of transistors that cannot be completely turned off. Historically, due to increasing clock frequencies dynamic power has been a major component of the total power. However, in deep-submicron technologies (90nm and below), leakage power has become a significant fraction of the total power [21].

Most modern processors support a form of dynamic voltage and frequency scaling (DVFS) as a performance-power trade-off mechanism. As given in Equation 1.1, the dynamic power of a processor depends on the supply voltage(V), and the clock frequency(f) of the processor. The maximum clock frequency of the processor depends on the supply voltage of the processor. Additionally, to a first order, the performance of the processor depends linearly on its clock frequency (ignoring the effect of clock frequency on main memory latency). Hence, by reducing the supply voltage, and thereby reducing the processor clock frequency, one can obtain cubic reductions in dynamic power (a quadratic effect due to reduction in voltage and a linear effect due to reduction in clock frequency) with only a linear reduction in processor performance. By modulating the supply voltage and clock frequency of the processor, DVFS helps trade-off dynamic power for performance.

DVFS has been the traditional performance-power trade-off mechanism in all modern processors. However, the rate of supply voltage scaling has slowed significantly in recent process technologies. A key reason for the slow rate of voltage

scaling is a type of leakage power called sub-threshold leakage. When supply voltage is reduced sub-threshold leakage increases exponentially [21]. Designers have slowed the rate of voltage scaling to keep sub-threshold leakage power under check. This slow rate of voltage scaling has significantly limited the effectiveness of DVFS as a performance-power trade-off mechanism, and has created a dire need for alternatives to DVFS. In this dissertation, we explore processor composability as an architectural alternative to DVFS.

Through experimental results we compare the performance and power trade-offs offered by three distinct mechanisms: (1) pure frequency scaling, (2) processor composability, and (3) voltage and frequency scaling. Our results show that composability achieves as good performance-power trade-offs as pure frequency scaling (only changes in frequency with no change in voltage), and much better trade-offs when compared to voltage and frequency scaling (both frequency and voltage can change). Our results indicate that among the mechanisms of frequency scaling, composability, and DVFS, when scaling up performance and power, pure frequency scaling is the best option as it provides a one-to-one trade-off between performance and power. Once the point where the frequency cannot be increased further without increasing the voltage is reached, processor composability provides the next best performance-power trade-off. Adding more physical cores to the logical TFlex processor buys a linear increase in performance with linear increases in power similar to pure frequency scaling. Beyond a tipping point, which is application-dependent, where adding more TFlex cores does not provide a linear performance-power trade-off, voltage and frequency scaling should be used as a last resort to improve performance. DVFS must be the last resort because a linear increase in performance with DVFS comes with a cubic increase in power. On the other hand, when performance and power are scaled down, the above steps should be applied in reverse. First, DVFS must be applied as it can provide cubic reductions in power for only

a linear drop in performance, followed by scaling down the number of cores using composability, which can be finally followed by pure frequency scaling. We also present a case for combining DVFS with composability. Our results show that this combination widens the operating regime of the composable system when operating under fixed performance or power targets.

1.2.4 Additional Performance Mechanisms for TFlex

Next, we explore the effects of additional performance-improving techniques for the TFlex system on its energy efficiency. Researchers have proposed a variety of techniques for improving the performance of the TFlex system. These include: (1) block mapping techniques to trade off data locality with concurrency [88]; (2) predicate prediction [27]; and (3) operand multi-cast/broadcast mechanism [64]. We examine each of these mechanisms in terms of its effect on the energy efficiency of TFlex, and our experimental results demonstrate the effects of operand communication, and speculation on the energy efficiency of TFlex.

1.2.5 Fine-grained Power Management Policies

Finally, this dissertation evaluates a set of fine-grained power management (FGPM) policies for TFlex: exploiting instruction criticality and controlled speculation. These policies rely on a temporally and spatially fine-grained dynamic voltage and frequency scaling (DVFS) mechanism for improving power efficiency. Because modulating the power supply voltage requires adjusting the board-level voltage regulator module (VRM), DVFS has been historically applied to an entire chip as a whole. However, recent research has proposed on-chip VRMs that provide a spatially and temporally fine-grained DVFS mechanism [60]. A Composable Lightweight Processor (CLP) like TFlex combined with such a fine-grained DVFS mechanism opens up exciting opportunities for power management policies. In this dissertation, we

explore two such fine-grained policies for the TFlex platform.

The first policy exploits the concept of instruction criticality to improve energy efficiency. We exploit the general principle that critical computation of the application determines execution speed, and non-critical computation in an application can be slowed down to exploit the inherent slack. Using a critical path model for the TFlex microarchitecture we compute the criticality of instruction blocks, and use this criticality to decide the DVFS setting that the block gets mapped to. Our policy maps highly critical blocks to higher frequencies whereas non-critical ones get mapped to lower frequencies so that the average power dissipation of the system can be reduced without significantly affecting performance.

The second policy explores the use of controlled speculation. Modern processors employ various forms of speculation like branch prediction, and memory disambiguation. to extract better performance from applications. Although essential for high performance, aggressive speculation could potentially lead to wasted instructions and resources, and thus, wasted energy. The TFlex system also employs speculation by executing multiple blocks in parallel via branch prediction. The second policy attempts to minimize the energy wasted by aggressive speculation in TFlex using a technique called branch confidence prediction. Branch confidence prediction attempts to predict how confident we are about a given branch prediction. If branch confidence prediction accuracy is very high, we can easily identify high and low confidence branch predictions. Instruction blocks that have high branch confidence are mapped to higher frequencies while ones that have lower branch confidence are mapped to lower frequencies. Our experimental results indicate that idealized instruction criticality and controlled speculation policies improve the operating range and flexibility of the TFlex system. However, when the actual overheads of fine-grained DVFS, especially energy conversion losses of voltage regulator modules (VRMs), are considered the power efficiency advantages of these

idealized policies quickly diminish. Our results also indicate that the current conversion efficiencies of on-chip VRMs need to improve to as high as 95% (current efficiencies range from 80% to 89%) for the realistic policies to be feasible.

1.3 Leakage Power

As discussed in Section 1.2.3, processor power consumption consists of two components: 1) dynamic power and 2) leakage power. Leakage power arises due to leakage of transistors that cannot be completely turned off. Although there are several sources of leakage, the two major sources are sub-threshold and gate-tunneling leakage currents [119]. Researchers have extensively studied techniques to reduce these sources of leakage, and have proposed solutions at the process technology, circuit, and microarchitectural levels. For example, Intel has introduced metal gates and high-K dielectric materials in its 45nm process technology [72]. These changes provide signification reduction in gate-tunneling leakage current. Techniques like dual-VT transistors [45], Gated V_{dd} and MTCMOS with sleep transistors [15, 85], body biasing [56], drowsy caches [32], and leakage-biased bitlines [44] have been studied to reduce sub-threshold leakage. In this proposal, we do not explore techniques that directly address leakage: most of the above techniques are directly applicable to the TRIPS and the TFlex microarchitectures. However, we explore a fine-grained DVFS mechanism which modulates the voltage and the frequency of elements on a multi-core chip. This technique addresses sub-threshold leakage power in two ways. First, as leakage power is governed by the equation VI_{off}, reduction in voltage reduces leakage power. Second, due to short-channel effects, reduction in supply voltage also reduces I_{off} which in turn reduces leakage power [32].

1.4 Dissertation Layout

This dissertation is organized as follows. Chapter 2 provides an overview of the TRIPS ISA and the TRIPS microarchitecture. We also qualitatively discuss the advantages and disadvantages of the TRIPS ISA and microarchitecture compared to conventional processors in that chapter. Next, we present an overview of Composable Lightweight Processors (CLPs) and the TFlex microarchitecture in the same chapter.

Chapter 3 first describes how we build our architectural power models for the TRIPS microarchitecture. We examine the accuracy of our architectural power models by comparing against power measurements from a TRIPS hardware prototype. We also describe how we improve the accuracy of our power models by leveraging hardware power measurements, and detailed RTL power models.

Chapters 4 and 5 present a detailed performance and power comparison of TRIPS and TFlex with that of ARM and Power4 microarchitectures. First, we describe our power modeling methodology for all platforms involved, and also describe how we normalize the power models of all platforms to ensure a fairer comparison. Next, we compare the performance, power, and energy efficiency of all the platforms. Finally, we present a detailed power breakdown of various hardware structures in all the platforms. This chapter highlights the advantages and disadvantages of the TRIPS and TFlex microarchitectures compared to other processors.

Chapter 6 compares the power-performance trade-offs offered by composability to that of DVFS with the goal of examining if composability can be an effective alternative to DVFS. Chapter 7 evaluates the energy efficiency of three different mechanisms that improve the performance of the TFlex system. These include: 1) block mapping techniques to trade off intra-block concurrency with operand communication across the operand network. 2) predicate prediction and 3) operand multi-cast/broadcast mechanism.

Chapter 8 provides an overview of the fine-grained power management techniques to improve TFlex power efficiency. We briefly discuss related work in Chapter 9. Chapter 10 concludes this dissertation by summarizing our key experimental results and observations, and by discussing possible future work for further improving energy efficiency of EDGE architectures.

Chapter 2

ISA and Microarchitectures Overview

This chapter provides a brief overview of the TRIPS ISA, and the TRIPS and the TFlex microarchitectures. This chapter also lists the advantages and disadvantages of the TRIPS ISA and the TRIPS microarchitecture, followed by a discussion of composable features provided by the TFlex microarchitecture.

2.1 TRIPS ISA

In this dissertation we examine the TRIPS ISA, an EDGE architecture [14], which is implemented by the TRIPS and the TFlex microarchitectures. The ISA enables compiler-generated dataflow graphs to be mapped to distributed, technology-scalable microarchitectures. The TRIPS ISA is characterized by two key features: (1) block-atomic execution and (2) direct communication among instructions within a block. Both these features allow an efficient dataflow-style execution of instructions.

2.1.1 Block-Atomic Execution

The TRIPS ISA obeys a block-atomic execution model, in which a block (a group of instructions) is fetched, executed, and committed as one entity. The ISA can group up to 128 instructions into a single TRIPS block. This block-atomic model amortizes many per-instruction bookkeeping overheads across a large number of instructions. The model also reduces the number of branch predictions, and the number of times control decisions such as fetch and commit have to be made.

The TRIPS compiler constructs the blocks out of individual instructions, and also assigns a location within the block to each instruction. Each TRIPS block can contain anywhere from two to five 128-byte chunks. The first 128-byte chunk of every block is called the `header` block, which can encode up to 32 `read` and 32 `write` instructions. The TRIPS ISA partitions the architectural register file into four register banks according to the register specifier. The ISA restricts the number of read and write instructions in a given block to a maximum of eight for each register bank (meaning each bank can support up to 8 register read and 8 register write instructions for a total of up to 32 read and 32 write instructions per block). Each read instruction accesses the architectural register value from its corresponding bank and sends the value to its consumers within the block. The write instruction, on the other hand, receives outputs from instructions within the block and commits them to the architectural register file. The header chunk also holds three types of control state for the block: a 32-bit "store mask" that indicates which of the possible 32 memory instructions are stores, block execution flags that indicate the execution mode of the block, and the number of instruction "body" chunks in the block.

A TRIPS block can contain up to four body chunks of 128 bytes each. Each body chunk contains up to 32 instructions, thus giving a maximum of 128 instructions (32 instructions per chunk x 4 body chunks) per TRIPS block. In addition to the maximum number of read and write instructions, the TRIPS ISA enforces

additional restrictions to easily detect block completion on the distributed execution substrate. Of the total 128 instructions, each TRIPS block can only contain up to 32 load or store instructions. The "store mask" in the header chunk identifies which of these 32 instructions is a store, and is a key component in the block commit protocol [92]. Furthermore, the TRIPS compiler employs a technique called predication [102] to construct large blocks. In order to easily detect the block completion event through all paths taken through predicated code, the TRIPS ISA also enforces that every predicated path must produce the same number of block outputs (stores, register writes and one branch). The TRIPS compiler ensures that the generated blocks conform to these constraints [101].

2.1.2 Direct Instruction Communication

The second key feature of the TRIPS ISA is the direct instruction communication within each TRIPS block. Using this feature, the producing instructions in a block directly send their results to their dependent consumers in a dataflow-style. Conventional superscalar processors typically devote monolithic, power-hungry hardware resources to identify all consumers of a given instruction. The dataflow-style execution enabled by the TRIPS ISA significantly simplifies the underlying microarchitectures. Any microarchitecture that implements the TRIPS ISA assigns physical coordinates to each of the 128 instructions in a TRIPS block. Using these physical coordinates, the microarchitecture precisely determines the location of all consumers of a given producing instruction, and can forward the produced values to the consumers.

2.1.3 ISA Advantages and Disadvantages

The TRIPS ISA has several advantages over conventional ISAs. Since the TRIPS ISA is block-based, any microarchitecture that implements this ISA fetches, exe-

cutes, and commits instructions as a block, in contrast to conventional designs that work with individual instructions. Second, the direct instruction-to-instruction communication in the TRIPS ISA relieves the hardware from rediscovering dependence among instructions again at runtime, which in turn simplifies the microarchitectures that implement the TRIPS ISA. Additionally, since all temporary values produced and consumed entirely within a block are directly communicated to the consuming instructions, the TRIPS ISA reduces the number of accesses to the global register file.

However, the block-oriented nature and the direct instruction communication of the TRIPS ISA have their disadvantages too. As mentioned in Section 2.1, every TRIPS block necessarily includes a header chunk with the register read and write instructions along with other metadata needed for executing that block. The body contains the actual "compute" instructions from the block. The presence of the header block is a clear disadvantage of the TRIPS ISA with respect to instruction cache efficiency and capacity. The direct instruction communication can be an additional overhead when a given producing instruction has many consumers. The TRIPS ISA employs "move" instructions to build a software fan-out tree for delivering values to all consumers, which is another disadvantage of the ISA. Furthermore, the TRIPS ISA utilizes dataflow-style predication [102] to build large blocks that can be mapped to the microarchitecture. Predication helps the compiler to build large blocks, and thereby reduce book-keeping overheads including instruction fetch and commit. However, the instructions within a block that are present on the wrong path of the predicate value consume precious resources like instruction cache lines, and reservation stations, and do not contribute to the actual output of the block.

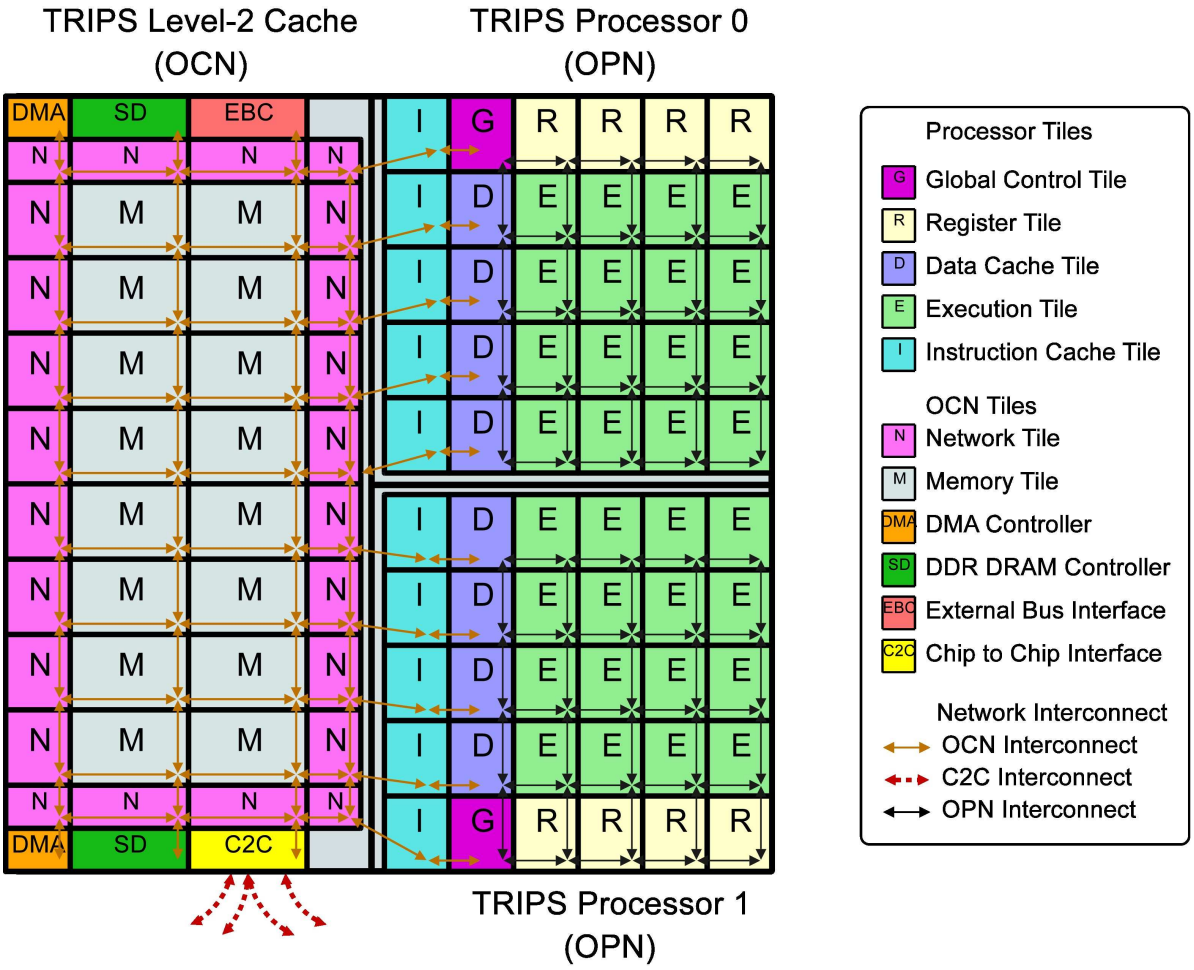


Figure 2.1: Block Diagram of the TRIPS Chip

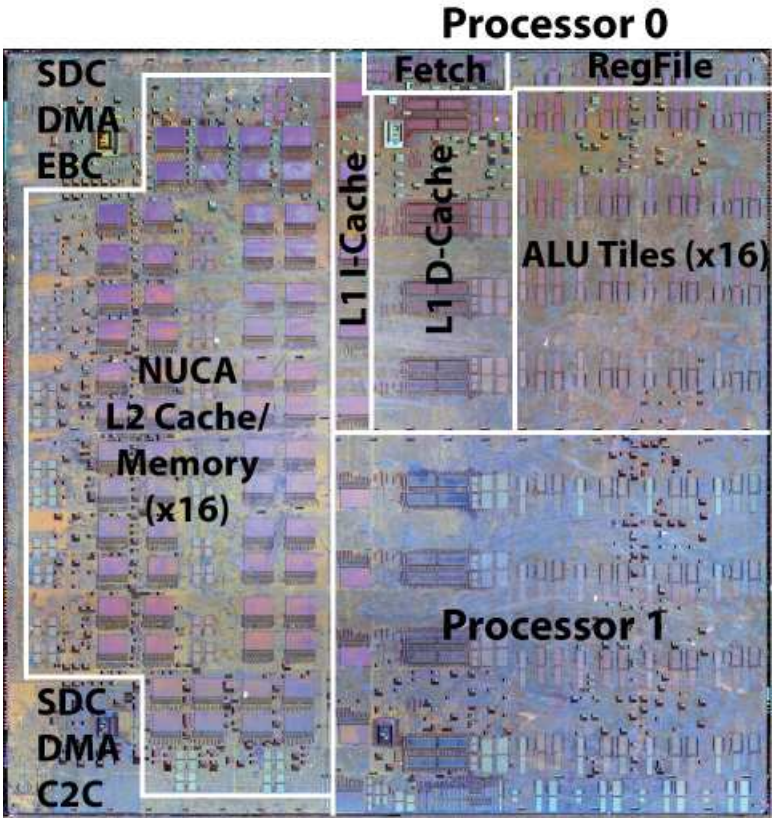


Figure 2.2: Die Photo of the TRIPS Chip

2.2 TRIPS Microarchitecture

The TRIPS microarchitecture implements the TRIPS ISA and provides technology-scalable performance on single-threaded workloads as well as exploits parallelism at different levels of granularity. Figures 2.1 and 2.2 show a block diagram of the TRIPS microarchitecture and an annotated die photo of the TRIPS prototype chip respectively. Each TRIPS chip consists of two processor cores (marked as Processors 0 and 1) and a 1-MB Non-Uniform Cache Access (NUCA) L2 cache organized into 16 memory banks [58]. The block diagram also shows the major microarchitectural units of a single TRIPS processor, including the register file, global control unit, L1 instruction, and data caches, and the 4x4 array of execution units. Each of these units is partitioned into smaller identical tiles, which are marked using various letters in the block diagram. The tile marked `G` is the Global Control Tile that orchestrates global block fetch and commit protocols. The register tiles (marked `R`) implement the partitioned register file banks and the register read and write queues that implement register forwarding between blocks. The data and instruction caches are implemented by tiles marked as `D` and `I` respectively. The 4x4 grid of execution units are marked with the letter `E`. The various tiles in the processor communicate with each other using well-defined micronetworks. One such network is the operand network (OPN) used for communicating operands (register values, ALU results, and load results) among various tiles. Each processor core also implements six additional micronetworks for orchestrating other distributed control and data protocols [92].

The processors and the NUCA L2 caches communicate using an on-chip network. The chip additionally has several data controller tiles, including two SDRAM controllers (SDC), two Direct Memory Access (DMA) controllers, an External Bus Controller (EBC) and a Chip-to-Chip (C2C) controller. The TRIPS chip is designed in a 130 nm IBM ASIC process with about 170 million transistors. To simplify pre-silicon verification, the TRIPS chip does not implement any power management

features like clock gating or dynamic voltage and frequency scaling [98].

Since the TRIPS ISA is block-based, the TRIPS microarchitecture fetches, executes, and commits instructions as a block of instructions. As mentioned in Section 2.1, the TRIPS compiler constructs large (up to 128 TRIPS instructions) single-entry multiple-exit blocks of instructions that are similar to hyperblocks [101]. The TRIPS microarchitecture fetches and maps these blocks of instructions onto the grid of 16 execution units (E-Tiles). All instructions within a TRIPS block execute in a dataflow-order. The TRIPS microarchitecture supports up to eight TRIPS blocks in flight, of which seven blocks are speculative and one is non-speculative. These eight blocks are stored in the reservation stations distributed across the 16 E-Tiles, and constitute a large 1024-entry instruction window (8 blocks x 128 instructions). The distributed nature of the microarchitecture allows such a large instruction window to be constructed without adversely affecting processor cycle time. The TRIPS microarchitecture supports both single-threaded and multi-threaded modes [92]. In the multi-threaded mode, the TRIPS microarchitecture supports up to four independent threads by partitioning its resources (register banks, reservation stations) equally among the four thread slots. In this dissertation, we only exercise the single-threaded mode where a single application is run across the entire processor core.

Microarchitecture Advantages and Disadvantages: The TRIPS microarchitecture has been implemented using small distributed tiles connected via well-defined micronetworks. This tiled design has been primarily motivated by ever-worsening on-chip wire delays, which are even exacerbated by superscalar designs that employ power-hungry monolithic hardware structures. In addition to mitigating wire delays, the tiled nature of the TRIPS microarchitecture has the additional benefit of energy efficiency. Conventional superscalar processors that sustain high issue widths (of four or greater) employ multi-ported register files, and Content-Addressable Memory (CAM) based instruction windows to extract better perfor-

mance. In contrast, the TRIPS microarchitecture extracts better performance using its distributed substrate. For instance, instead of employing multi-ported register files to support multiple in-flight accesses, the TRIPS microarchitecture partitions a large register file into four register banks (`R` tiles), each of which has only two read ports and one write port. Similarly, the dataflow-style execution of TRIPS ISA greatly simplifies the hardware for identifying the dependent consumers instead of relying on power-hungry CAM structures.

On the flip side, the distributed nature of TRIPS microarchitecture comes with certain energy overheads as well. The tiled microarchitecture relies on well-defined micronetworks and distributed protocols for various operations like block fetches, flushes, and commits. Such distributed protocols are an energy overhead of the TRIPS microarchitecture. Furthermore, the TRIPS microarchitecture relies on an operand network (OPN) for communicating data among the tiles of a TRIPS core. The OPN replaces the conventional bypass networks, and is a key determinant of TRIPS performance. The OPN implementation in TRIPS is a 5x5 mesh network connecting 25 tiles. All of these 25 instances of the OPN router consume energy for clocking the input FIFOs, arbitration logic, and for driving control and data packets onto wires connecting to the neighboring routers. Thus, the OPN could be a significant energy overhead for the TRIPS microarchitecture. Our experimental results in later chapters clearly quantify the effects of the advantages and disadvantages of the TRIPS ISA and microarchitecture.

2.3 TFlex Overview

In this section, we provide an overview of a new class of microarchitectures called Composable Lightweight Processors (CLPs) and the TFlex microarchitecture, a type of CLP that implements the TRIPS ISA. The TFlex microarchitecture supports composability - a feature that enables dynamic aggregating of physical cores into

a logical processor. In this section, we also describe the various features of TFlex microarchitecture, and the features of the TRIPS ISA and TFlex microarchitecture that support composability.

Composable Lightweight Processors (CLPs) are a new class of microarchitectures that are characterized by two key features: they are highly configurable at runtime to adapt to resource needs of applications and are distributed microarchitectures built using simple cores or tiles. CLPs provide a flexible solution to one of the key questions in chip multiprocessor (CMP) design: how many cores to place in each chip, and how big should each core be? Different CMPs take differing approaches to this question depending on their market segment, which we refer to as *bulldozers*, *chainsaws*, and *termites*. *Bulldozer* designs typically are composed of fewer but complex cores of larger area whereas *termites* are composed of lots of simpler cores, and *chainsaws* fall somewhere in between *bulldozers* and *termites*. As an example, at one end of the spectrum are *bulldozer* designs like AMD quad-core Barcelona, which consists of four complex 4-issue, out-of-order processors each measuring $36mm^2$ at 65nm [23]. At the other end, Intel's Polaris chip is an example of a bunch of termite processors, each measuring about $2.5mm^2$, and is aimed at very parallel workloads like network processing. We also have chainsaw designs like the Sun's single-issue in-order Niagara-2, which exploits abundant thread-level parallelism (TLP) in transaction processing workloads.

While bulldozer designs are better suited for mining ILP, termite and chainsaw designs are better suited for mining TLP. Thus, these CMP designs are better optimized for exploiting one type of parallelism or the other at design time, and are inflexible when the available parallelism in workloads does not match their assumptions. The goal of CLPs is to increase the flexibility of CMPs by building termites or chainsaws, and to dynamically aggregate them together to build bulldozers as and when needed.

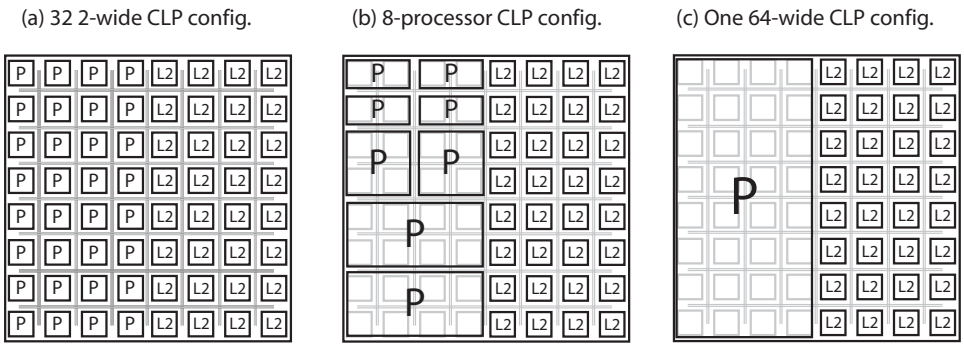


Figure 2.3: TFlex Microarchitecture Overview

The key advantage of CLPs is the flexibility offered to software layers, especially the Operating System (OS) or the hypervisor and the compiler. Depending on the system metric to be optimized for, and the inherent concurrency available in workloads, the OS can choose the right granularity of logical processors to allocate to those workloads. For example, if an application has high inherent concurrency (Instruction-Level Parallelism or ILP), the OS can aggregate many physical cores into a logical processor, allocate that processor to the application, and thus, optimize for performance. On the other hand, if an application has lower concurrency, the OS can allocate fewer physical cores to that application, and thus, optimize for power efficiency. Because CLPs provide adaptability in the number and granularity of processors, their flexibility not only enables efficient execution of single-threaded workloads, but also multi-threaded workloads by extracting TLP.

2.4 TFlex Microarchitecture

The TFlex microarchitecture [57] belongs to this new class of microarchitectures (CLPs) for the following reasons: 1) TFlex is configurable because it can dynamically aggregate multiple physical cores into a logical processor and 2) TFlex is a distributed microarchitecture built using simple, dual-issue cores. In this dissertation, we evaluate a specific TFlex microarchitecture through simulation that has 32

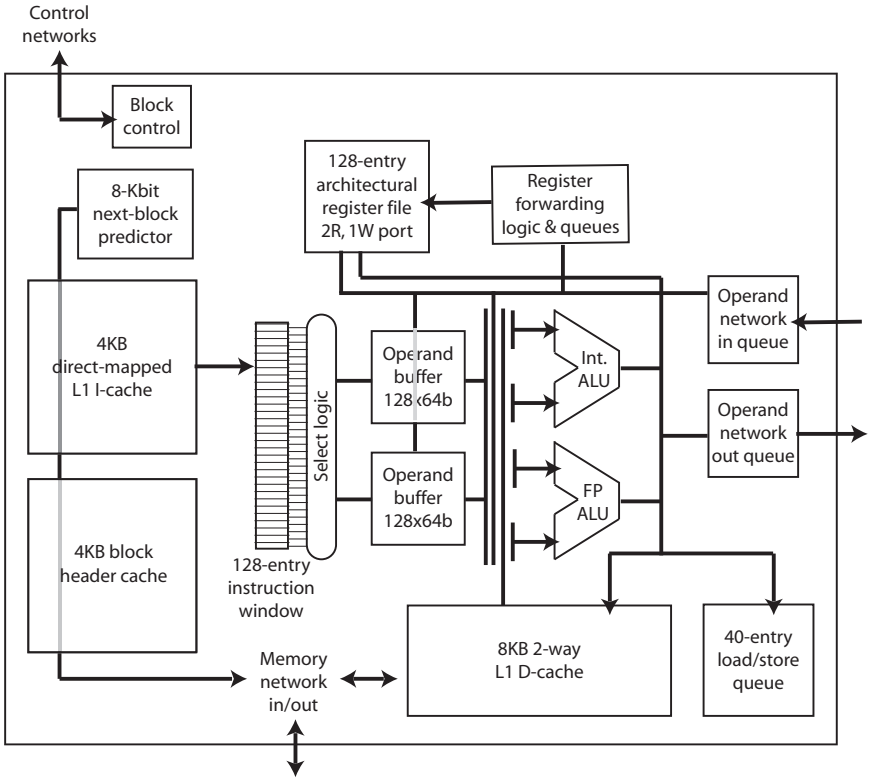


Figure 2.4: Illustration of microarchitectural components of a single TFlex core. The TFlex chip consists of 32 such TFlex cores and NUCA L2.

simple dual-issue cores along with 16 banks of NUCA L2 cache. The OS can configure the TFlex microarchitecture into many possible configurations, each having a different number and granularity of processors. Figure 2.3 shows three sample configurations out of the many possible configurations: (a) 32 1-core processors, (b) a mix of processors composed of different numbers of cores and (c) one 32-core processor. Depending on the needs of the application and the need to optimize for performance, power-efficiency or throughput, the right number of cores can be allocated to a process.

Figure 2.4 details the microarchitecture of the TFlex processor. The TFlex system we use in this dissertation consists of 32 such TFlex cores, and 16 L2 banks of 64KB each for a total of 4 MB of L2 cache capacity. Each TFlex core has enough resources (Instruction-Cache, Data-cache, instruction windows, operand buffers, load-store queues, register files, and an operand router) to fully execute only one TRIPS block. There are certain key differences between the TRIPS and the TFlex microarchitectures. First, the TFlex microarchitecture avoids the maximally-sized solution employed in the load-store queue design of TRIPS. TFlex instead employs a negative acknowledgement based load-store queue design to avoid fully-sized LSQs [97]. Second, each OPN router in TFlex enjoys twice the bandwidth when compared to its counterpart from the TRIPS design. Each TFlex router employs two sets of OPN channels and FIFOs compared to single channel and set of FIFOs in TRIPS. Third, each TFlex core is capable of issuing two instructions in each cycle. Each core can either issue two integer instructions or one integer and one floating point instruction every cycle. This is in contrast to the E-Tiles of the TRIPS design which can issue only one instruction per cycle. We refer the reader to [57] for a more thorough and detailed description of the TFlex microarchitecture.

2.5 Support for Composability

The TFlex processor leverages support from the TRIPS ISA and the microarchitecture to provide a composable execution substrate. First, the TRIPS ISA being block-oriented greatly reduces many book-keeping overheads like instruction fetch, and commit [14]. Since each TRIPS block can have up to 128 instructions, the frequency of key control decisions like fetching new blocks, committing executed blocks, and predicting the next block can be reduced. Second, since the TRIPS ISA is an EDGE architecture the communication between instructions within a block is explicit. This support from the ISA facilitates the process of composing physical cores into logical processors. The explicit encoding of the targets of an instruction (the physical co-ordinates specifying the core ID and the instruction queue ID) are interpreted "differently" depending the "logical" size of the composed TFlex system. Thus, the direct instruction communication feature of the TRIPS ISA eliminates the need for operand broadcast otherwise required for composing cores.

The CLP microarchitectures partition structures by address whenever possible, and avoid physically centralized microarchitectural structures completely. Specifically, the TFlex microarchitecture supports composability by physically distributing various structures including the register file, instruction window, L1 caches, and operand bypass network. When a TFlex core operates as a single logical processor, all of the microarchitecture structures are local. However, when multiple cores are aggregated into a logical processor, the logical instruction window, register file, instruction cache, data cache, and branch predictor are address-interleaved across all the participating cores. Since interleaving is controlled by bit-level hash functions, all logical processor sizes must be a power-of-two number of cores (up to 32 cores in our TFlex system).

The TFlex microarchitecture currently uses three distinct hash functions for address leaving across three different classes of hardware structures: (1) Block Start-

ing Address, (2) Instruction ID within a block, and (3) Data address. The next-block predictor tables and the I-Cache tags utilize a hash function based on the virtual address of a block, which corresponds to the Program Counter in conventional architectures. Each TRIPS block contains up to 128 instructions. Depending on the block-mapping mode (described further in Chapter 7), these 128 instructions are either striped across all the participating cores or all mapped to a single owner core. Finally, the data caches and the load-store queues are partitioned based on a hash function of the load-store addresses, and the register files are address-interleaved based on the lower order bits of the register ID.

In addition to partitioning of various structures, the microarchitecture implements various distributed protocols using micronetworks to implement instruction fetch, execute, commit, speculation recovery, and other processor actions [57]. Each TRIPS block is assigned an owner core based on a hash function of the virtual address of the block. This owner core is in charge of initiating the fetch of the block, and also predicting the next block. Once the address of the next predicted block is known, a message is sent to the owner core of the predicted block to initiate fetch. The owner core is also responsible for initiating flush messages caused by misspeculations, and also detecting when the block is complete and committing it. Similar to the TRIPS microarchitecture, the TFlex microarchitecture maintains many blocks in flight depending on the number of cores in the composed system. When the logical TFlex processor is composed of just one core, only one TRIPS block is in flight as each core only has enough resources to execute a single block. When more than one physical core is composed into a logical processor, the TFlex microarchitecture maintains as many blocks in flight as the number of physical cores. Only one of these in-flight blocks is non-speculative, and the others are speculative.

Chapter 3

TRIPS Power Modeling and Validation

Designers typically construct architectural power models with cycle-level performance simulators to investigate power-performance trade-offs early in the design cycle. The most commonly used power model in academic architectural studies is Wattch [12]. Other high-level analytical power models are listed in the survey by Najm [76]. Despite substantial effort by researchers to build such power models, validating these models has proven difficult at best. The absolute power estimates of Wattch are validated to within 30% for three industrial designs [12]. Despite such validation efforts, applying such models to novel architectures such as TRIPS and new process technologies invariably results in errors.

In this chapter, we describe our methodology for building initial architectural power models for the TRIPS architecture. Next, we describe our methodology for validating these power models with feedback from RTL power models and hardware power measurements. We show that applying common power modeling methodologies to the TRIPS architecture underestimates the hardware power by 65% on an average (This means that the measured hardware power is almost 3 times that of the

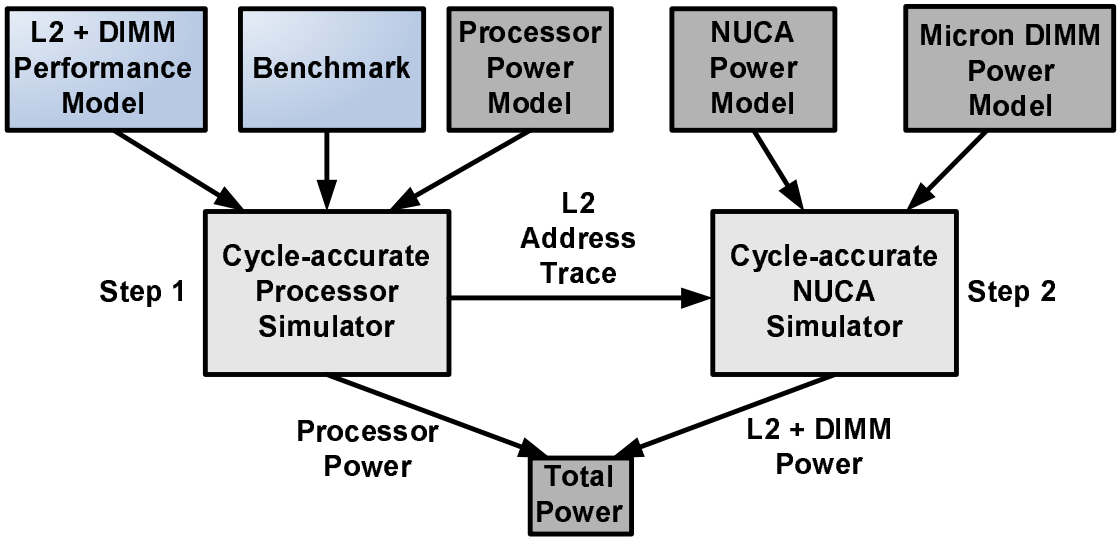


Figure 3.1: Baseline Architectural Power Models

initial architectural power estimate). Using a detailed power breakdown obtained from a validated Register Transfer Level(RTL) power model of the same processor (which has 6% average error), we identify, classify, and quantify the major sources of inaccuracy in the architectural power models. Using feedback from the hardware and RTL models, we reduce the accuracy gap between the baseline architecture power model and the hardware. Despite poor absolute accuracy, the baseline architectural power models have good *relative accuracy* to begin with (10%) and in the improved architectural power models the relative accuracy improves with absolute accuracy (to 3%).

3.1 Architectural Power Models

Figure 3.1 describes our baseline architectural power modeling methodology for TRIPS, which matches the state-of-the-art methodology in academia for building architectural power models. First, we run the benchmark binary on a cycle-level

simulator that models the TRIPS processor core alone (excluding the L2). This simulation produces access counts of various microarchitectural structures in the core and a trace of all generated L2 addresses. Second, we run this L2 address trace through a cycle-level NUCA L2 simulator to obtain access counts of the structures in the L2 subsystem. We follow this two-step methodology to ease the process of validation and correlation with RTL power models. Since RTL simulation speed is orders of magnitude lower than architectural simulations, we adopt a two-step RTL simulation methodology to run reasonable benchmarks. We use the same unified L2 and DIMM model for both architectural and RTL simulators of the processor core.

The TRIPS base architectural power is derived via commonly used power modeling methodologies. We build CACTI [110] models for all major structures such as caches, SRAM arrays, register arrays, branch predictor tables, load-store queue CAMs, and on-chip network router FIFOs to obtain a per-access energy for each structure. This per-access energy combined with the access counts from the architectural simulator provides the overall energy dissipated in these structures. The power models for integer and floating point ALUs and the clock tree are derived from Wattch [12] using linear technology scaling from the built-in 350nm technology of Wattch. We model global clock drivers, global clock tree interconnect, pre-charge transistors and pipeline latches as part of the clock tree. We estimate the number of latches in each tile based on a detailed microarchitecture specification. The per-latch capacitance estimates are derived from Wattch as well.

Analytical estimation of combinational or control logic power is challenging at the architectural level. As one of the key contributions of this work, we propose simple rules-of-thumb to estimate control logic power. We assume that the total gate count for a TRIPS tile is a constant (about four) times the number of latches in the tile. Table 3.1 shows the gate-to-latch ratio of various TRIPS tiles based on a detailed analysis of the post-synthesized netlist. We observe that the rule,

Tile Name	G/L Ratio
Chip-to-Chip Controller	3.56
DMA Controller	14.23
External Bus Controller	9.50
Instruction Cache	2.23
SDRAM Memory Controller	3.96
Data Cache and Load-Store Queues	5.54
Execution Tile (Issue Logic, ALUs)	8.57
Global Control Tile	4.15
L2 Cache Banks	3.45
Register Tile	5.19
L2 Router Tile	4.41

Table 3.1: Control Logic Ratios

despite being simple, holds for most of the TRIPS tiles with notable exceptions being DMA (Direct Memory Access Controller), EBC (External Bus Controller) and the Execution Tile, which are control-logic intensive and have relatively less storage when compared to other tiles. Excluding the DMA and the EBC, which are not used in this study, the arithmetic mean of the gate-to-latch ratio is 4.56 with a standard deviation of 1.8. Although such simple rules must be fine-tuned before applying to other architectures, the key take-away is that applying even simple rules-of-thumb for control logic improves the accuracy of architectural power models significantly.

Given the gate counts, we use another rule-of-thumb, similar to equation (2) in [79], to estimate the total gate capacitance. The value of C_{avg}, a high-level estimate of the average gate capacitance, is obtained from the documentation of IBM 130nm ASIC process. Using these gate capacitance estimates and models based on Rent's rule [106], we estimate the control logic and interconnect access energies of the various tiles. These energies combined with various event counts of the tiles provide the total control logic and interconnect energies. We build leakage power models for all array structures based on HotLeakage [127], and leakage models for non-array structures are based on gate-count estimates and average transistor

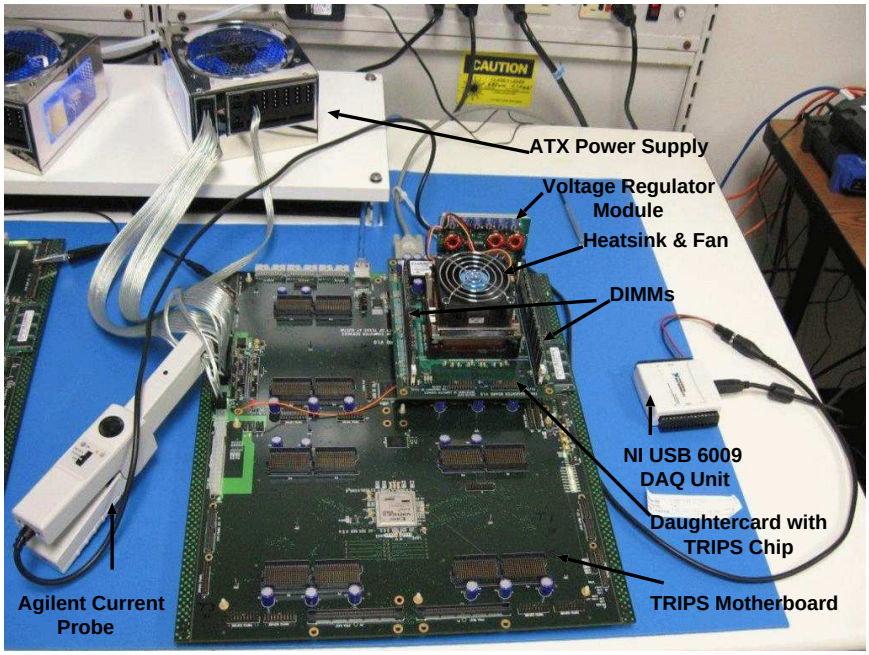


Figure 3.2: TRIPS Circuit Boards and Power Measurement Infrastructure.

density estimates. We use an analytical power model for the DIMMs obtained from Micron for both architectural and RTL power models [71].

3.2 Power Model Validation

We now describe how we use real hardware power measurements on the TRIPS prototype chips and detailed RTL power models to validate the baseline architectural power models (described above).

3.2.1 Hardware Power Measurement

Figure 3.2 shows a photograph of our power measurement infrastructure and the TRIPS prototype system. Each TRIPS motherboard can support up to four TRIPS chips. Each chip is mounted on the motherboard via a daughtercard. The daugh-

tercard contains one Voltage Regulator Module (VRM) that steps down the 12 V ATX power supply to 1.5 V for the TRIPS chip, a heat-sink and fan assembly, and two 1-GB DDR SDRAM DIMMs. The DIMMs receive a 2.5 V power supply from the regulator. We use the following system parameters for our validation experiments: 1.5V chip power supply, 366 MHz chip clock frequency and 133/266 MHz for the DIMMs. We use an Agilent 1146A clamp-on current probe for measuring the power consumption of the TRIPS daughtercard. The voltage output of the probe is sampled by a National Instruments (NI) USB 6009 Data Acquisition System at the rate of 10 KHz and is logged to a Linux Workstation using the NI Data Logger program.

Using a set of carefully-designed experiments, we isolate the power consumed by the TRIPS motherboard and the DRAM DIMMs from the measured hardware power. We remove the TRIPS daughtercard from the motherboard and note the measured power. This power is 3.3 Watts and is attributed to the TRIPS motherboard alone. To measure the power consumed by the DDR DIMMs, we take two power measurements: one when the DIMMs are unplugged from the daughtercard and a second when the DIMMs are plugged in and a sequence of random memory reads and writes are being performed. The difference between these two readings, about 3.6 Watts, is attributed to the DIMMs. After this isolation, we also account for 90% rated conversion efficiency of the VRM when converting from 12V to 1.5V needed for the TRIPS chip. This provides the measured hardware power for the TRIPS chip. Finally, we attempt to isolate the clock tree portion of the total power. To this end, we run the chip in the idle mode at 100 and 366 MHz and measure the dissipated power. Since the chip is idle in both cases, we use the linear dependence between clock frequency and power with these two data points to isolate the clock tree power. We interpolate the clock tree power model and confirm that it matches the measured power at 200 MHz. In total, we estimate the clock

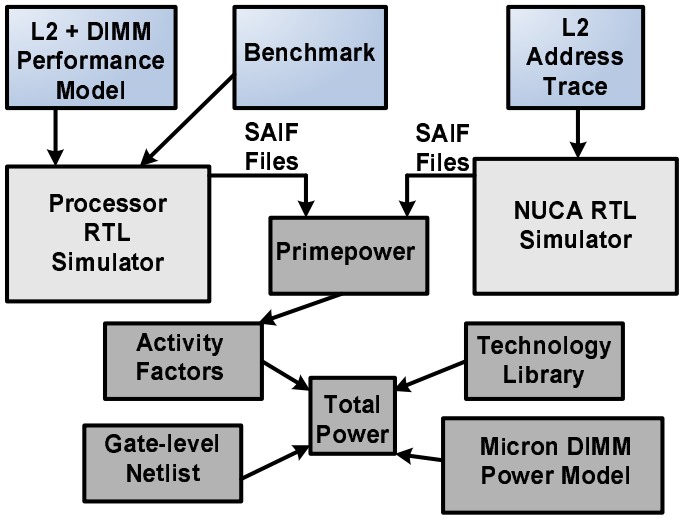


Figure 3.3: RTL Simulation Methodology

tree to consume 18.3 Watts at 366 MHz.

We run a set of microbenchmarks, which have key loops extracted from the SPEC CPU2000 [43], for our validation experiments. We run these benchmarks on real TRIPS hardware, TRIPS architectural simulator, and on TRIPS RTL simulator. We use these microbenchmarks to mitigate the slow RTL simulation speeds. We then compare the measured hardware power to the estimated architectural power. We find that on an average the architectural power model underestimates the total power by about 65%.

3.2.2 RTL Power Models

Having observed the inaccuracy of our baseline architectural power models, we use a detailed RTL power model of the TRIPS design to identify, quantify and rectify the various sources of inaccuracy. Figure 3.3 describes our RTL power modeling methodology. First, we run the benchmark on a Synopys VCS [108]-based processor-level RTL simulator, which uses a pre-synthesized RTL netlist of the design. This simulation produces a set of Switching Activity Interchange Format (SAIF) files.

Next, we feed the L2 address trace obtained from the architectural simulations (Figure 3.1) to the NUCA RTL simulator to obtain the L2 cache SAIF files.

The SAIF files represent the toggle counts of the various nodes in the pre-synthesized netlist of the design. We use Synopys Primepower [107] to propagate these toggle counts to a post-synthesized, gate-level netlist and obtain an average switching activity (is benchmark-dependent) for each tile in the core and the L2 subsystem. Combining this average activity factor for each tile with the total capacitance estimate from the gate-level netlist and the IBM Standard Cell library, we estimate the average dynamic power. We obtain the capacitance of the gates and global clock buffers from IBM cell library. We estimate the interconnect capacitance using Rent's Rule as published in [106]. We also obtain the PFET and the NFET widths of various IBM cells from the library to estimate the leakage power.

3.2.3 Validation Results

Figure 3.4 compares the power estimates of the baseline architectural models (the bar labeled Base) to RTL estimates (labeled RTL) and hardware power (labeled HW). We observe that the baseline architectural power model underestimates the total power by 65% compared to the hardware power. We also find that the RTL power estimates are much more accurate and within 6% of the measured hardware power. As shown in Table 3.2, we break down the RTL power estimates into power categories not visible via hardware measurement to identify the root source of errors in the baseline architecture power model and to derive improvements to the power model.

Sources of Inaccuracy: Table 3.2 shows a breakdown of the average power estimate of the microbenchmarks into major categories like dynamic power due to combinational logic, array structures, ALUs, interconnect, clock tree including the latches and clock buffers, leakage power and power dissipated in the DIMMs along

Category	Arch(W)	RTL(W)	Fraction of Total Error
Control Logic + Arrays + ALUs	1.91	5.94	0.21
Interconnection	0.47	1.27	0.04
Clock Buffers	0.13	3.30	0.16
Latches	4.21	14.56	0.54
Leakage	1.36	1.91	0.03
DIMMs	3.44	3.61	0.01
Total	11.52	30.84	1.00

Table 3.2: Detailed Power Breakdown

with the fraction of the total error caused by each category in Column 4. Using this breakdown, we focus our attention on the major sources of error: latches, clock buffers, and control logic power. Errors in these categories of power can stem from underestimates in counts (latch counts, gate counts, etc) and underestimates in capacitances.

Latch Counts: We estimate the number of latches based on microarchitecture specifications for each tile in the TRIPS design. Upon a detailed analysis, the architectural model underestimates the latch counts by 53%. The key reasons for this underestimate are twofold. First, certain structures in the TRIPS design like load-store queue Content-Addressable Memories (CAMs), and FIFOs, which were expected to be custom SRAM arrays, had to be implemented out of discrete latches due to a lack of suitable dense structures in the ASIC library. These latches, which account for 40% of the actual latch count, are not included in the initial architectural estimates. The D-Tile (data tiles) in the TRIPS design is the most prominent source of this error - each D-Tile contains a maximally-sized load/store CAM with 256 entries.

After accounting for these additional latches, the architectural latch estimates still underestimate the latch count by 13%. The second key reason for the

initial underestimates is as follows. The microarchitectural specifications, that form the basis for our initial latch counts, invariably change during actual RTL implementation and the specifications are not kept up to date to reflect the actual design. Additionally, when preparing specifications, designers typically do not anticipate the use of temporary latches that are needed during actual design. This mismatch between the tile specification and the actual design causes the additional mismatches in latch counts. In our experience, although we find this specification mismatch in most tiles we find it to be severe in E-Tile and the M-Tile in the TRIPS design. Since the TRIPS design has many instances of the E-Tile (32) and M-Tile (16), even small underestimates on a single instance of such tiles quickly adds up at the chip level.

Latch Capacitance: The architectural latch capacitance estimates come from Wattch, after suitable technology scaling and the RTL estimates come from the IBM Standard Cell library. The architectural models underestimate the per-latch capacitance by 40%. The errors in latch counts and latch capacitances contribute 54% of the overall error (Row 4 in Table 3.2).

Clock Buffer Counts: The base architectural models underestimate the number of clock buffers in the design by 33%. Additionally, IBM requires Level-Sensitive Scan Design (LSSD) based latches for testability [24]. Due to this requirement, the final TRIPS clock tree has many clock-splitters [24] (about 30K), which are not accounted for in the initial architectural power estimates. This mismatch in the number of clock-splitters causes an average error of about 16% in the total power estimate (Row 3 in Table 3.2).

Control Logic Power: Estimating control logic power at the architectural level is challenging because of the inherent difference in the level of abstraction between architectural and RTL models [59]. We estimate the control logic capacitance based on rules-of-thumb for gate-counts and gate capacitances, which are described

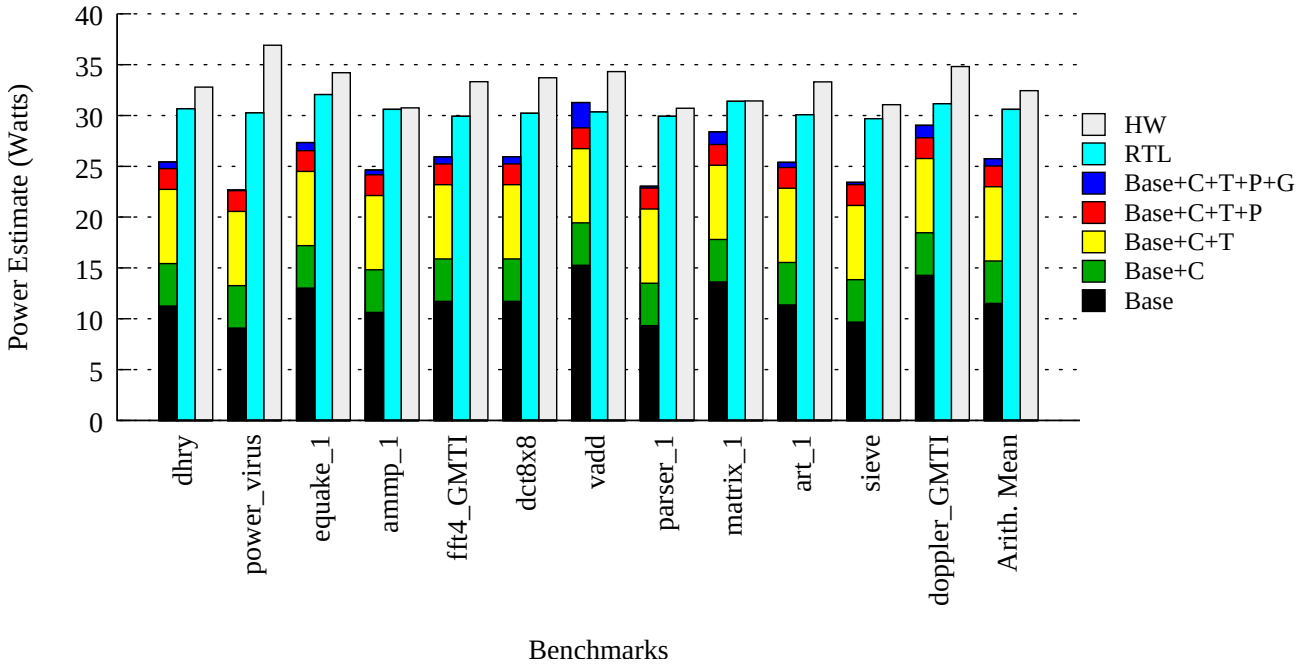


Figure 3.4: TRIPS Estimated and Measured Power.

in Section 3.1. A detailed analysis shows that the capacitance estimates based on rules-of-thumb underestimate the actual capacitance by 35%. We use an event-based model in the architectural simulator to estimate the activity factor of control logic. This event-based model, although accurate, is clearly not a substitute for the more accurate bit-level switching activity factors from the RTL power models. These differences combined cause a 21% error attributed to both the control logic and the array power (Row 1 in Table 3.2).

3.3 Improved Architectural Models and Relative Accuracy

Using the above analysis, we evaluate a series of architecture power models that incrementally fix classes of errors to improve accuracy. Figure 3.4 shows the power estimates of the architectural power models for the microbenchmark suite. For rea-

sons of clarity, the figure only plots 12 of the 24 benchmarks in the suite, and the arithmetic mean is shown for all 24 benchmarks. Appendix A presents the validation results for all 24 benchmarks in our suite. In Figure 3.4, for each benchmark, the graph shows three bars: architectural power estimates, RTL power estimates, and measured hardware power. The architectural bar has five segments, each representing a different architectural power model. **Base** represents our baseline architectural power model described in Section 3.1, which underestimates total power by 65%.

In the **Base+C** model, we fix most of modeling errors introduced by latch and clock-splitter counts. However, we do not include the underestimate of latches (13%) due to differences between the specifications and the RTL, and the underestimate of buffers in the clock tree (33%). The **Base+C+T** model fixes all the technology scaling errors in the latch capacitance and clock buffer capacitance. In the **Base+C+T+P** model, we include the additional 13% latches and 33% clock buffers to fix all errors in the clock tree power. In the **Base+C+T+P+G** model, we replace the gate count estimates for various tiles based on rules-of-thumb by the actual gate counts of the tiles.

The **Base+C** model, which fixes the modeling errors related to the clock tree, reduces the overall error by 13% compared to **Base**. Fixing the technology scaling errors in the **Base+C+T** model reduces the overall error by an additional 22% compared to the **Base+C** model. The **Base+C+T+P** model with a perfect clock tree model reduces the overall error by 6%. Finally, the actual gate counts in the **Base+C+T+P+G** model reduces the error by a small amount of 2%. The marginal reduction in error in the **Base+C+T+P+G** model is due to two reasons: (1) the original rules-of-thumb for control logic capacitance estimation are reasonably accurate, and (2) the actual gate counts for a few tiles are less than the rule-of-thumb estimates, which tends to negate the accuracy improvement of actual gate counts.

Thus, power estimates obtained using the **Base+C+T+P+G** model are within 21% of measured hardware power for the microbenchmark suite. We also apply the **Base+C+T+P+G** models to the EEMBC suite and observe that on an average the architectural estimates are within 24% of hardware power. We further fine-tune the **Base+C+T+P+G** models, by mainly experimenting with different scaling factors for control logic switching activity, to bring the final accuracy to within 15% of measured hardware power. This scaling factor is an attempt to address the lack of fine-grained bit-level switching activity in the architectural simulator. Such support could be added to the simulator, but only at the expense of increased simulation times. Further differences in the power models for control logic switching activity, interconnects, leakage, and the DIMMs cause the remaining discrepancy between modeled and measured power.

Our experience shows that applying commonly used power modeling methodologies to TRIPS results in a more than a factor of two underestimate in absolute power consumption. The underestimate stems from errors in estimating latch count, gate count, clock tree, and logic gate capacitance. While refining these estimates with feedback from the final design improves the accuracy to within 21%, yet more empirical data from the final design is needed. These results point to the difficulty in building architecture power models from the ground up and provide guidance on where to focus attention in architecture-level power models: better clock tree models, accurate technology models and better models for unstructured, random logic.

Our experience also indicates that power-modeling tools like Wattch [12] and CACTI [110] are very valuable in early-stage architectural power modeling. However, understanding the limitations of such tools is critical in achieving better accuracy. For example, although Wattch and CACTI have been validated against real industrial designs, such validation efforts have been performed at older technol-

ogy nodes. One must be careful when applying such tools to novel architectures such as TRIPS, and to newer process technologies. Considering that companies like Intel have introduced drastically new manufacturing technologies [72], more frequent validation efforts such as ours are needed to make these tools more useful and relevant to current technologies. Additionally, the CACTI [110] tool is extremely valuable in modeling regular array structures such as caches, SRAM arrays, and register files. However, such tools might be less useful in modeling unstructured random logic, and custom-designed array structures, which tend to be heavily circuit-optimized in industrial designs, and thus, cannot be modeled accurately with the analytical models of CACTI.

While estimating absolute power consumption is particularly difficult, we did find that the relative power from the architecture models tracked the power measured in hardware reasonably well across the programs in our benchmark suite. We measure this *relative* accuracy by measuring relative increase or decrease in power on a benchmark from the arithmetic mean across all the benchmarks for both the power models and the hardware. The results show that all the architectural power models track the hardware results very closely, and that on average **Base** tracks the hardware to within about 10%. The average relative accuracy improves to within 3% with **Base+C+T+P+G** model. This observation bodes well for architecture studies that seek to compare relative power consumption across different applications and architecture configurations as long as the modeling, abstraction, and technology modeling errors in the power models are shared in common mode across the configurations.

3.4 Lessons

Our power model validation work in this dissertation has provided us with some invaluable lessons, which are as follows. First, developing accurate architectural

power models is a very challenging exercise, especially for novel architectures such as TRIPS. Designers must exercise caution when applying pre-existing power models to novel architectures, and new technology nodes. Understanding the true limitations of tools like CACTI and Wattch is very critical to the final model accuracy. Second, while achieving absolute accuracy with architectural power models is very challenging for any architecture, we observe that architectural power models fare well in relative accuracy–the relative increase or decrease in power across microarchitectural configurations and benchmarks. This observation bodes well for architectural power models, which are typically used for early-stage design explorations.

Chapter 4

Performance and Power Comparison Methodology

This chapter describes our experimental methodology for comparing the TRIPS and the TFlex microarchitectures with ARM and PowerPC platforms. We justify the choice of these platforms for our comparison and describe the simulators used for the different platforms. We outline our efforts to normalize the respective power and performance models to ensure a fair comparison of the architectures. We list the benchmark suites used for our comparison. This chapter also discusses the microarchitectural parameters of all the platforms included in our study: ARM, PowerPC, TRIPS and TFlex.

4.1 Experimental Platforms

We use a cycle-level, execution-driven simulator, called TFlex simulator [57], capable of simulating both the TRIPS and the TFlex microarchitectures for our comparison experiments. Although the TRIPS microarchitecture is multi-threaded, and can execute up to four threads [92], we only run the TRIPS processor in the single-

threaded mode running a single application thread.

To assess the relative advantages and disadvantages of TRIPS and TFlex microarchitectures, we use two microarchitectures, Power4 [54] and XScale [20], as our comparison platforms. We model the Power4 microarchitecture with a trace-driven simulator called Turandot, which incorporates parameterizable processor power models called PowerTimer [73]. We use "version II" of the Turandot/PowerTimer models for our experiments. We leverage the XTREM performance and power simulator for modeling the XScale microarchitecture [20].

Choice of Experimental Platforms: We compare TRIPS and TFlex against Power4 and XScale for the following reasons. First, these platforms represent two categorical extremes in the performance and power spectrum – a high-end system optimized for performance and a low-end system optimized for power, providing a good range for our comparison to the EDGE architectures. Second, as mentioned in Chapter 3, the TRIPS architectural power models have been validated against hardware power measurements and accurate RTL power models [36]. Thus, the baseline TRIPS power models are based on the 130nm node used for the TRIPS prototype implementation [36]. The original Power4 processor has been implemented at the 180nm technology [54]. The Turandot simulator and its power models have been validated against low-level RTL models of the Power4 processor [73]. Similarly, the XTREM power models have been validated against real hardware [20]. Since all platforms have been through some form of validation, our methodology eliminates many uncertainties of unvalidated architectural power models. While it would be interesting to compare TRIPS and TFlex power and performance to other high-performance, power-optimized platforms like Intel Core-2 [81] or AMD Opteron [23] microarchitectures, we avoid such an attempt because using real hardware for one platform and a simulator for another injects many uncontrollable unknowns into our experiments. Furthermore, to understand the relative advantages and disad-

vantages of TRIPS and TFlex compared to these new platforms, detailed access to fine-grained power and performance measurements is essential and is not readily available outside of simulation.

4.2 Power Models

A comparison of disparate platforms such as TRIPS, TFlex, Power4 and XScale is challenging because they all implement different instruction sets, and are based on different process technologies and design methodologies (custom vs. ASIC). To mitigate these challenges and to enable a fair comparison, we tune the baseline Power4 and XTREM power models to reasonably match that of TRIPS. As a first step, we ensure the same process technology (65nm), supply voltage (1.2 Volt) and core frequency (2 GHz) in all platforms. Although TRIPS, TFlex, and Power4 could potentially operate at higher frequencies at 65nm we choose 2 GHz to be a reasonable operating frequency for the XScale platform. Additionally, we ensure that all platforms use the same main memory latency (cycles). The baseline TRIPS power models at 130nm and Power4 and XTREM power models at 180nm are suitably scaled down to the 65nm using linear technology scaling. The Turandot simulator accepts the number of fan-out-of-4 (FO4) delays of the design as a parameter to adjust its performance and power models. Assuming a FO4 delay value of 22 picoseconds (which is equal to $0.3 * 65$nm (feature size)) and the desired 2 GHz frequency, we use 23 FO4 delays in the simulator.

4.2.1 TRIPS Power Models

The TRIPS power models are based on the **Base+C+T+P+G** power models (described in Section 3.3). We incorporate these power models into the TFlex simulator, and place the simulator in the TRIPS mode for our comparison. The power model uses CACTI [110] models for all major structures such as instruction

and data caches, SRAM arrays, register arrays, branch predictor tables, load-store queue CAMs, and on-chip network router FIFOs to obtain a per-access energy for each structure. These per-access energies combined with access counts from the architectural simulator provides the energy dissipated in these structures. The power models for integer and floating point ALUs are derived from both Wattch [12] and the TRIPS design database.

The combinational logic power in various microarchitectural units is modeled based on detailed gate and parasitic capacitances from the TRIPS design databases, and activity factor estimates from the simulator. The **Base+C+T+P+G** models do not include clock gating support to conform with the TRIPS prototype, which does not implement clock gating [98]. However, we add clock gating support (explained below) to the TFlex simulator for all of our experiments.

4.2.2 Turandot and ARM Power Models

The baseline Power4 models use detailed circuit-level simulations of various circuit macros of the processor. The XTREM models use the models (version 1.0 [123]) for various array structures. For a consistent comparison, we replace these in-built power models for all array structures in Turandot and XTREM with CACTI power models for the same structures, ensuring that TRIPS, XScale and Power4 all utilize CACTI models for all array structures. To ensure further consistency, we replace built-in ALU power models of Power4 and XTREM with those of the TRIPS model with one key difference. The Power4 microarchitecture supports a fused-multiply-add floating point instruction, which performs two operations (a multiply and an add) in one instruction. Hence, we rely on the baseline Turandot FPU power models for Power4 to correctly model this fused instruction. The combinational logic power in Turandot is modeled similar to TRIPS: capacitances come from the Power4 design database, and activity estimates come from the Turandot simulator.

4.2.3 More Normalization Efforts

L2 Caches: The three simulators support different L2 cache organizations. While TRIPS implements a Non-Uniform Cache Access (NUCA) L2 cache [58], Turandot and XTREM support a unified, banked L2 cache. In all the experiments of this chapter, we ignore the L2 power – both dynamic and static – to focus solely on the efficacy of composability and EDGE architectures. Since the focus of this dissertation is on the advantages and disadvantages of EDGE architectures vis-a-vis conventional architectures, and not NUCA versus non-NUCA cache organization, we assume the above simplification for our experiments. However, for the sake of completeness we do present the overall chip power estimates for all platforms in Chapter 5.

Clock Tree: We model global clock drivers, global clock tree interconnect, pre-charge transistors and pipeline latches as part of the TRIPS clock tree. The TRIPS clock models are based on validated latch count estimates from the TRIPS design and accurate per-latch capacitances from the design library. Since the TRIPS prototype chip does not implement clock gating [98], we add the clock gating support to the TRIPS simulator for our experiments. Our clock gating model keeps track of utilization factors in all microarchitectural units. If a unit is active during a clock cycle, our model attributes the full ungated clock power to that unit. However, if a unit is idle during a cycle our model does not completely clock-gate that unit to be realistic. Instead, our model assigns a fixed percentage (10%) of the ungated clock power to the idle unit. Our power models also clock gate the OPN routers in the TRIPS microarchitecture.

The Turandot clock power models are similarly based on measurements on circuit-level macros [78]. However, the baseline Turandot clock model has two key differences with the TRIPS model. First, the Turandot model assumes perfect clock gating – any gateable unit is completely clock gated. We adjust this clock gat-

ing style to match the realistic gating style of TRIPS (fixed 10% overhead for idle units). Second, the power consumption of higher levels of the clock tree (clock buffers, and splitters) is not modeled in the Turandot model. Assuming the same clock-tree design style across the two systems, and using the clock tree power models from the TRIPS design, we add a fixed percentage overhead to the baseline Turandot clock power. Our analysis of the TRIPS clock tree power shows that the clock-tree overhead with varying clock-gating patterns is around 20% for the TRIPS microarchitecture. However, to be conservative, we only add around 10% clock-tree overhead to the Turandot models. We also add clock gating support to the XTREM simulator. The simulator keeps track of activity in the pipeline latches, and all idle latches and the associated last level of clock buffers are assumed to be clock-gated. The rest of clock tree is always left on.

Leakage Models: We obtain detailed transistor width estimates for all the TRIPS tiles from the TRIPS design database. We use the sub-threshold current values predicted by Zhao et al. [128]. For gate-leakage current, we use gate-oxide thickness values from [128] and the work that relates gate-oxide thickness to gate-leakage density [19]. These unit-width leakage current values combined with transistor width estimates provide a simple area-based leakage power models for the TRIPS processor. The total leakage power of the TRIPS system is estimated as the sum of leakage powers of all the constituent tiles in a TRIPS processor. Although the TRIPS prototype chip consists of two TRIPS processors, we assume that only one TRIPS processor is used to run the benchmark, and that the other TRIPS processor is completely turned off and it does not contribute to any dynamic or static power. To normalize the leakage power models across all platforms, we replace the built-in Turandot model, which estimates leakage power as a fixed percentage of dynamic power, and XTREM models with our area-based leakage models after adjusting for the area estimates of the Power4 and the XScale core.

4.2.4 TFlex Power Models

As mentioned in Section 4.1, we use a cycle-accurate simulator called the TFlex simulator, capable of simulating both the TRIPS and the TFlex microarchitectures, for our experiments. The TFlex power model is incorporated into the TFlex simulator in a manner similar to Wattch [12], and is derived from the validated TRIPS power models, the **Base+C+T+P+G** models, described in Section 3.3. The power model for an individual TFlex core consists of components from the TRIPS power model and clock tree power scaled by the ratio of TRIPS to TFlex core latch counts. The power model uses CACTI [110] models for all major structures such as instruction and data caches, SRAM arrays, register arrays, branch predictor tables, load-store queue CAMs, and on-chip network router FIFOs to obtain a per-access energy for each structure. These per-access energies combined with access counts from the TFlex simulator provides the energy dissipated in these structures. The power models for integer and floating point ALUs are derived from both Wattch [12] and the TRIPS design database. The combinational logic power in various microarchitectural units is modeled based on detailed gate and parasitic capacitances from the TRIPS design databases, and activity factor estimates from the simulator.

The TFlex power models employ the same clock gating models as described in Section 4.2. TFlex has two additional enhancements over the TRIPS microarchitecture. First, each TFlex core is dual-issue processor compared to the single-issue E-Tile in TRIPS [57]. Second, each TFlex core enjoys twice the OPN bandwidth compared to the E-Tile in TRIPS by using two sets of channels, FIFOs, etc. We incorporate these enhancements into our TFlex power models by including the additional ports in the reservation station, additional ALU and instruction selection logic needed for dual issue, and the additional channels and FIFOs required in the OPN router.

The leakage power models for TFlex are based on core area estimates and

Parameter	Configuration
Hand-optimized Benchmarks	2 Kernels, 6 EEMBC benchmarks
Compiled Benchmarks	4 microbenchmarks, 6 kernels, 23 EEMBC and 15 SPEC CPU 2000 benchmarks currently supported (7 Integer, 8 FP), simulated with single simpoint of 100 million instructions [100]
Process Technology	65 nm
Supply Voltage	1.2 volts
Clock Frequency	2 GHz
Memory Frequency	667 MHz

Table 4.1: Benchmarks and Experimental Configuration

unit-area based leakage models from Section 4.2. The total leakage power of the TFlex system is estimated as the sum of leakage powers of all cores in the composed system. We assume that when two TFlex cores are composed into a logical processor, the total power consumption is the sum of the dynamic and the static power dissipated in those two cores; we assume that unused TFlex cores can be completely power- and clock-gated.

4.3 Experimental Configuration

In this section, we describe our experimental configuration used for the comparison study: list of benchmarks, and the microarchitectural configurations for various platforms, and a detailed area estimate for each platform at 65nm technology node.

4.3.1 Benchmarks

Table 4.1 lists the various benchmarks used in this study and the experimental configuration. Although processors built in 45nm technology, and even 32nm technology, are being shipped in 2010, we assume the 65nm process technology for our

experiments. We expect our experimental results to hold if more recent process technologies were used. The key difference would be that 45nm process would enjoy reduced leakage power levels because of transition to metal gates and high-K dielectric material as in [72]. The benchmarks fall under two categories. The first category of benchmarks are compiled by the TRIPS compiler, and are additionally hand-optimized. These include 2 kernels and 6 EEMBC [2] benchmarks. The second category of benchmarks are purely compiled by the TRIPS compiler. These include 6 kernels, 4 microbenchmarks, 23 EEMBC benchmarks, and 15 SPEC [43] benchmarks with a single simpoint of 100 million instructions [100]. We include both categories of benchmarks because hand-optimized benchmarks typically exhibit better performance on TRIPS than compiled ones. All these benchmarks are compiled with the XLC compiler on an AIX system for the Power4 platform. Code for the ARM platform is generated using a gcc cross-compiler.

For SPEC benchmarks, we generate the instruction traces corresponding to the same simpoints on an AIX machine, and feed these traces to the Turandot simulator. We were forced to exclude the rest of SPEC suite for one or more of the following reasons:

- The TRIPS compiler infrastructure, as of late January 2010, currently fails to generate reliable binaries for four benchmarks (197.parser, 200.sixtrack, 255.vortex and 254.gap);
- The ARM simulators (functional and performance) fail output checks for the SIMPOINT regions on others (253.perlbmk and 301.apsi).

4.3.2 Microarchitectural Parameters

We tabulate the microarchitectural parameters of TRIPS, XScale and Power4 in this section along with a detailed area estimates of various microarchitectural structures in 65 nm process technology. Tables 4.2 and 4.3 compare the microarchitectural

features of TRIPS and TFlex to that of the ARM and Power4 microarchitectures respectively. The table also shows the corresponding area estimates for each microarchitecture at 65nm. The area estimates for TRIPS are obtained from the TRIPS design database; the Power4 area estimates are from the Turandot simulator, and the ARM estimates come from [20]. The original area estimates for each platform has been suitably scaled down to 65nm using linear technology scaling with a fixed 10% scaling overhead. Although the L2 cache capacity is the same across all platforms, the L2 cache organization is very different: while TRIPS uses the NUCA L2 caches ARM and Power4 use banked L2 caches. As mentioned in Section 4.2 we ignore the power consumed by the L2 cache banks for our comparison, and we also ignore the differences in their area estimates. We argue that this is a reasonable assumption because it is relatively straight forward to implement the same L2 cache organization on all platforms having similar areas.

The original Power4 processor can issue up to eight instructions [54] whereas the Power4 in this study has been scaled down to issue up to four instructions. The scaled-down Power4 avoids the high complexity and the power overhead of the original Power4. The number of ports in the register files, and issue queues in Power4 have been suitably adjusted to match the new issue width of four. We have also disabled the multi-threading functionality present in original Power4 processor [54], and have suitably adjusted the area estimates, the performance and the power models in the Turandot simulator.

Similar to TRIPS, the TFlex power models have been suitably scaled to the 65nm process technology to ensure a fair comparison across all platforms: TRIPS, TFlex, ARM and Power4. We use the exact set of simulators, benchmarks, voltage and frequency settings, and power models for the other platforms (TRIPS, ARM and Power4) as mentioned in Sections 4.1 and 4.2. The TFlex system utilizes the 4MB NUCA L2 cache similar to the TRIPS system. However, we only use the power

Structures	TRIPS (16-issue)		Arm (1-issue, in-order)		Power4 (4-issue)	
	Size	Area (mm^2)	Size	Area (mm^2)	Size	Area (mm^2)
Fetch (B.Pred., I-cache)	Block Predictor (64Kbit) [87] 80KB I-cache	 2.1	BTB in [20] 32KB I-Cache	 0.69	Predictor in [54] 64KB I-cache	 1.72
Reg. Files	512 entries	0.83	16 entries	0.12	80-entry INT 72-entry FP	0.63
Exec. resources (issue window, ALUs)	1024 entries SRAM-based issue window INT(16),FP(16)	10.82	Single Issue 1 INT-ALU, 1 MAC	0.36	82-entry CAM-based issue window (partitioned) 1 combined, INT+MEM Unit 1 FP, 1 Branch 1 Logical	2.86
L1 D-cache (D-cache, LSQ)	32KB D-cache 1K-entry LSQ	9.20	32 KB D-cache Write and Fill Buffers	0.62	32KB D-cache Two 36-entry queues	2.99
Routers	OPN [38]	3.03	N/A	0.0	N/A	0.0
L2 Caches	4-MB NUCA Cache	–	4-MB Banked Cache	–	4-MB Banked Cache	–
Total		25.98		1.79		8.2

Table 4.2: TRIPS Microarchitecture Comparison

Structures	TFlex-1 (2-issue)		Arm (1-issue, in-order)		Power4 (4-issue)	
	Size	Area (mm^2)	Size	Area (mm^2)	Size	Area (mm^2)
Fetch (B.Pred., I-cache)	Block Predictor in [57] 8KB I-cache	 1.06	BTB in [20] 32KB I-Cache	 0.69	Predictor in [54] 64KB I-cache	 1.72
Reg. Files	128 entries	0.25	16 entries	0.12	80-entry INT 72-entry FP	0.63
Exec. resources (issue window, ALUs)	128-entry SRAM-based issue window 2-INT ALU, 1-FP ALU	1.06	Single Issue 1 INT-ALU, 1 MAC	0.36	82-entry CAM-based issue window (partitioned) 1 combined, INT+MEM Unit 1 FP, 1 Branch 1 Logical	2.86
L1 D-cache (D-cache, LSQ)	8KB D-cache 44-entry LSQ	0.99	32 KB D-cache Write and Fill Buffers	0.62	32KB D-cache Two 36-entry queues	2.99
Routers	Dual OPN [57]	0.27	N/A	0.0	N/A	0.0
L2 Caches	4-MB NUCA Cache	–	4-MB Banked Cache	–	4-MB Banked Cache	–
Total		3.63		1.79		8.2

Table 4.3: TFlex Microarchitecture Comparison

dissipated in the TFlex cores for our comparison studies.

The TFlex microarchitecture supports various block mapping policies aimed at balancing concurrency and communication across the distributed substrate [88]. We provide a brief overview of these block-mapping policies in Chapter 7, and evaluate the combined performance and power characteristics of some of these policies. In our comparison experiments involving TFlex, we use "deep" block-mapping strategy [88], which maps each TFlex block to single owner core. In contrast, the original TFlex implementation uses the "flat" mapping strategy [57] that stripes each TFlex block across multiple participating cores.

Chapter 5

Performance and Power Comparison Results

This chapter presents the results for performance and power comparison of all the platforms: TRIPS, ARM, Power4, and TFlex. The simulators, power models, microarchitectural configurations, and benchmarks used in these experiments are described in Chapter 4. To simplify the discussion, we first compare TRIPS to Power4 and ARM in terms of metrics like performance, power, energy-delay, and energy-delay-squared product. Next, we present a detailed structure-by-structure power breakdown analysis of TRIPS and Power4. This analysis provides insights into the overheads of the respective systems. Finally, we compare the TFlex microarchitecture to all the previous platforms, and discuss the effects of composability on performance and energy efficiency.

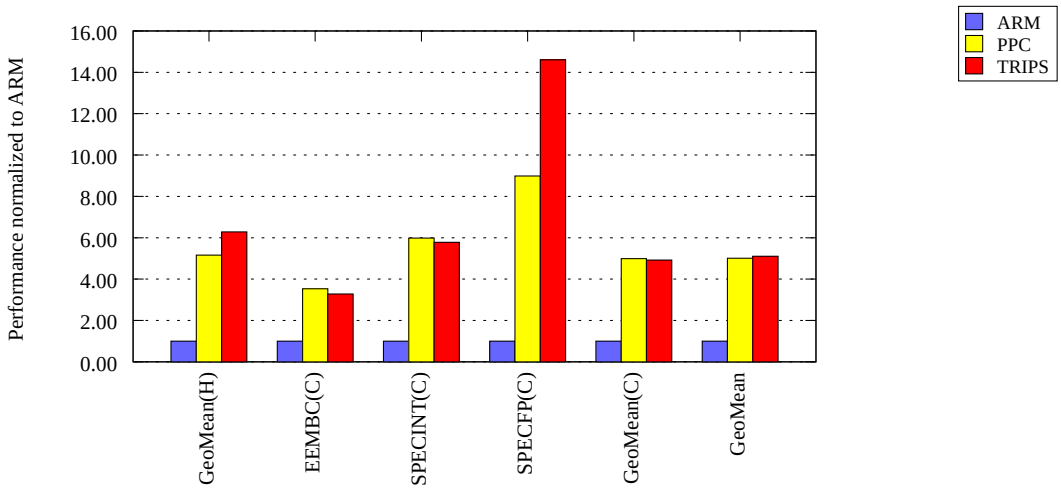


Figure 5.1: TRIPS vs. Other Platforms: Performance Comparison @ 1.2V, 2 GHz, 65nm

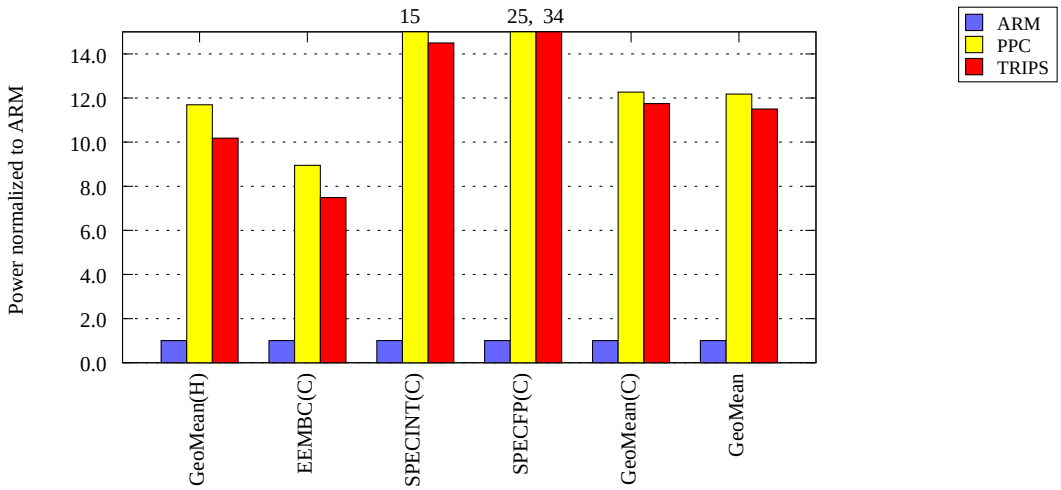


Figure 5.2: TRIPS vs. Other Platforms: Power Comparison @ 1.2V, 2GHz, 65nm

5.1 TRIPS Comparison

5.1.1 Performance and Raw Power

Figures 5.1 and 5.2 compare performance and power respectively of the three platforms: ARM, Power4, and TRIPS. In these graphs, both performance and power of all platforms are normalized to that of ARM. The following experimental configuration is used to generate these results: voltage supply of 1.2 Volts, 2GHz clock frequency and 65nm technology. The y-axis of Figure 5.1 plots performance (inverse of number of execution cycles); higher bars equate to fewer execution cycles, and better performance. The y-axis of Figure 5.2 plots power dissipation in Watts with higher bars indicating higher power dissipation. The x-axis of both figures show the various types of benchmarks. The graphs show benchmarks that are compiled and then hand-optimized on TRIPS (denoted by the letter 'H') separately from the ones that are purely compiled by the TRIPS compiler (denoted by 'C'). We separate these benchmarks because hand-optimized benchmarks typically exhibit better performance on TRIPS than compiled ones. The graphs also report the overall geometric mean of normalized performance and power for all platforms.

On average, we observe that TRIPS achieves five times better performance compared to the ARM platform, and approximately the same performance as the Power4. On the SPEC floating point (FP) benchmarks (SPECFP), TRIPS and Power4 outperform the ARM platform mainly because the simulated ARM platform lacks hardware FP support and emulates FP code in software. The TRIPS microarchitecture implements 16 FP units compared to one FP unit (with support for fused multiply and add) in Power4. By better extracting the parallelism in FP code using its 16 FP units, TRIPS outperforms Power4 in the SPEC FP category. In addition, due to better code quality of hand-optimized benchmarks, TRIPS outperforms Power4 in those benchmarks as well. Since the other compiled benchmarks are control-intensive, the quality of code generated by the TRIPS compiler

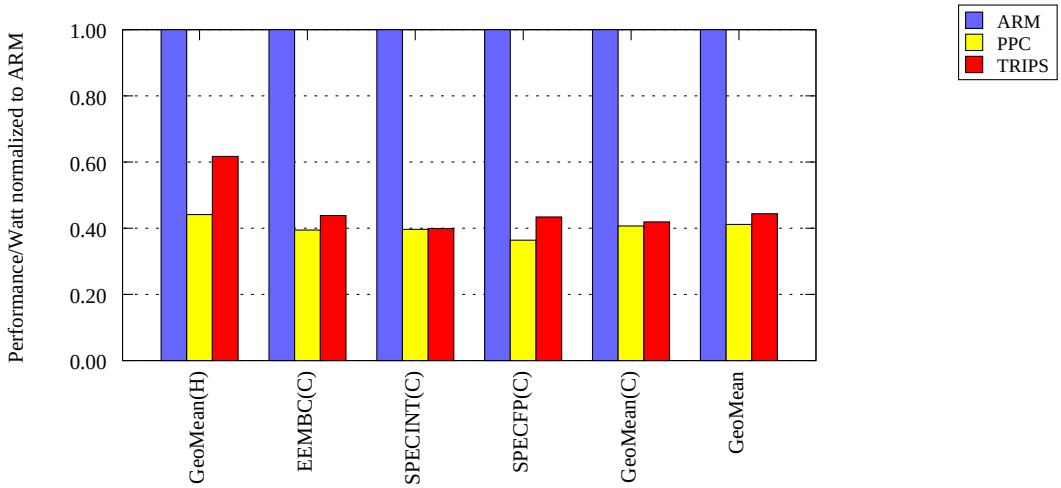


Figure 5.3: TRIPS vs. Other Platforms: Inverse PDP Comparison @ 1.2V, 2GHz, 65nm

prevents TRIPS from achieving better performance. As a result, the Power4 slightly outperforms TRIPS in such benchmarks.

In terms of raw power dissipation, we observe that on average TRIPS consumes approximately 11.5 times more power than ARM. Power4 fares similarly consuming about 12 times more power compared to ARM. The ARM microarchitecture has a single-issue pipeline, and has been heavily optimized for the embedded domain. On the other hand, both TRIPS and Power4 target higher performance, and are not necessarily optimized for the lowest power. Despite achieving the same performance level as that of Power4, TRIPS consumes slightly lower power on average compared to Power4. As mentioned above, TRIPS achieves much better performance than Power4 on SpecFP code, by better utilizing the 16 FP ALUs. Due to this, TRIPS consumes almost 40% more power than Power4 in SPEC FP code.

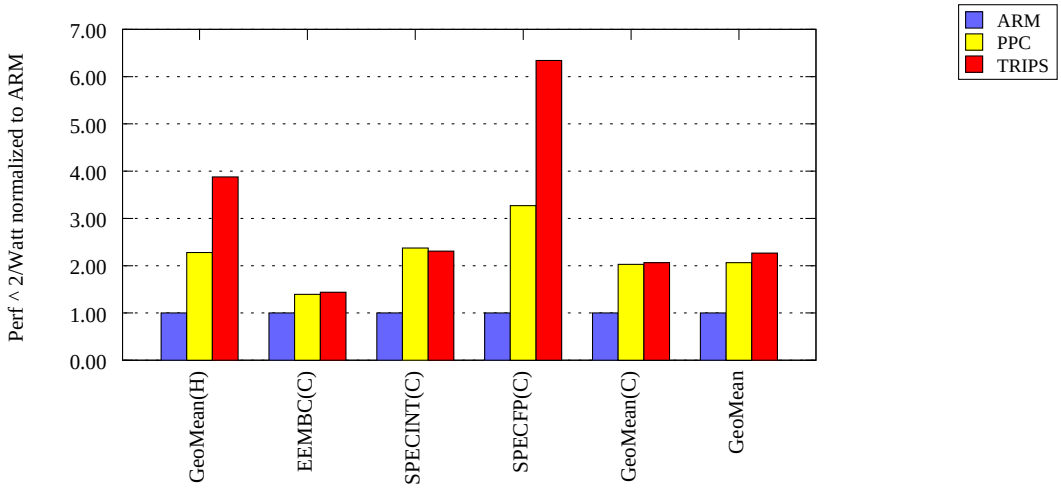


Figure 5.4: TRIPS vs. Other Platforms: Inverse EDP Comparison @ 1.2V, 2GHz, 65nm

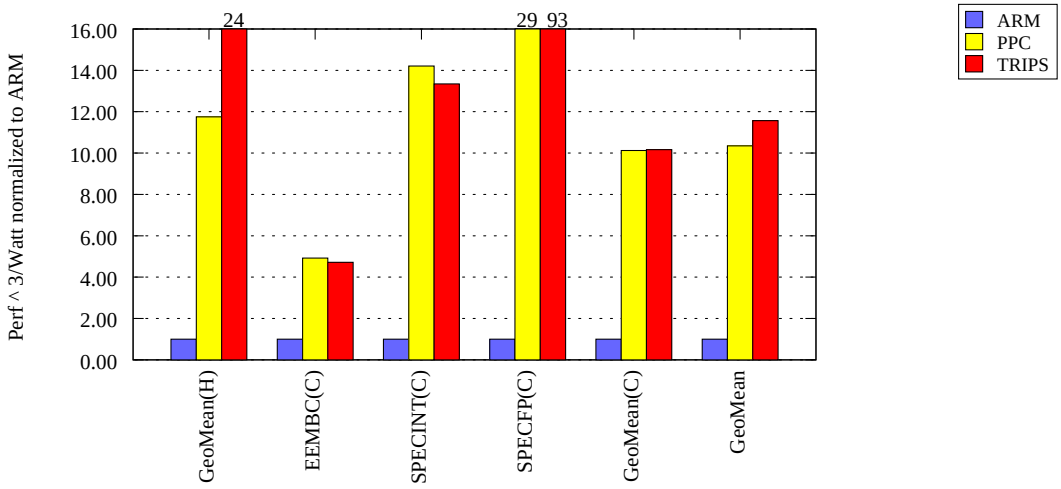


Figure 5.5: TRIPS vs. Other Platforms: Inverse ED^2P Comparison @ 1.2V, 2GHz, 65nm

5.1.2 Energy-Delay-Product and Energy-Delay^2 Product

Researchers use different metrics to compare the energy efficiency of processors. These metrics include, but are not limited to, power-delay-product (PDP), energy-delay-product (EDP), and energy-delay^2-product (ED^{2}P) [13]. Hofstee suggests the use of a metric called Energy-Performance Ratio (EPR) to evaluate techniques that maximize performance within strict power budgets [46]. For every design parameter that affects performance and energy (circuit techniques, pipelining, and voltage scaling are some examples), the EPR metric measures the ratio of percent change in energy-per-operation per percent change in performance. Hofstee also classifies the metrics of PDP, EDP, and ED^{2}P according to the EPR values of 0, 1, and 2 respectively. Each of these metrics (PDP, EDP, and ED^{2}P) can be used for energy efficiency comparison under different scenarios. The PDP metric and its inverse, performance/watt, represent the total energy consumed, and are used for comparison in the mobile computing domain. In this domain, maximizing the battery life takes precedence over performance. Hence, the PDP metric favors designs with lower power consumption regardless of their performance levels. In contrast, the EDP and the ED^{2}P metrics place equal or more emphasis on performance compared to energy respectively, and is typically used in high-end designs [13]. The PDP metric is useful in the narrow regime of mobile computing (sub 10-watt regime), where maximizing battery life is of utmost importance. However, we mainly use the ED^{2}P metric in this dissertation for comparing energy efficiency of the various platforms because this metric favors designs with maximal performance within strict power budgets [11, 13, 69, 103, 130]. For the sake of completeness, we also compare the various platforms using the PDP and the EDP metrics.

Figure 5.5 compares the inverse ED^{2}P metric of ARM, Power4, and TRIPS platforms. The y-axis of the graph plots performance^3/watt normalized to that of ARM, and the x-axis shows various types of benchmarks. In this graph, a higher

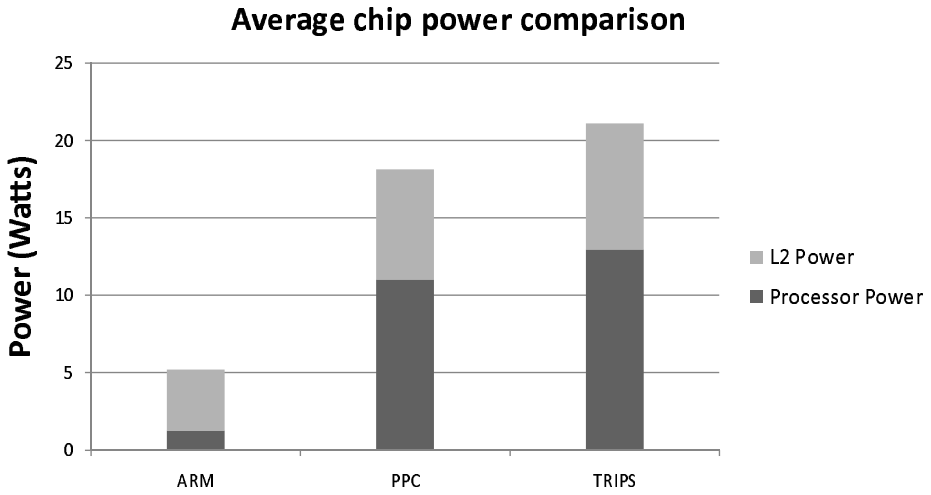


Figure 5.6: TRIPS vs. Other Platforms: Chip Power Comparison

value for performance^3/watt means higher energy efficiency of the system. Both TRIPS and Power4 achieve better inverse ED^{2}P compared to ARM because they achieve better performance. In comparison to Power4, TRIPS achieves approximately 12% better ED^{2}P, as they both have similar performance and TRIPS has slightly lower power dissipation. Figures 5.3 and 5.4 compare the inverse PDP, and EDP respectively of ARM, Power4 and TRIPS. The y-axis of the graphs plots performance/watt, and performance^2/watt normalized to that of ARM respectively. As discussed above, when the PDP metric is used for comparison, the ARM platform performs about 2.5 times better than Power4 and TRIPS. In terms of performance/watt, both Power4 and TRIPS do not achieve a one-to-one increase in performance and power, despite the better performance compared to ARM. So, Power4 and TRIPS do not fare as well as ARM using the PDP metric.

5.1.3 Comparison of Chip Power

Figure 5.6 compares the average chip power consumption of ARM, Power4, and the TRIPS platforms. Each bar shows the average chip power consumed by each platform, and also shows the fraction of the chip power contributed by the processor core and the L2. As mentioned before, both ARM and Power4 designs model a 4-MB banked cache whereas the TRIPS design uses a 4-MB NUCA cache. As the ARM processor we study is single issue, it experiences lower L2 utilization than other platforms, and hence, consumes the least chip power. TRIPS, on the other hand, experiences a higher L2 utilization than Power4, and hence, consumes about 14% more power in the L2 than Power4. Overall, the TRIPS chip consumes the most power (as both the TRIPS core and NUCA L2 consume more power than the Power4 and banked L2 respectively), and is almost 16% more than Power4.

5.1.4 TRIPS: Detailed Power Breakdown

Table 5.1 presents average power (arithmetic mean) breakdowns of XScale, Power4 and TRIPS configurations across all benchmarks. Table 5.1 breaks down the total processor power into categories including leakage, clock tree, and different microarchitectural units. The power for each category is reported in milliWatts (mW). Since XScale is an in-order, single-issue microarchitecture that has been optimized for power, many structures including the load-store queues, issue windows and register renamers are absent. Such categories are attributed 0% power for XScale.

In our analysis, we mainly focus the comparison on the power overheads of the superscalar Power4 and the distributed TRIPS microarchitectures. Power4 is a typical superscalar processor that employs large, multi-ported register files, complex register renamers, power-hungry issue windows supporting associative searches to extract better performance. TRIPS, on the other hand, completely eliminates monolithic, centralized structures, and uses partitioning and replication to achieve

Category	XScale (milliWatts)	Power4 (milliWatts)	TRIPS (milliWatts)
LEAKAGE	44 (5%)	234 (2%)	632 (5%)
CLOCK-TREE	56 (6%)	479 (4%)	441 (3%)
FETCH	239 (27%)	1026 (10%)	472 (4%)
DECODE	21 (2%)	1284 (12%)	73 (1%)
DCACHE	144 (16%)	451 (4%)	412 (3%)
FP-ALUs	0 (0%)	1246 (12%)	1745 (13%)
INT-ALUs	369 (41%)	2887 (27%)	4967 (38%)
ISSUE-Q	0 (0%)	1848 (17%)	545 (4%)
REGFILE	23 (3%)	130 (1%)	26 (0%)
REG-RENAMER	0 (0%)	875 (8%)	303 (2%)
LSQ	0 (0%)	304 (3%)	2232 (17%)
BLOCK-CONTROL	0 (0%)	0 (0%)	85 (1%)
OPN-ROUTER	0 (0%)	0 (0%)	1030 (8%)
TOTAL POWER	896	10764	12964

Table 5.1: TRIPS Power Breakdown Comparison

Structure	Per-Access Energy (pJ)		Avg. Number of Accesses (millions)		Avg. Power (mW)	
	Power4	TRIPS	Power4	TRIPS	Power4	TRIPS
Reg. File	15.79	5.685	78.6	14.2	130	26
Reg. Renamer	401.72	224.13	78.6	61.8	875.0	303.0
Issue Window	94.42	8.91	882	319	1848.0	545.0
LSQ	153.18	1348.06	20.0	22.64	304.0	2232.0

Table 5.2: Detailed Power Breakdown

better performance. Our comparison clearly illustrates the power advantages and disadvantages of either approach.

All power categories in Table 5.1 are common to both TRIPS and Power4 except for "Block-Control" and the "OPN-Router". Block control refers to the power spent in the block fetch and commit protocols in the distributed TRIPS microarchitecture, and OPN-Router refers to the power consumed by the operand network router. Table 5.2 tabulates the per-access energy, average number of accesses, and the average power consumed by important structures in Power4 and TRIPS. The per-access energy comes from CACTI-based models, and does not include clock or control logic power. However, the average power column includes both clock and control logic power for each structure.

Clock Tree and Leakage: As can be seen from Table 5.1, the in-order, single-issue XScale microarchitectural is power-optimized compared to TRIPS and Power4. Since both Power4 and TRIPS are larger designs, the clock tree power for Power4 and TRIPS are almost 8 times that of ARM. TRIPS being the largest of the three designs (in terms of area) has the most leakage power, which is about 3x that of Power4 which in turn is 5x that of XScale.

Power4 is a complex design, compared to TRIPS, with larger caches, multiple register files with higher number of ports and CAM-based issue queues, which require larger clocking support. Also, as described in Section 4.2 we perform two normalization steps to the baseline clock models of Power4 – realistic clock gating and clock tree overhead. Realistic clock gating adds a fixed percentage (10%) of ungated clock power as overhead to an idle unit. Clock tree overhead adds the power dissipated in the clock tree (clock buffers) to the baseline clock power (a fixed percentage of 10%). These two normalization steps combined with more complex design cause Power4's clock tree power comparable to that of TRIPS.

Register Files: The TRIPS microarchitecture implements 128 architectural

registers per thread slot, which are partitioned into four R-Tiles [92]. The register files in each R-Tile have only 1 read and 1 write ports. The Power4 baseline, on the other hand, implements separate floating point and integer register files. The FP and the Integer register files have 3 read and 1 write, and 2 read and 1 write ports respectively, which in turn increases the per-access power for the Power4 register files.

Furthermore, the EDGE ISA eliminates many register accesses due to direct intra-block instruction communication (5 times less register accesses compared to Power4), as indicated in Table 5.2. Due to these factors the register file power in Power4 is 5 times that of TRIPS. Power4 supports multiple concurrent register accesses through additional ports, whereas TRIPS concurrent register access through partitioned register files with fewer ports. Simple and partitioned register files and direct dependence encoding are clear advantages for the TRIPS microarchitecture.

Register Renamer: The Power4 microarchitecture maps architectural register names to physical registers using integer and FP register renamers. Since the Power4 baseline can dispatch up to 4 integer operation and 4 FP operations per cycle to the issue queues, the individual register renamers should have the ability to map a maximum of 12 integer (4 * 2 Sources + 4 * 1 Destination) and 12 FP registers, which greatly increases the complexity of the renamer. The TRIPS microarchitecture implements register renaming between blocks using read and write queues in each R-Tile [92]. The partitioned implementation of the read and write queues decreases the number of required ports, and achieves a 3x reduction is register forwarding power compared to that of Power. Again, the partitioned implementation of register forwarding is an advantage for the distributed TRIPS microarchitecture.

Issue Windows: Power4 microarchitecture dynamically rediscovers dependencies between producer and consumer instructions in a program using the wake-up/select logic in the issue windows. Power4 employs power-hungry, content-

addressable memories (CAMs) in the integer and FP issue windows. Every result broadcasts its tag to all instructions in the issue window and an associative search is performed to find the correct dependent instructions. TRIPS employs an ISA, which explicitly encodes the dependence between a producer and a consumer, thereby avoiding the need for this associative search at runtime. TRIPS has a power-efficient, RAM-based issue window which significantly reduces the energy consumption. The associative issues windows of Power4 (integer and FP combined) consume 3x more power than that of TRIPS.

Load-Store Queues: Modern microprocessors typically include load-store queues for dynamic memory disambiguation [96]. The Power4 microarchitecture employs a 36-entry load queue and a 36-entry store queue for memory disambiguation. The TRIPS microarchitecture supports a large instruction window (1024 instructions) with a total of 8-inflight TRIPS blocks [92]. Due to restrictions in the TRIPS ISA [92] each TRIPS block is allowed up to 32 load or store instructions. This means a total 256 load or store instructions (32 instructions x 8 blocks) could potentially be in-flight. TRIPS employs a brute-force solution to attack this problem: each D-Tile implementing the load-store queue in TRIPS has a 256-entry load-store queue [92]. This ensures correctness in the worst case where all 256 load or store instructions map to a single D-Tile (because of address partitioning). This brute-force LSQ design in the TRIPS is a significant area and power overhead compared to Power4. Our results indicate that TRIPS LSQs consume almost 7x the power of Power4 LSQs. Researchers have proposed solutions for solving this problem using negative acknowledgements on the operand network [97]. The TFlex microarchitecture employs this solution to avoid maximally sizing the LSQs, and thus mitigates this disadvantage significantly.

Operand Network and Block Control: As shown above, the distributed TRIPS microarchitecture achieves better power efficiency by avoiding multi-ported,

monolithic hardware structures, and by partitioning various hardware structures. On the downside, TRIPS must spend significant energy for communication across the distributed substrate. The TRIPS microarchitecture implements a host of micronetworks for both communicating data and sending control messages to orchestrate distributed execution. The operand network (OPN), a key micronetwork, replaces the conventional bypass networks in the TRIPS microarchitecture and transports results of an instruction to all of its consumers. Our results indicate that almost 9% of total processor power is spent in the distributed micronetworks (OPN-Router and Block-Control categories in Table 5.1).

Additionally, as described in Section 2.1, another overhead of the TRIPS ISA is the move instruction. TRIPS compiler employs the move instruction to create a software fan-out tree whenever a producer has more consuming instructions than supported by a single instruction. Analysis of simulation results show that roughly 20% of all TRIPS instructions consists of moves. The energy spent by TRIPS in these communication micronetworks, and the limited fan-out of the move instruction are power overheads.

I-Cache Efficiency: Each TRIPS block has a header block that holds has the register read and write instructions and other metadata. This header block is an overhead of the TRIPS ISA with respect to the instruction cache capacity and hit rate. Although the ISA has support for variable-sized blocks, the TRIPS microarchitecture expands non-full blocks into NOPs when fetching blocks from the L2 cache to I-caches [68]. These overheads affect the overall I-cache hit rate, thus impacting performance. For instance, the average I-cache hit rate for SPEC benchmarks in Power4 is about 99% whereas it is 97% in TRIPS. These efficiency issues can be addressed by reducing the header-block size in the ISA and by supporting variable-sized blocks in the I-cache. However, exploring techniques to improve I-cache efficiency is outside the scope of this dissertation.

5.1.5 Summary of TRIPS Results

Using a detailed power breakdown of XScale, Power4 and TRIPS, we have discussed the power advantages and overheads of the TRIPS ISA and the TRIPS microarchitecture. Despite the overheads of the TRIPS ISA and the distributed TRIPS microarchitecture, TRIPS achieves similar performance and slightly lower power compared to Power4. Thus, TRIPS achieves slightly better energy efficiency, approximately 12% better ED^2P, compared to the Power4. XScale consumes much lower power than both TRIPS and Power4, but also achieves poor performance, and thereby, achieving poor energy efficiency compared to other platforms.

The TFlex microarchitecture, which also implements the TRIPS ISA, shares a few overheads of the TRIPS ISA like predication and I-cache efficiency. However, by better matching hardware resources to needs of applications, TFlex is expected to achieve better energy efficiency compared to TRIPS. Additionally, TFlex employs the NACK LSQ mechanism to avoid maximally-sized LSQs found in TRIPS.

5.2 TFlex Results

In this section, we present our experimental results comparing performance, power, and energy efficiency of the TFlex platform to that of ARM, Power4 and TRIPS. First, we present the comparison results of TFlex 1-core and 2-core configurations with that of ARM and Power4. Next, we present a detailed power breakdown analysis of 1-core TFlex configuration, and compare it with the breakdowns of Power4. Finally, we describe the effects of composability (all TFlex configurations, up to 32 cores) on performance, power and energy efficiency. We conclude the chapter with a comparison of all platforms discussed so far: TRIPS, ARM, Power4, and all TFlex configurations.

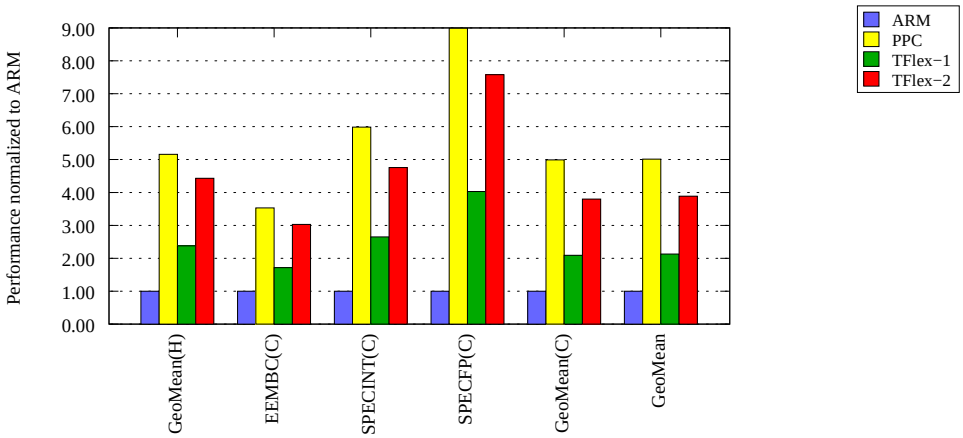


Figure 5.7: Performance comparison of 1-core and 2-core TFlex configurations with ARM and Power4 @ 1.2V, 2GHz and 65nm.

5.2.1 TFlex 1-Core and 2-Core Configurations

Figures 5.7 and 5.8 compare the performance and power of Power4, XScale, TFlex 1-core and 2-core configurations all normalized to that of XScale ("ARM" bar) for different benchmark types. We show both TFlex 1-core and 2-core configurations here because each is similar in performance to XScale and Power4 respectively. As discussed in Chapter 4, we separate the benchmarks that are hand-optimized for TFlex from the purely compiled ones because hand benchmarks typically exhibit better performance on TFlex than compiled ones. Power4 out-performs TFlex 1-core by 2.3x on average and has better performance in all benchmark categories due to its larger level-1 caches and higher issue width. Additionally, Power4 supports a fused multiply-add instruction that combines a multiply and add operation into one instruction. On the other hand, dual-issue, out-of-order TFlex 1-core outperforms the single-issue, in-order XScale core by achieving 2.1x better performance. Power4 outperforms TFlex 2-cores only by 30%, but TFlex 2-cores achieves almost 4 times better performance than ARM.

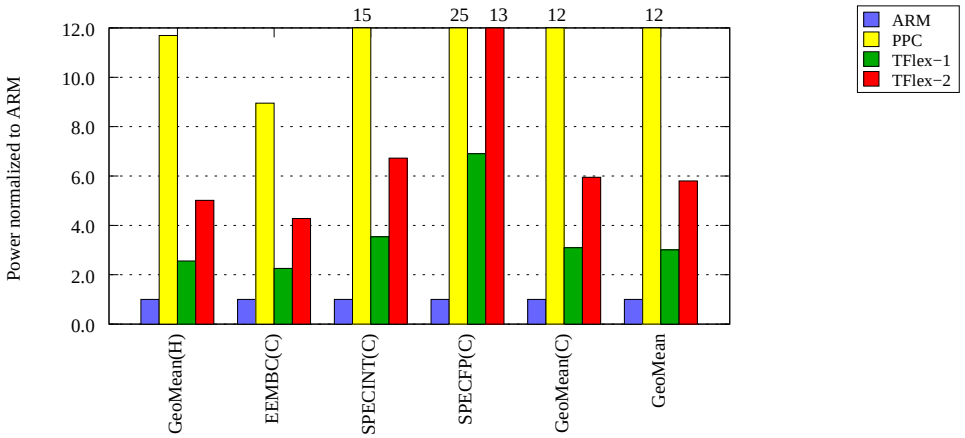


Figure 5.8: Power comparison of 1-core and 2-core TFlex configurations with ARM and Power4 @ 1.2V, 2GHz and 65nm.

Figure 5.8 shows the geometric mean of power for Power, TFlex-1, and TFlex-2 all normalized to XScale. As discussed in Chapter 4, we only report the power consumed by the respective processor cores in all platforms, and exclude the L2 power to limit the discussion to relative advantages and disadvantages of EDGE. While Power4 performs 2.4 times better than 1-core TFlex, Power4 also consumes 4 times more power than 1-core TFlex. TFlex 1-core outperforms XScale by 2.1x while consuming 3x more power, achieving 30% worse energy efficiency. The key reason for higher power consumption of Power4 is its use of many power-inefficient structures to extract better performance.

Figure 5.9 compares the inverse ED^2P metrics of Power4, TFlex-1 and TFlex-2 all normalized to XScale. The y-axis of the graphs plots $performance^3/watt$ normalized to ARM and the x-axis shows various types of benchmarks. On average, Power4 achieves 3.2x better ED^2P than TFlex 1-core, a direct result of better performance of Power4 compared to TFlex. Additionally, Power4 only achieves a 2% improvement in ED^2P over TFlex 2-cores despite consuming 2x the power of TFlex

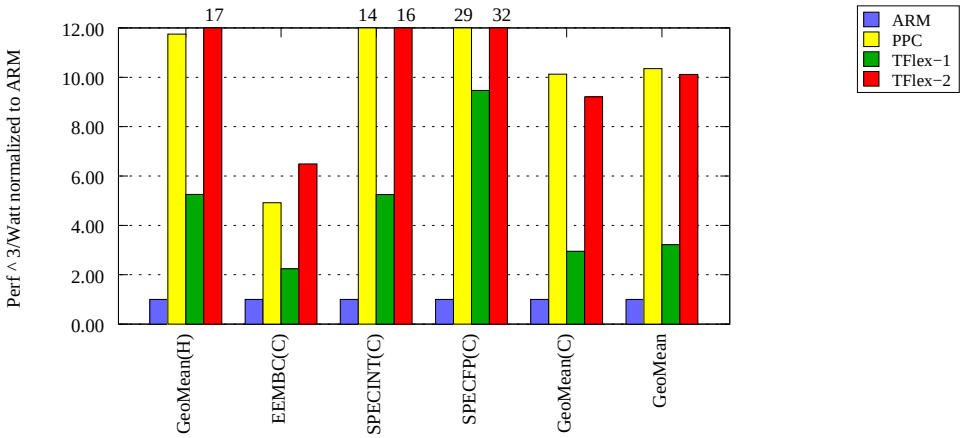


Figure 5.9: ED^{2}P comparison of 1-core and 2-core TFlex configurations with ARM and Power4 @ 1.2V, 2GHz and 65nm.

2-cores.

5.2.2 Power Breakdown Analysis of TFlex 1-Core

Table 5.3 presents detailed average power statistics of XScale, Power4 and TFlex 1-core configurations across all benchmarks. This table has the same format as Table 5.1 but reports the power breakdown of TFlex 1-core configuration as opposed to TRIPS. As in the case of TRIPS breakdown, we restrict our analysis to Power and TFlex. Figure 5.10 presents the same data for both Power4 and TFlex 1-core (as in Table 5.3) in the form of a pie-chart. In addition, Table 5.4 tabulates the per-access energy, average number of accesses, and the average power consumed by important structures in Power4 and TFlex.

Clock Tree and Leakage: TFlex 1-core has a smaller area compared to Power4, and is closer to the ARM core in its power consumption. The clock tree and leakage power of Power4 is about 5 times and 2.3 times that of TFlex 1-core.

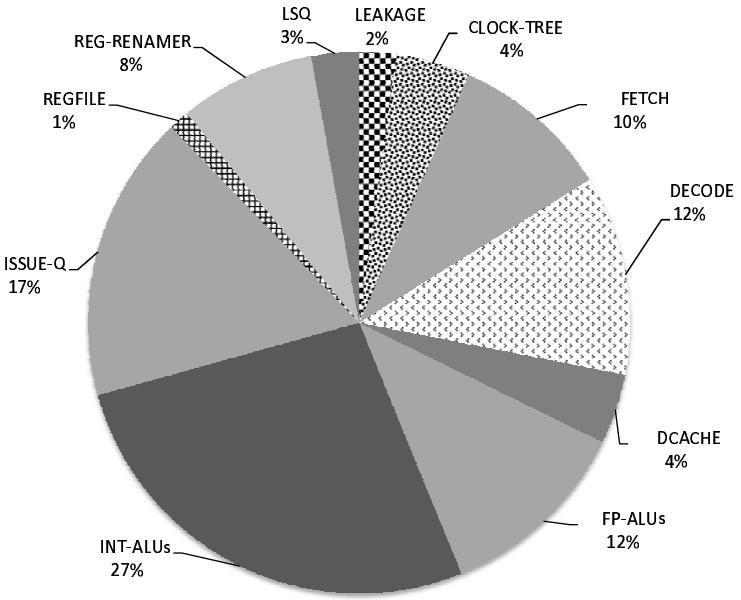
Register Files and Renamers Each TFlex core implements a unified ar-

Category	ARM (milliWatts)	Power4 (milliWatts)	TFlex 1-Core (milliWatts)
LEAKAGE	44 (5%)	234 (2%)	98 (3%)
CLOCK-TREE	56 (6%)	479 (4%)	97 (3%)
FETCH	239 (27%)	1026 (10%)	307 (11%)
DECODE	21 (2%)	1284 (12%)	32 (1%)
DCACHE	144 (16%)	451 (4%)	124 (4%)
FP-ALUs	0 (0%)	1246 (12%)	488 (17%)
INT-ALUs	369 (41%)	2887 (27%)	1336 (47%)
ISSUE-Q	0 (0%)	1848 (17%)	95 (3%)
REGFILE	23 (3%)	130 (1%)	14 (0%)
REG-RENAMER	0 (0%)	875 (8%)	87 (3%)
LSQ	0 (0%)	304 (3%)	103 (4%)
BLOCK-CONTROL	0 (0%)	0 (0%)	48 (2%)
OPN-ROUTER	0 (0%)	0 (0%)	17 (1%)
TOTAL POWER	896	10764	2848

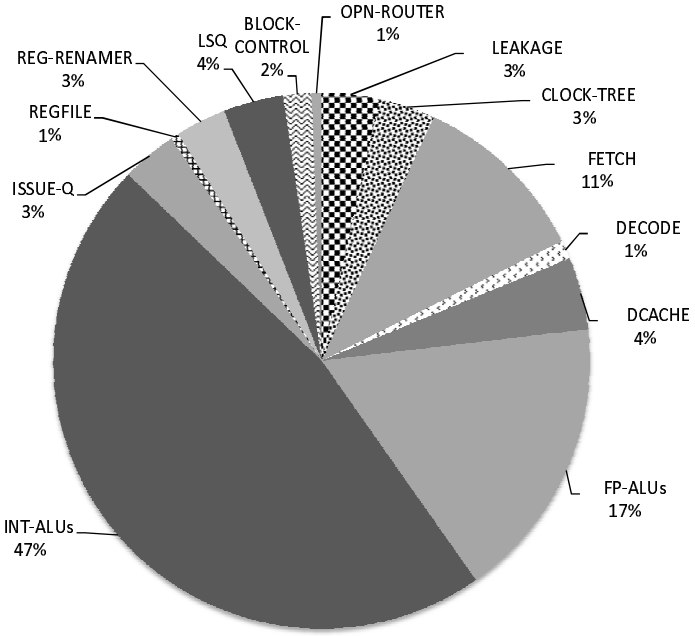
Table 5.3: TFlex 1-Core Power Breakdown Comparison

Structure	Per-Access Energy (pJ)		Avg. Number of Accesses (millions)		Avg. Power (mW)	
	Power4	TFlex-1	Power4	TFlex-1	Power4	TFlex-1
Reg. File	15.79	11.27	78.6	21.6	130.0	14.0
Reg. Renamer	401.72	223.37	78.6	23.0	875.0	87.0
Issue Window	94.42	20.1	882	272.0	1848.0	95
LSQ	153.18	269.2	20.0	19.3	304.0	103.0

Table 5.4: Detailed Power Breakdown



(a) Power4 Power Breakdown



(b) TFlex 1-core power breakdown

Figure 5.10: Power Breakdown

chitectural register file containing 128 architectural registers with 2 read and 1 write ports [57]. As discussed before, Power4 employs multi-ported integer and FP register files, and register renamers to extract better performance. Similar to TRIPS, TFlex enjoys the reduced register file accesses (3 times less compared to Power4 as shown in Table 5.4) due to direct instruction encoding of the TRIPS ISA. These factors cause the register file in TFlex to consume about 9 times less power compared to Power4. Similarly, register renaming implemented in TFlex consumes almost 10 times less power than Power4 register renaming.

Issue Windows: TFlex employs an efficient RAM-based issue window, which is enabled by the direct instruction encoding of the ISA. Power4, which employs power-hungry CAM structures for its issue windows, consumes 19 times more power than TFlex.

LSQs: The TRIPS microarchitecture implemented a maximally-sized LSQ as a brute-force solution to its memory disambiguation problem. As a result, the LSQs are a significant source of power and area overhead in the TRIPS design. TFlex, on the other hand, avoids maximally-sized LSQs by implementing unordered LSQs along with negative acknowledgements (NACKs) in the OPN [97]. These unordered LSQs significantly reduce the power overhead of the LSQs for TFlex (about 20x compared to TRIPS).

Operand Network and Block Control: In the 1-core mode, TFlex only executes a single block at a time. Hence, TFlex does not incur any distributed control messages overhead in the 1-core mode. Similarly, there is no communication across the operand network in TFlex. Although our power models implement clock gating in the OPN routers, the input FIFOs are not completely clock-gated. The power model attributes a 10% overhead in all hardware structures during idle cycles. This clock-gating overhead for the router shows up as "OPN-ROUTER" in the table, and is less than 1% of the total processor power.

Useful Instructions: Although the 1-core TFlex configuration spends a significant fraction of its power in the ALUs (a total of 64%), not all of these executed instructions are useful in producing the final block outputs (register writes, stores, or branches). Our analysis indicates that about 20% of all the executed instructions are useless in that they do not contribute to the final block outputs. Additionally, about 36% of all the fetched instructions are useless. These instructions are rendered useless by the predication model of the TRIPS ISA. Instructions that are fetched but do not receive a matching predicate or all of their operands are not executed at all. Other instructions that are speculatively executed are predicated out later in their dependence chain. Both these instruction types (fetched but not executed, and executed but not used) contribute to useless instructions, which consume approximately 20% of the total TFlex 1-core energy on an average.

Move Instructions: As described before, whenever a producer instruction has many dependent consumers the TRIPS ISA uses a software fanout tree to distribute results to the consumers. This fanout tree is built out of a special "move" instruction in the TRIPS ISA. While the explicit dependence encoding of the TRIPS ISA reduces the number of accesses to the global register file, the presence of these extra move instructions adds overhead to TFlex execution. On average, about 25% of the executed instructions in TFlex 1-core configuration are moves. Since conventional architectures like Power4 use the global register file to communicate operands from producers to consumers (ignoring operand bypassing), it is interesting to compare the energy required for a register file access in Power4 and the move instruction in TFlex. Our CACTI models show that a single access to the operand buffer (a structure that stores the operands for each instruction) requires an energy of 11.2 picoJoules (pJ), whereas a single access to the combined FP and integer register file in Power4 costs approximately 15.8 pJ. Because the move instruction must perform two accesses to the operand buffer to deliver the result to actual consumer (one

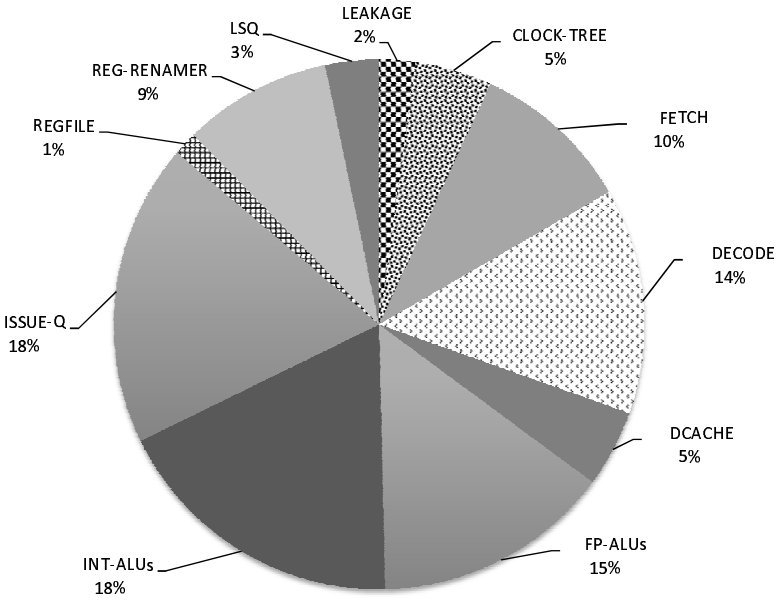
Category	Power4 (milliJoules)	TFlex 1-Core (milliJoules)
LEAKAGE	6	8
CLOCK-TREE	13	8
FETCH	27	23
DECODE	38	2
DCACHE	13	10
FP-ALUs	40	43
INT-ALUs	50	93
ISSUE-Q	51	7
REGFILE	4	1
REG-RENAMER	25	7
LSQ	9	8
BLOCK-CONTROL	0	4
OPN-ROUTER	0	1
TOTAL ENERGY	276	215

Table 5.5: Comparison of Energy Breakdowns: Power4 and TFlex 1-core

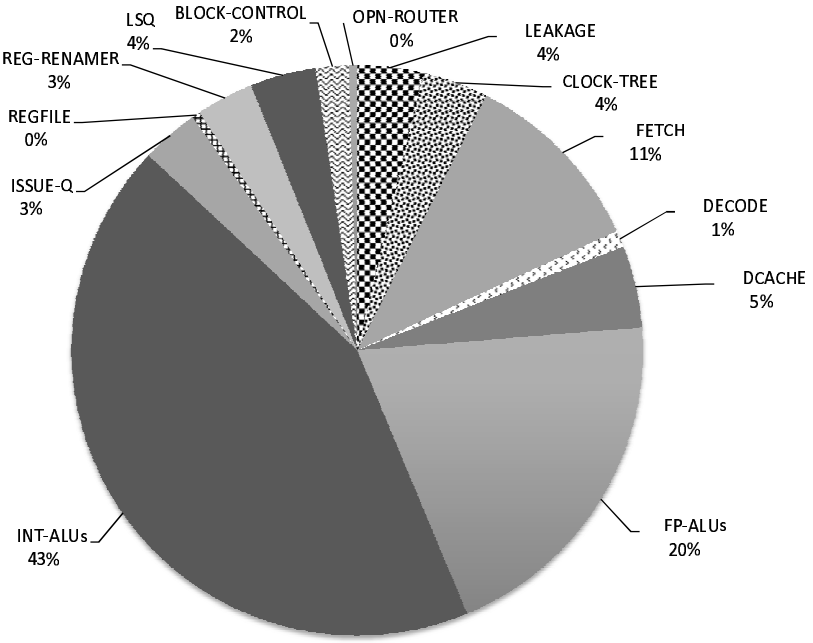
access for reading the result value and another for storing the result in the entry corresponding to the consumer), it consumes a total of 22.4 pJ for one consumer, and 11.2 + 11.2 * n pJ, in general, where n is the number of targets supported by the move instruction.

5.2.3 Energy Breakdown Analysis

One of the main reasons for the higher power consumption of Power4 compared to TFlex 1-core is that the performance of Power4 is better than TFlex 1-core. Power4 completes execution of the benchmarks in a shorter time interval compared to TFlex 1-core, and because power is measured as energy consumed per unit time, the average power dissipation is higher for Power4. In order to discount the effects of execution time in the comparison, we look at the average (arithmetic mean) energy breakdown of Power4 and TFlex 1-core. Table 5.5 presents this average energy breakdown for Power4 and TFlex 1-core in milliJoules. The same data is represented as a pie-chart



(a) Power4 Energy Breakdown



(b) TFlex 1-core Energy Breakdown

Figure 5.11: Energy Breakdown

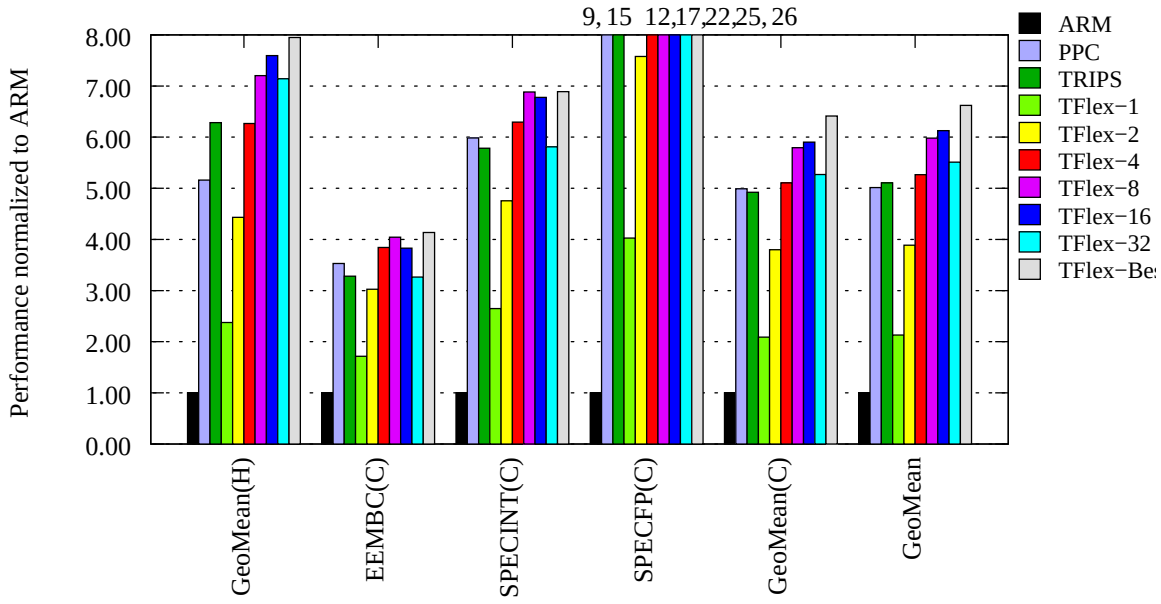


Figure 5.12: Performance comparison of all TFlex configurations with other platforms @ 1.2V, 2GHz and 65nm.

in Figure 5.11. From these results, we observe that the Power4, on an average, consumes about 30% more energy than the TFlex 1-core configuration.

5.2.4 Performance and Power Comparison of Composability

In this section we examine the effects of composability on performance, power and ED^2P of the TFlex system, scaling from 1 to 32 cores. We also plot the same metrics for the other platforms considered so far: TRIPS, ARM and Power4. All of these metrics have been normalized to that of ARM. Figures 5.12, 5.13, and 5.14 plot normalized performance, power, and inverse ED^2P of all TFlex configurations along with other platforms. As discussed earlier, we mainly use the ED^2P metric for energy efficiency comparison as it favors designs that achieve maximal performance within a power budget. Figure 5.16 and 5.15 compare the inverse PDP and EDP

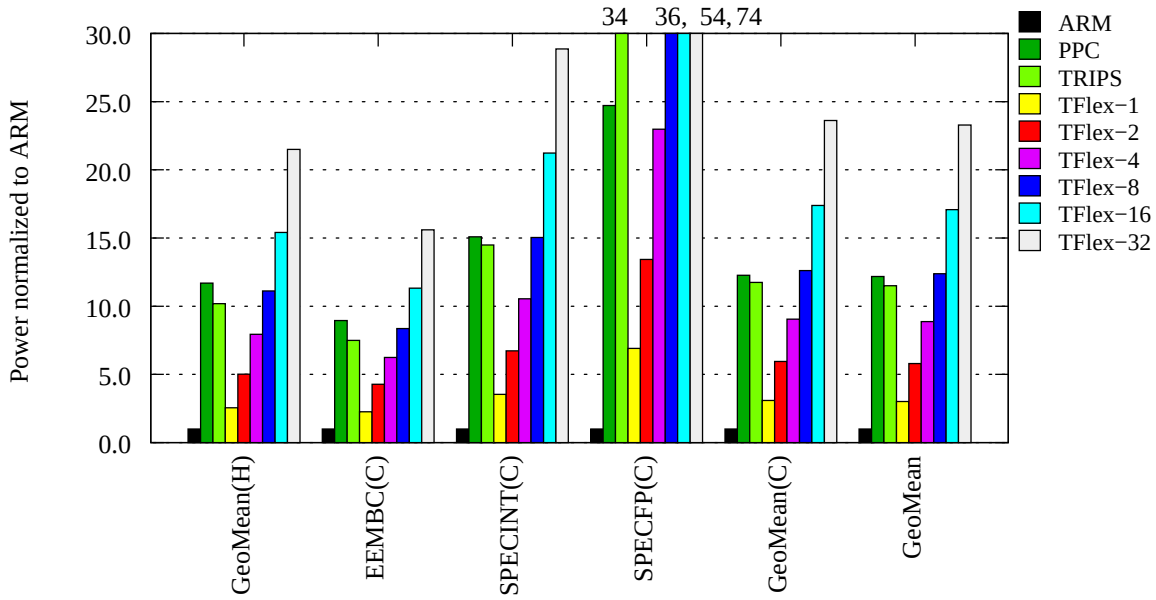


Figure 5.13: Power comparison of all TFlex configurations with other platforms @ 1.2V, 2GHz and 65nm.

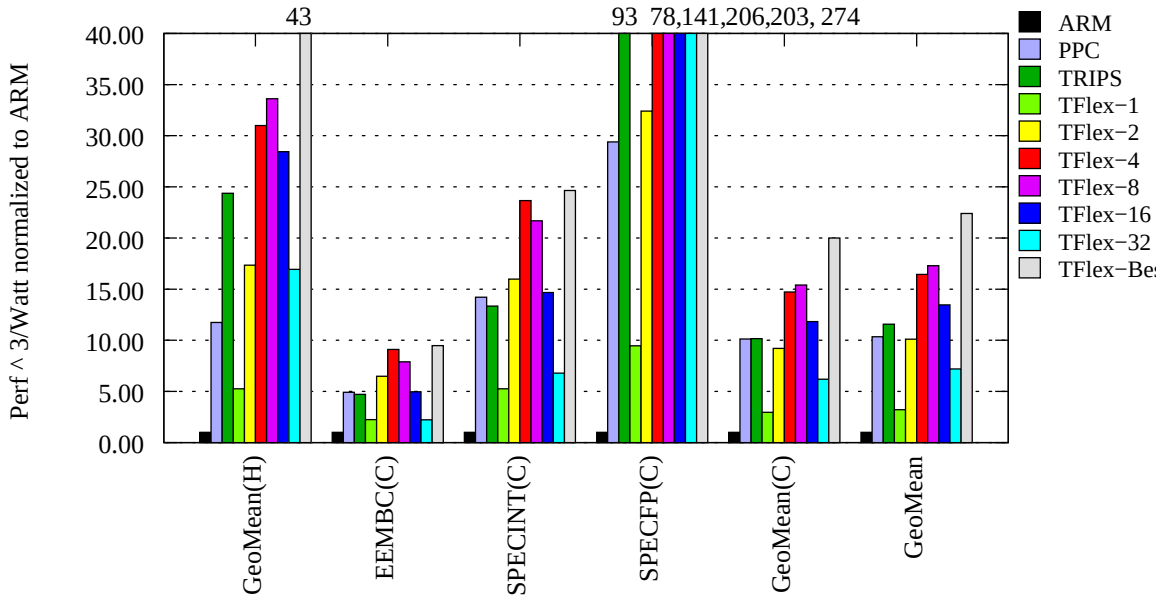


Figure 5.14: ED^{2}P comparison of all TFlex configurations with other platforms @ 1.2V, 2GHz and 65nm.

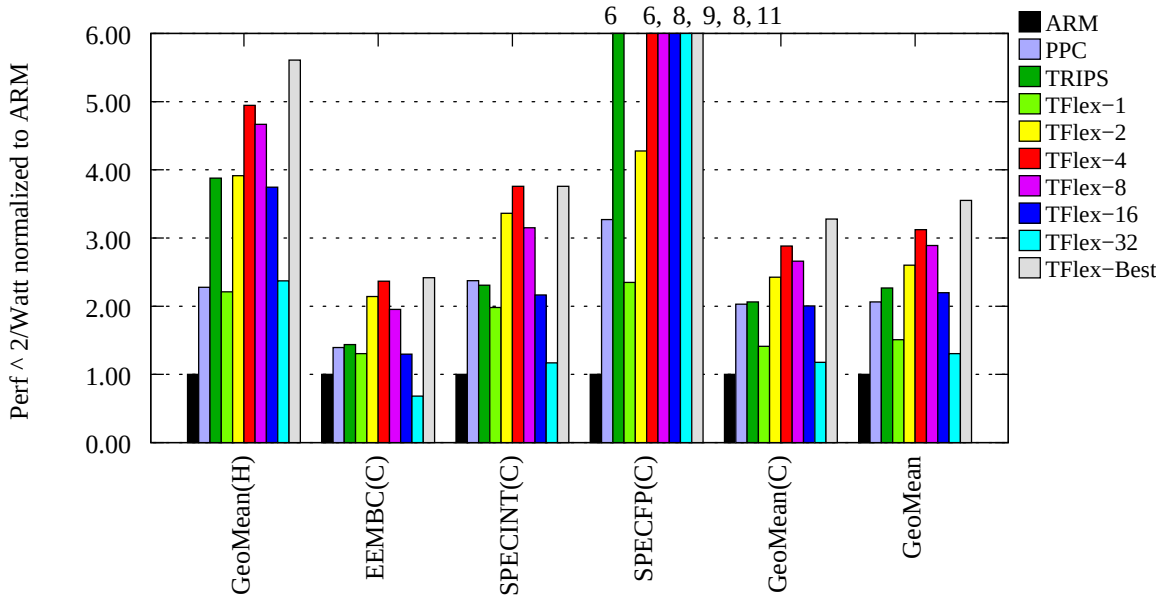


Figure 5.15: EDP comparison of all TFlex configurations with other platforms @ 1.2V, 2GHz and 65nm.

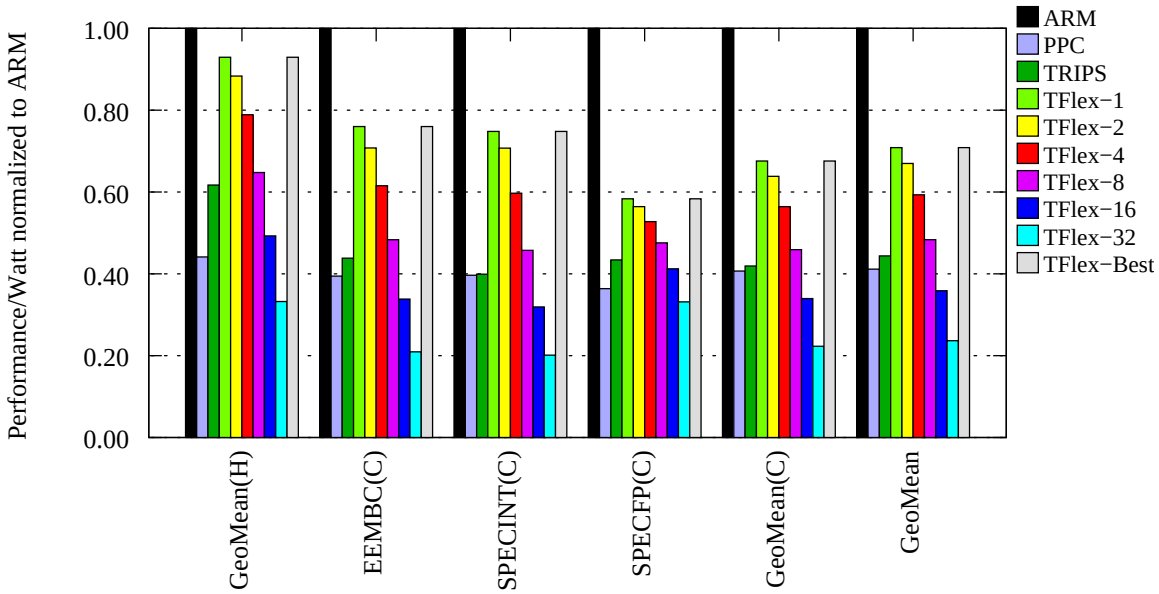


Figure 5.16: PDP comparison of all TFlex configurations with other platforms @ 1.2V, 2GHz and 65nm.

metrics of various platforms, and are presented here for the sake of completeness. The various TFlex bars represent the TFlex configurations with increasing number of cores. Since TFlex is flexible enough to use different core configurations for different applications, we also plot the "TFlex-Best" bar which represents the best-performing TFlex configuration for each application.

From these results, we observe that TFlex 16-cores is the best performing configuration on average, and the optimal TFlex configuration, in terms of performance, varies with each benchmark and benchmark types. For example, performance peaks at TFlex 16-cores for Integer SPEC programs whereas performance scales up to 32 cores in FP SPEC benchmarks. We also observe that TFlex 16-core configuration achieves roughly 20% better performance compared to Power4 and TRIPS configurations. As we increase the number of cores in the TFlex system, its power consumption also increases with increasing clock power, leakage, and communication overheads with TFlex 32-cores consuming almost 8 times more power than TFlex 1-core. Presence of fine-grained clock gating in TFlex, and the relative reduction in OPN traffic in the "deep" block mapping mode compared to the "flat" mode (please refer to Chapter 7.1 for a discussion of block-mapping policies), keep the TFlex power consumption under check as we increase the number of cores.

In terms of inverse ED^2P, TFlex 8-cores fares best compared to other TFlex configurations, and platforms. Although TFlex 16-cores has better performance than TFlex 8-cores, it also consumes more power than TFlex 8-cores and 4-cores. Hence, the optimal configuration in terms of ED^2P shifts to TFlex 8-cores, which achieves 67% and 50% better ED^2P compared to Power4 and TRIPS. The "TFlex-Best" configuration achieves 29% better ED^2P than the best performing static TFlex configuration (TFlex 8-cores). The TFlex-Best also achieves 2.1x and 1.93x better ED^2P compared to Power4 and TRIPS respectively. Thus, by better matching execution resources to the needs and parallelism profiles of applications, the TFlex

Category	TFlex-1 (mW)	TFlex-2 (mW)	TFlex-4 (mW)	TFlex-8 (mW)	TFlex-16 (mW)	TFlex-32 (mW)
LEAKAGE	98 (3%)	196 (3%)	392 (4%)	784 (6%)	1568 (9%)	3136 (13%)
CLOCK-TREE	97 (3%)	189 (3%)	341 (4%)	602 (5%)	1074 (6%)	1969 (8%)
FETCH	307 (11%)	618 (11%)	977 (11%)	1386 (11%)	1880 (10%)	2073 (9%)
DECODE	32 (1%)	63 (1%)	95 (1%)	126 (1%)	150 (1%)	152 (1%)
DCACHE	124 (4%)	248 (4%)	411 (5%)	624 (5%)	935 (5%)	1386 (6%)
FP-ALUs	488 (17%)	930 (16%)	1451 (16%)	2037 (16%)	2653 (15%)	3124 (13%)
INT-ALUs	1336 (47%)	2526 (45%)	3782 (42%)	4967 (38%)	5782 (32%)	5844 (25%)
ISSUE-Q	95 (3%)	177 (3%)	271 (3%)	378 (3%)	499 (3%)	627 (3%)
REGFILE	14 (0%)	26 (0%)	43 (0%)	71 (1%)	122 (1%)	218 (1%)
REG-RENAMER	87 (3%)	236 (4%)	449 (5%)	751 (6%)	1162 (6%)	1735 (7%)
LSQ	103 (4%)	207 (4%)	320 (4%)	434 (3%)	549 (3%)	623 (3%)
BLOCK-CONTROL	48 (2%)	91 (2%)	156 (2%)	268 (2%)	459 (3%)	844 (4%)
OPN-ROUTER	17 (1%)	138 (2%)	347 (4%)	706 (5%)	1195 (7%)	1927 (8%)
TOTAL POWER	2848	5645	9035	13135	18028	23658

Table 5.6: TFlex 1-Core Power Breakdown Comparison

platform produces better energy efficiency than the other platforms, and TRIPS microarchitecture.

5.2.5 Composability: Power Breakdown Analysis

Table 5.6 shows the average power breakdown in milliwatts for all TFlex configurations. The leakage power doubles as we double the number of cores because transistor counts grow with area. With increasing cores clock tree power increases as well, but grows sub-linearly due to fine-grained clock gating in the cores. All the other power categories show a steady increase from 1 to 32 cores due to higher number of in-flight blocks, and higher microarchitectural activity. The largest source of power overhead for TFlex is operand communication over the network. Despite the presence of clock-gating, the OPN routers consume roughly 8% of the total core power at the 32-core configuration. Similarly, the increased overhead caused by the

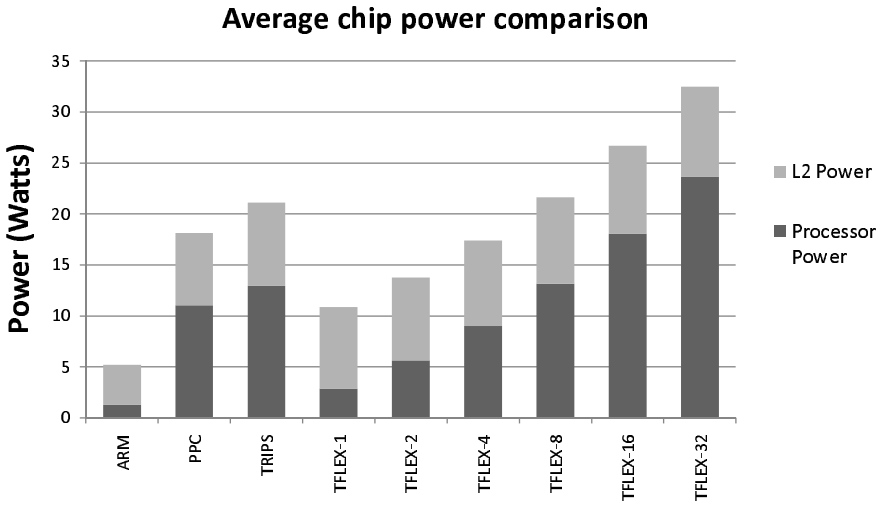


Figure 5.17: Chip Power Comparison

control messages becomes noticeable at 32 cores (about 4% of total power).

Figure 5.17 compares the average chip power consumption of all the platforms so far. We observe that the NUCA L2 cache consumes slightly more power than its banked L2 counterpart in Power4. This is especially true for higher TFlex configurations with the 32-core configuration consuming 24% more power than the L2 in Power4. This higher power is mainly caused by the increased issue width in higher core configurations, and hence, the increased L2 utilization. Additionally, the additional clocking and routing support needed in the network tiles in the NUCA L2 increase the energy overhead slightly.

5.2.6 Summary of TFlex Results

Our experimental results can be summarized as follows:

- The TFlex microarchitecture exhibits great flexibility by operating both in high-performance and low-power regimes. For example, TFlex 1-core configuration has similar power profile as that of an optimized XScale platform while

achieving better performance. On the other hand, TFlex 2-core configuration performs 30% within the performance of a 4-issue, out-of-order Power4 configuration while consuming 2x lower power than Power4.

- By better matching hardware execution resources to the needs and parallelism profiles of applications through its composability, TFlex achieves better energy efficiency both compared to the high-performance, out-of-order Power4 and non-composable TRIPS microarchitectures. For example, the "TFlex-Best" configuration achieves almost 2x better ED^2P compared to Power4 and TRIPS respectively across a wide range of applications.

5.2.7 Lessons

Based on our experience in comparing conventional platforms with TRIPS and TFlex, we have learned that EDGE architectures are promising candidates for energy-efficient extraction of performance. With composability as a mechanism to improve performance or to reduce power when desired, the TFlex architecture shows enormous promise for improved energy efficiency compared to fixed, non-composable architectures. As the demand for energy-efficient computing increases and as leakage power begins to dominate overall power, the ability to match underlying computational resources to needs of applications becomes extremely critical to better energy efficiency.

Chapter 6

DVFS and Composability: A Comparison

In this chapter, we examine the efficacy of processor composability - the ability to dynamically aggregate physical cores into a logical processor - as a performance-power trade-off mechanism. In particular, we examine composability with respect to the performance-power trade-offs offered by Dynamic Voltage and Frequency Scaling (DVFS) mechanism. First, we provide a brief overview of DVFS, and explain how it trades performance for power. We also motivate the need for potential alternatives to DVFS, and how composability can serve as this alternative. Next, we describe our experimental methodology for this chapter. Finally, we present our experimental findings, and describe under what circumstances DVFS and Composability are useful.

6.1 Introduction

As described by Equation 1.1, the power consumption of a processor has two components: 1) dynamic power caused by charging and discharging capacitance as the

processor operates, and 2) leakage power which is caused by transistors not completely turning off. Equation 1.1 describes how DVFS acts a mechanism to trade performance for power. Processor performance is linearly proportional to the operating frequency, f, to a first order. Certain types of workloads are mostly memory-bound (meaning the processor is often waiting for memory requests to complete), and do not experience this linear trend. Ignoring such workloads and the effects of main memory on performance, we typically see a linear boost in the performance when frequency is increased and vice versa. On the other hand, processor dynamic power is linearly dependent on the frequency, f, and quadratically dependent on the supply voltage, v. The maximum operating frequency of the processor depends on the current supply voltage. Hence, reducing the supply voltage allows reduction in operating frequency of the processor. The combined reduction of supply voltage and the operating frequency provides a cubic reduction of dynamic power (quadratic reduction due to drop in voltage and a linear reduction due to drop in frequency). Performance, as discussed above, experiences only a linear drop because frequency has been reduced linearly. In summary, DVFS provides an efficient mechanism to achieve cubic reductions in power with only linear drop in performance. Thus, DVFS has been the *mainstay* performance-power trade-off mechanism for a long time, and is universally implemented in current processor designs.

Figure 6.1 provides an overview of the performance-power trade-offs offered by DVFS. The graph plots performance on the y-axis, and power on the x-axis. We use several terms in the chart which are explained as follows. V_{min} is the minimum supply voltage supported by a given process technology and processor design. Of the different frequencies supported at V_{min}, f_{min} represents the minimum frequency. f_{maxeff}, on the other hand, is the maximum frequency supported at V_{min}. The maximum supply voltage is V_{max}, and the maximum frequency supported at V_{max} is given by f_{max}. As shown in Figure 6.1, DVFS provides two distinct modes

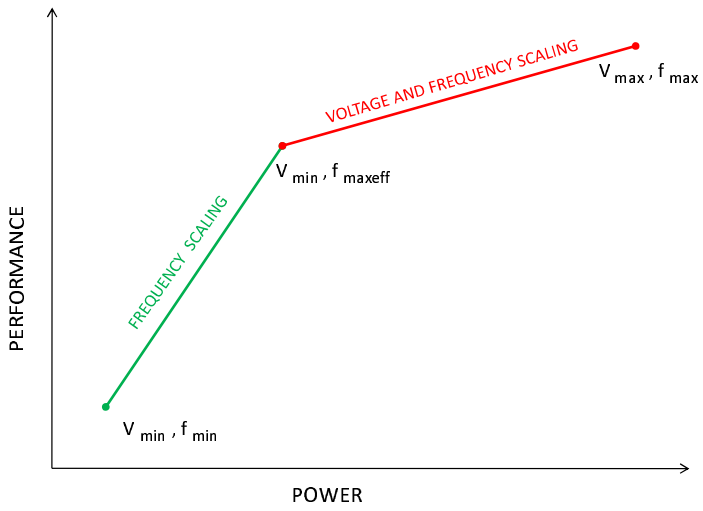


Figure 6.1: DVFS: Performance vs. Power

or regimes of operation: 1) pure frequency scaling and 2) dynamic voltage and frequency scaling. Several modern processors exhibit these regimes [1, 37]. Table 1 of [37] lists the various P-states (DVFS regime) and the T-states (frequency scaling regime) supported by the Intel 5160 processor in an IBM HS21 blade server. Additionally, Figure 1 from [37] clearly demonstrates the performance and power trade-offs provided by both DVFS and frequency scaling.

In the pure frequency scaling regime only the operating frequency of the processor changes, and the supply voltage is held constant. In the example shown in the chart, the supply voltage V_{min} remains the same, and the processor frequency is allowed to change from f_{min} to f_{maxeff} in the frequency scaling regime. In this regime, DVFS typically achieves a linear performance and power trade-off because only the frequency is modulated: a linear increase in frequency buys a linear increase in performance at the cost of linear increase in power.

The second regime is the dynamic voltage and frequency regime where both the supply voltage and the clock frequency of the processor are allowed to change. In our example, the voltage can range from V_{min} to V_{max} whereas the frequency can range from f_{maxeff} to f_{max}. In this regime, an increase in frequency additionally requires a proportional increase in voltage resulting in a cubic increase in power (linear factor from the frequency increase, and quadratic factor from the voltage increase) with only a linear increase in performance (resulting from the increased frequency). Hence, the slope of the performance-power curve in this regime is lower than that of the frequency scaling regime. In other words for every additional watt expended by the system frequency scaling regime achieves better performance than DVFS regime.

Although ED^2P metric has been used in earlier chapters for comparing energy efficiency, we use the line plots as shown in Figure 6.1 containing performance on the y-axis and power on the x-axis throughout this chapter. We use such plots primarily to highlight the difference in performance and power trade-offs provided by techniques such as frequency scaling and voltage scaling. As can be seen in Figure 6.1 and as results presented later in this chapter indicate, pure frequency scaling provides a one-to-one increase in performance and power, whereas the rate of increase in power is much higher in the voltage scaling regime. Experimental results presented later in this chapter compare the performance and power trade-offs of composability to that of frequency and voltage scaling.

In summary, the p-state, (V_{min}, f_{maxeff}), represents a fundamental constraint for the DVFS mechanism. At this p-state, the system can no longer extend the linear performance-power trade-off offered by pure frequency scaling as the maximum frequency supported at V_{min} has already been reached. Further increases in frequency necessitate an increase in voltage that results in a worse performance-power trade-off than frequency scaling.

6.2 DVFS Alternatives

Despite being the mainstay power management scheme in modern processors, the flexibility of DVFS is fast becoming limited due to narrowing voltage margins. A recent paper by Watanabe et al. [121] illustrates that the dynamic range supported by DVFS mechanisms has narrowed significantly. This narrow range of voltage scaling motivates the need for potential alternatives to DVFS. One of the key reasons for this slowing rate of voltage scaling is leakage power. When supply voltage is reduced sub-threshold leakage, a type of leakage power, increases exponentially [21]. As a result, the rate of voltage scaling has slowed down significantly to keep the increasing leakage power under check.

Given the current strict chip-level power budgets and the projected power trends, we can expect systems to be operated at the minimum voltage point V_{min}. With voltage held at V_{min} one can keep increasing frequency to get linear performance and power until we reach the (V_{min}, f_{maxeff}) point. At this point, further increases to frequency are accompanied by further increases to voltage thereby substantially increasing power dissipation for a modest increase in performance. Thus, conventional designs implementing DVFS lack alternative mechanisms that provide performance-power trade-offs similar to frequency scaling. Furthermore, processor designers have backed away from ultra-high frequency designs because power grows superlinearly as circuits are successively optimized for higher and higher clock frequency. Thus, the maximum frequency (f_{max}) for integrated circuits is likely to scale slowly in future technologies. To achieve single-thread performance beyond the range of f_{max}, systems need new microarchitectural mechanisms that provide the same or better performance-power trade-offs as frequency scaling.

We argue that processor composability is such an alternative mechanism that provides almost as good performance-power curves as frequency scaling, and much better curves compared to DVFS. A composable system can scale up or down the

number of physical cores to match the needs of the application at hand. If the application has abundant parallelism the system can aggregate many physical cores to exploit the parallelism at the cost of additional power. On the other hand, the system can reduce the alloted physical cores to an application with less parallelism (lower performance) and thereby reduce the power. Thus, processor composability provides a completely orthogonal dimension for performance-power trade-offs by scaling the number of cores up or down, and could be a potential alternative to DVFS. Additionally, composability provides a mechanism to further improve single-threaded performance beyond the range of f_{max}. This chapter examines the efficiency of composability vis-a-vis DVFS.

This study shows that composition, coupled with frequency scaling, can provide a dynamic range of power and performance for single threads that other architectures cannot, scaling from low-performance, high power efficiency to power-efficient high performance. Using that efficiency as a baseline, composing a moderate number of cores (four to eight dual-issue cores) can provide increased performance at roughly the same energy efficiency as strict frequency scaling, thus providing a more efficient solution than increasing the supply voltage.

Figure 6.2 illustrates the kind of experimental results that we aim to achieve in this chapter. The figure includes the frequency scaling and the DVFS regimes, discussed earlier in this chapter, along with the composability regime. To achieve the best power efficiency while scaling performance, one should first scale frequency while maintaining the voltage at V_{min} until the maximum frequency at that voltage (which we term f_{maxeff})is reached. One should then compose more cores while still at V_{min} and f_{maxeff} until diminishing returns are reached at eight or sixteen cores. This diminishing point is labeled as the "tipping point" in the above figure. The performance-power slopes achieved by composability are expected to be approximately linear like frequency scaling. Hence, the slope of the composability regime is

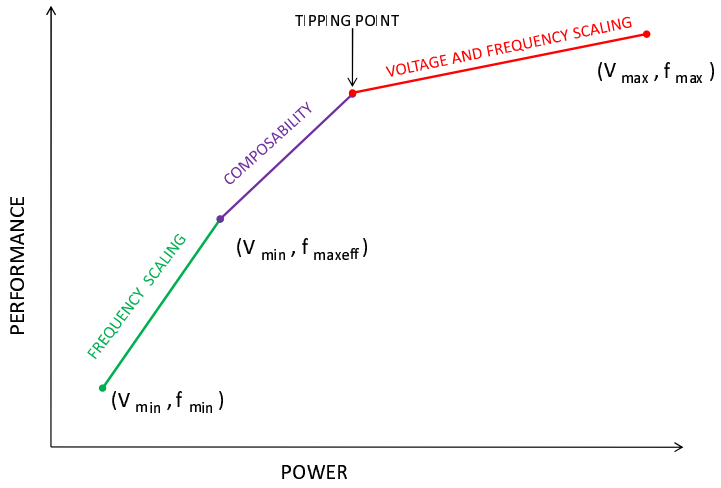


Figure 6.2: DVFS vs. Composability

slightly smaller than that of frequency scaling. As a last resort to further improve performance, voltage and frequency can be scaled together. Similarly, when scaling performance and power down, one should perform the above steps in reverse order: scale voltage and frequency first, scale down the number of cores next followed by frequency scaling alone.

6.3 Methodology

We use the TFlex platform as our experimental vehicle to study the efficiency of composability and DVFS. We use the TFlex simulator and the power models as described in Chapter 4. Since TFlex already supports composability, we merely add the support for the two DVFS regimes in the TFlex simulator: 1) pure frequency scaling and 2) dynamic voltage and frequency scaling. In addition, we also add DVFS support to the other experimental platforms, ARM and Power4, to compare

them with TFlex. Since TFlex supports both composability and DVFS, and the other platforms only DVFS, this comparison allows us to discern the benefits of composability by itself, and when composability is combined with DVFS.

We limit our experiments to *static* DVFS where the voltage and frequency are set *before* running the application and not *dynamic* DVFS where the settings are changed *while* the application is running (Chapter 8 discusses experiments with *dynamic* DVFS). Thus, we do not need to model the reconfiguration costs associated with changing the settings in all the platforms. This approach is fair because we compare static DVFS to static composability, that is composing the cores *before* an application runs. Static composability also does not incur the reconfiguration costs similar to static DVFS. If one were to compare *dynamic* DVFS configurations with *dynamic* composability modeling the reconfigurations costs for both DVFS and composability would be a fairer comparison.

In our DVFS experiments we assume a fixed set of voltages and frequencies supported at these voltages (given later in Table 6.1). We also assume that our platforms can be designed to run at the maximum frequency, 2 GHz in our case, and that these designs can be scaled down to run at lower frequencies. For modeling power with DVFS, we scale the dynamic power by the applied voltage and frequency and leakage power by the applied voltage. We also adjust the memory latency (in cycles) based on the current processor frequency. We measure performance in these DVFS experiments as the inverse of total time taken (in seconds) by the benchmark.

6.4 Experimental Results

6.4.1 Composability Results

To examine the relationship between performance and power for the TFlex composable core processor, we vary the core count from 1 to 32 (while running a single

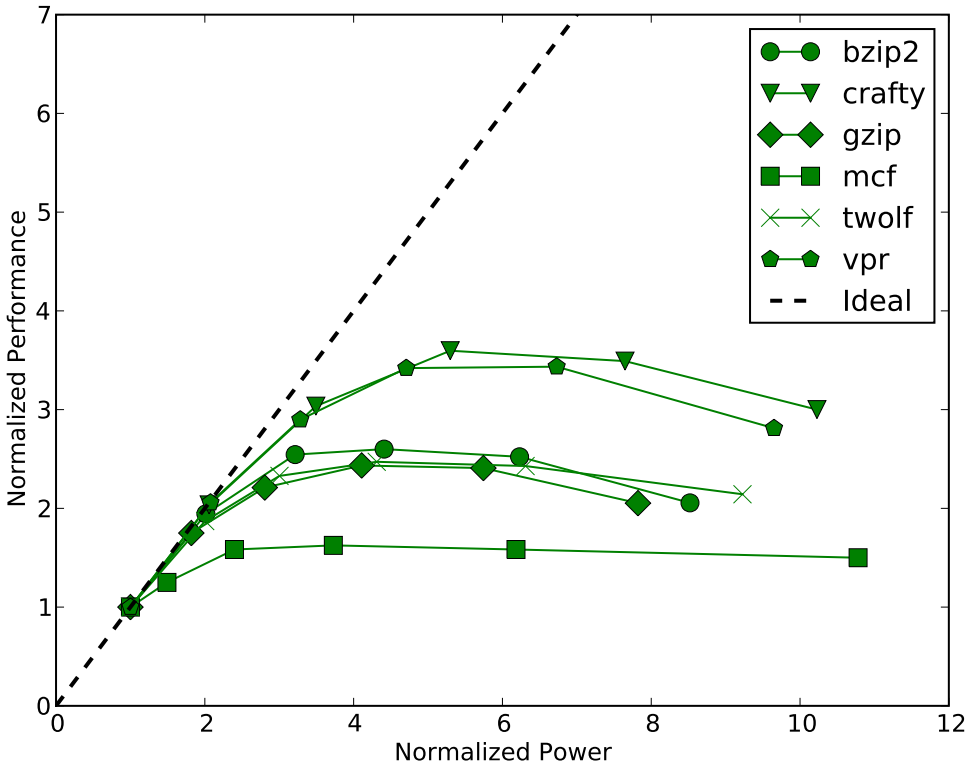


Figure 6.3: TFlex Performance Scalability: SPEC-INT

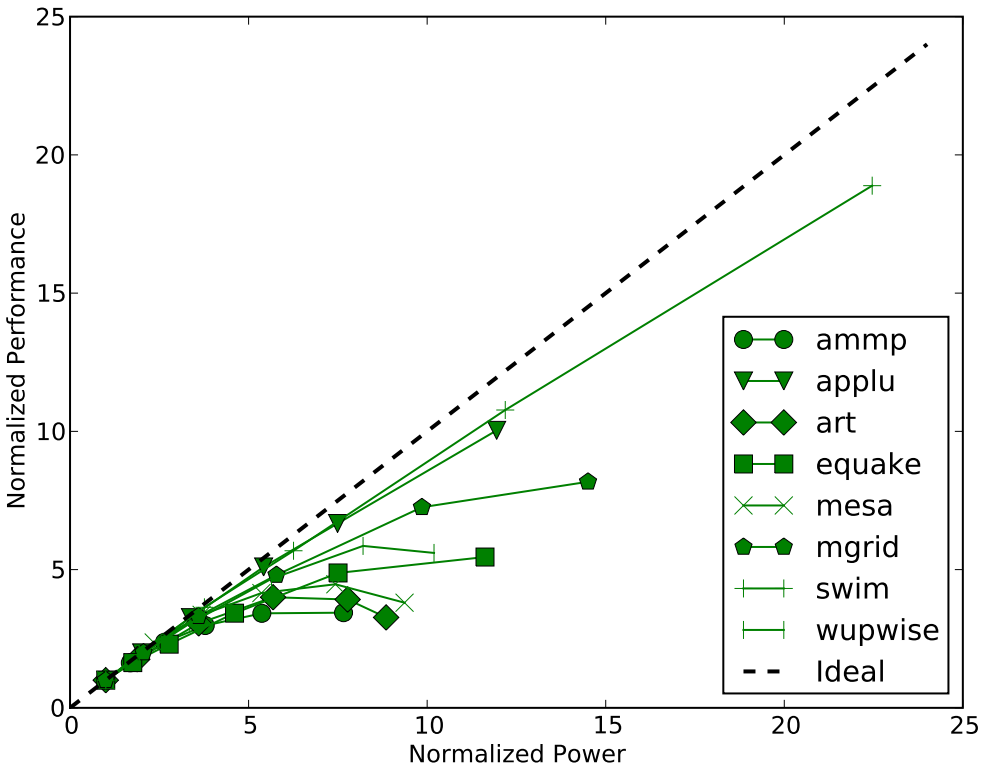


Figure 6.4: TFlex Performance Scalability: SPEC-FP

thread that employs all of the provided cores) and measure the improvement in performance and power relative to a 1-core configuration. Figures 6.3 and 6.4 plot these performance and power results for SPECINT and SPECFP applications respectively. Both graphs plot power normalized to TFlex 1-core on the x-axis and performance normalized to TFlex 1-core on the y-axis. Both graphs also plot a dotted line labeled "Ideal". This line has a slope of 1 and represents the linear performance-power trade-off (a unit increase in performance causes a unit increase in power and vice versa). Comparing the different TFlex lines with this ideal line provides insights into the regimes where composability offers a near-linear performance and power trade-off.

For most of the SPECINT benchmarks, performance and power increase at equal rates in the range of 1–4 cores. Performance reaches a peak at 8 cores and then drops off at 16 and 32 cores. These applications have insufficient instruction level parallelism to effectively use more than 8 cores; beyond that point, the increasing distribution of the architecture (with higher number of cores) increases communication overheads and causes performance to drop. In this regime, power increases but sublinearly relative to core-count increases because clock gating reduces power consumption in the idle cores. The benchmark mcf performs particularly poorly as it is memory bound and reaches its limit with just a few cores. Adding more cores to such workloads has little or negative impact on the overall performance. However, addition of more cores increases power dissipation. Composability fares worse than the ideal curve for such workloads.

On the other hand, the SPECFP benchmarks have abundant instruction level parallelism, and show better scalability with core count increases. Most SPECFP benchmarks are able to use up to 16 cores effectively, and provide almost near-ideal performance and power trade-offs. In particular, benchmarks like applu and swim have ample parallelism and easily scale up to 32 cores, providing the same trade-

Index	Vdd (volts)	Frequency (MHz)
1	1.2 (V_{max})	2000 (f_{max})
2	1.1	1750
3	1.0	1500
4	0.9	1250
5	0.8	1000 (f_{maxeff})
6	0.8	800
7	0.8	600
8	0.8 (V_{min})	400 (f_{min})

Table 6.1: DVFS Configurations for all platforms.

offs as the ideal curve. In addition, power and performance scale nearly linearly in the regime of 1–16 cores for SPECFP benchmarks, and for most applications performance drops when moving to 32 cores.

6.4.2 Composability and DVFS

To model voltage and frequency scaling we assume a 65nm process and the power states listed in Table 6.1 for our experiments. The table shows that we assume a moderate f_{max} of 2 GHz. Indices from 1 through 5 represent the DVFS regime, where both the operating voltage and frequency can change. Indices from 5 through 8 represent the frequency scaling regime where only the operating frequency changes and the voltage remains constant at 0.8 volts. We include both voltage/frequency scaling and just frequency scaling regimes to account for limited reduction in V_{min} in future process technologies. We operate Power4 and XScale models at all the states in Table 6.1. The TFlex processor is operated at all composable configurations – 1, 2, 4, 8, 16 and 32 cores – at each of the DVFS states, resulting in a total of 48 configurations (6 core configurations x 8 DVFS states).

Figures 6.5 and 6.6 show normalized performance (Y-axis) and normalized power (X-axis) of various TFlex configurations, Power4, and ARM for the SPEC-INT and SPEC-FP applications. The performance and power of all the configurations are normalized to that of TFlex 1-core configuration operating at V_{min} and f_{min} (index 8 in Table 6.1). Since performance is plotted on the y-axis and power

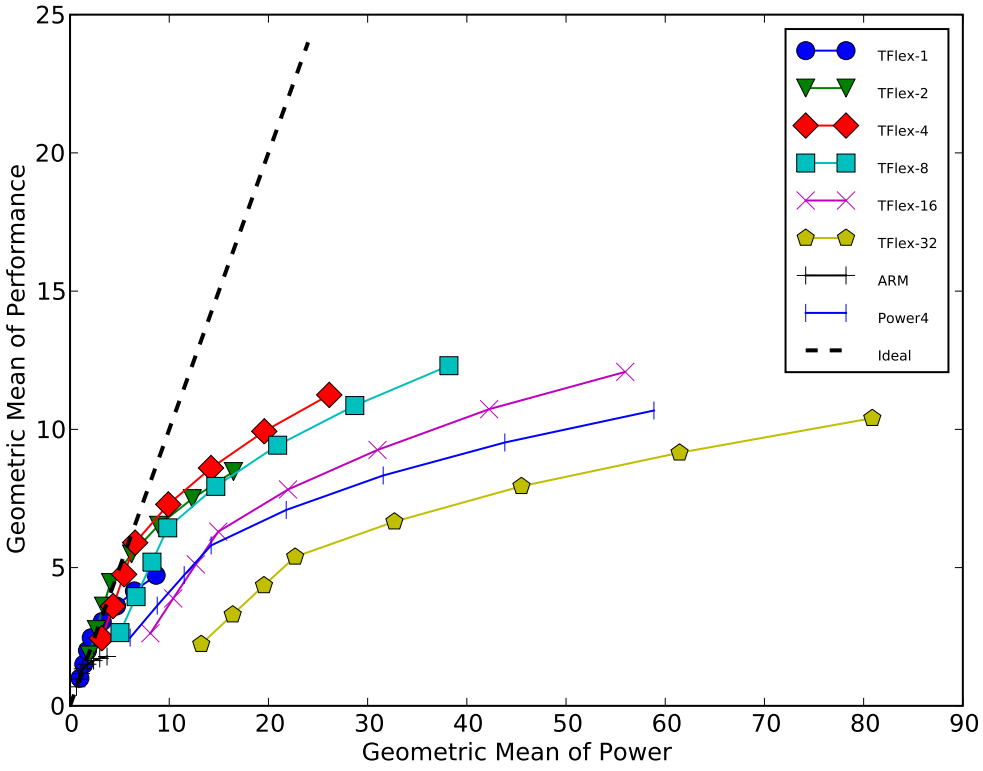


Figure 6.5: Normalized Performance and Power: SPEC-INT

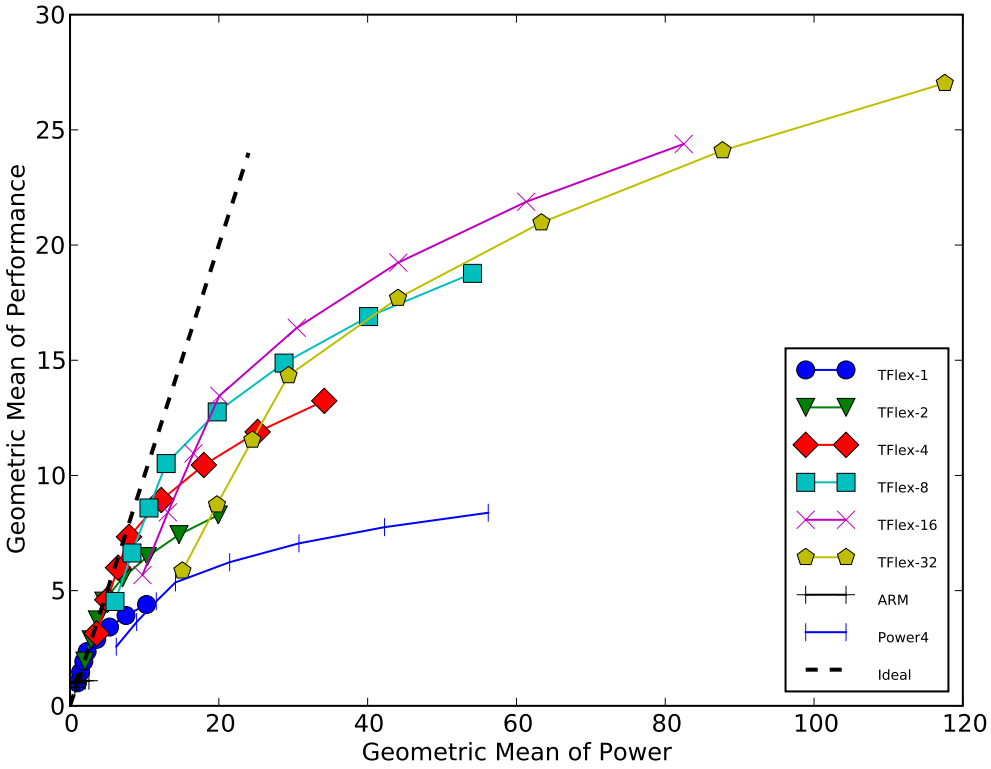


Figure 6.6: Normalized Performance and Power: SPEC-FP

on the x-axis, configurations that have higher y values and lower x values are more energy efficient than other configurations – such configurations expend less energy and achieve better performance compared to other configurations. Each TFlex curve represents a different core count (1–32 cores) and the points on each curve cover all the DVFS settings in Table 6.1. The graphs also plot the various DVFS settings for both ARM and Power4 platforms. The graphs also include a dotted line labeled "Ideal", which represents a linear performance and power trade-off (a slope of 1).

The ARM and the Power4 curves contain points from the frequency scaling and DVFS regimes whereas the TFlex curves include points from three regimes: frequency scaling, composability, and DVFS. A key insight from these graphs is that combining the composability with DVFS increases the operating range of TFlex significantly. For instance, Power4 and ARM curves include 8 operating points corresponding to the 8 indices in Table 6.1. On the other hand, TFlex curves include a total of 48 configurations (6 TFlex cores x 8 DVFS points). This observation could be exploited by the operating system (OS) that attempts to maximize system throughput subject to a given power budget. The OS chooses from a wide variety of TFlex configurations that either provide different performance levels at a given power or different power levels for a given performance.

From these graphs, we observe that in the frequency scaling regime (indices 5 through 8 in Table 6.1), each TFlex configuration achieves approximately a one-for-one improvement in performance when additional power is applied. The ARM and the Power4 platforms also perform similarly in the frequency scaling regime. For all platforms involved, the points in the frequency scaling regime of the graphs represent the iso-energy configurations–meaning all configurations of a given platform consume the same amount of energy (equal increase or decrease in both performance and power) in this frequency scaling regime. The performance-power curves for almost all of the platforms in this regime is parallel to the "linear" line (that is they achieve

the same slope) plotted in the graph, which proves the one-to-one performance and power relationship provided by frequency scaling. However, when the V_{min} and f_{maxeff} p-state is reached, conventional platforms like Power4 and ARM have no other mechanism to further improve performance except to increase the voltage. TFlex, on the other hand, provides composability to further improve performance without adversely incurring power overhead of voltage scaling.

In the SPECINT benchmarks, although Power4 performs as good as or better than many TFlex configurations, most Power4 configurations incur more power overhead at similar performance levels. By providing a rich range of feasible configurations, TFlex optimizes for power efficiency in SPECINT benchmarks when lower performance levels could be tolerated. As discussed in Section 6.4.1, since SPECFP benchmarks have abundant parallelism, TFlex achieves much better performance-power trade-offs than in SPECINT workloads. Figure 6.6 shows that TFlex achieves better performance at similar power levels and lower power at similar power levels compared to Power4.

Figures 6.7 and 6.8 show the power and performance scaling curves for ARM, PowerPC, and TFlex, all normalized to the ARM core at the lowest voltage and frequency we examine (index 8 from Table 6.1). Both ARM and PowerPC show a linear relationship between power and performance while frequency is scaled up at V_{min}. The TFlex curves plot the pareto-optimal configurations from all three regimes, frequency scaling, composability, and DVFS. These configurations produce the best measured power and performance compared to other TFlex, Power4, and ARM configurations. Like ARM and PowerPC, frequency scaling at V_{min} and one TFlex core produces the best power and performance until reaching f_{maxeff}. At this point, keeping voltage and frequency constant while scaling the core count up to eight produces the best results. When core scaling reaches diminishing returns, returning to voltage and frequency scaling produces greater performance, albeit at

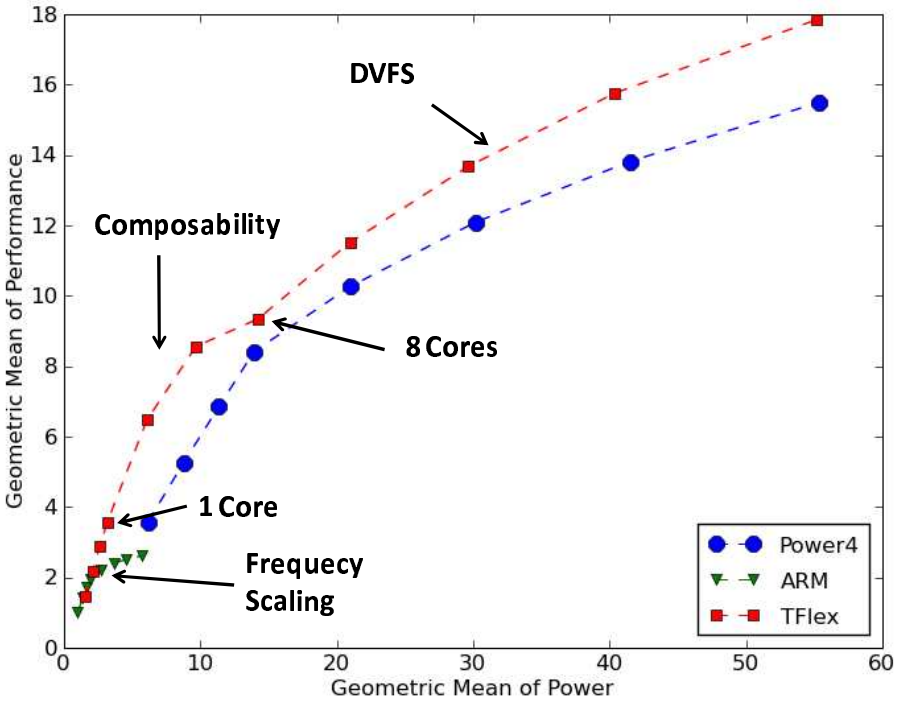


Figure 6.7: TFlex Pareto-Frontier: SPEC-INT

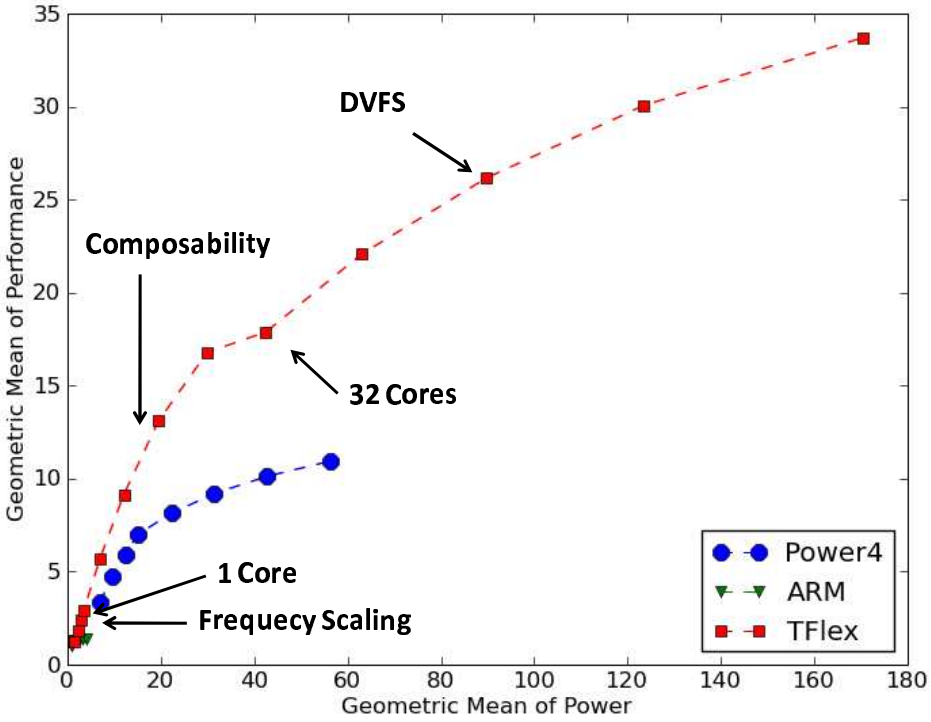


Figure 6.8: TFlex Pareto-Frontier: SPEC-FP

a higher power cost. The graph also shows the potential for composability as a means of bridging across the spectrum of architectures with different power and performance optimization points. Although ARM and PowerPC are designed for very different points in the spectrum, the composabilty of TFlex allows different configurations to operate in either regime.

In summary, for scaling up performance and power from a single TFlex core operating at v_{min} and f_{min}, the first best choice is to scale frequency only. Once v_{min} and f_{min} is reached, the second best option that provides the biggest performance benefit for the power cost is processor composability. This concurs with the observation from Section 6.4.1 that performance and power trade nearly one-for-one for at least a subset of the composable configurations. Once the tipping point of composability is reached after which adding cores ceases to improve performance, voltage and frequency scaling combined should be applied. Such scaling should be reserved for last as reaching f_{max} through voltage increases is power inefficient. For the purposes of scaling performance and power down, one must apply the above regimes in reverse order. In the regime where the supply voltage is greater than v_{min} and TFlex has more than one core, DVFS provides the best reduction in power. When v_{min} is reached, scaling down the number of cores provides the next best possible power reduction followed by pure frequency scaling.

6.4.3 Summary

Obtaining higher performance generally requires expending more power. For a conventional microprocessor chip that supports voltage and frequency scaling, improving performance with the best efficiency first comes by scaling the frequency while keeping the processor at its minimal possible voltage (v_{min}). Eventually, scaling will reach the maximum possible frequency attainable at this lowest voltage, which we term f_{maxeff}. If the system requires greater performance, frequency and voltage

must increase in concert, with each step being less power efficient than the last. Likewise, when aiming for lower power with the least drop in performance, conventional systems should perform the reverse of these steps: first scale down voltage and frequency to v_{min}, and then scale frequency down alone. Our experimental results show that composability offers an additional step, namely keeping voltage and frequency the same and changing the number of cores. In general, in a composable TFlex system, one can start with a single core at v_{min} and f_{min}, and keep increasing frequency to get better performance. This trend can continue until we reach the maximum frequency f_{maxeff} at V_{min}. At this point, to further improve performance, we can keep adding more cores until we reach the point of diminishing returns. Only after reaching this point does it make sense to increase the supply voltage to further increase the performance because of higher power overheads of increasing voltage. Additionally, combining DVFS with composability provides a richer trade-off space than with DVFS or composability alone, which can be exploited by system software policies.

6.5 Lessons

Increasing demand for energy efficiency requires mechanisms at all layers of system stack to trade off performance for power. Dynamic voltage and frequency scaling is a circuit-level technique, and has been the main-stay power management scheme in modern processors. Recent technology trends have significantly slowed down the rate of supply voltages, thus limiting the flexibility of DVFS. Our experience indicates that composability, a microarchitectural technique, is complementary to DVFS, and their combination can significantly improve the performance and power trade-offs provided by either technique. Going forward, we expect a deeper cooperation among all layers of the system stack, namely, silicon technology, circuits, microarchitecture, architecture, system software, and applications, to enhance energy efficiency.

Chapter 7

TFlex Performance Mechanisms: An Evaluation

This chapter evaluates a set of performance-enhancing mechanisms for the TFlex microarchitecture with respect to energy efficiency. Prior research has proposed a set of mechanisms to address certain overheads of the TFlex system and to subsequently improve its performance. We evaluate the effects of two such techniques on the energy-efficiency of TFlex: 1) Block Mapping policies that balance concurrency and communication and 2) Predicate prediction, a speculative technique that predicts the value of a predicate chain to improve performance. Finally, as mentioned in Section 5.2, the TFlex microarchitecture suffers from the overhead caused by "move" instructions used for fanning out operands. To address the performance overheads of such "move" instructions, researchers have evaluated an operand multicast/broadcast policy that aims to improve performance as well as energy efficiency. We summarize their findings with respect to the performance and energy improvement here as well.

7.1 Block Mapping Policies

In distributed architectures like Composable Lightweight Processors (CLPs), there exist two key determinants of good performance: program concurrency and data locality. TFlex composes many physical cores into a logical processor and exploits concurrency by mapping program blocks to different cores. On the other hand, when blocks are mapped to different physical cores, TFlex has to efficiently manage all inter-core communication on the operand network (OPN). For example, a producing instruction must be mapped closer to its consuming instruction to minimize communication over the OPN. Likewise, consumers of read and load instructions should be ideally mapped closer to the cores containing the register banks and data caches respectively. The OPN, a characteristic feature of distributed microarchitectures like TRIPS [92], RAW [111], and Intel Tera-Scale [118], is key to good performance and power efficiency in such distributed microarchitectures. Hence, reducing communication across the OPN could potentially improve both performance and energy efficiency in distributed microarchitectures.

Block-mapping policies for the TFlex microarchitecture determine how various TFlex blocks from a program binary are mapped to participating cores, as described by Robatmili et al. [88]. These policies provide a flexible software mechanism for TFlex to manage program concurrency and data communication across the OPN. Block-mapping policies are broadly classified into two categories: 1) fixed policies and 2) adaptive policies. While fixed policies use the same block-mapping technique for all blocks in a program, adaptive policies use a per-block mapping technique. Fixed block-mapping policies are further classified into 1) flat mode and 2) deep mode. The TRIPS and the original TFlex microarchitectures use the flat mapping mode [57] shown in Figure 7.1(a). In this example, the TFlex system has four participating cores and hence, four in-flight blocks. Each of the four blocks is striped equally across all four cores. In the deep mapping mode, each in-flight block

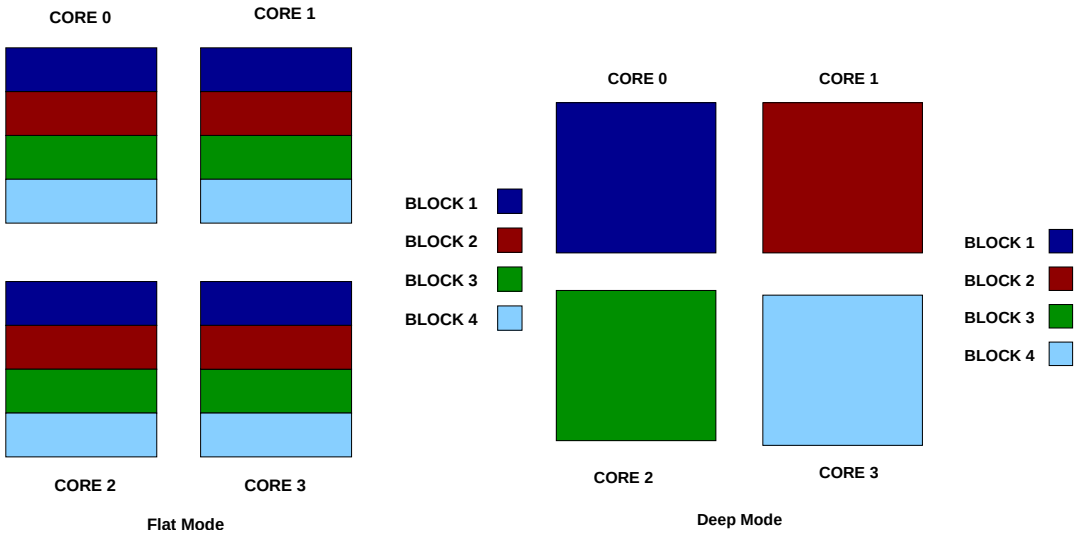


Figure 7.1: Block Mapping Policies

is assigned to a single participating core. As shown in Figure 7.1(b), each of the four in-flight blocks is mapped to a single core in the TFlex system. However, the register files, the data caches, and the load/store queues are partitioned equally across all participating cores in both the flat and the deep modes. Figure 7.2 reproduced from the paper by Robatmili et al. [88] is a concrete example that shows how two blocks, B0 and B1, are mapped differently in the flat mode (sub-figure (b)) and in the deep mode (sub-figure (c)) in a TFlex system with four cores. We note that although the instructions in the two blocks ('a' through 'h') are mapped differently in the deep and the flat modes, the registers (R0, R1, and R3) and data cache banks (D3) are mapped in the same manner in both the modes.

The flat and the deep modes have their own advantages and disadvantages in balancing extraction of concurrency and data communication across OPN. The flat mode can extract intra-block concurrency by mapping instructions from a single block to different cores. However, since each block is striped across multiple cores

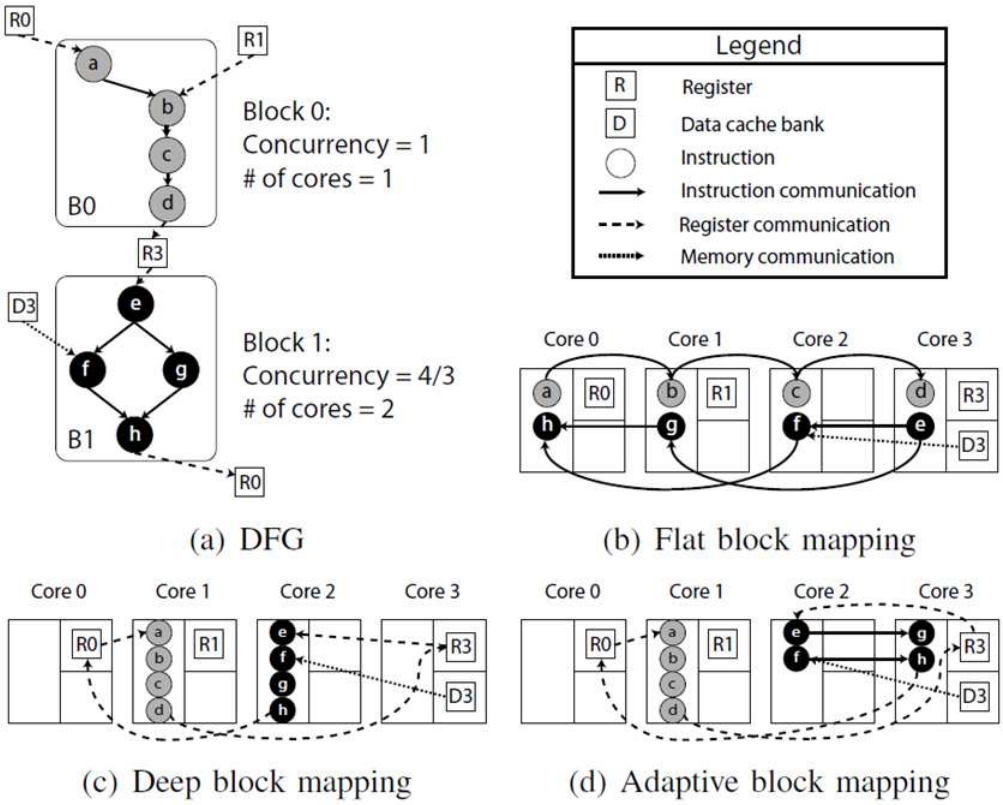


Figure 7.2: Block Mapping Example. Reproduced from the paper by Robatmili et al. [88]

the flat mode has more communication across the network, especially in higher core counts. On the other hand, since all instructions of the same block are mapped to the same core in the deep mode this mode decreases OPN traffic significantly at the expense of lesser intra-block concurrency.

Researchers have also proposed adaptive policies that determine the mapping technique on a per-block basis with the help of the compiler [88]. The TFlex compiler analyses each block for its intra-block concurrency. After this analysis, the compiler embeds a numeric value denoting the estimated concurrency into the block header. The concurrency value serves as a hint to the dynamic block mapper (an OS scheduler) to determine the right block-mapping policy and the number of allotted cores for this block. Sub-figure (d) in Figure 7.2 illustrates the adaptive block mapping policy. Various block-mapping policies have different concurrency versus OPN communication trade-offs and clearly have an impact on the performance and power profiles of the TFlex microarchitecture. Results from Robatmili et al. [88] indicate that the adaptive policy provides the best performance among all policies and is about 18% better than the best fixed policies. The results from Robatmili et al. [88] also show that as issue width of a TFlex core increases the performance boosts from the adaptive policy decreases, making the complexity of the adaptive policy not worth the effort.

For our analysis in this chapter, we consider dual-issue TFlex cores, and the effects of various fixed block-mapping policies on both the performance and energy-efficiency of TFlex. Since the performance boost from the adaptive policy decreases as we move from single-issue to dual-issue TFlex cores, we do not consider the adaptive policy for our comparison. However, we expect the adaptive policy to provide the best performance with lower power than fixed policies by virtue of its better trade-off between concurrency and OPN traffic. As mentioned in Section 4.1, we use the cycle-accurate TFlex simulator for our experiments with the block-

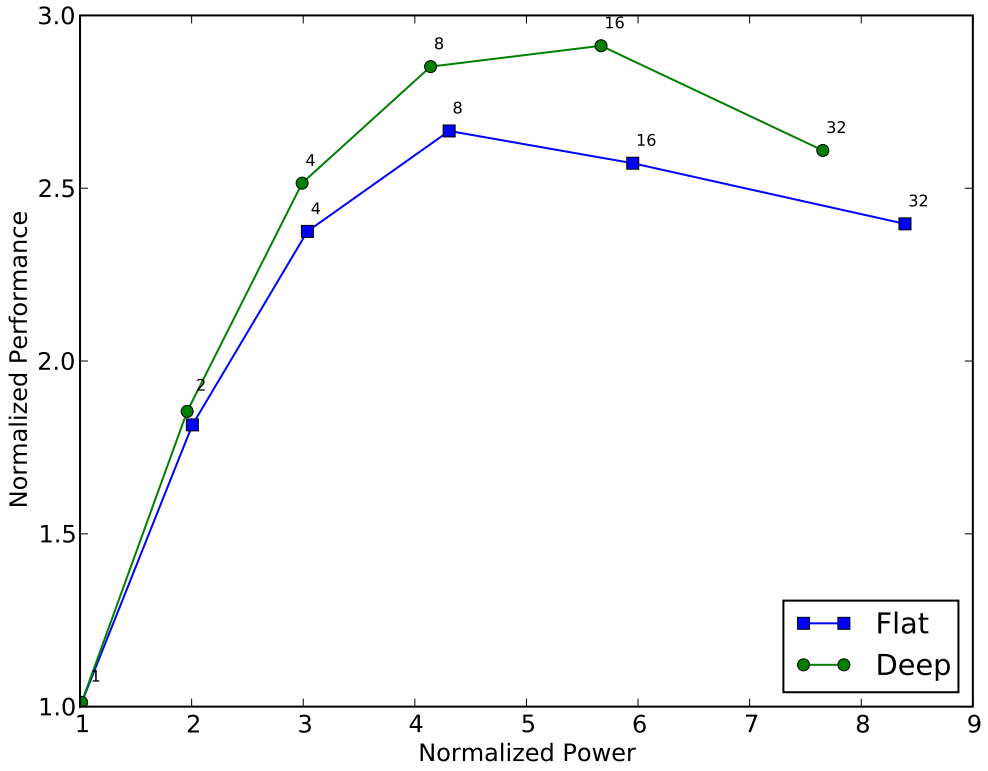


Figure 7.3: Flat vs. Deep Modes: Normalized Performance and Normalized Power

mapping policies. The TFlex simulator supports both the "flat" and the "deep" mapping policies, and we obtain the performance and power results for both of these policies. We use the same TFlex microarchitecture as described in Table 4.3, and the benchmarks listed in Table 4.1 for our evaluation.

7.1.1 Results

Figure 7.3 plots the geometric mean of normalized performance and normalized power for both the flat and the deep block-mapping modes. Normalized power is plotted on the x-axis and normalized performance is plotted on the y-axis. Both power and performance have been normalized to that of the 1-core configuration

of the flat mode. Each point on the flat and deep curves represents a TFlex configuration with a specific core count increasing from 1 to 32 cores (a total of six configurations for each mapping mode). We observe that for lower core configurations (1 and 2 cores), both the flat and deep modes are approximately similar to each in both performance and power. This is because with lower core counts communication over the operand network does not dominate the overall performance, and hence, both flat and deep modes perform similarly.

With increasing core counts, we observe that the deep mode outperforms the flat mode both in terms of performance and power. For example, in the 16-core configuration the deep mode performs about 15% better than the flat mode while consuming 3% less power. As the core count increases, the flat mode experiences increased OPN traffic among the cores (intra-block operands, registers, loads and stores). The performance loss due to this increased OPN traffic completely offsets the performance gains due to extraction of intra-block concurrency in the flat mode. On the other hand, since each TFlex core is dual-issue, the deep mode exploits good intra-block concurrency while experiencing reduced OPN communication. Thus, the deep mode outperforms the flat mode both in terms of performance and energy efficiency.

We plot the inverse Energy-Delay^2 Product (ED^{2}P) (perf^3/watt) for both deep and flat modes to assess the combined effects of the block-mapping policies on both performance and power. Overall, we observe that the deep mode achieves better EDP than the flat mode for all TFlex configurations except 1-core configuration. In the 1-core configuration, the deep and the flat mapping modes behave exactly the same due to absence of any inter-core communication. Thus, both block-mapping modes achieve exact same performance, power and EDP. With increasing core counts the deep mode either achieves better performance with same power or better performance with lower power compared to the flat mode, achieving better

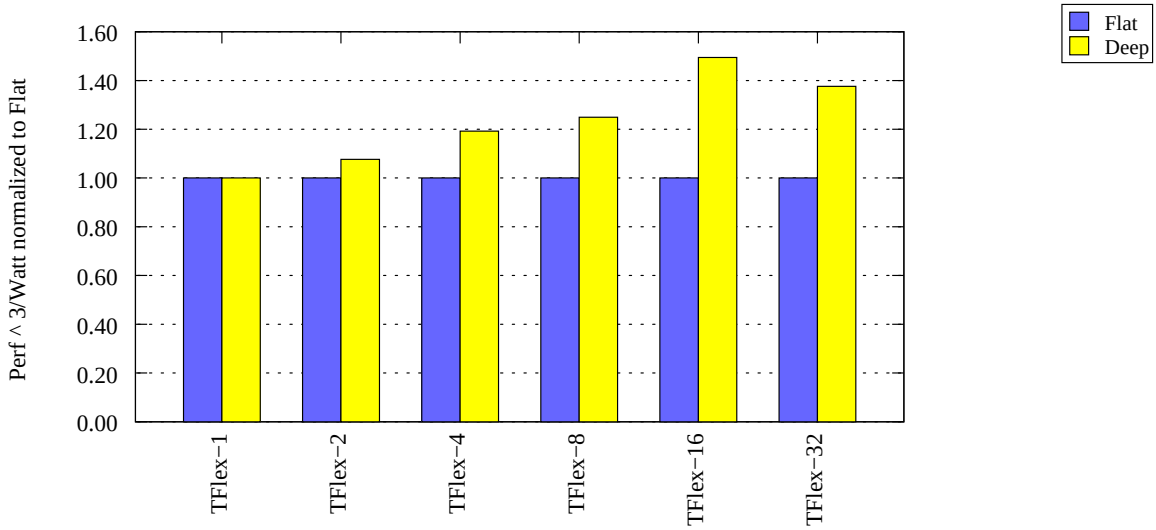


Figure 7.4: Flat vs. Deep Modes: Inverse Energy-Delay^2 Product Comparison

EDP. For instance, the 16-core TFlex configuration of deep mode achieves the best EDP of all TFlex configurations of both block mapping modes, and is approximately 50% better than its flat-mode 16-core counterpart.

To summarize, inter-core communication across distributed substrates like TFlex play a key role in both performance and energy efficiency. Higher level software policies that perform block allocations to programs must balance two key aspects of the TFlex system: program concurrency and traffic across the OPN. Our results indicate that in TFlex configurations with higher core counts the deep block mapping mode significantly reduces OPN traffic compared to the flat mode, and achieves better energy efficiency.

7.2 Predicate Prediction

Sustaining good instruction fetch bandwidth is a key determinant of good processor performance. This is especially true for processors like TFlex which implement a large instruction window. The TRIPS ISA, which the TFlex microarchitecture

implements, supports large instruction blocks containing up to 128 instructions. The TRIPS compiler employs techniques like if-conversion [101] to construct large hyperblocks, which are mapped to the distributed execution substrate in TFlex. Ideally, difficult to predict branches are converted to predicates, thereby converting the control dependence on the branch to a data dependence on the value of the predicate. Thus, the technique of predication enables the compiler to aggressively construct large hyperblocks, which is key to sustaining a good fetch bandwidth. On the downside, the newly introduced predicates must wait until execution time for their values to resolve. This additional waiting time for the predicate values could negatively impact performance. In order to minimize the performance impacts of predication, researchers have proposed the technique of predicate prediction. This technique predicts if a given predicate evaluates to true or false, and speculatively executes the instructions predicted to be on "true" path. Prior work has applied the concept of predicate prediction to the distributed TFlex microarchitecture [27]. In that work, the authors propose predicate prediction mechanisms that balance accuracy of prediction with the complexity of prediction in the distributed TFlex substrate. Like with any speculative technique, predicate prediction must be used with caution with respect to energy efficiency. While predicate prediction is essential to improve performance in TFlex, we seek to ensure that the energy overheads of the prediction mechanism – energy consumed by the prediction tables and components, and reissuing of instructions when the prediction is wrong – do not outweigh the benefits provided by prediction.

In the work by Esmaeilzadeh et al. [27] the authors evaluate the impact of predicate prediction on performance of the 16-core TFlex configuration. In this section, we evaluate the combined effects of predicate prediction on both the performance and power of the TFlex system. We compare two different versions of the TFlex system: 1) a baseline system with no predicate prediction and 2) a system that

supports predicate prediction. The TFlex simulator is augmented to support the **2KB GPHR.exit ⊕ CLPHR** predicate predictor as described by Esmaeilzadeh et al. [27], which achieved better prediction accuracy than other techniques, and within a reasonable area overhead. This predictor augments the base GEHL predictor in each TFlex core with a core-local prediction table for predicate prediction. The core-local table is indexed by the core-local predicate history register (CLPHR), and the global tables in the GEHL predictor are indexed using the Global Predictor History Register (GPHR) in each core. The predictions from the core-local table, the global and local GEHL tables are added up to make the prediction. The sign of the resultant sum provides the actual prediction whereas the confidence of the prediction is given by the absolute value of the sum. The **GPHR.exit** in this predictor stands for the way of constructing the global histories with the exit IDs of each branch prediction [27]. In this scheme, the compiler statically assigns suitable branch IDs to capture the predicate path leading to those branches.

We augment the baseline TFlex power models with CACTI-based power models for the predictor tables and power models of adders needed for prediction, and additional leakage power dissipated due to the increased area. Since the integer benchmarks of the SPEC suite are more control-intensive, and thereby result in more aggressively predicated code than floating point benchmarks, we restrict our analysis to the SPECINT benchmarks.

7.2.1 Results

Figure 7.5 plots the geometric mean of normalized performance and normalized power for four different configurations: `flat`, `deep`, `flat + predp`, and `deep+predp` modes, where `flat` refers to the flat mapping mode where all blocks are equally striped across participating TFlex cores, and `deep` refers to the deep mapping mode where each block is assigned to a single TFlex core in the system. `flat + predp`

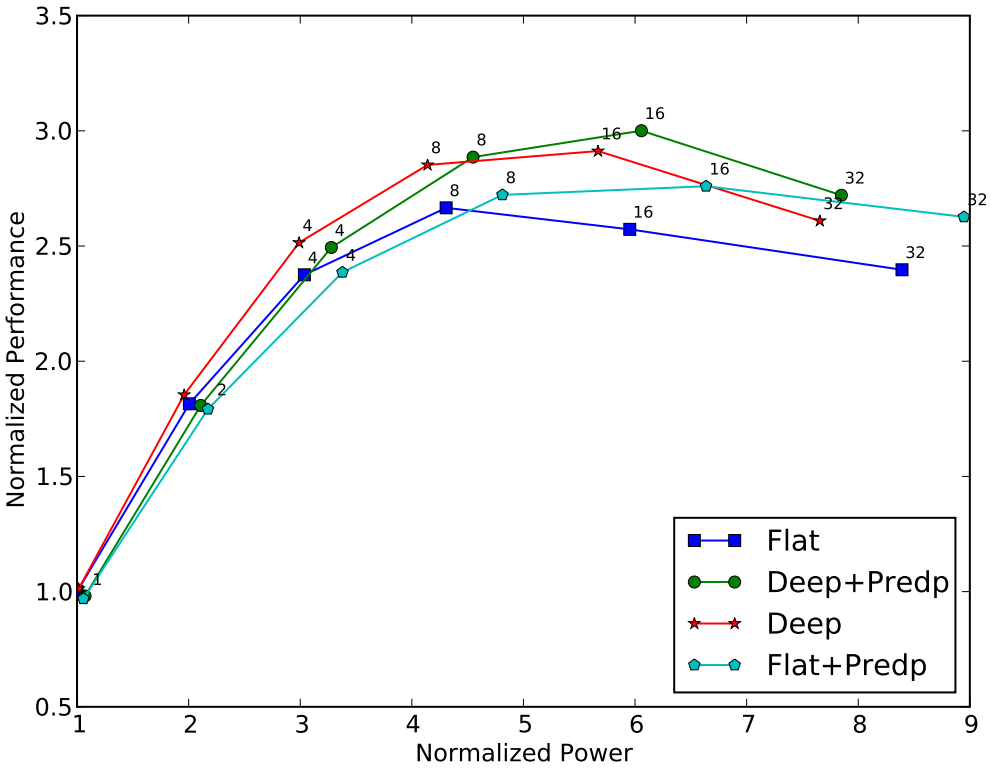


Figure 7.5: Deep vs. Deep + Predicate Prediction: Normalized Performance and Normalized Power

mode refers to predicate prediction enabled in the flat mode, and `deep + predp` refers to the deep mode with predicate prediction. Normalized power is plotted on the x-axis and normalized performance is plotted on the y-axis. Both power and performance have been normalized to that of the 1-core configuration of the `flat` mode. Each point on the `flat`, `deep`, `flat + predp`, and `deep + predp` curves represents a TFlex configuration with a specific core count increasing from 1 to 32 cores (a total of six configurations for each mapping mode). The curves are annotated with the core counts near each point. We observe that for configurations up to two TFlex cores, all the four modes are approximately the same in terms of both performance and power. At four cores, the modes with predicate prediction achieve approximately the same performance as their non-predicate prediction counterparts but at the expense of slightly higher power. From eight cores and upwards, predicate prediction shows marginal performance improvements over both the deep and the flat mode but at higher power levels. For example, the 32-core configuration in predp mode achieves about 4% better performance than its deep mode counterpart while consuming about 2% more power than the deep mode. Additionally, we observe that predicate prediction achieves better performance in the `flat` mode than the `deep` mode. We hypothesize that the predicate prediction accuracy is higher in the `flat` mode than in the `deep` mode, given the way instructions are mapped to different cores, and thus, predicate prediction achieves better performance boosts in the `flat` mode.

We plot the inverse EDP ($perf^3$/watt) for the four modes to assess the combined effect of predicate prediction on both performance and power. Overall, we observe that the `deep` mode achieves better EDP than the `deep + predp` mode for all TFlex configurations except the 16-core and 32-core configurations. This observation is also true with the `flat` mode. In the 32-core configuration, `deep + predp` mode achieves the maximum performance boost over the `deep` mode with

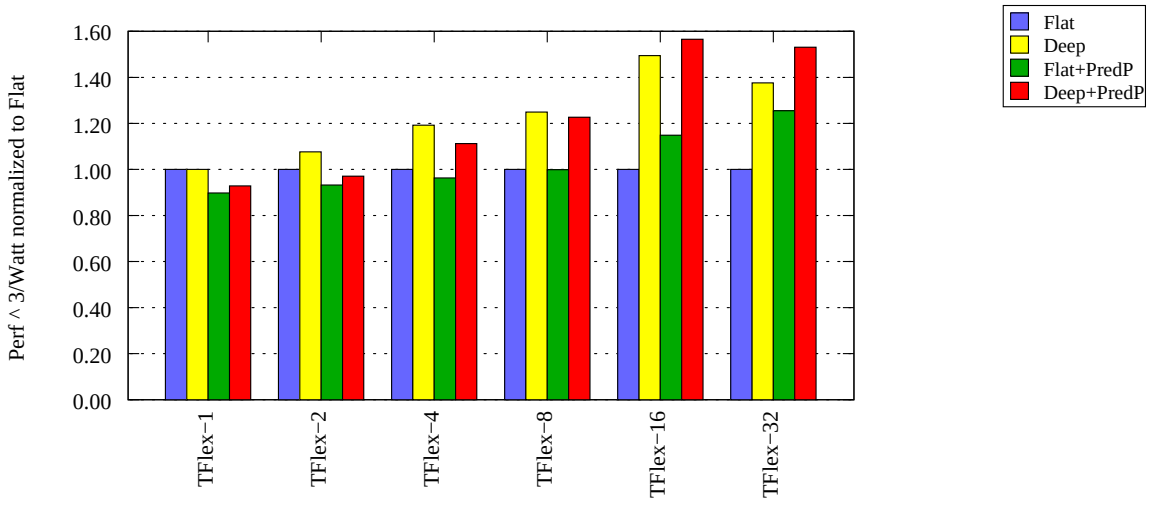


Figure 7.6: Deep vs. Deep + Predicate Prediction: Inverse Energy-Delay^2 Product Comparison

a minimal increase in power, thus achieving a 11% boost in ED^{2}P over the `deep` mode. Similarly, in the 32-core configuration, `flat + predp` mode achieves a 25% boost in ED^{2}P compared to the `flat` mode. With the other core counts the predicate prediction either achieves poor performance at similar power levels or better performance with even worse power levels.

Our experimental results indicate that predicate prediction, a speculative technique, does not always improve performance over a baseline without predicate prediction. For instance, predicate prediction provides only marginal performance improvement in the `deep` mode while adding energy overheads to the baseline. Hence, when energy efficiency is considered, predicate prediction achieves improvements in energy efficiency only with higher core counts. At lower core counts, the increased energy overheads of predicate prediction combined with only marginal performance improvements do not justify predicate prediction.

7.3 Operand Multicast

Another key overhead in the EDGE architecture implemented in TFlex stems from move instructions required to fan out values that are used by many consumers [68]. Unlike a conventional architecture which can employ the register file to broadcast values, a compiler for an EDGE architecture must build a software fanout tree of move instructions which can result in instruction execution overheads of up to 20% [68].

To target this overhead, researchers have examined a compiler-assisted hybrid instruction communication mechanism that augments a token-based instruction communication model with a small number of architecturally exposed broadcasts within the instruction window [64]. A narrow CAM allows high-fanout instructions to send their operands to their multiple consumers. All other instructions, which have low-fanout, rely on the point-to-point token communication model. The compiler determines statically which instructions use tokens and which use broadcasts. Few tags are required as the compiler can reuse broadcast identifiers for non-overlapping live range broadcasts.

Figures 7.7 and 7.8, reproduced from the results in [64], show the performance and power respectively of the hybrid broadcast scheme running single-threaded code on a composed 16-core TFlex system relative to a system without support for broadcast. With four overlapping broadcasts per hyperblock, the hybrid scheme achieves about a 5% speedup and 10% reduction in power for SPEC-INT. For SPEC-FP, the speedup and power reductions are mixed, due largely from the fewer number of move fanout trees in these benchmarks.

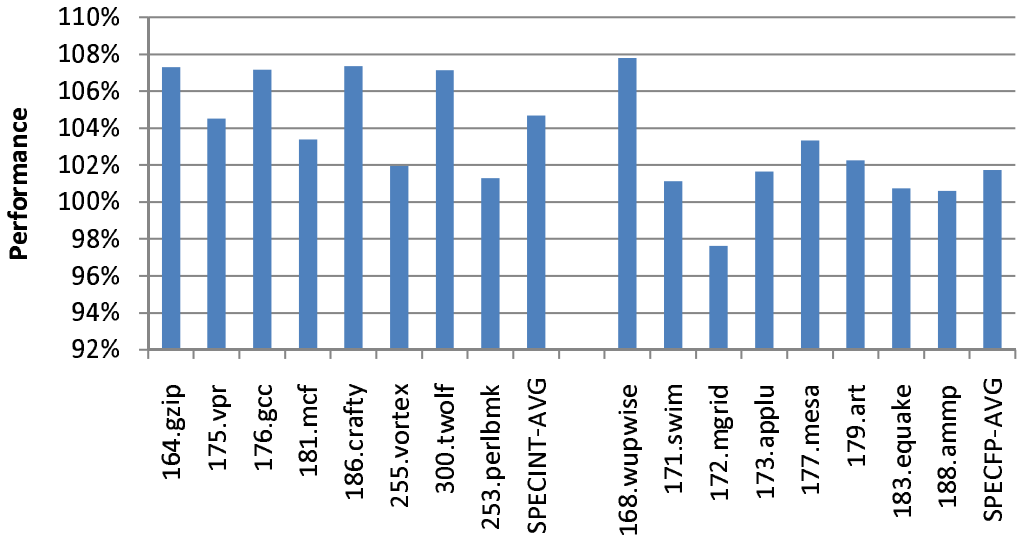


Figure 7.7: Performance of Limited broadcast support in TFlex. Figure reproduced from [64].

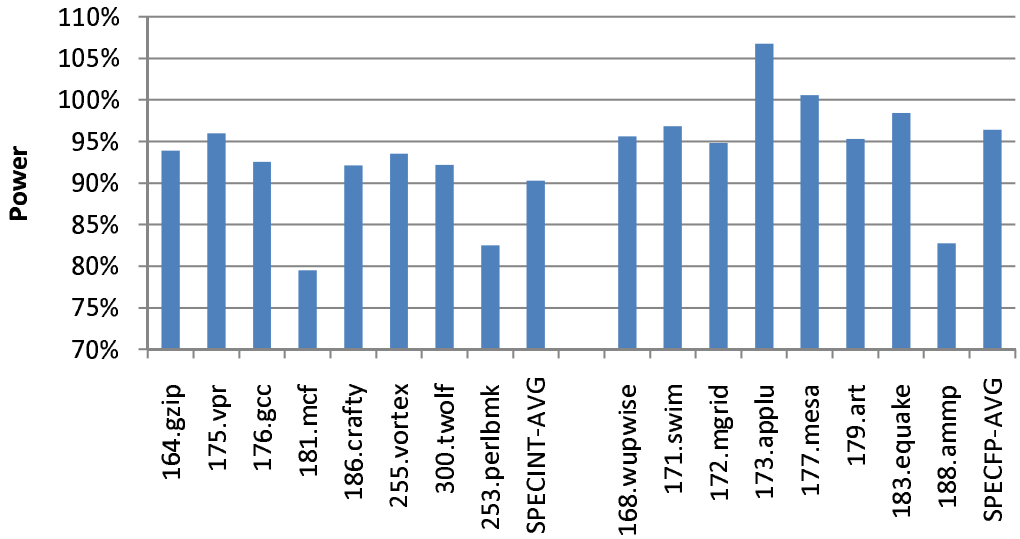


Figure 7.8: Power of Limited broadcast support in TFlex. Figure reproduced from [64].

7.4 Summary

In this chapter, we evaluate a set of performance-boosting mechanisms for the TFlex microarchitecture with respect to energy efficiency. These mechanisms include flexible block-mapping policies, predicate prediction and multicast/broadcast support for operands on the TFlex microarchitecture. Block-mapping policies trade off program concurrency and communication across the operand network in the distributed TFlex substrate. Predicate prediction, a speculative technique, aims to improve the performance of TFlex by predicting the value of predicate instructions in programs. The multicasting/broadcasting mechanism mitigates the performance impact by the additional `move` instructions used for operand communication. We examine the effects of these mechanisms in terms of both performance and power, and summarize the lessons learnt from our evaluation below.

- Data communication across distributed microarchitectures such as TFlex is a key determinant of both performance and power. The block-mapping modes, `flat` and `deep`, balance intra-block parallelism with communication across the operand network. By significantly reducing operand communication across the network, the `deep` mode achieves better performance `flat` mode. As an added advantage, by virtue of reduced communication the `deep` mode reduces power dissipation compared to `flat` mode. Hence, the `deep` block-mapping mode is a "win-win" strategy with respect to performance and power for the distributed TFlex microarchitecture.

- Like with any technique of speculation, predicate prediction should be implemented with caution. Although predicate prediction significantly improves performance at higher TFlex core counts in both `flat` and `deep` modes, the performance improvements are only marginal or sometimes negative at lower core counts. Considering the added energy overheads caused by this form of speculation, the marginal performance improvements at lower core counts do not justify the inclusion of predicate prediction.

- The TRIPS ISA employs the `move` instruction to fan out operands when a producer has more consumers than allowed by the ISA. These additional `move` instructions increase the dependence height between producers and consumers. The broadcast technique decreases the dependence height by adding hardware support for limited broadcast. While the technique itself adds more power consumption, it achieves an overall reduction in power due to reduced number of `move` instructions.

An interesting future research direction could look at the combined effects of all these mechanisms both in terms of performance and power. This dissertation evaluates these mechanisms mostly individually. However, because these mecha-

nisms are orthogonal to one another it would be interesting to study the combined effects of all mechanisms. Similar to the flexibility offered by the composability of the TFlex substrate, higher levels of system software combined with adequate compiler support could pick and choose various combinations of mechanisms that achieve the best performance and power characteristics for each application.

Chapter 8

Fine-Grained Power Management Policies

As described in Chapter 1, power dissipation is governed by the equation $P = CV^2f + VI_{off}$. The first term in this equation is the dynamic power component, and the second term is leakage. Various forms of power management exist in modern microprocessors including clock gating, power gating, and frequency throttling [119] that reduce power consumption by modulating different variables in the power equation. One such popular technique is dynamic voltage and frequency scaling (DVFS). The dynamic component of power dissipation is quadratically dependent on the operating voltage (V), and linearly dependent on the frequency (f). Since DVFS simultaneously modulates both the voltage and the frequency of the processor, DVFS typically achieves a cubic reduction in power combined with a linear reduction in performance.

Several researchers have used DVFS to trade performance for power in chip multiprocessors (CMPs) and in Multiple Clock-Domain (MCD) processors [49, 51, 65, 94]. Because modulating the power supply voltage requires adjusting the board-level voltage regulator circuit, DVFS has been historically applied to an entire chip

as a whole. Although the work by Isci et al. [49] assumes a fine-grained per-core DVFS implementation, they do not explore the design issues of on-chip Voltage Regulator Modules (VRMs), and the interplay between the policies and the fine-grained DVFS mechanism. Researchers at Harvard have recently shown the potential to build multiple on-chip VRMs, enabling the modulation of DVFS settings of individual cores/elements on a chip and providing very fast DVFS transitions (a few nanoseconds) [60]. These on-chip VRMs enable a temporally and spatially fine-grained DVFS mechanism. Distributed microarchitectures like TFlex, when combined with such a fine-grained DVFS mechanism, open up novel opportunities for fine-grained power management policies.

In this chapter, we explore the feasibility of two such fine-grained power management policies for the TFlex microarchitecture as novel contributions of this dissertation. The first policy seeks to run critical computation within a program at the highest performance levels, and slows down non-critical computation by mapping them to lower performance levels to improve overall power efficiency. The second policy explores the idea of using controlled speculation to achieve a better trade-off between performance and power. CLPs like TFlex provide an extremely flexible substrate to manage speculation, and thereby, trade off performance for power. In particular, we explore the idea of using the confidence of a branch prediction (block prediction in TFlex) to decide the DVFS state of the block. If a branch prediction has high confidence, we map the block to higher DVFS states, and vice-versa. For each of the above policies, we discuss the general principle underlying each policy, which is applicable to any architecture, followed by the implementation details on the TFlex microarchitecture in the sections below.

Although the DVFS mechanism has been extensively studied in prior research, the novelty of our contributions stems from applying a temporally and spatially fine-grained DVFS mechanism to CLPs like TFlex. Since the DVFS mech-

anism is spatially fine-grained, it can be applied to individual TFlex cores. This feature combined with the ability of TFlex to dynamically aggregate cores has the potential to enable better power management policies. For example, critical and non-critical components of a single-threaded program can be mapped to cores with different DVFS settings, and hence, different performance levels. Similarly, non-speculative and speculative components of the program can be mapped to cores with varying DVFS/performance levels. The block-oriented TRIPS ISA provides additional benefits by amortizing DVFS transition overheads among many instructions. However, combining fine-grained DVFS with TFlex opens up several challenges as well. Our novel contributions include exploring the detailed interplay between the policies and the fine-grained DVFS mechanism, and analyzing the various sources of overheads of the fine-grained DVFS mechanism.

In the rest of this chapter, we first discuss the fine-grained DVFS mechanism and its associated design issues. Next, we explore the critical policy with a limit study for gauging the upper bound on its potential. In this limit study, we idealize many parameters of the system design. Next, we explore a series of realistic designs where we incrementally remove the idealistic assumptions of the limit study. Next, we explore the policy that maps blocks based on branch confidence and wrap up the chapter with a summary of our findings and directions for future research.

8.1 DVFS Mechanism

The fine-grained power management policies discussed in this section utilize an underlying per-core DVFS mechanism. Conventional DVFS designs rely on VRMs located on the motherboard and only provide a temporally and spatially coarse-grained DVFS mechanism. In contrast, we consider on-chip VRMs that enable a temporally and spatially fine-grained DVFS mechanism. The key VRM characteristics are its energy conversion efficiency, load transient response, DVFS transition

time, and area [60]. Since these characteristics are highly interdependent, we require design-space explorations to finalize the VRM design for the TFlex microarchitecture. In this section, we outline the major challenges in the design of the fine-grained DVFS mechanism. In the following sections, we present the results of VRM design space exploration that was conducted in collaboration with Harvard researchers. Our primary focus for this chapter will be on the policies that leverage the DVFS mechanism and the challenges in making such a design work. Hence, we evaluate a few solution points from the VRM design space results, and we explore the two proposed policies.

8.1.1 Implementation Challenges

Area: Our fine-grained DVFS mechanism uses on-die VRMs to supply power to the TFlex cores. Since the VRMs are located on the die, the area of the VRM dictates the number of VRMs on the chip, and hence, the number of TFlex cores that can share a single VRM. At one end of the design spectrum, each TFlex core can have a dedicated VRM. At the other end, all TFlex cores can share a single VRM. The first option has the highest area overhead but enjoys enormous flexibility in terms choosing the right DVFS state for each core. The second option has the lowest area overhead but has limited flexibility in terms of DVFS mapping. To place the area overheads in perspective, our estimates show that if all 32 TFlex cores share a single VRM, it could occupy 12.5 mm^2 in 65nm technology, which is about 5% of the TFlex chip area. On the other hand, if each TFlex core has its dedicated VRM, the total area overhead would be 82 mm^2, which is 32% of the chip area.

DVFS transitions: During the transition from a DVFS state to another, the voltage transitions are not instantaneous, but occur gradually. Typical voltage transitions occur in the nanonseconds range (for example, 5 to 40 millivolts per nanosecond have been reported) [60]. These gradual voltage transitions result in

both performance and energy overheads in a real fine-grained DVFS system as the voltage cannot transition instantaneously. However, just like in prior work [30, 60], we assume that the processor clocks are kept running during the voltage transition and that frequency changes are effectively instantaneous.

Efficiency: Both on-chip and off-chip VRMs supply power for the operation of circuits by converting a higher voltage to lower voltages. This process of conversion incurs losses due to inefficiencies in the voltage regulator. Efficiency of VRMs, specifically on-chip VRMs, is a key characteristic, and is heavily interdependent on the area of the VRM. On-die VRMs with higher conversion efficiency typically occupy a larger area whereas VRMs with smaller area have lower efficiency. Any system using on-chip VRMs must carefully trade off energy efficiency for on-chip area. Efficiencies of 80-90% have been reported for on-die VRMs [60].

Synchronization: When fine-grained DVFS is combined with a CLP like TFlex, cores operating at different voltage/frequency levels need to communicate among themselves. This cross-domain communication requires a mechanism of synchronization across voltage/frequency boundaries, and is a key challenge that needs to be addressed in such a design. The synchronization issue has been encountered in various designs where different frequency domains interact [104], including Globally Asynchronous Locally Synchronous (GALS) processors [51]. The problem is mitigated by the use of synchronizers (synchronizing latches and dual-clocked FIFOs [82]) at the edge of the domains when data crosses from one frequency to another. While synchronizers greatly reduce the probability of the metastable state [104], they introduce extra latencies in the communication path. Since all TFlex cores communicate among themselves using the operand network (OPN), which replaces the conventional bypass networks, any extra latency in the OPN will directly affect the performance and will counterbalance the power advantages of DVFS.

Voltage (volts)	Frequencies (MHz)	
1.1	3000	2400
1.0	2000	1500
0.9	1000	800
0.8	500	400

Table 8.1: DVFS settings used

8.2 Experimental Setup

We use the TFlex platform as our experimental vehicle for exploring the fine-grained DVFS policies on CLPs. Our experiments described in this chapter use the cycle-level TFlex simulator described in Chapter 4.1 after necessary modifications. We add the support for modeling the interaction among different architectural components (cores, network routers, and L2 banks) each running at a different clock frequency. We also model the synchronization delays that are needed when communication happens across frequency domains. Finally, we also model the finite voltage transition delay penalty that occurs when shifting voltage levels. In our limit study, which gauges the upper bound on the DVFS benefits, we assume ideal synchronization delays, DVFS transition timings, and regulator efficiency. We assume the 65nm process technology, and the voltage and clock frequencies as described in Table 8.1, which differ from those used in Chapter 6. As listed in Table 8.1, each voltage level is assumed to support two different frequencies. The simulator models events at a frequency equal to the lowest common multiple of all frequencies supported by the system. Hence, we chose these settings to model a reasonable range of voltages and frequencies, and also, to minimize the performance impact on the TFlex simulator. We use all benchmarks as described in Table 4.1 except that we were only able to include 10 SPEC benchmarks (5 INT and 5 FP). For the other SPEC benchmarks, a few of the TFlex core configurations face long simulations times, and hence, were excluded from the study.

8.3 Instruction Criticality

We first evaluate an idea that runs computation critical to the performance of the program at the maximum performance level, and slows down non-critical computation by mapping it to lower performance levels. This idea finds many embodiments in prior research. Seng et al. [95] use a critical path predictor to predict if each instruction is critical to the overall program execution. If the instruction is critical, the microarchitecture steers that instruction to a high-performance, higher power pipeline. If the instruction is predicted non-critical, it is steered towards a low-performance, lower power pipeline. Other researchers have exploited this principle of criticality for better performance as opposed to reducing power. For example, Fields et al. [29] use a token-based critical path predictor for better instruction scheduling and improved value prediction.

TFlex Implementation: We evaluate an embodiment of this principle on the tiled and distributed TFlex microarchitecture. We leverage a simulation-based critical path tool designed for the TFlex microarchitecture [89] for identifying critical blocks in a program. Figure 8.1 describes the design of our criticality-based block mapping heuristic on the TFlex substrate. Although our implementation uses the concept of instruction criticality for power efficiency like others, the similarities end there. The fine-grained DVFS mechanism is expected to provide very fast DVFS transitions for each participating TFlex core. Also, since TFlex implements the block-based TRIPS ISA, we expect TFlex to better amortize DVFS transition overheads among many instructions. These aspects distinguish our work from prior applications of instruction criticality for power efficiency.

For each benchmark, we perform two runs on the TFlex simulator. In the first phase, we leverage the critical path tool described in [89] to categorize all committed blocks as either critical or non-critical. The tool uses a critical path model of the TFlex microarchitecture, similar to [75] but applied to TFlex, and

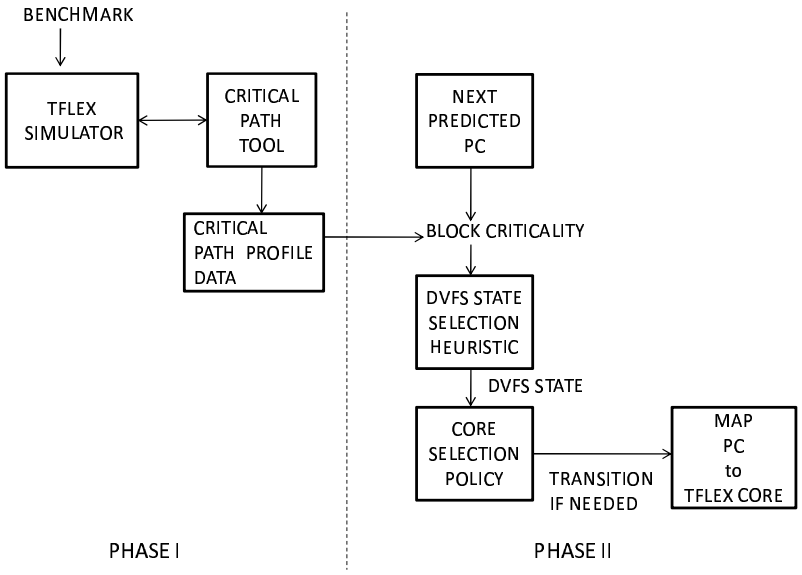


Figure 8.1: Criticality-based DVFS Mapping

analyzes the various microarchitectural events to identify the critical path for the benchmark. The critical path includes all the instructions that contribute to the longest path from the first block to the last block in the benchmark. We extend the notion of criticality to the block level, and classify every block that contains at least one critical instruction or event as critical. All the other blocks are classified as non-critical. Although there are many possible definitions of block-based criticality, we use the above definition because it is a natural and simple extension of instruction criticality to the block level.

The profile data (critical and non-critical blocks) from the first phase is passed onto the second phase. Typical studies that use profile data do so by using training data from a different input set than the actual input set. In our methodology, we leverage the training data from the same input set, which helps us gauge the upper bound on the power efficiency of this approach. During the second phase, each time the TFlex microarchitecture fetches a new block, the profile data from the first phase is consulted to check if this block is critical or not. If a block is predicted critical, then the block is mapped to a TFlex core with a higher DVFS setting. On the other hand, if the block is predicted non-critical, it is mapped to a TFlex core with a lower DVFS setting and is run slower with lesser power and to exploit the slack. We explore different heuristics that use different DVFS settings for critical and non-critical blocks, and study the effects on overall performance and power.

8.4 Limit Study

To begin our limit study, we first analyze the percentage of committed blocks that are categorized as critical by the critical path tool using the maximum voltage and frequency point in Table 8.1. Figure 8.2 shows the average percentage of critical (committed) blocks in different benchmark categories. The different benchmarks are plotted in the x-axis, and percentage is plotted in the y-axis. We note that only

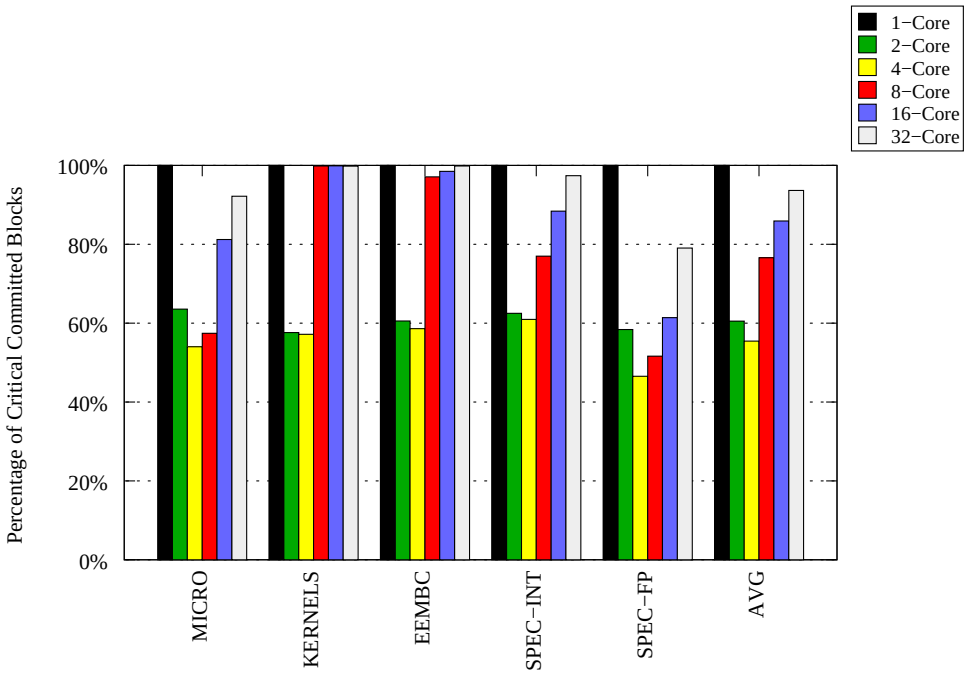


Figure 8.2: Percentage of Critical Blocks

committed blocks are considered by the critical path tool for this categorization, and all mis-speculated blocks are ignored in this analysis. For each benchmark category, the figure reports the percentage of critical blocks in all possible TFlex configurations ranging from 1 to 32 cores.

Overall, we observe that percentage of criticality starts at 100% for the 1-core configuration, and falls down to 60% in the 4-core configuration and increases to about 94% in the 32-core configuration. In the 1-core configuration, every block is run on a single core, and hence, every block ends up in the critical path. The percentage of critical blocks decreases in the 2-core and 4-core configurations due to increased parallelism in the system. However, when the core count increases beyond eight, the inter-core communication starts to dominate the latencies despite the increased extraction of parallelism. This causes the percentage of critical blocks

to increase with higher core counts. According to these results and according to Amdahl's law, we can theoretically improve the power efficiency by 40%, in the best case, for the 4-core TFlex configuration – assuming that the performance is unaffected, and all critical blocks are mapped to the maximum p-state and that the non-critical blocks consume no power. This theoretical upper limit varies with the number of cores, according to the percentage of critical blocks. In a realistic setup, the non-critical blocks must be executed to complete the program, and hence, the overall performance will drop down. Furthermore, the non-critical blocks must be executed at some non-zero frequency, however low that might be. All these factors would mean that in a realistic setup the improvement in power efficiency using this policy is likely to be far lower than 40% in the 4-core case, and even lower in other configurations.

We idealize all the parameters of the fine-grained DVFS system for the purposes of this limit study. This includes perfect or zero-cycle synchronization across frequency domains, perfect or instantaneous voltage transitions, and perfect regulator conversion efficiency. Additionally, we assume that there are unlimited number of VRMs to provide the fine-grained DVFS for all the components of the chip (cores, L2 banks, network routers, etc) as needed. These idealizations provide an upper bound for the benefits expected from fine-grained DVFS.

8.4.1 Effects on Performance and Processor Power

The first policy we consider maps all critical blocks to the maximum frequency (1.1 volts and 3 GHz) and all non-critical blocks to the state with 1.0 volts and 2 GHz (called C-Ideal policy). We compare this policy to two static policies: one which runs all the blocks at the 3 GHz (called as 3GHz policy) and another that runs all blocks at 2 GHz (2GHz policy). This comparison shows how much performance and power difference is expected from this policy when two frequencies are available at

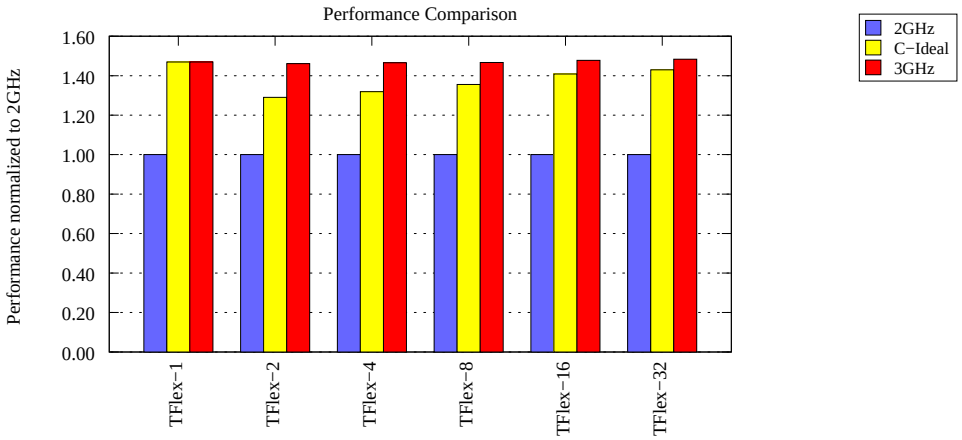


Figure 8.3: Performance Comparison with C-Ideal policy

hand. The 3GHz and 2GHz policies operate the entire chip (all the cores, L2 cache banks and network routers) at 3 GHz and 2 GHz respectively, whereas the C-Ideal operates the rest of the chip (L2 cache banks and network routers) at 3 GHz, and the cores at the desired frequency (3 GHz for critical blocks and 2 GHz for non-critical ones).

Figures 8.3, 8.4, and 8.5 compare the performance, processor power, and performance/watt of the three policies: 3GHz, C-Ideal, and 2GHz policies. The X-axis shows the various TFlex configurations, and the Y-axis shows performance, processor power, and performance/watt, each normalized to that of the 2GHz policy for each TFlex core configuration. Since the voltage conversion efficiency of VRMs is a key design criteria, we focus on the energy metric or its inverse, performance/watt, throughout this chapter. As Figure 8.2 indicates, all blocks are critical in the 1-core configuration. Hence, the C-Ideal policy maps all blocks to 3 GHz, and is identical to the 3GHz policy in all respects. Except for the 1-core configuration, the 3GHz policy outperforms the other two policies but at the cost of consuming the maximum power of all policies. The 2GHz policy, on the other hand, consumes the minimum

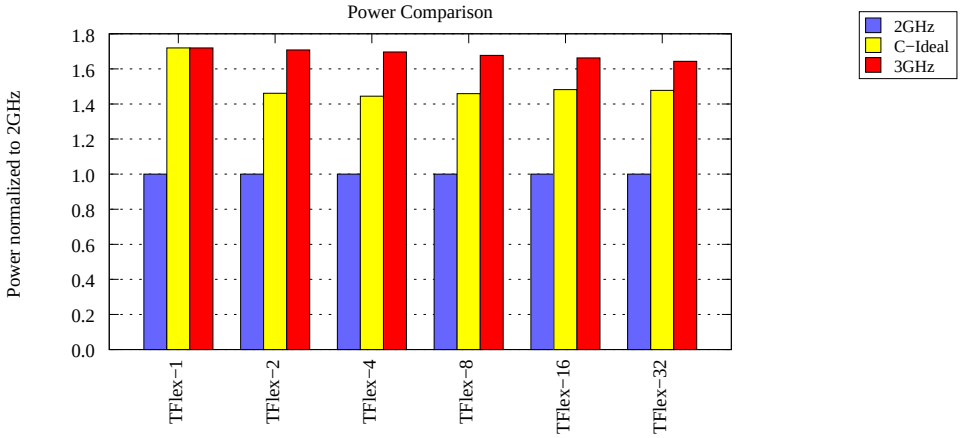


Figure 8.4: Power Comparison with C-Ideal policy

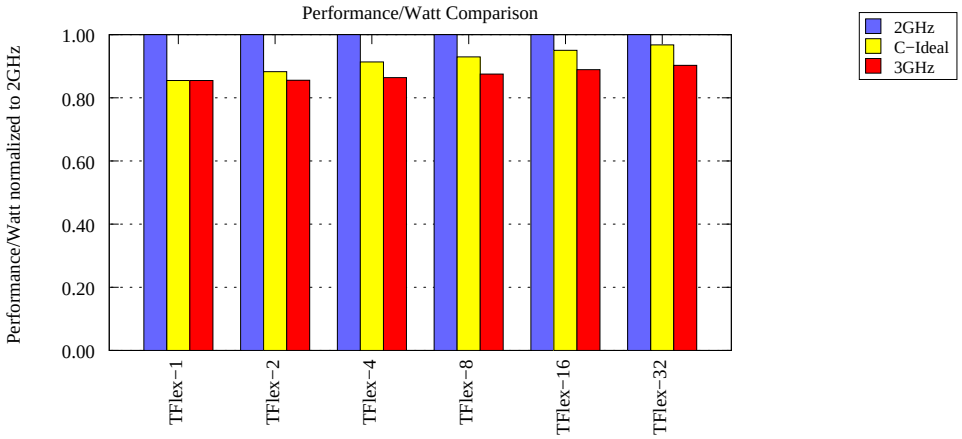


Figure 8.5: Performance/Watt Comparison with C-Ideal policy

power of all the policies, but also performs poorly compared to the other two. The C-Ideal policy, however, performs better than the 2GHz policy while consuming less power than the 3GHz policy. As the core count increases, as shown in figure 8.2, a higher percent of blocks become critical, and thus, the C-Ideal policy maps more blocks to 3 GHz, and so approaches the 3GHz policy in all respects.

Challenges: Ideally, the C-Ideal policy must achieve similar performance levels as the 3GHz policy because all the critical blocks are mapped to 3 GHz in the C-Ideal policy. However, from the above results, we observe that this is not the case, and C-Ideal policy falls short of the 3GHz policy. There are two key reasons for this performance drop. First, TFlex is a distributed microarchitecture where all hardware structures including the register file, data cache banks, and instruction windows are equally partitioned across the participating cores (when there is more than one core in the composed TFlex system). While the C-Ideal policy maps all critical blocks to 3 GHz the non-critical blocks are mapped to 2 GHz. Because many non-critical blocks execute concurrently with the critical blocks, certain cores are mapped to 3 GHz and certain other cores are mapped to 2 GHz. The critical blocks running at 3 GHz do need to communicate with cores running at 2 GHz mainly because the register files and the data cache banks are address-interleaved across all the participating cores. Thus, even though the critical blocks are run at 3 GHz they are invariably slowed down because of the necessary communication with the register file banks and data cache banks running at 2 GHz. From our analysis, this observation is the key challenge in marrying fine-grained DVFS policies with distributed architectures like TFlex.

Second, in the ideal case, when the non-critical blocks are slowed down and are run at 2 GHz one should not see any drop in performance. However, in reality, we do observe performance drops when non-critical blocks are run at 2 GHz. Typically, programs consist of multiple paths that are slightly shorter than the actual critical

path through the program. When the non-critical blocks are slowed down, it is possible that paths which are non-critical originally become critical. Due to this, the C-Ideal policy could experience performance drops compared to the 3GHz policy.

Effects on Performance/Watt: In terms of the inverse of energy or performance/watt, the 2GHz policy outperforms the 3GHz and C-Ideal policies in all configurations. This is because both 3GHz and C-Ideal policies rely on expensive voltage scaling to go from 2 GHz to 3 GHz (a voltage increase from 1.0 volt to 1.1 volts). This voltage increases brings along a quadratic increase in power but only with a linear increase in performance. So, in terms of performance/watt, 2 GHz policy shines. However, because the C-Ideal policy maps only a certain percentage of blocks to 3 GHz, it is marginally (4-6%) better than the 3GHz policy in terms of performance/watt.

Although the C-Ideal policy does not provide a linear improvement in performance and power like pure frequency scaling (a linear increase in power results in a linear increase in performance), the C-Ideal policy improves the flexibility and operating range of the TFlex system. These results indicate that the C-Ideal policy is only marginally successful in exploiting the slack from non-critical blocks. For example, the C-Ideal policy only provides a 15% drop in processor power and 11% drop in performance compared to the static 3GHz policy. In comparison, our ideal-case theoretical upper limit for the power reduction is 40% in the 4-core case with no change in performance. The distributed TFlex microarchitecture requires communication across cores running at different frequencies. Hence, the critical blocks are invariably slowed down compared to the static 3GHz policy. Furthermore, these results assume ideal synchronization, ideal voltage transitions with perfect on-die VRM efficiency. We expect these marginal improvements to dwindle further if these ideal assumptions are relaxed.

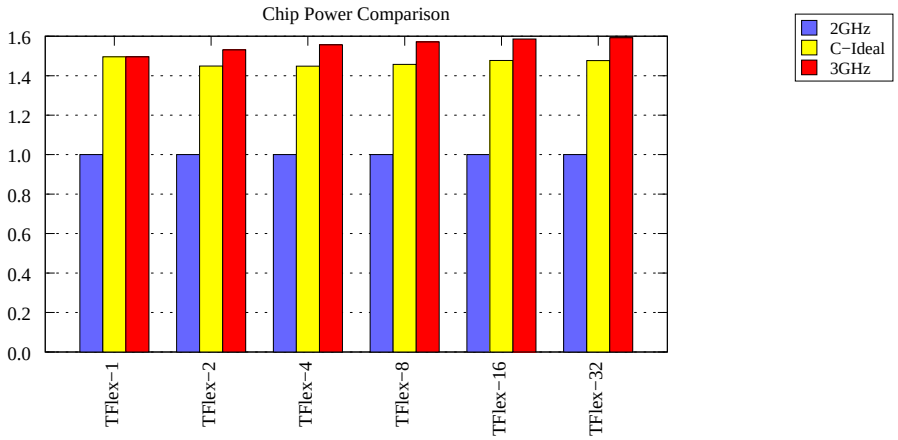


Figure 8.6: Total Chip Power Comparison with C-Ideal policy

8.4.2 Effects on Chip Power

The previous figures plot only the processor power and the performance/watt metric calculated with the processor power. The C-Ideal policy actively modulates the DVFS settings of the cores only, and these processor power metrics provide us a clear picture of performance and power differential of the critical policy. Figures 8.6 and 8.7 compare the total chip power and the performance/watt metric (inverse of energy) calculated with the chip power. We observe that the power and performance/watt advantage of the C-Ideal policy over the 3GHz policy diminishes when the total chip power is used for comparison. This is because both the 3GHz and C-Ideal policies run the L2 cache banks and network routers at 3 GHz, and the C-Ideal policy only modulates the frequencies of the cores. So, the only reduction in the overall chip power comes from the cores, and the amount of power reduction decreases when chip power metric is used.

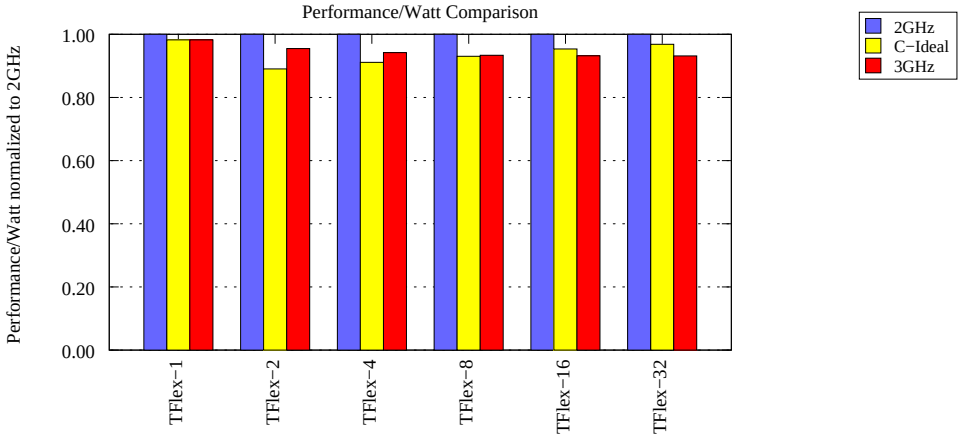


Figure 8.7: Total Performance/Watt Comparison using Chip Power with C-Ideal policy

8.4.3 Limits of Criticality-based Slack

Next, we explore a set of criticality-based policies to fully gauge how much slack we can exploit from the non-critical blocks. We study two more policies called C-IdealM and C-IdealL, which are similar to the C-Ideal in that they map critical blocks to 3 GHz but map the non-critical ones to 1 GHz and 0.8 GHz respectively. By mapping non-critical blocks to lower frequencies, these policies aim to further exploit the slack from non-critical blocks.

Figures 8.8, 8.9, and 8.10 compare the performance, processor power, and performance/watt of all the policies: 3GHz,C-Ideal,C-IdealM,C-IdealL, and 2GHz. The C-IdealM and C-IdealL policies have lower power consumption that the C-Ideal policy, but also perform worse. In the 2-core and 4-core configurations, the C-IdealM and C-IdealL policies consume either the same power or slightly more power compared to the 2GHz policy. However, the performance of these policies in the 2-core and 4-core case is also worse than the 2GHz policy. Hence, in terms of performance/watt metric, the C-IdealM and C-IdealL policies are outperformed

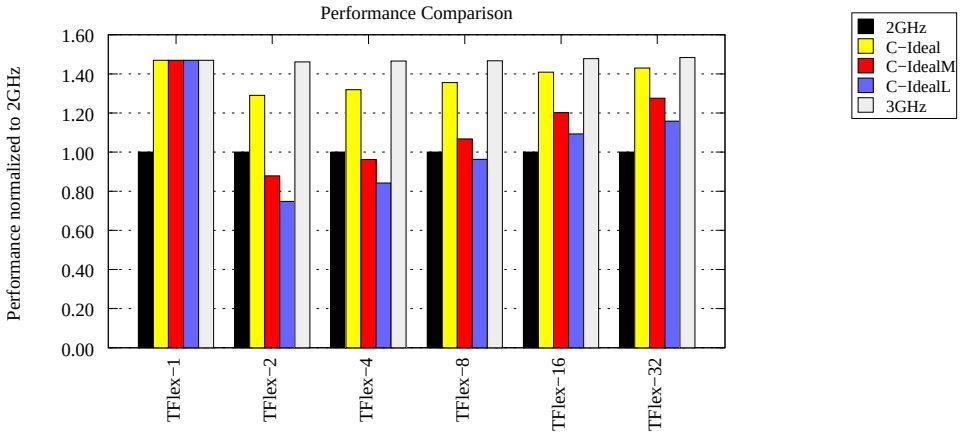


Figure 8.8: Performance Comparison with C-Ideal, C-IdealM and C-IdealL policies

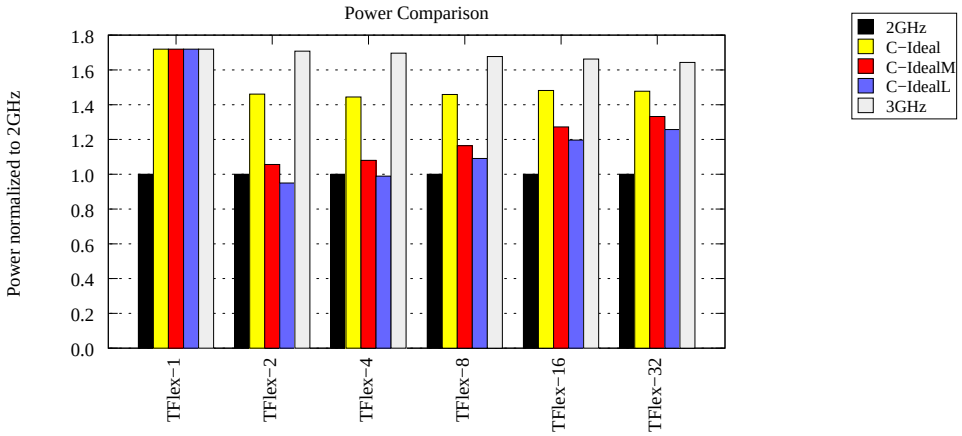


Figure 8.9: Processor Power Comparison with C-Ideal, C-IdealM and C-IdealL policies

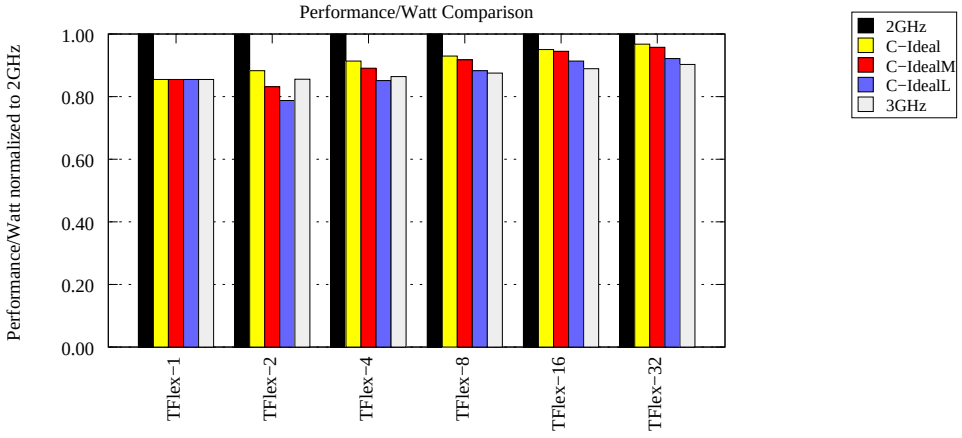


Figure 8.10: Performance/Watt Comparison with C-Ideal, C-IdealM and C-IdealL policies

by the 2GHz policy, and even by the 3GHz policy. At higher core counts, the C-IdealM and C-IdealL catch up to the 3GHz policy in terms of performance/watt, but only beat it marginally. This result adds further testimony to the fact that even by mapping non-critical blocks to lower and lower frequencies, the criticality-based policies are unable to exploit any slack. This, we hypothesize, is mainly due to the percentage of critical blocks at various core counts, which is noticeably high.

8.4.4 L2 Caches

All the policies studied above modulate only the core frequencies by assuming that the L2 cache banks are run at the maximum frequency. Additional power benefits could be reaped if the L2 cache banks are operated at lower frequencies. To estimate the possible power savings by potentially running L2 at slower frequencies, we explore two other policies called C-Ideal-L2 and C-Ideal-L2-Low, which area similar to the C-IdealM policy except that the L2 cache banks are operated at slower frequency of 1.5 GHz and 1.0 GHz respectively.

Figures 8.11, 8.12, and 8.13 show the performance, total chip power, and per-

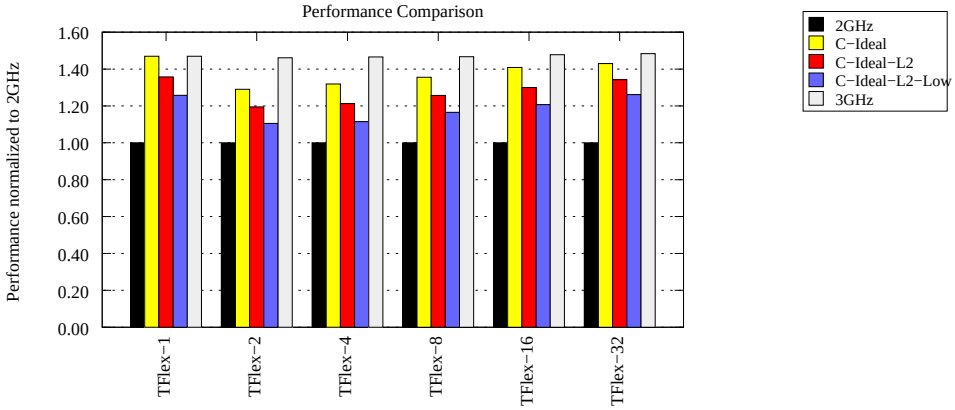


Figure 8.11: Performance Comparison with C-Ideal-L2 policy

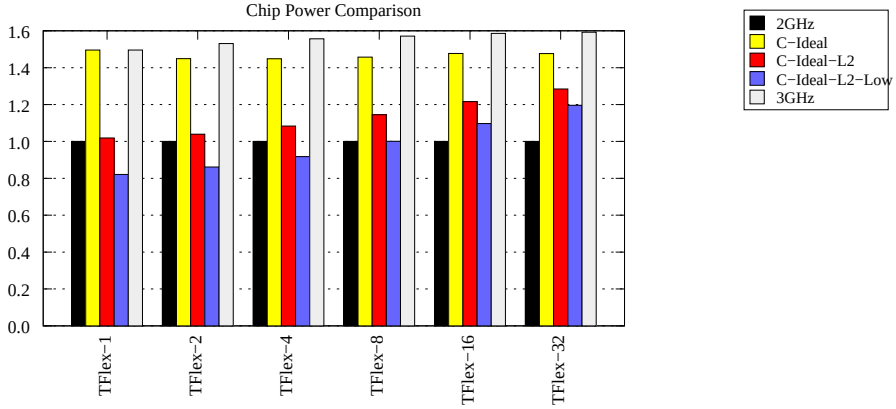


Figure 8.12: Chip Power Comparison with C-Ideal-L2 policy

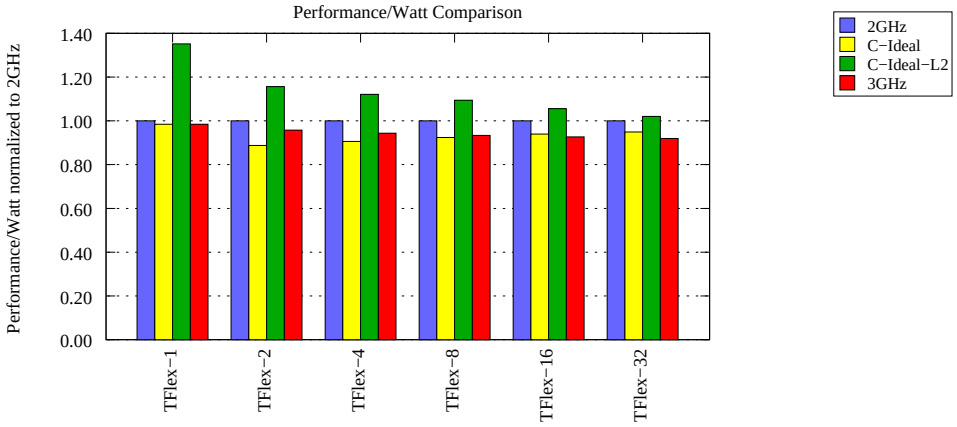


Figure 8.13: Performance/Watt Comparison with C-Ideal-L2 policy

formance/watt for C-Ideal-L2 and C-Ideal-L2-Low policies. From these results, we observe that C-Ideal-L2 and C-Ideal-L2-Low provide further reduction in total chip power compared to C-Ideal with good improvements in performance/watt. The C-Ideal-L2-Low and C-Ideal-L2 policies outperform C-Ideal, 2GHz and 3GHz policies, by simultaneously exploiting instructions criticality and slack in the L2 sub-system. These results point to the advantage of having an independent VRM controlling the L2 cache banks, and the flexibility and improvement in power efficiency offered by such a design. By operating the L2 cache banks at even lower frequencies or different L2 banks at different frequencies according to demand, more power efficiency benefits are expected. Fully exploring the design space of modulating the processors and the L2 cache banks is beyond the scope of this dissertation.

Overall, the ideal critical policies extend the operating range of TFlex, and marginally outperform the static 2GHz policy while consuming marginally lower power than the static 3GHz policy. In terms of overall energy usage or its inverse performance/watt, all the ideal critical policies outperform the static 3GHz policy, which has the best performance, but is outperformed by the 2GHz policy due to the usage of costly voltage scaling in their criticality-based mapping. Also, when operat-

ing under tight chip-level power budgets, such policies are necessary to extract more performance from the system within the power budget. Our experimental results show that having an additional VRM to operate the L2 cache banks independently improves the power efficiency of the overall chip in terms of the performance/watt metric.

8.5 Realistic Synchronization

The previous section performs a limit study of the various criticality-based policies with ideal synchronization, ideal voltage transition, unlimited number of VRMs, and ideal regulator efficiency. In this section, we relax the assumption of ideal synchronization across frequency domains and study the ensuing effect on performance and power of the TFlex system. We assume the presence of dual-clocked synchronizing FIFOs [82] at the boundary of the different frequency domains. Whenever the transported data crosses frequency domains, for example from a core to its attached operand network router or from a core to its attached L2 cache bank router, we assume that the data is transmitted and received using these dual-clock FIFOs. Although the synchronizing latencies could vary [104], a synchronizing delay of at least one clock cycle is necessary for virtually all synchronizers. We model a synchronization cost of one cycle when data crosses different frequency domains. We further assume that when data is transported across the same frequency we do not have to pay the synchronization penalty. Finally, we ignore the area and power overheads of the dual-clocked synchronizing FIFOs.

All TFlex cores are tightly coupled by the operand network (OPN) which transports different types of operands among the TFlex cores: register, memory, and intra-block operands. Because the OPN replaces the conventional bypass networks, any extra latency in the OPN will have a major impact on TFlex performance. To mitigate this performance impact, we explore two different solutions. The first

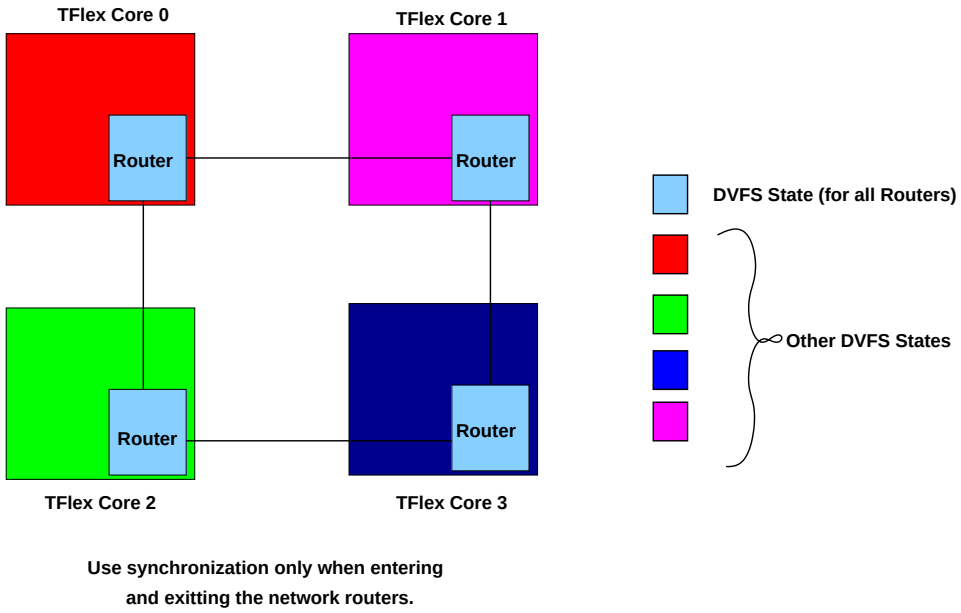


Figure 8.14: Synchronization Optimization

solution is to use the "deep" block-mapping strategy as discussed in Chapter 7. The "deep" strategy maps every block to a dedicated TFlex core, thereby minimizing the intra-block communication overhead compared to the "flat" strategy. As results in Chapter 7 show, the "deep" strategy outperforms the "flat" strategy both in terms of performance and energy efficiency. Hence, using the "deep" strategy is a win-win situation: better performance and energy efficiency overall as well as reduced synchronization with fine-grained DVFS policies.

The second technique to mitigate synchronization delays is to decouple each OPN router in a TFlex core and to run the OPN router in a power and frequency domain different from that of the cores. Figure 8.14 describes this design where in the worst case (where every TFlex core is at a different frequency), a data packet would incur a 2-cycle bubble throughout its entire path (1-cycle bubble each during entry and exit from the network) as opposed to a 1-cycle bubble for every hop.

To study the effects of realistic synchronization, we relax the assumption of ideal synchronization with a penalty of 1 cycle to cross from one frequency domain

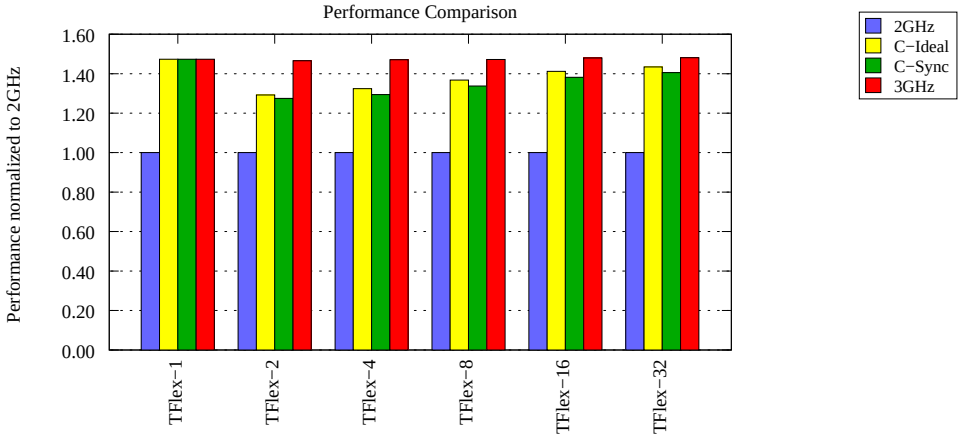


Figure 8.15: Performance Comparison with C-Sync policy

to another. We also apply the optimization described in Figure 8.14 to our design, and study the effects with C-Ideal policy.

Figures 8.15, 8.16, and 8.17 describe the effects of realistic synchronization on performance, processor power, and performance/watt respectively. These figures compare the 2GHz,C-Ideal,C-Sync and 3GHz policies, where C-Sync represents the C-Ideal policy but with real synchronization. We observe that C-Sync policy performs almost the same as the C-Ideal policy and is mostly with 2% of the performance and 1% of power of that of the C-Ideal policy. The network optimization described in Figure 8.14 proves extremely effective in mitigating the performance impact of real synchronizations. These results indicate that the extra overhead of having an independent VRM for the network routers, and providing dual-clocked FIFOs at network boundaries is definitely worth the overhead.

8.6 Realistic DVFS Transition Times

This section further relaxes the idealistic assumptions of our initial limit study by examining the effects of real voltage transition times. The previous sections assume

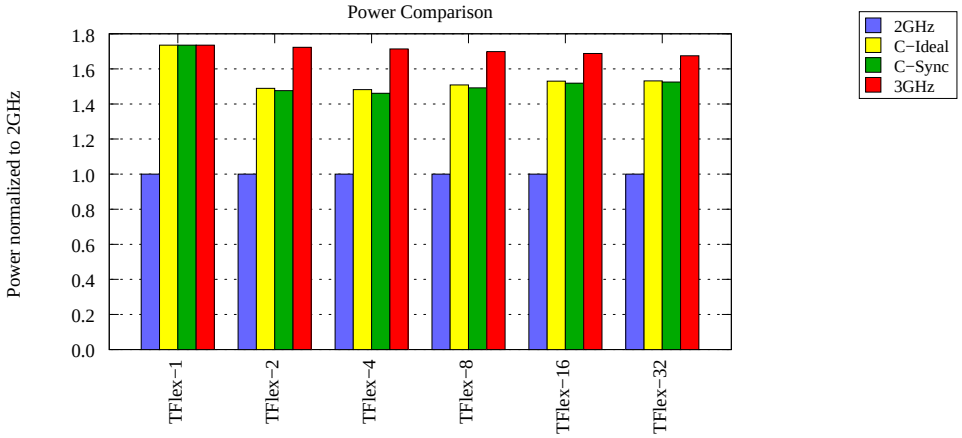


Figure 8.16: Processor Power Comparison with C-Sync policy

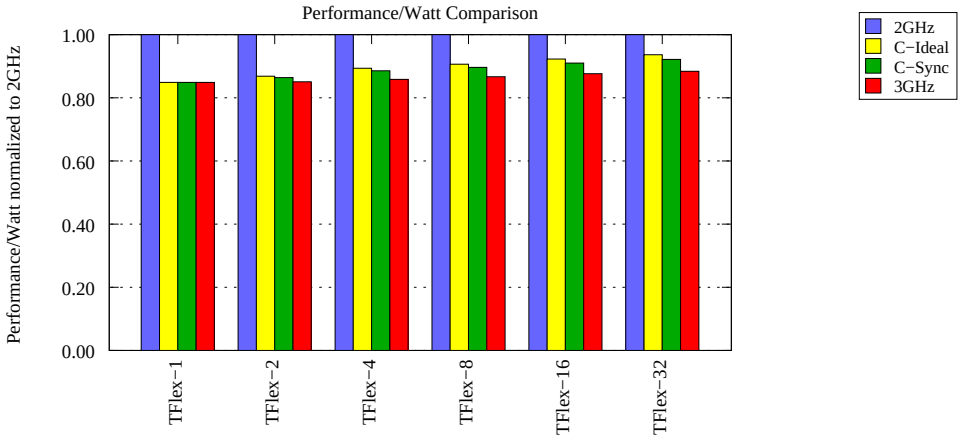


Figure 8.17: Performance/Watt Comparison with C-Sync policy

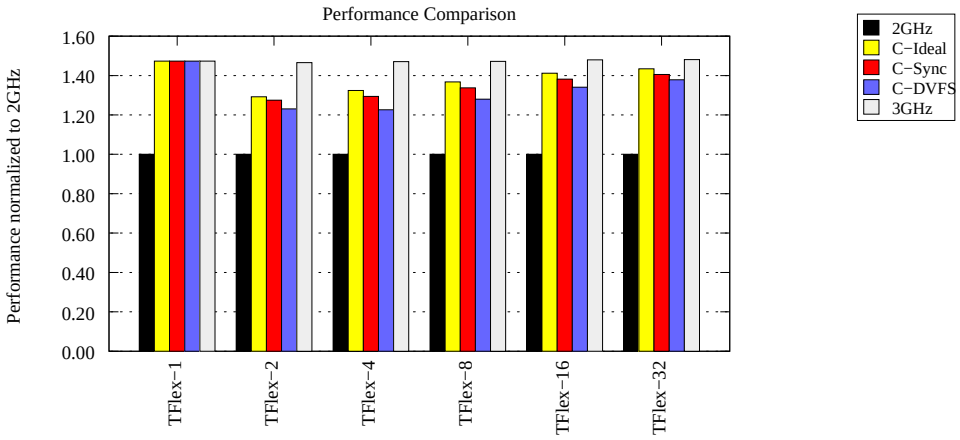


Figure 8.18: Performance Comparison with C-DVFS policy

an instantaneous transition in voltage levels when moving from one DVFS level to another. Voltage transition times of conventional off-chip VRMs are in the range of several microseconds to milliseconds whereas on-chip VRMs enable very fast DVFS transitions in the range of few tens of nanoseconds [60]. For our experiments in this section, we assume a very conservative voltage transition time of 5 millivolts per nanosecond. With this transition slope, it would take 20 nanoseconds for the voltage to transition a full 100 millivolts. We plug in this voltage transition time into the TFlex simulator to study the effects of realistic voltage transitions. As mentioned before, during an upward voltage transition our models ramp up the voltage first before ramping up the frequency. Likewise, during a downward voltage transition our models first ramp down the frequency followed by a ramp-down of the voltage. This ensures that the circuits do not encounter timing errors during voltage transitions [60].

Figures 8.18, 8.19, and 8.20 describe the effects of realistic voltage transition times on performance, processor power, and performance/watt on the C-Ideal policy. These graphs plot the various policies for reference: 2GHz, 3GHz, and C-Ideal

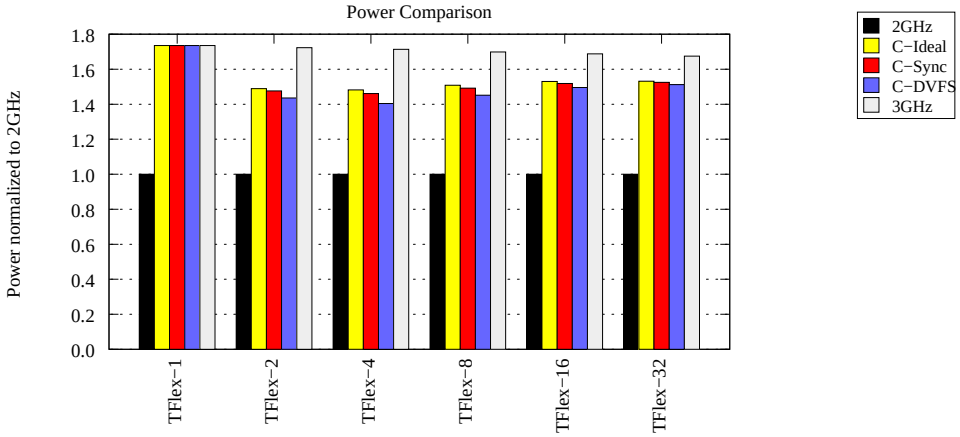


Figure 8.19: Processor Power Comparison with C-DVFS policy

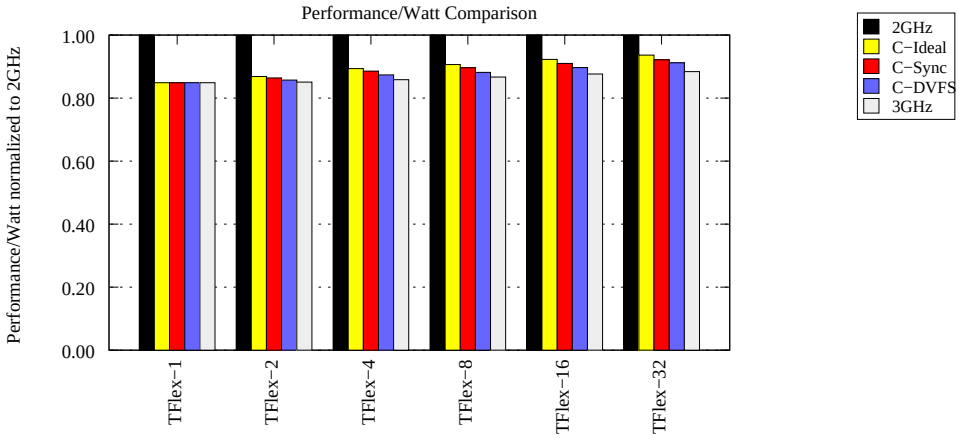


Figure 8.20: Performance/Watt Comparison with C-DVFS policy

are the policies discussed above. C-Sync is the C-Ideal policy with realistic synchronization, and C-DVFS is the C-Sync policy with realistic voltage transitions. These results indicate that the realistic C-DVFS policy achieves about 4-5% of both the performance and power of the C-Sync policy. Similarly, the C-DVFS policy is within 8% and 5% of the performance and power the C-Ideal policy respectively. These results indicate the potential of on-chip VRMs, and their fast transition times. Even a conservative voltage transition time of 5 millivolts per nanosecond enables the C-DVFS policy to achieve performance and power levels of within 5% of a similar policy that assumes instantaneous voltage transitions.

8.7 VRM Area versus Efficiency

In this section, we analyze the area overheads and energy efficiency overheads of on-chip VRMs and the interdependence between their area and energy efficiency. We explore the design space of on-die VRMs with the help of our collaborators at Harvard. Figure 8.21 is a scatter plot of VRM area overhead and VRM loss at different voltages of interest. The VRM area overhead is plotted in the y-axis in $mm^2/Watt$ (the watts in the denominator denotes average dissipated power), and the VRM loss is plotted in the x-axis as a percentage. If the VRM loss is, say, 20%, and if the power delivered to the VRM is 20 watts, then only 16 watts (80% of 20 watts) is actually available for the VRM load and the remaining 4 watts is lost in the process of voltage conversion in the VRM. The scatter plot clearly shows the trade off involved in VRM area and conversion loss. If we wish to minimize the area overheads of on-die VRMs (lower y values), we should pay for the choice with higher conversion losses of the VRM (higher x values). On the other hand, if we attempt to minimize regulator losses, large area overheads must be incurred.

From these scatter plot results, we choose two different VRM designs for our evaluation. The first design minimizes the area overhead of the VRM while suffering

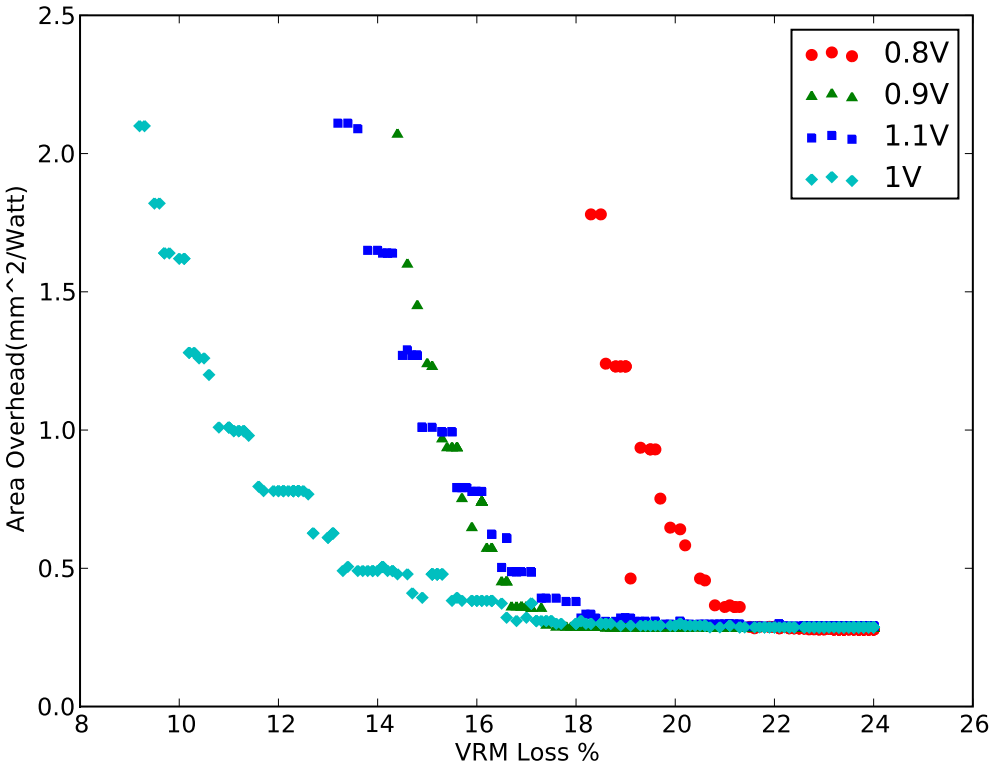


Figure 8.21: VRM Design Space: Area vs. Efficiency

VRM Category	Area Overhead ($mm^2/Watt$)	Voltage (volts)	VRM Loss (%)
Small Area	0.5	1.1	17%
		1.0	13%
		0.9	17%
		0.8	19%
Large Area	1.0	1.1	15%
		1.0	11%
		0.9	15%
		0.8	19%

Table 8.2: VRM Area vs Loss Comparison

Number of VRMs	Small Area		Large Area	
	VRM Area (mm^2)	VRM Area Over-head (%)	VRM Area (mm^2)	VRM Area Over-head (%)
1	12.5	5.0	25.0	9.9
2	17.1	6.8	34.2	13.6
4	24.3	9.7	48.6	19.3
8	33.2	13.2	66.5	26.4
16	66.4	26.4	132.8	52.8
32	81.9	32.6	163.8	65.1

Table 8.3: VRM Area vs Loss Comparison

from higher losses. The second design has one of the lowest losses but incurs a high area overhead. Table 8.2 tabulates the area overheads and conversion losses of these VRMs at different voltages. As we observe from this table, the conversion losses of the VRMs is dependent on the voltage level. Typical VRMs, both off-chip and on-chip, attain their peak efficiency at a certain voltage level and their efficiency falls off at other voltages. The exact voltage at which the peak efficiency is reached depends on the type of the voltage regulator and its individual characteristics. We observe this trend in Table 8.2, where both the VRM designs reach their peak efficiency (minimum loss) at 1.0 volts and their efficiency drops off at other voltages.

To fully evaluate the area overheads of on-die VRMs we use the 32-core TFlex microarchitecture with 4-MB of NUCA L2 cache. Our area models indicate that this chip occupies an area of 252 mm^2 in the 65nm process technology. Table 8.3 tabulates the area estimates and the ratio of VRM area to the chip area (without the VRMs) with varying number of on-die VRMs (ranging from 1 to 32). The table includes estimates for both types of VRM designs: smaller area-lower efficiency

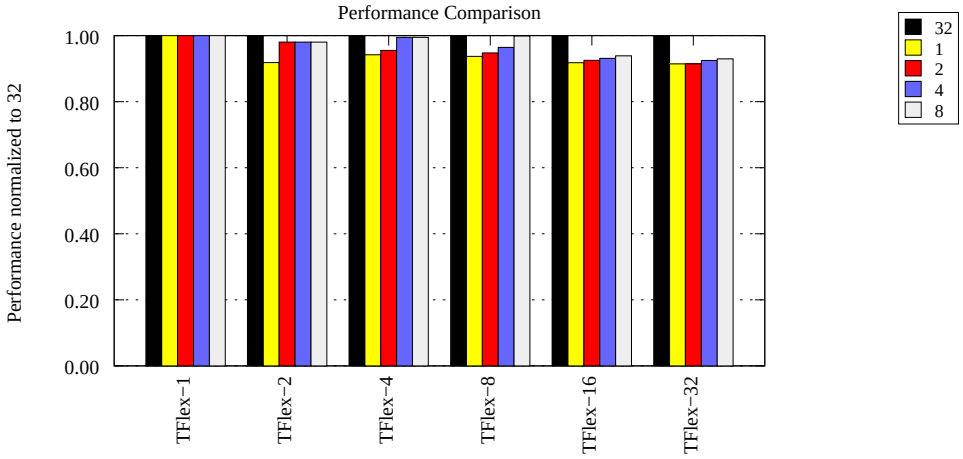


Figure 8.22: Performance Comparison with limited VRMs

and larger area-higher efficiency. If the chip includes only one VRM for the cores, the area overhead is minimal (around 5% for the smaller area design). However, this design is inflexible in that when one core needs to transition to a new DVFS setting, all the 32 cores have to undergo the transition as well. At the other end of the spectrum, if the chip includes 32 independent VRMs, one for each core, the design is more flexible as each core can independently transition its DVFS setting. However, this option has a huge area overhead, which can be up to 65% in the large area VRM design. We study a series of designs where we vary the number of on-die VRMs from 1 to 8, and compare these designs to the 32-VRM design. This experiment gauges the performance and power impact of having limited number of VRMs on the chip to minimize the area overhead.

Figures 8.22, 8.23, and 8.24 report the performance, power, and performance/watt of the designs with varying number of VRMs. The bars are labeled with numbers which represent the number of VRMs in the design. From these results, we observe that the limited VRM designs suffer from performance losses, especially with higher TFlex core counts, compared to the 32-VRM design. At the

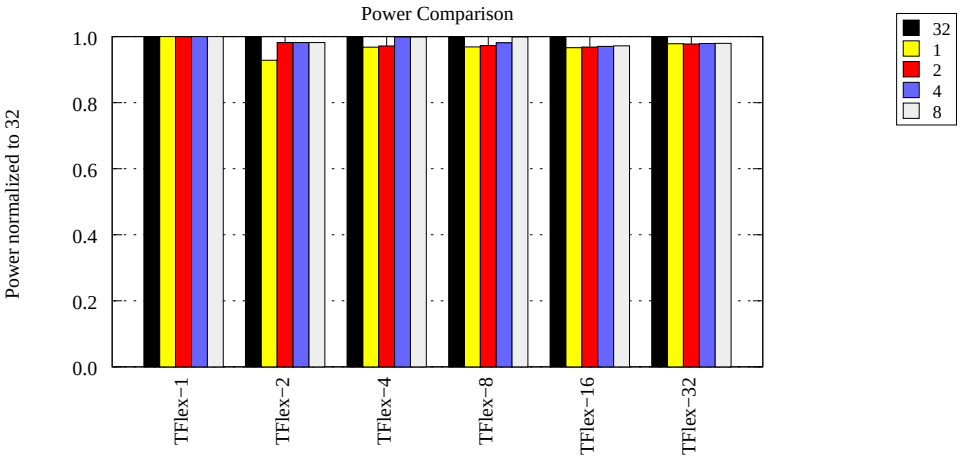


Figure 8.23: Processor Power Comparison with limited VRMs

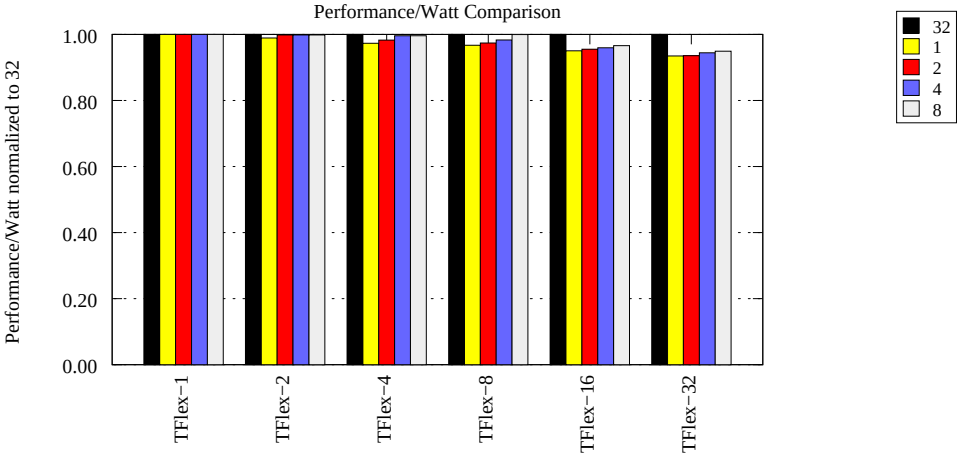


Figure 8.24: Performance/Watt Comparison with limited VRMs

Voltage (volts)	Efficiency (%)
1.1	90
1.0	89
0.9	88
0.8	87

Table 8.4: Off-Chip VRM Losses

same time, the limited VRM designs also achieve a marginal reduction in processor power. The limited VRM designs achieve within 5% of performance/watt compared to the 32-VRM design. Only the 16-core and 32-core configurations experience about 5% drop in performance/watt whereas all the other configurations achieve virtually the same performance/watt as 32 VRMs. Considering these results, and the area overheads of the VRMs, a realistic TFlex design could potentially include four or eight on-die VRMs for the TFlex cores.

Finally, to assess the effects of VRM efficiency on the overall power efficiency of the system we use the efficiency values tabulated in Table 8.2. We study the effects of both the large area-high efficiency and small area-low efficiency VRM designs reported in the table. We compare the overall power efficiency of the C-DVFS policy (real synchronization with realistic voltage transitions) with that of the 3GHz and 2GHz policies. The C-DVFS policy includes 32 on-chip VRMs, one for each core, and two additional VRMs, one for the operand network and another for L2. The 3GHz and 2GHz policies assume an off-chip VRM to supply their voltages. In order to do a fair comparison between on-chip and off-chip VRMs, we add efficiency models to the off-chip VRMs as well. Table 8.4 tabulates the efficiencies we assume for the off-chip VRM. Similar to the on-chip VRMs we assume that the efficiency of off-chip VRMs peak at a given voltage (1.1 volts in this case) and drop off at other voltages.

Figures 8.25 and 8.27 compare the processor power and performance/watt

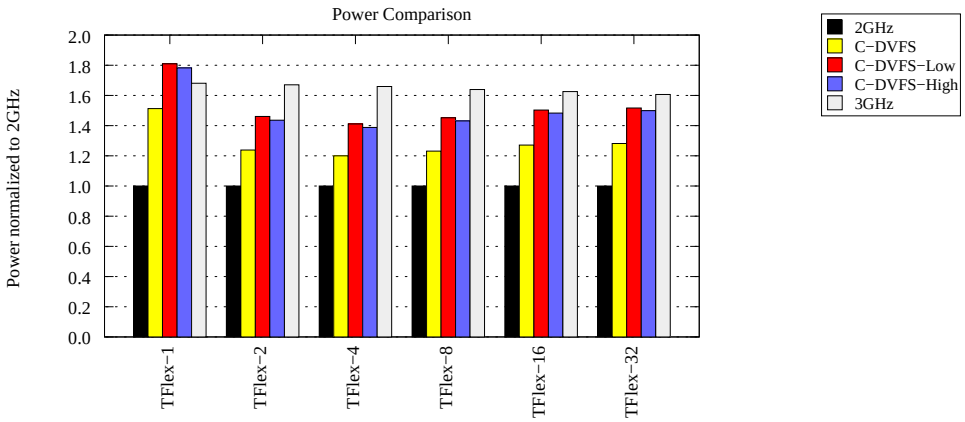


Figure 8.25: Processor Power Comparison with real VRM efficiency

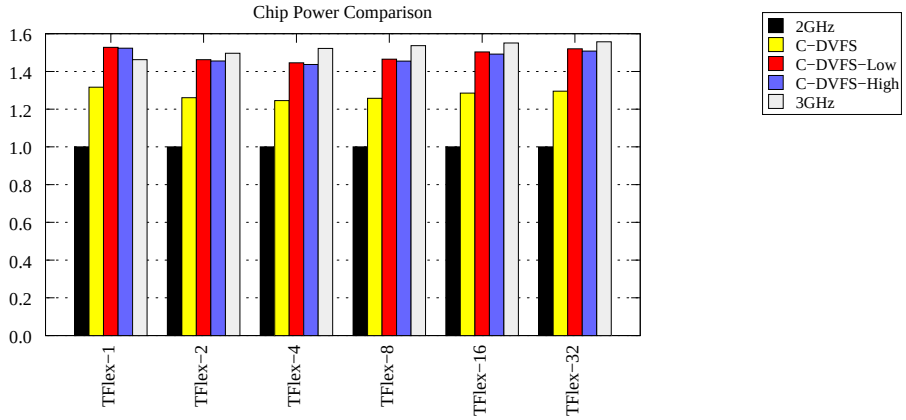


Figure 8.26: Chip Power Comparison with real VRM efficiency

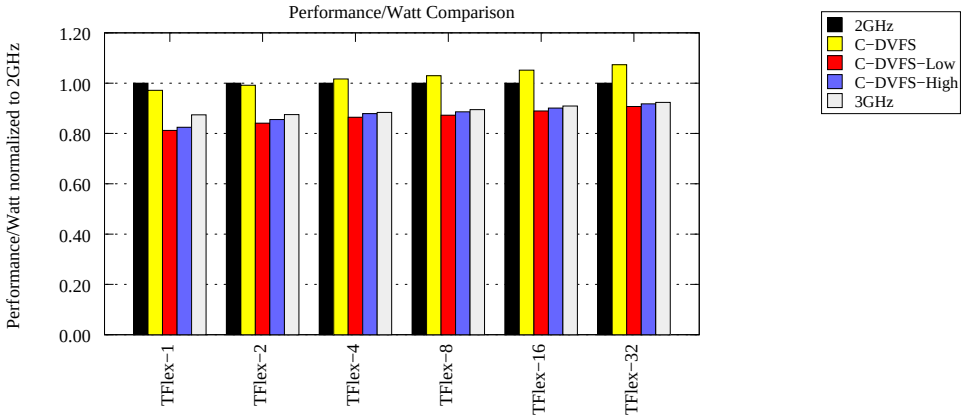


Figure 8.27: Performance/Watt Comparison with real VRM efficiency

of the various policies when VRM efficiency is taken into account. Figure 8.26 compares the overall chip power when VRM efficiency is taken into account. The 3GHz and 2GHz policies assume an off-chip VRM with efficiency values as in Table 8.4 whereas the C-DVFS policy assumes both the smaller area-lower efficiency VRMs (labeled as C-DVFS-Low) and the larger area-higher efficiency VRM design (labeled as C-DVFS-High).

These results indicate that even with the best possible on-chip VRM efficiencies the marginal processor power and performance/watt improvements observed in the C-Ideal policies rapidly diminish when compared to the off-chip-VRM-powered static policies. We additionally observe that the higher efficiency designs only provide a marginal reduction in power compared to the lower efficiency designs. This is already demonstrated by the reported efficiency values in Table 8.2. In terms of the performance/watt metric, the C-DVFS policy is outperformed by the static policies in most TFlex configurations except in 16- and 32-core configurations where the C-DVFS policy achieves almost the same performance/watt as 3GHz policy. When the overall chip power is considered, the marginal improvements achieved by C-Ideal policy are diminished even further because the inefficiencies of the VRM powering

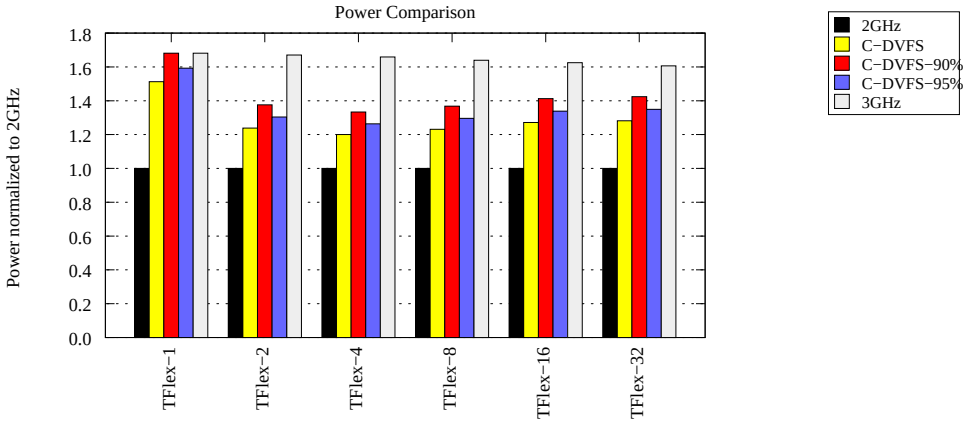


Figure 8.28: Processor Power Comparison with hypothetical VRM efficiencies

the L2 cache banks are included as well.

The above result is one of the key observations of this chapter, and represents a fundamental challenge in the wide-spread adoption of on-chip VRMs in future chip multiprocessors and CLPs. Our collaborators at Harvard, Wonyoung et al., indicate that by using more sophisticated on-package inductors the efficiency of on-chip VRMs by about 15% better than their current values [124]. Even with these improved efficiency values, the C-Ideal policies, at least as studied here, would be outperformed by the off-chip VRM designs.

To wrap up, we examine two hypothetical on-chip VRM designs with higher efficiencies than the ones used above to find out the efficiencies at which on-chip VRMs used with our criticality-based block mapping would exceed the overall power efficiency of off-chip VRMs. We examine the VRMs with efficiency of 90% and 95% respectively. Figures 8.28 and 8.29 compare the processor power and performance/watt of all the VRMs including the hypothetical VRMs with efficiency of 90% and 95%. The bars labeled C-DVFS correspond to the VRM with ideal or 100% efficiency whereas the bars labeled 90% and 95% correspond to the hypothetical designs with 90% and 95% efficiency respectively. These results indicate that

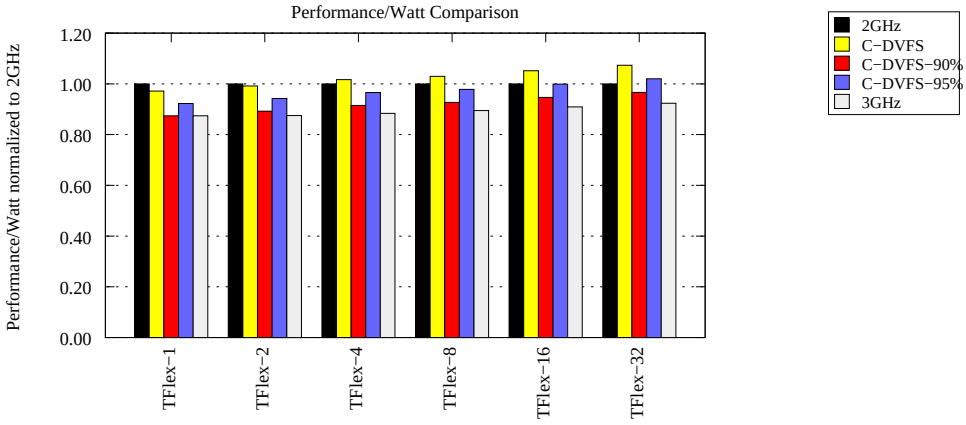


Figure 8.29: Performance/Watt Comparison with hypothetical VRM efficiencies

with our current criticality-based block mapping policies, the on-chip VRMs should achieve very high efficiencies, in the range of 95%, to be able to achieve better overall power efficiencies than the off-chip VRMs studied here.

8.8 Controlled Speculation

The next principle we explore employs controlled speculation to improve energy efficiency. Modern microprocessors typically employ various speculation techniques including control and data speculation to extract better performance. While speculative techniques generally improve performance and in some cases are crucial to good performance (for e.g., branch prediction), speculation can be a source of wasted energy when not done correctly. For example, during a branch misprediction all pipeline stages with the wrong instructions must be flushed before correct execution can begin. The costs of misspeculation are even greater for block-oriented architectures like TFlex which have to potentially flush hundreds of instructions. To achieve a better trade-off between performance and energy, we evaluate the principle that processors must employ controlled speculation. Many researchers have

evaluated implementations of this principle in different contexts. Manne et al. [67] evaluate the concept of pipeline gating for superscalar processors. They use confidence estimates of branch instructions to gate or throttle the fetch pipeline. If the current branch prediction has high confidence, the fetch pipeline continues normally as usual. If the current prediction has a low confidence estimate, the fetch pipeline is gated, thereby preventing any further fetching of instructions and reducing wasted energy.

TFlex Implementation: We evaluate this principle on the TFlex microarchitecture using fine-grained DVFS. Figure 8.30 describes our TFlex implementation based on branch confidence. Our branch confidence predictor is based on the confidence predictor described in [53], and is incorporated into the block-based branch predictor in the TFlex design [57]. In the deep block mapping mode, each in-flight TFlex block gets mapped to a different TFlex core depending on the number of cores allocated to the program. We map speculative blocks to specific cores with a given DVFS setting based on the branch confidence estimates. The TFlex microarchitecture uses an exit predictor, a version of the branch predictor adapted for block-oriented architectures, to predict the next block to be fetched and executed [57]. We augment this exit predictor to also provide a confidence estimate for each prediction. The confidence estimate is a value ranging from 0 to 15 as the confidence predictor uses 4-bit counters. We evaluate a set of policies which choose the DVFS settings of the core based on the branch confidence value. Two bimodal policies split the confidence values into two ranges, and maps the higher range to the maximum DVFS setting and the lower range to 2 GHz and 1 GHz respectively. For example, assuming good confidence prediction, if a specific branch prediction has high confidence, the corresponding TFlex block is almost guaranteed to be in the correct path and can be mapped to a core with the highest DVFS setting. If the confidence in the estimate is lower, this block could potentially be in the wrong

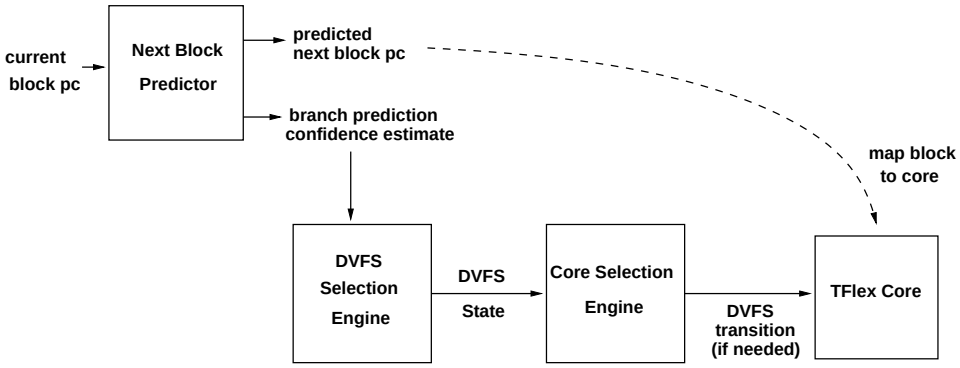


Figure 8.30: Controlled Speculation Technique

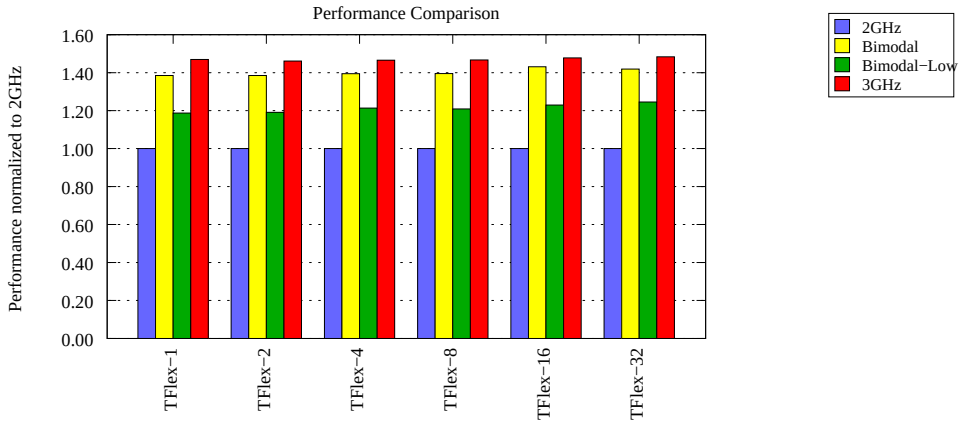


Figure 8.31: Performance Comparison with Branch Confidence based mapping

path and will be mapped to a lower DVFS setting. Thus, the above policies could potentially help TFlex to intelligently manage the trade-off between performance via speculation and energy dissipation.

8.9 Branch Confidence: Results

As in Section 8.4, we first assume idealized parameters to gauge the potential of branch confidence-based block mapping. Similar to criticality-based block mapping,

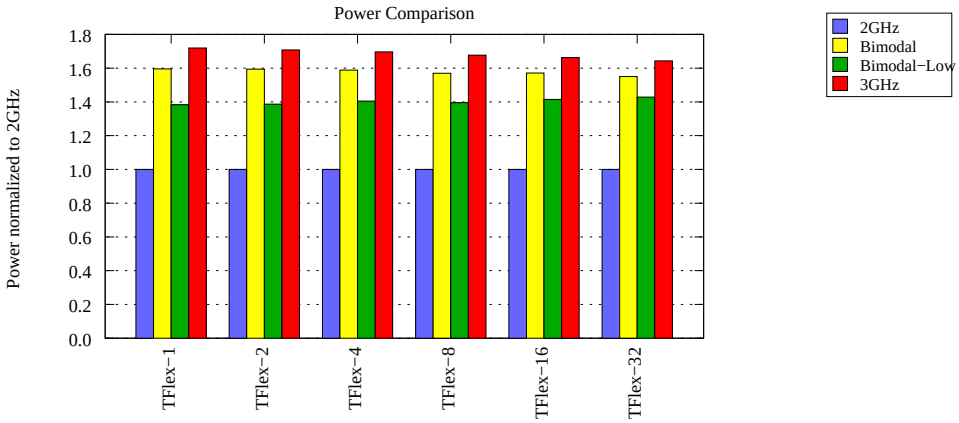


Figure 8.32: Processor Power Comparison with Branch Confidence based mapping

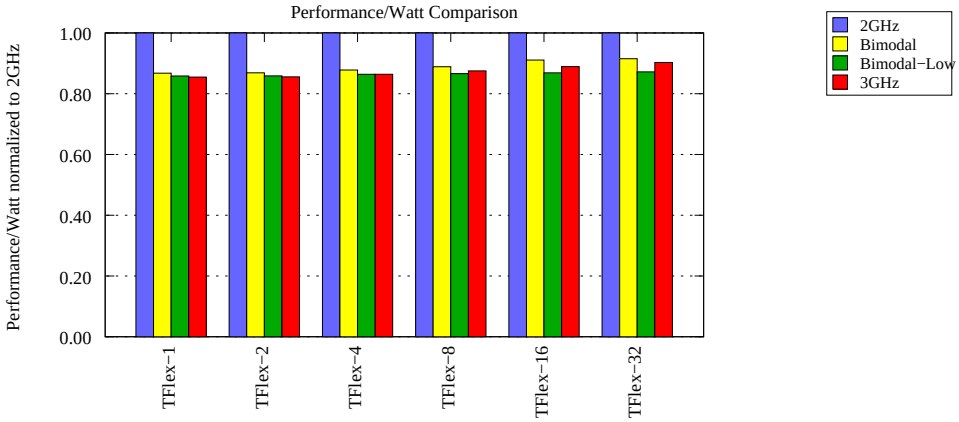


Figure 8.33: Performance/Watt Comparison with Branch Confidence based mapping

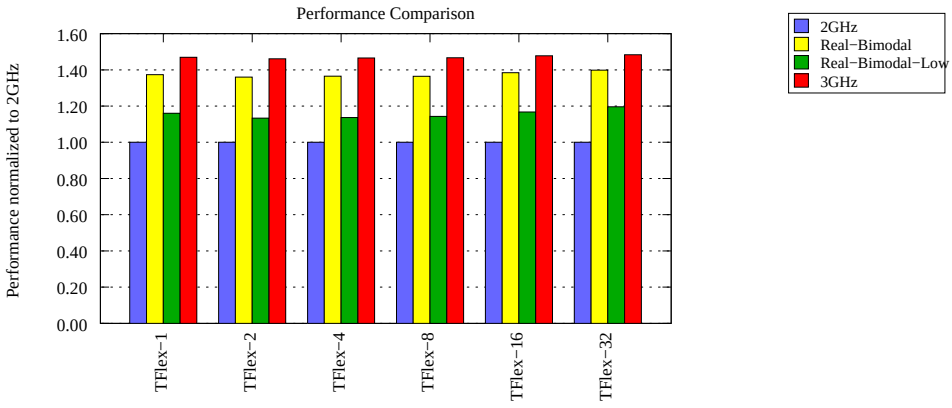


Figure 8.34: Performance Comparison with Realistic Branch Confidence based mapping

we compare the various branch confidence to static policies that map all blocks in a program to a specific frequency. Figures 8.31, 8.32, and 8.33 compare the performance, processor power, and performance/watt of the static 3GHz, 2GHz policies with that of the branch confidence policies. These results show that the various branch confidence policies achieve better performance than the 2GHz policy, but do not achieve commensurate drop in processor power. Similarly, the branch confidence policies perform worse than the 3GHz policy while consuming lower power than 3GHz policy. In terms of performance/watt, the various branch confidence policies do not exhibit the marginal improvements seen by the C-Ideal policy in the previous section. At best, the various branch confidence policies achieve the same performance/watt as the 3GHz. Additionally, our results indicate that adding the synchronization overhead, voltage transition times and regulator efficiency, the performance/watt of the various branch confidence policies become significantly worse compared to the static policies.

Figures 8.34, 8.35, and 8.36 compare the performance, processor power, and performance/watt of the static 3GHz, 2GHz policies with that of realistic branch

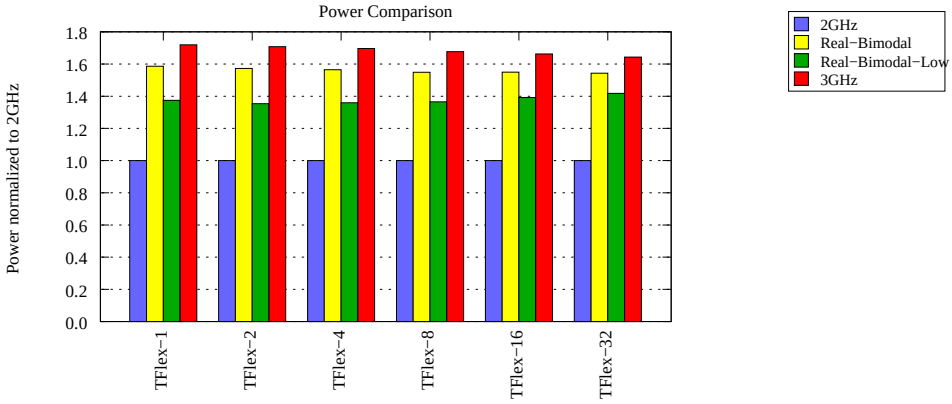


Figure 8.35: Processor Power Comparison with Realistic Branch Confidence based mapping

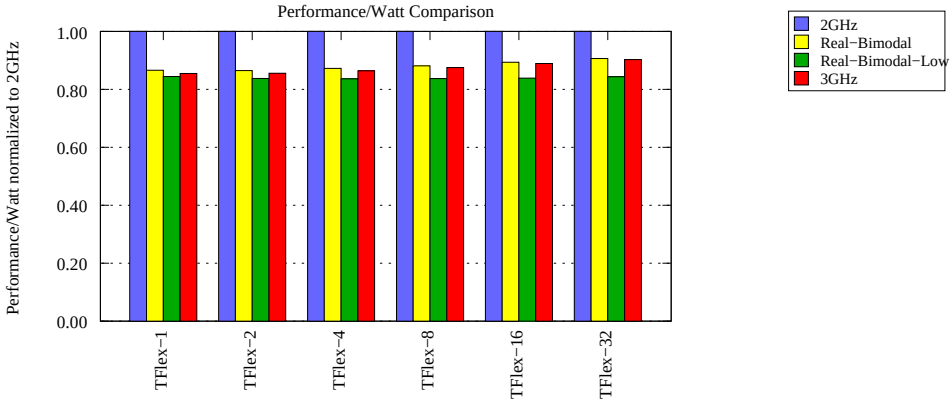


Figure 8.36: Performance/Watt Comparison with Realistic Branch Confidence based mapping

confidence policies, which assume realistic synchronization and DVFS transition times. As with the criticality-based policies, the overhead caused by realistic synchronization and DVFS times are very minimal. However, when we take into account the VRM conversion efficiency, the confidence-based policies quickly lose their power advantage over the static 3GHz policy.

Further analysis indicates that the simple, low-complexity branch confidence estimation mechanism we evaluate suffers from inaccuracies. On average, the branch confidence predictor achieves only 83% accuracy across all the TFlex configurations. Since TFlex implements a block-based EDGE ISA, it is inherently challenging to achieve same levels of confidence accuracy as in conventional processors. This drop in confidence accuracy is the key reason for the poor performance of all branch confidence policies even with ideal DVFS parameters: ideal synchronization, ideal DVFS transitions, and ideal VRM conversion efficiencies. More sophisticated branch confidence estimation techniques as reported in [5,41], could potentially enhance the feasibility of the branch confidence based block mapping policies we have studied so far. However, it would be challenging to adapt these sophisticated branch-confidence estimation techniques to a distributed block-based architecture such as TFlex.

8.10 Future Work and Conclusions

In this section, we explored two fine-grained power management policies on the distributed TFlex substrate. These policies rely on a temporally and spatially fine-grained DVFS mechanism enabled by Voltage Regulator Modules (VRMs) built on the same die as the TFlex processor. The first set of policies we evaluate is based on the concept of instruction criticality. Using profiling results from a simulation-based critical path prediction tool, we categorize the committed blocks of a program as critical and non-critical. By mapping the critical blocks to the maximum frequency, and non-critical ones to lower frequencies, this set of policies attempts to reduce the

average power consumption while minimizing the performance loss.

Our initial limit study, which assumes ideal synchronization, ideal DVFS transitions and ideal conversion efficiency, shows that the critical policies increase the operating range and flexibility of the TFlex system. Given two different DVFS settings, one with higher frequency and another with lower frequency, the critical policies achieve better performance than a static policy which maps all blocks to the lower frequency, and lower power than the static policy which maps all blocks to the higher frequency. However, our experience indicates that our current criticality-based policies are barely able to exploit the slack from non-critical blocks.

Our next set of experiments assess the effects of real synchronization and real DVFS transitions on the overall performance and power of the TFlex system. Our results indicate that real synchronization and real DVFS transitions have very minimal impact on system performance and power. Our optimization of decoupling the operand network routers from the TFlex cores, and running the routers at the maximum frequency proves to be very effective in minimizing the effects of real synchronization. Furthermore, even with a conservative voltage transition time of 5 millivolts per nanosecond, the on-chip VRMs we consider provide very fast DVFS transitions with minimal impact on performance and power.

Our experiments involving VRM conversion efficiencies and area overheads identify the real challenge in making the on-chip VRMs feasible for TFlex and other CMPs. Our design-space exploration studies clearly demonstrate the tight trade off between area overheads of VRMs and their efficiencies. If one wishes to minimize area overhead, we must compromise on the VRM efficiency. On the other hand, if one wishes to maximize VRM efficiency, a severe area penalty must be incurred, sometimes more than 50% of the chip area. Even the best VRMs we evaluate have maximum efficiencies less than 90%, and sometimes have efficiencies as low as 80%. All on-chip VRM based policies we study are outperformed by static policies

based on off-chip VRMs. This is mainly because the on-chip VRMs achieve lower conversion efficiency than their off-chip counterparts. Although current research is looking at using on-package inductors to achieve higher efficiencies than possible today, our results indicate that with our current policies the on-chip VRMs must achieve efficiencies as high as 95% to be better than the off-chip policies. Future research attempting to use criticality-based policies must look at improving things from multiple directions. First, one should attempt to use on-chip VRMs with higher efficiencies. Second, one could attempt to use different "definitions" for the notion of block criticality. In the current scheme or definition, as much as 60% of the blocks are categorized as critical and are mapped to the maximum frequency. Due to this, the theoretical upper limit according to Amdahl's law for the power reduction is only 40%. Future research could actually rank critical blocks in terms of number of cycles contributed to the critical path, and in turn use the ranking to decide the DVFS mappings. By reducing the number of blocks being mapped to the maximum frequency, such a policy could expect better power savings than the policies studied here.

Furthermore, we have explored only a fraction of the interesting trade-offs among choosing the number of independent voltage and frequency domains, suitable voltage and frequency values for these domains, and the different block-mapping policies. The distributed TFlex architecture with various modular components such as the cores, OPN routers, and L2 cache banks offers an excellent substrate for studying this design space, and further investigation of this trade-off space definitely merits more attention.

Finally, the second policy we evaluate is branch confidence based block mapping. In this policy, we estimate the confidence of every branch prediction made by the TFlex microarchitecture. Depending on the confidence estimate, the policy map blocks with high confidence values to higher frequencies and vice versa. We eval-

uate a simple confidence predictor incorporated along with the block predictor in the TFlex microarchitecture. Our experimental results indicate that even idealized confidence policies do not provide any improvements in performance/watt compared to the 3GHz policy, and that real confidence policies are outperformed by the 2GHz policy. The current branch confidence predictor achieves only an accuracy of 83%, which causes many blocks to be mapped to lower frequencies. Consequently, the static policies outperform the confidence policies. Implementing more sophisticated confidence prediction schemes such as perceptron-based predictors could potentially improve the confidence prediction accuracy, and hence, the overall power efficiency.

8.11 Lessons

The key lesson we glean from our fine-grained DVFS work is that designers should carefully account for all overheads when evaluating new ideas for their feasibility. Our overall goal in the fine-grained work is to reduce power consumption without drastically affecting performance. Our idealized limit study indicates that combining fine-grained dynamic voltage and frequency scaling (DVFS) with the TFlex architecture can be useful. However, the limit study also indicates the fundamental challenge in getting such a design to work effectively. In the distributed TFlex architecture, the register files, and the data cache banks are distributed across multiple participating cores. So, when one core is slowed down to improve energy efficiency the overall performance is affected as well. Due to this challenge, the fine-grained mechanism sacrifices performance commensurate to the energy savings, and thus, provides only marginal benefits. Furthermore, when realistic system overheads such as synchronization and finite DVFS transitions are considered, the already-marginal benefits of the fine-grained policies diminish even further. We conclude that for the two policies we study the overheads of the fine-grained DVFS mechanism might not justify its inclusion for single-threaded workloads. However, more research is needed

to study if different policies might be useful for single-threaded workloads, and for parallel and multi-programmed workloads.

Chapter 9

Related Work

9.1 Energy Efficiency

Researchers have extensively studied the area of energy efficiency, and have proposed various solutions for the same in different fields of computing including algorithms, applications, compilers, architectures, microarchitectures, circuit design and process technologies. A survey by Vasanth et al. [119] lists the techniques at various levels of abstraction for power efficient designs. A catalog of power efficiency techniques in different fields can also be found in a recent book [39]. Another recent compilation by Kaxiras and Martonosi provides an in-depth survey of techniques to improve power efficiency [55]. This book discusses several power modeling techniques, the metrics required to evaluate energy efficiency, and several techniques for achieving energy efficiency. Since the volume of power-related work is large, we restrict our discussion to research that is most relevant and related to our work. We refer the reader to the survey and the books discussed above for a more thorough treatment of the techniques to improve energy efficiency.

9.1.1 Dynamic Power

Researchers have studied many techniques at different layers of the system stack to reduce dynamic as well as leakage power [55]. Techniques to reduce dynamic power attempt to reduce at least one of the four main factors that dynamic power depends on: the voltage of the processor, its clock frequency, the capacitance of the design, and the switching activity. Dynamic voltage and frequency scaling (DVFS) modulates the voltage and frequency of the processor, and trades off performance for power, as described in Chapter 6. The power efficiency book by Kaxiras and Martonosi discusses the application of the DVFS technique at different levels: system-level [33, 122], program-level [47, 50, 93, 125], and the hardware level [26]. DVFS can also be applied for Multiple-Clock Domain (MCD) processors, which is discussed in Section 9.4. Techniques have also been studied to reduce the overall capacitance of the processor by leveraging clustered microarchitectures [16, 28], partitioned, and modular NUCA caches [58], and tiled CMPs [57, 120]. These and several related ideas tend to avoid large, monolithic hardware structures that do not scale well to wire-delay dominated process technologies, and thus, provide sizeable reductions in the total capacitance of the design. A plethora of research has looked at reducing the net switching activity of the processor in order to reduce dynamic power. Table 4.1 of the book by Kaxiras and Martonosi clearly categorizes such techniques by the type of switching activity reduced by each technique, such as 1) clock-gating idle functional units 2) support for narrow-width operands and ALUs 3) adaptively resizing hardware structures based on workload demand 4) serializing or eliminating parallel lookup activity in caches 5) use of caching or memoization to prevent repetitive operations 6) techniques to reduce speculative activity and 7) different encoding schemes to reduce switching activity on buses [55].

9.1.2 Leakage Power

Recently, starting with the 90nm technology node, leakage power has steeply increased in prominence. Researchers have extensively studied techniques to reduce different sources of leakage, and have proposed solutions at the process technology, circuit, and microarchitectural levels. The two major sources of leakage are sub-threshold and gate-tunneling leakage currents [119]. Intel has introduced metal gates and high-K dielectric materials in its 45nm process technology [72]. These changes provide signification reduction in gate-tunneling leakage current. Techniques like dual-VT transistors [45], Gated V_{dd} and MTCMOS with sleep transistors [15, 85], body biasing [56], drowsy caches [32], and leakage-biased bitlines [44] have been studied to reduce sub-threshold leakage.

9.1.3 ISA and Compiler Support

Prior work has investigated support from the instruction set and the compiler for improving energy efficiency. Like EDGE architectures that are described in this dissertation, VLIW architectures [31] also aim to simplify the microarchitecture by shifting the burden of extracting parallelism from the hardware to the compiler. The task of identifying the dependence between instructions rests entirely with the compiler. The underlying microarchitecture avoids much of the complexity found in superscalar processors to dynamically identify the dependence among instructions. A few VLIW architectures are also block-based like EDGE architectures because they employ hyperblocks. However, the key difference between VLIW and EDGE architectures is the data-flow style execution within an EDGE block and the dynamic issue of instructions in EDGE compared to the static issue in VLIW architectures. Energy-exposed ISA work by Asanovic et al. aims to improve energy efficiency by using an energy-aware instruction set to enable better compiler energy optimizations [39]. While their work retains a conventional ISA with only select

modifications for energy efficiency, EDGE architectures rely on a completely novel ISA. Being block-atomic, EDGE architectures provide a broader and general framework for amortizing book-keeping overheads compared to software restart markers in Energy-exposed ISAs. Additionally, EDGE ISAs support explicit dependence encoding between producer and consumer instructions. In contrast, energy-exposed ISAs work exposes the internal accumulator registers to software to eliminate some additional register accesses between producers and consumers [39].

Many researchers have investigated using the compiler to improve power efficiency [63, 117, 126] and using compiler hints to improve performance. The work by Valluri and John attempts to find if compiling for performance and power or energy are identical [116]. These authors conclude that compiler optimizations which improve performance by reducing the amount of work done reduces the total energy. On the other hand, optimizations that improve the IPC or the overlap of executing instructions tend to increase the average power dissipation [116]. Various block mapping policies discussed in Chapter 7 leverage the compiler to both improve performance and power. The compiler analyzes each TFlex block for its concurrency and communicates this concurrency to the hardware via an encoding in the block header. The compiler additionally optimizes for operand locality by placing producers and consumers near by on the TFlex grid of cores.

9.2 Power Modeling

We distinguish our power model validation from prior work by using all three levels for our validation–architectural models, RTL power models, and hardware power measurements. While prior work that compares across any two levels exists, our work compares across all levels to gain additional insights. The work by Kim et al. [59] discusses the challenges for architectural power modeling, and provides guidelines for architectural power modeling. While our work has some similarities, we

additionally quantify the various sources of inaccuracies in architectural power modeling by comparing with real hardware.

Chen et al. [18] present a technique to validate architectural-level power estimation of a processor, consisting of a 16-bit Digital Signal Processing (DSP) engine and a 32-bit Reduced Instruction Set Computing (RISC) core. Their technique validates their architectural power models by comparing the estimated values with those from a gate level power simulator. Our approach uses more realistic benchmarks for our study as opposed to simple ones. Natarajan et al. [77] built a validated power model for Alpha 21264 processor to analyze the energy implications of speculation and pipeline over-provisioning. They leverage detailed power breakdowns of Alpha 21264 published in literature for their model validations. The original Wattch work by Brooks et al. [12] also validates architectural power models against fine-grained capacitance estimates and published industrial power data for many designs. Our work is similar in that we leverage a detailed, post-synthesized netlist to validate our architectural power models. More importantly, our observation that architectural power models provide better relative accuracy than absolute accuracy concurs well with a similar observation by Brooks et al. [12].

Shafi et al. [99] performed hardware power measurements on PowerPC based systems with the intent of validating their higher level power models. They used various microbenchmarks to understand the energy consumed by various events of interest like a cache miss, an ALU operation, etc. We categorize the power dissipated by various components on the hardware, but use a slightly different approach as described in Chapter 3. However, we do use certain hand-crafted benchmarks for studying the power dissipation of a few hardware subsystems in the TRIPS design similar to their work. The authors of the XTREM simulator for the XScale microarchitecture validate their architectural power models with hardware power measurements on a development board [20]. These authors employ hand-crafted

stress tests to exercise specific components of the XScale system to validate their power models. Our work also utilizes hand-crafted stress tests in hardware power measurements to isolate power dissipated by different system components.

Brooks et al. perform a validation study by using two power estimation models: Wattch and Powertimer [10]. They compare the power modeling methodologies of Wattch and Powertimer and also discuss the various modeling errors possible with Wattch. While our work also quantifies a few inaccuracies in the Wattch model, we do so by comparing against a gate-level simulator and the real hardware.

Many authors have used real hardware power measurements before. One such work was done by Wu et al. [125], where the main goal in using hardware power measurements is to study power and thermal management, and to analyze power impact of compilation. On the other hand, we use hardware power measurements to validate our gate-level and architectural-level power models. The work by Mesa-Martinez et al. [70] validates architectural power models by using thermal models built with an infrared camera. Our work is similar in that we use real hardware for validation, but we also use RTL power models for validation.

9.3 Composability

The TFlex microarchitecture provides composability as a mechanism to match the resource needs of applications to hardware resources. There have been many proposals in the literature which aim at matching resource needs of applications to hardware. The fundamental difference between TFlex and most such approaches is that sharing of any centralized hardware structure is completely eliminated by the distributed TFlex microarchitecture. This, in turn, provides seamless scalability of TFlex cores, up to 32 in our design. Most of the related approaches share some hardware structures among the composed cores, and hence, suffer from scalability issues. On the downside, TFlex relies on a novel EDGE ISA to support its composability,

and hence, does not maintain binary compatibility like other approaches.

The most closely related approach to composability is the Core Fusion work by Ipek et al. [48]. Core Fusion provides support for dynamically aggregating out-of-order cores into a larger logical core, when the application requires more resources. The key motivation of Core Fusion is to provide a hardware substrate that can dynamically accommodate software diversity (single-threaded, and multi-threaded workloads), and to provide a gradual path from single-threaded code to pervasive parallelism. The key differences between Core Fusion and TFlex are twofold. First, TFlex uses support from the EDGE ISA to support composability while Core Fusion supports conventional ISAs, thereby providing software compatibility with existing binaries. Second, since Core Fusion does not leverage any support from the ISA for composing cores, it relies on centralized register renaming and synchronized fetch mechanisms that limit the scalability of the technique to four dual-issue cores (a maximum issue width of eight instructions). In contrast, since TFlex relies on the EDGE ISA support and a distributed microarchitecture, it can provide scalability up to 32 cores in SPEC-FP applications and up to 16 cores in SPEC-INT workloads.

In the core federation technique [109], multiple in-order cores are fused to form a larger out-of-order core. However, each TFlex core supports dual-issue and is out-of-order, and can be aggregated to form larger logical cores. Another related approach to composability is the Voltron architecture by Zhong et al. [129]. In this architecture, a set of VLIW cores can operate in two different modes: coupled and decoupled. In the coupled mode, which is similar to the composed mode in TFlex, a set of cores act as a wide-issue VLIW machine executing a single logical stream of instructions. In the decoupled mode, each VLIW core executes an independent thread that uses fine-grained synchronization to communicate with other threads running on different cores. The Voltron architecture is similar to the TFlex architecture in that both leverage an operand network to communicate intermediate

values to other cores. On the other hand, because Voltron is a statically exposed microarchitecture, applications typically require recompilation when targeted to a different microarchitecture to maintain good performance compared to the original microarchitecture.

Several researchers have explored the idea of clustered microarchitectures, where multiple clustered execution units are combined to form a larger superscalar processor [16,28]. Such approaches address the increased complexity of unclustered, high-issue width superscalar designs. Hardware structures like register files, register renamers, and issue windows become increasingly complex when the issue width of processors is increased. Such microarchitectures employ multiple execution clusters, for example, one integer cluster and one floating point cluster, to provide higher issue widths with lower complexity. A recent paper by Watanabe et al. [121] leverages a clustered microarchitecture to provide an array of execution units (EUs), and can dynamically compose two adjacent EUs to adapt to workloads with different parallelism profiles. Although such clustered microarchitectures reduce the complexity of supporting higher issue width, they require an instruction steering mechanism to direct instructions to a given cluster. Such a mechanism must strive to steer dependent instructions to the same cluster to minimize inter-cluster communication. Additionally, such microarchitectures must cope with physical register files that are partitioned across the clusters, and must implement mechanisms such as copy instructions to move register values from one cluster to another.

Adaptive processing techniques dynamically resize various hardware structures in a superscalar processor to adjust to varying phases in the workload [6]. These techniques add microarchitectural support to resize various structures like caches, and instruction issue windows, and expose a set of control registers to software to trigger these adaptations. These techniques involve mechanisms to dynamically resize caches [7], issue window entries [34], issue windows combined with

load-store queues [84], and the issue width of the processor along with the number of functional units [8]. Most of such adaptive processing techniques are orthogonal to EDGE architectures, especially the TFlex architecture, in that they can be easily combined with the TFlex implementation. For example, the selective cache ways technique [7] could be easily combined with the TFlex microarchitecture to provide a fine-grained adaptation of instruction and data cache capacity. By virtue of composability, TFlex already provides a coarse-grained cache size trade off by aggregating or disaggregating different cores. Another key difference between composability and such adaptive techniques is that they do not provide a fine-grained trade-off between hardware resources and varying number of threads.

Kumar et al. [61] describe a heterogeneous multi-core architecture that implements the same instruction set. The multi-core chip consists of a mix of processors, some possibly from older technology nodes and some of which are simple cores consuming lower power and providing lower performance while others are more complex cores consuming more power but with better performance. Depending on the phase of the application or the type of the workload, the chip runs the workload either on the simpler cores or on the complex cores. For example, if the workload has a lot of threads it could be run on multiple simple cores. On the other hand, if there is enough ILP to be extracted from the application it could be run on a larger core. Although providing heterogeneous cores on a chip increases flexibility provided to an operating system, such a design is relatively inflexible in that the choice of the various processors is made at design time, and not at runtime like TFlex. For example, heterogeneous cores could hurt efficiency when running multiple identical threads on these cores or when the hardware requirements of workloads do not exactly match that provided by the heterogeneous cores.

Conjoined-core CMPs [62] is a related approach in which adjacent cores share some hardware structures to reduce area and power complexity. The authors study

sharing the floating point unit, L2 crossbar ports, instruction and data caches among the adjacent cores. By intelligently sharing these resources between adjacent cores, this approach aims to match requirements of workloads to actual resources. However, due to large wiring overheads in sharing the resources, it would be prohibitive to scale this approach to more than two adjacent cores. In contrast, the TFlex microarchitecture explicitly addresses the growing wire delay issues by distributing the cores, and by elegantly sharing these cores to provide scalability.

9.4 Dynamic Voltage and Frequency Scaling

Dynamic voltage and frequency scaling has been an effective power management technique, and has been extensively studied in literature. Researchers have used DVFS to trade performance for power in the context of chip multiprocessors (CMPs). For example, the work by Isci et al. [49] explores the use of fine-grained, per-core DVFS mechanism for global chip-level power management. Their work assumes a global power manager that has access to fine-grained core-level power data. The global power manager uses these fine-grained statistics to decide the DVFS setting of individual cores, and thereby maximizes the throughput of the system under a strict chip-level power budget. Although this work assumes a fine-grained per-core DVFS implementation, they do not explore design issues of on-chip Voltage Regulator Modules (VRMs) and the interplay between the policies and the fine-grained DVFS mechanism. Li et al. explore the trade-off between number of allocated cores and DVFS settings of a CMP in the context of parallel workloads [52]. Their adaptation technique dynamically searches a two-dimensional space, number of cores allocated to the parallel workload, and the DVFS setting of these cores, to maximize power savings within specified levels of performance.

DVFS has been extensively studied in the context of Multiple Clock-Domain (MCD) processors and Globally Asynchronous, Locally Synchronous processors [51,

65, 94]. Such designs decouple different stages of a processor, and place each stage in a different frequency domain. For example, the work by Iyer et al. [51] partitions a conventional processor into five different frequency domains: i-cache and branch predictors, decode and rename, integer issue queues and ALUs, floating point issue queues and ALUs, and memory issue queues and data cache. When data crosses frequency domains, it is communicated via dual-clocked queue entries at the edge of each frequency domain. By limiting the size of each frequency domain, these designs better tolerate the problem of distributing a global clock signal with minimal skew. Additionally, these MCD designs have the flexibility of operating the frequency domains at different frequencies, and thereby, slow down parts of the processor on demand. MCD and GALS designs take advantage of the inherent slack in the processor pipeline, and slow down parts of the pipeline to achieve better energy efficiency. Although our fine-grained power management policies are very similar to the GALS approach, we apply the voltage and frequency domains at the granularity of individual TFlex cores, and not within a single TFlex core. Our fine-grained policy results indicate the potential for incorporating a GALS-type design in the context of TFlex, by running the register files and data cache banks in a different frequency domain from other hardware structures inside a core. However, the GALS approach is too fine-grained for reasonable implementation in a chip multi-processor. The number of on-die VRMs is primarily dictated by the area of the VRMs [60]. Hence, when we combine on-die VRMs with GALS, the number of required VRMs might become prohibitive (essentially number of cores x number of microarchitecture units within a GALS core).

Because modulating the power supply voltage requires adjusting the board-level voltage regulator circuit, DVFS has been historically applied to an entire chip as a whole. We leverage recent research at Harvard that has shown the potential to build multiple on-chip VRMs, enabling modulation of DVFS settings of individual

cores/elements on a chip and providing very fast DVFS transitions (a few nanoseconds) [60]. These on-chip VRMs enable a temporally and spatially fine-grained DVFS mechanism. The combination of such a fine-grained DVFS mechanism with the distributed composable TFlex architecture provides novelty for our work. The generalized principles that underlie our fine-grained DVFS policies have been implemented in conventional designs [67, 95]. Manne et al. [67] use the idea of branch confidence to throttle the fetch unit of a processor, and thereby to save energy. Although we borrow the idea of branch confidence for saving energy, we use the confidence estimate to decide a suitable DVFS setting for a TFlex core. Similarly, we borrow the idea of instruction criticality for power efficiency from Fields et al. [29] and Seng et al. [95]. The distributed TFlex microarchitecture provides an interesting substrate to implement the token-based critical-path model from [29]. Additionally, we map blocks of instructions to a TFlex core after setting that core to a DVFS level. This helps us amortize the cost of DVFS transitions over many instructions.

Chapter 10

Conclusions

Increasing power dissipation is one of the most serious challenges facing the designers in the microprocessor industry. Current and projected power trends point to a world where all transistors on a microprocessor chip cannot be operated at the maximum possible frequency, and worse, all of them may not be switched on simultaneously. The expected growth in the number of data centers and the volume of consumer electronics is likely to worsen the problem [90]. Hence, the demand for energy efficiency in all domains of computing is on the rise.

Technology trends like increasing power dissipation, increasing wire delays [4], and design complexity have forced designers to switch from conventional super-scalar processors to multi-core architectures or chip multi-processors (CMPs). While CMPs mitigate the increasing wire delays and the design complexity, they do not directly address single-threaded performance. Also, programs must be parallelized (automatically or manually) to fully exploit the performance of CMPs. In addition, according to Amdahl's law, the overall efficiency of a parallel system is eventually constrained by the sequential region of the code.

As an alternative to conventional CMPs, researchers have recently proposed an architecture called Explicit Data Graph Execution (EDGE) [14]. EDGE archi-

tectures enable technology-scalable microarchitectures that can potentially provide good single-threaded performance while exploiting other types of parallelism including data-level and thread-level parallelism [91]. EDGE architectures are characterized by two key features. First, they are block-oriented architectures which fetch, execute, and commit instructions as a group in contrast to conventional architectures that work with individual instructions. Second, within a block the dependence between a producing and a consuming instruction is explicitly encoded in the producing instruction. This relieves the hardware from rediscovering this dependence at runtime and greatly simplifies the underlying microarchitectures.

In this dissertation, we examine the energy efficiency of EDGE architectures, specifically the TRIPS Instruction Set Architecture (ISA) and two microarchitectures– TRIPS and TFlex–that implement the TRIPS ISA.

10.1 Dissertation Contributions

10.1.1 TRIPS: A Detailed Power Analysis

First, this dissertation presented a thorough power analysis of the TRIPS microarchitecture. We described how we develop architectural power models for TRIPS and how we improved the accuracy of the architectural power models using hardware power measurements on the TRIPS prototype system combined with detailed Register Transfer Level (RTL) power models. Using these refined architectural power models, we performed a detailed performance and power comparison of the TRIPS microarchitecture with two different processors: a low-end processor designed for energy efficiency (ARM/XScale) and a high-end superscalar processor designed for high performance (a variant of Power4). This detailed power analysis provided key insights into the advantages and disadvantages of the TRIPS ISA and the TRIPS microarchitecture compared to other processors like XScale and Power4. Our re-

sults indicate that the TRIPS microarchitecture achieves 11x better energy efficiency compared to ARM, and 12% better energy efficiency than Power4, in terms of the Energy-Delay-Squared (ED^2) metric.

10.1.2 TFlex Power Analysis and Comparison

Second, this dissertation analyzed the power efficiency of the TFlex microarchitecture, which also implements the TRIPS ISA. TFlex belongs to a class of microarchitectures called Composable Lightweight Processors (CLPs). CLPs are distributed microarchitectures composed of simple cores. Additionally, CLPs are highly configurable at runtime to adapt to resource needs of applications. This dissertation showed how we developed power models for the TFlex microarchitecture based on the validated TRIPS power models. Next, we evaluated the energy efficiency of the TFlex microarchitecture by comparing to the ARM and Power4 platforms, and the TRIPS microarchitecture. Our quantitative results showed that by better matching execution resources to the needs of applications, the composable TFlex system can operate in both regimes of low power (similar to ARM) and high performance (similar to Power4). We also showed that composability of TFlex helps achieve a signification improvement (2x) in energy-efficiency compared to TRIPS in terms of ED^2.

10.1.3 Composability vs. DVFS: Comparison of Performance/Power Mechanisms

Third, we examined the efficacy of processor composability–the ability of dynamically composing physical cores into a logical processor–as a potential performance-power trade-off mechanism, using TFlex as our experimental platform. Traditionally, a technique called Dynamic Voltage and Frequency Scaling (DVFS) has served as the mainstay performance-power trade-off mechanism in processors. By reduc-

ing the supply voltage, and thereby reducing the processor clock frequency one can obtain cubic reductions in dynamic power (quadratic reduction due to reduction in voltage and a linear reduction due to reduction in clock frequency) with only a linear reduction in processor performance. By modulating the supply voltage and clock frequency of the processor, DVFS helps trade off dynamic power for performance. However, the rate of supply voltage scaling has slowed significantly in recent process technologies, to keep increasing leakage power under check [21]. This slow rate of voltage scaling has significantly limited of DVFS, and has created a dire need for alternatives to DVFS. As one of the key contributions of this dissertation, we explored processor composability as an architectural alternative to DVFS. Through experimental results, this dissertation showed that processor composability achieves almost as good performance-power trade-offs as pure frequency scaling (no changes in supply voltages), and much better performance-power curves compared to voltage and frequency scaling (both supply voltage and frequency change). This dissertation also presented a case for combining DVFS with composability. Our results clearly indicated that this combination widens the operating regime of the composable system when operating under fixed performance or power targets.

10.1.4 Additional Performance Mechanisms for TFlex

Next, we explored the effects of additional performance-improving techniques for the TFlex system on its energy efficiency. Researchers have proposed a variety of techniques for improving the performance of the TFlex system. These include: 1) block mapping techniques to trade off data locality with concurrency 2) predicate prediction and 3) operand multi-cast/broadcast mechanism. We examine each of these mechanisms in terms of its effect on the energy efficiency of TFlex, and our experimental results demonstrate the effects of operand communication, and speculation on the energy efficiency of TFlex.

10.1.5 Fine-grained Power Management Policies

Finally, this dissertation evaluated a set of fine-grained power management (FGPM) policies for TFlex: exploiting instruction criticality and controlled speculation. These policies rely on a temporally and spatially fine-grained dynamic voltage and frequency scaling (DVFS) mechanism for improving power efficiency. The first policy exploited the concept of instruction criticality to improve energy efficiency. We exploited the general principle that computation that is critical to the speed of the application must execute faster, and non-critical computation in an application can be slowed down to exploit the inherent slack. Using a critical path model for the TFlex microarchitecture, we computed the criticality of instruction blocks, and use this criticality to decide the DVFS setting that the block gets mapped to. Our policy mapped highly critical blocks to higher frequencies whereas non-critical ones get mapped to lower frequencies so that average power dissipation of the system can be reduced without adversely affecting performance.

The second policy explored the use of controlled speculation. Modern processors employ various forms of speculation to extract better performance from applications. Although essential for high performance, aggressive speculation could potentially lead to wasted instructions, and hence, wasted energy. The TFlex system also employs speculation by executing multiple blocks in parallel via branch prediction. The second policy attempted to minimize the energy wasted in aggressive speculation in TFlex by using a technique called branch confidence prediction. Branch confidence prediction attempts to estimate the confidence of a given branch prediction. If branch confidence prediction accuracy is very high, we can easily identify high (likely to be on the correct path) and low confidence branch predictions (likely to be on the wrong path). Instruction blocks that have high branch confidence are mapped to higher frequencies while ones that have lower branch confidence are mapped to lower frequencies.

Our experimental results indicated that idealized instruction criticality and controlled speculation policies improve the operating range and flexibility of the TFlex system. However, when the actual overheads of fine-grained DVFS, especially energy conversion losses of voltage regulator modules (VRMs), are considered the power efficiency advantages of these idealized policies quickly diminish. Our results also indicated that the current conversion efficiencies of on-chip VRMs need to improve to as high as 95% (current efficiencies range from 80% to 89%) for the realistic policies to be feasible.

10.2 Future Directions

We conclude this dissertation with a discussion about possible future directions of research for further improving the performance and power efficiency of EDGE architectures, specifically the TFlex microarchitecture.

- In this dissertation, for the purposes of power model validation, we considered power models for a TRIPS chip running a single thread on a single TRIPS processor. It would be interesting as well as challenging to extend our power models to the entire TRIPS motherboard consisting of four TRIPS chips, and further to a set of TRIPS motherboards running a mix of parallel, single-threaded, and multi-programmed workloads. Such a power model would require accurate power estimates of both intra- and inter-motherboard communication across the chip-to-chip interfaces, as well as amount of data communicated across the chips and boards, communication pattern among the workloads, and the voltage and frequency setting of individual chips and chip-to-chip links.

- Current EDGE architectures suffer from code bloat and I-cache efficiency issues mainly because of a separate header block for encoding register read and

write instructions, and because partially filled blocks are expanded to their full size in the Level-1 instruction caches with NOPs. ISA changes to minimize the header overhead, and compiler and microarchitectural support for variable-sized blocks would go a long way in improving the I-cache efficiency of EDGE architectures. Researchers are currently looking at supporting variable-sized blocks in TFlex [66].

- Composable architectures like TFlex open up excellent research opportunities at the Operating System or Hypervisor level. Researchers have analyzed the impact of various OS scheduling algorithms on the TFlex architecture and their benefits in [42]. However, this work only considers overall system throughput, and not power efficiency. It would be extremely interesting to explore ways to maximize the overall system throughput under a strict chip-level power budget. Furthermore, adding the mechanism of DVFS and even on-chip fine-grained DVFS opens up exciting opportunities as well as challenges for the Operating System or Hypervisor to maximize system throughput under a power budget.

- Since TFlex microarchitecture tightly integrates the network routers with the cores as well as the L2 cache banks, more exploration is required in the area of power-optimized network routers. Policies that implement intelligent power management features like power gating, and link-gating are necessary to manage the power dissipated in the routers. Additionally, such policies must be tightly integrated with the dynamic block-mapping schemes to maximize locality with minimal impact on performance.

- Although the fine-grained power management policies we study provided only marginal benefits, there remain lots of avenues for further exploration of fine-grained policies. For example, one idea is matching the capacity and speed of the L2 cache banks to that of composed cores of varying sizes. Since the

L2 cache banks in TFlex are NUCA-based, they form a neat, modular design where fine-grained DVFS can be applied to further improve power efficiency.

- The concept of hardware accelerators is very popular, and for any given task, a dedicated hardware accelerator produces one of the best possible energy efficiencies. The TFlex microarchitecture, with its routed mesh network, provides an ideal eco-system for integrating on-chip accelerators tightly along with the TFlex cores. By using the operand network router and/or additional control networks, the integrated hardware accelerators can share access to other cores as well as L2 cache banks, and can execute portions of the code that can be optimized on the accelerator. The identification and mapping of the special regions of code can be done with compiler and OS support.

- Last but least, the EDGE architectures evaluated so far show good promise for power efficiency when compared to their superscalar counterparts. However, in order to become mainstream, the EDGE architectures must address the issue of x86 software compatibility. More research is needed in adapting techniques like binary translation and just-in-time compilation to provide x86 compatibility with minimal impact on performance and power. Another interesting research direction could look at the potential for augmenting mainstream processors with EDGE hardware accelerators. Vast amount of research is needed into CUDA-style [80] ecosystems to get such heterogeneous systems containing EDGE accelerators to work well with mainstream processors.

Appendices

Appendix A

Power Validation Results

This appendix presents the power model validation results for all 24 benchmarks used in this dissertation. Section 3.3 presented these results only for 12 benchmarks for the sake of clarity. In this appendix, we present the results for all benchmarks, including the 12 that were presented in Section 3.3.

In Figure A.1, for each benchmark, the graph shows three bars: architectural power estimates, RTL power estimates, and measured hardware power. The architectural bar has five segments, each representing a different architectural power model. For a description of the various architectural power models, please refer to the discussion in Section 3.3.

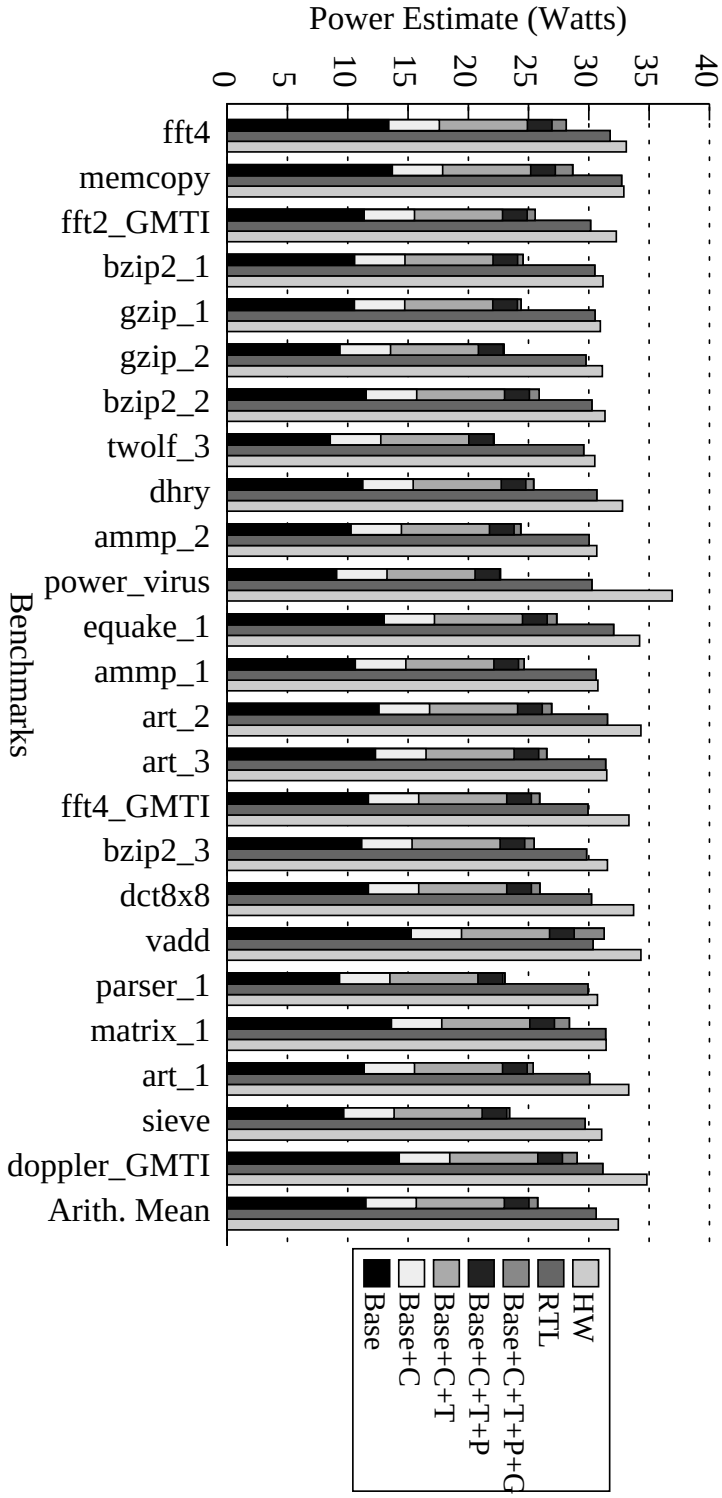


Figure A.1: TRIPS Power Validation Results: All Benchmarks

Appendix B

Power Density Comparison

As microprocessors operate the power dissipated generates heat that must be safely removed for continued reliable operation of the chip. Computing systems utilize a variety of techniques including active and passive heatsinks, and cooling fans for dissipating the generated heat. A metric called **power density**, power dissipated per unit area, is commonly used to compare the heat generated by various processors. The higher the power density of a microprocessor, the more heat is generated per unit area in the overall system, and thus, the cooling system needs to work more efficiently to remove the heat. Therefore, microprocessors with lower power densities are preferred because they result in lower packaging and cooling costs than ones with higher power densities.

Table B.1 tabulates the average power densities of all platforms examined so far. From these results we observe that all TFlex configurations have either comparable or lower power densities compared to ARM and TRIPS, and have much lower power densities compared to the Power4. The average power consumption in the TFlex system increases with the number of cores as does the total core area. The sub-linear growth in average power combined with linear growth in core area causes the power density of TFlex to decrease as more cores are added to the system.

Configuration	Avg. Power (Watts)	Area (cm^2)	Avg. Power Density (Watts/cm^2)
ARM	0.90	0.0179	50.34
Power4	10.78	0.0820	131.44
TRIPS	12.96	0.2598	49.90
TFlex-1	2.85	0.0363	78.45
TFlex-2	5.65	0.0726	77.76
TFlex-4	9.03	0.1452	62.22
TFlex-8	13.14	0.2904	45.23
TFlex-16	18.03	0.5808	31.04
TFlex-32	23.66	1.1616	20.37

Table B.1: Power Density Comparison

Bibliography

[1] Bay Wolf's Speedstep FAQ. `www.bay-wolf.com/speedstep.htm`.

[2] EEMBC: The Embedded Microprocessor Benchmark Consortium. `http://www.eembc.org`.

[3] The Problem of Power Consumption in Servers, March 2009. http://www.drdobbs.com/215800830;jsessionid=3LQKHP2H3QX0ZQE1GHPCKHWATMY32JVN.

[4] Vikas Agarwal, M. S. Hrishikesh, Stephen W. Keckler, and Doug Burger. Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pages 248–259, May 2000.

[5] Haitham Akkary, Srikanth T. Srinivasan, Rajendar Koltur, Yogesh Patil, and Wael Refaai. Perceptron-based Branch Confidence Estimation. In *Proceedings of the 10th Annual International Symposium on High Performance Computer Architecture*, pages 265–277, February 2004.

[6] David H. Albonesi, Rajeev Balasubramonian, Steve Dropsho, Sandhya Dwarkadas, Eby G. Friedman, Michael C. Huang, Volkan Kursun, Grigorios Magklis, Michael L. Scott, Greg Semeraro, Pradip Bose, Alper Buyukto-

sunoglu, Peter W. Cook, and Stanley Schuster. Dynamically Tuning Processor Resources with Adaptive Processing. *IEEE Computer*, 36(12):49–58, 2003.

[7] D.H. Albonesi. Selective Cache Ways: On-Demand Cache Resource Allocation. In *Proceedings of the 32nd Annual International Symposium on Microarchitecture*, pages 248–259, December 1999.

[8] R. Iris Bahar and Srilatha Manne. Power and Energy Reduction via Pipeline Balancing. In *Proceedings of the 28th Annual International Symposium on Computer Architecture*, pages 218–229, July 2001.

[9] Robert D. Blumofe, Christopher F. Joerg, Bradley C. Kuszmaul, Charles E. Leiserson, Keith H. Randall, and Yuli Zhou. Cilk: An Efficient Multithreaded Runtime System. In *Proceedings of the 5th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, pages 207–216, July 1995.

[10] David Brooks, Pradip Bose, and Margaret Martonosi. Power-performance Simulation: Design and Validation Strategies. *SIGMETRICS Performance Evaluation Review*, 31(4):13–18, 2004.

[11] David Brooks, Margaret Martonosi, John david Wellman, and Pradip Bose. Power-Performance Modeling and Tradeoff Analysis for a High End Microprocessor. In *International Workshop on Power Aware Computing Systems (PACS) at ASPLOS-IX*, pages 126–136, November 2000.

[12] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: A Framework for Architectural-level Power Analysis and Optimizations. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pages 83–94, May 2000.

[13] D.M. Brooks, P. Bose, S.E. Schuster, H. Jacobson, P.N. Kudva, A. Buyuktosunoglu, J. Wellman, V. Zyuban, M. Gupta, and P.W. Cook. Power-aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors. *IEEE Micro*, 20(6):26–44, November/December 2000.

[14] D. Burger, S.W. Keckler, K.S. McKinley, M. Dahlin, L.K. John, Calvin Lin, C.R. Moore, J. Burrill, R.G. McDonald, and W. Yoder. Scaling to the End of Silicon with EDGE Architectures. *IEEE Computer*, 37(7):44–55, July 2004.

[15] Benton H. Calhoun, Frank A. Honore, and Anantha Chandrakasan. Design Methodology for Fine-grained Leakage Control in MTCMOS. In *Proceedings of the 2003 Annual International Symposium on Low power Electronics and Design*, pages 104–109, August 2003.

[16] Ramon Canal, Joan-Manuel Parcerisa, and Antonio González. A Cost-Effective Clustered Architecture. In *Proceedings of the 8th International Symposium on Parallel Architectures and Compilation Techniques*, pages 160–168, October 1999.

[17] Bradford L. Chamberlain. *The Design and Implementation of a Region-Based Parallel Language*. PhD thesis, The University of Washington, November 2001.

[18] Rita Yu Chen, Robert Michael Owens, Mary Jane Irwin, and Raminder Singh Bajwa. Validation of an Architectural Level Power Analysis Technique. In *Proceedings of the 1998 International Design Automation Conference*, pages 242–245, June 1998.

[19] Mike Goodwin Chris Bowen, Gerhard Klimeck and Dick Chapman. Dopant Fluctuations and Quantum Effects in Sub-0.1um CMOS, 1997. http://www.cfdrc.com/nemo/pubs/isdrs_html/isdrs_html.html.

[20] Gilberto Contreras, Margaret Martonosi, Jinzhang Peng, Guei-Yuan Lueh, and Roy Ju. The XTREM Power and Performance Simulator for the Intel XScale Core: Design and Experiences. *ACM Transactions in Embedded Computing Systems*, 6(1):4, 2007.

[21] Vivek De and Shekhar Borkar. Technology and Design Challenges for Low Power and High Performance. In *Proceedings of the 1999 Annual International Symposium on Low Power Electronics and Design*, pages 163–168, August 1999.

[22] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 2004 Usenix Symposium on Operating System Design and Implementation(OSDI)*, pages 137–150, December 2004.

[23] J. Dorsey, S. Searles, M. Ciraula, E. Fang, S. Johnson, N. Bujanos, R. Kumar, D. Wu, M. Braganza, and S. Meyers. An Integrated Quad-Core Opteron^{TM} Processor. In *IEEE International Solid-State Circuits Conference*, pages 102–103, February 2007.

[24] J.J. Engel, T.S. Guzowksi, A. Hunt, D.E. Lackey, L.D. Pickup, R.A. Proctor, K. Reynolds, A.M. Rincon, and D.R. Stauffer. Design Methodology for IBM ASIC Products. *IBM Journal of Research and Development*, 40(4):387–406, July 1996.

[25] United States Environmental Protection Agency Report to Congress on Server and Data Center Energy Efficiency, 2007. http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Report_Exec_Summary_Final.pdf.

[26] Dan Ernst, Shidhartha Das, Seokwoo Lee, David Blaauw, Todd Austin, Trevor Mudge, Nam Sung Kim, and Krisztian Flautner. Razor: Circuit-Level Cor-

rection of Timing Errors for Low-Power Operation. *IEEE Micro*, 24(6):10–20, November/December 2004.

[27] Hadi Esmaeilzadeh and Doug Burger. Hierarchical Control Prediction: Support for Aggressive Predication. In *Proceedings of the 2009 Workshop on Parallel Execution of Sequential Programs on Multi-core Architectures*, pages 71–80, June 2009.

[28] Keith I. Farkas, Paul Chow, Norman P. Jouppi, and Zvonko Vranesic. The Multicluster Architecture: Reducing Cycle Time Through Partitioning. In *Proceedings of the 30th International Symposium on Microarchitecture*, pages 149–159, December 1997.

[29] Brian Fields, Shai Rubin, and Rastislav Bodík. Focusing Processor Policies via Critical-Path Prediction. In *Proceedings of the 28th Annual International Symposium on Computer Architecture*, pages 74–85, July 2001.

[30] T. Fischer, J. Desai, B. Doyle, S. Naffziger, and B. Patella. A 90-nm Variable Frequency Clock System for a Power-Managed Itanium Architecture Processor. *IEEE Journal of Solid-State Circuits*, 41(1):218–228, January 2006.

[31] J.A. Fisher. Very Long Instruction Word Architectures and the ELI-512. In *Proceedings of the 10th Annual International Symposium on Computer Architecture*, pages 140–150, June 1983.

[32] Krisztián Flautner, Nam Sung Kim, Steve Martin, David Blaauw, and Trevor Mudge. Drowsy Caches: Simple Techniques for Reducing Leakage Power. In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, pages 148–157, May 2002.

[33] Krisztián Flautner, Steve Reinhardt, and Trevor Mudge. Automatic Perfor-

mance Setting for Dynamic Voltage Scaling. *Wireless Networks*, 8(5):507–520, 2002.

[34] Daniele Folegnani and Antonio González. Energy-Effective Issue Logic. In *Proceedings of the 28th Annual International Symposium on Computer Architecture*, pages 230–239, July 2001.

[35] Message Passing Interface Forum. A Message-Passing Interface Standard. Technical report, University of Tennessee, Knoxville, April 1994.

[36] Madhu Saravana Sibi Govindan, Stephen W. Keckler, and Doug Burger. End-to-End Validation of Architectural Power Models. In *Proceedings of the 2009 International Symposium on Low Power Electronics and Design (ISLPED)*, pages 383–388, 2009.

[37] Madhu Saravana Sibi Govindan, Charles Lefurgy, and Ajay Dholakia. Using On-line Power Modeling for Server Power Capping. In *Proceedings of the 2009 Workshop on Energy-Efficient Design (WEED)*, June 2009.

[38] Paul Gratz, Karthikeyan Sankaralingam, Heather Hanson, Premkishore Shivakumar, Robert McDonald, Stephen W. Keckler, and Doug Burger. Implementation and Evaluation of Dynamically Routed Processor Operand Network. In *Proceedings of the 1st International Symposium on Networks-on-Chip*, May 2007.

[39] Robert Graybill and Rami Melhem, editors. *Power Aware Computing*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[40] Ed Grochowski and Murali Annavaram. Energy-Per-Instruction Trends in Intel Microprocessors. http://www.intel.com/pressroom/kits/core2duo/pdf/epi-trends-final2.pdf.

[41] Dirk Grunwald, Artur Klauser, Srilatha Manne, and Andrew Pleszkun. Confidence Estimation for Speculation Control. *SIGARCH Computer Architecture News*, 26(3):122–131, 1998.

[42] D.P. Gulati, C. Kim, S. Sethumadhavan, S.W. Keckler, and D. Burger. Multitasking Workload Scheduling on Flexible-Core Chip Multiprocessors. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, pages 187–196, October 2008.

[43] John L. Henning. SPEC CPU2000: Measuring CPU Performance in the New Millennium. *IEEE Computer*, 33(7):28–35, July 2000.

[44] Seongmoo Heo, Kenneth Barr, Mark Hampton, and Krste Asanović. Dynamic Fine-grain Leakage Reduction using Leakage-biased Bitlines. In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, pages 137–147, May 2002.

[45] Yen-Te Ho and Ting-Ting Hwang. Low Power Design Using Dual Threshold Voltage. In *Proceedings of the 2004 Annual Conference on Asia South Pacific Design Automation*, pages 205–208, January 2004.

[46] H. Peter Hofstee. Power-Constrained Microprocessor Design. In *International Conference on Computer Design*, volume 0, pages 14–16, September 2002.

[47] Chung-Hsing Hsu and Ulrich Kremer. The Design, Implementation, and Evaluation of a Compiler Algorithm for CPU Energy Reduction. In *Proceedings of the 2003 ACM conference on Programming Language Design and Implementation (PLDI)*, pages 38–48, June 2003.

[48] Engin Ipek, Meyrem Kirman, Nevin Kirman, and Jose F. Martinez. Core Fusion: Accommodating Software Diversity in Chip Multiprocessors. In *34th*

Annual International Symposium on Computer Architecture, pages 186–197, June 2007.

[49] Canturk Isci, Alper Buyuktosunoglu, Chen-Yong Cher, Pradip Bose, and Margaret Martonosi. An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance For A Given Power Budget. In *Proceedings of the 39th Annual International Symposium on Microarchitecture*, pages 347–358, December 2006.

[50] Canturk Isci, Gilberto Contreras, and Margaret Martonosi. Live, Runtime Phase Monitoring and Prediction on Real Systems with Application to Dynamic Power Management. In *Proceedings of the 39th Annual International Symposium on Microarchitecture*, pages 359–370, December 2006.

[51] Anoop Iyer and Diana Marculescu. Power and Performance Evaluation of Globally Asynchronous Locally Synchronous Processors. In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, pages 158–168, May 2002.

[52] Li J. and Martinez J.F. Dynamic Power-Performance Adaptation of Parallel Computation on Chip Multiprocessors. In *Proceedings of the 12th Annual International Symposium on High-Performance Computer Architecture*, pages 77–87, February 2006.

[53] Erik Jacobsen, Eric Rotenberg, and J. E. Smith. Assigning Confidence to Conditional Branch Predictions. In *Proceedings of the 29th Annual International Symposium on Microarchitecture*, pages 142–152, December 1996.

[54] J.M.Tendler, J.S.Dodson, Jr J.S.Fields, H.Le, and B.Sinharoy. Power4 System Microarchitecture. *IBM Journal of Research and Development*, 46(1):5–25, 2002.

[55] Stefanos Kaxiras and Margaret Martonosi, editors. *Computer Architecture Techniques for Power-Efficiency, Synthesis Lectures on Computer Architecture*. Morgan and Claypool Publishers, 2008.

[56] C. Kim and K. Roy. Dynamic V_{th} Scaling Scheme for Active Leakage Power Reduction. In *Proceedings of the 2002 Annual Conference on Design, Automation and Test in Europe*, pages 163–167, May 2002.

[57] Changkyu Kim, Simha Sethumadhavan, M. S. Govindan, Nitya Ranganathan, Divya Gulati, Doug Burger, and Stephen W. Keckler. Composable Lightweight Processors. In *Proceedings of the 40th Annual International Symposium on Microarchitecture*, pages 381–394, December 2007.

[58] Chankyu Kim, Doug Burger, and Stephen W. Keckler. An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 211–222, October 2002.

[59] Nam Sung Kim, Todd Austin, Trevor Mudge, and Dirk Grunwald. Challenges for Architectural Level Power Modeling. In *Power Aware Computing*, pages 317–337. Kluwer Academic Publishers, Norwell, MA, 2002.

[60] Wonyoung Kim, Meeta Gupta, Gu-Yeon Wei, and David Brooks. System Level Analysis of Fast, Per-Core DVFS using On-Chip Switching Regulators. In *Proceedings of the 14th Annual International Symposium on High Performance Computer Architecture*, pages 123–136, February 2008.

[61] Rakesh Kumar, Keith I. Farkas, Norman P. Jouppi, Parthasarathy Ranganathan, and Dean M. Tullsen. Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction. In *Proceedings of*

the 36th Annual International Symposium on Microarchitecture, pages 81–92, December 2003.

[62] Rakesh Kumar, Norman P. Jouppi, and Dean M. Tullsen. Conjoined-Core Chip Multiprocessing. In *Proceedings of the 37th Annual International Symposium on Microarchitecture*, pages 195–206, December 2004.

[63] Chingren Lee, Jenq Kuen Lee, Tingting Hwang, and Shi-Chun Tsai. Compiler optimization on VLIW instruction scheduling for Low Power. *ACM Transactions on Design Automation of Electronic Systems*, 8(2):252–268, 2003.

[64] Dong Li, Behnam Robatmili, Madhu Saravana Sibi Govindan, Aaron Smith, Stephen W. Keckler, and Doug Burger. Compiler-assisted Hybrid Operand Communication. Technical Report TR-09-33, Department of Computer Sciences, The University of Texas at Austin, November 2009.

[65] Grigorios Magklis, Michael L. Scott, Greg Semeraro, David H. Albonesi, and Steven Dropsho. Profile-based Dynamic Voltage and Frequency Scaling for a Multiple Clock Domain Microprocessor. In *Proceedings of the 30th Annual International Symposium on Computer Architecture*, pages 14–27, June 2003.

[66] Bertrand Maher. *Atomic Block Formation for Explicit Data Graph Execution Architectures*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin, August 2010.

[67] S. Manne, A. Klauser, and D. Grunwald. Pipeline Gating: Speculation Control for Energy Reduction. *Proceedings of the 25th Annual International Symposium on Computer Architecture*, pages 132–141, July 1998.

[68] Mark Gebhart, Bertrand A. Maher, Katherine E. Coons, Jeff Diamond, Paul Gratz, Mario Marino, Nitya Ranganathan, Behnam Robatmili, Aaron Smith, James Burrill, Stephen W. Keckler, Doug Burger, and Kathryn S. McKinley.

An Evaluation of the TRIPS Computer System. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 1–12, August 2009.

[69] Alain J. Martin. Towards an energy complexity of computation. *Information Processing Letters*, 77(2-4), February 2001.

[70] Francisco Javier Mesa-Martinez, Joseph Nayfach-Battilana, and Jose Renau. Power Model Validation through Thermal Measurements. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, pages 302–311, June 2007.

[71] Micron Technology Incorporated: Calculating DDR Memory System Power. http://download.micron.com/pdf/technotes/ddr/TN4603.pdf, 2001.

[72] K. Mistry, C. Allen, C. Auth, B. Beattie, D. Bergstrom, M. Bost, M. Brazier, M. Buehler, A. Cappellani, R. Chau, C.-H. Choi, G. Ding, K. Fischer, T. Ghani, R. Grover, W. Han, D. Hanken, M. Hattendorf, J. He, J. Hicks, R. Huessner, D. Ingerly, P. Jain, R. James, L. Jong, S. Joshi, C. Kenyon, K. Kuhn, K. Lee, H. Liu, J. Maiz, B. Mclntyre, P. Moon, J. Neirynck, S. Pae, C. Parker, D. Parsons, C. Prasad, L. Pipes, M. Prince, P. Ranade, T. Reynolds, J. Sandford, L. Shifren, J. Sebastian, J. Seiple, D. Simon, S. Sivakumar, P. Smith, C. Thomas, T. Troeger, P. Vandervoorn, S. Williams, and K. Zawadzki. A 45nm Logic Technology with High-k+Metal Gate Transistors, Strained Silicon, 9 Cu Interconnect Layers, 193nm Dry Patterning, and 100% Pb-free Packaging. *International Electron Devices Meeting*, pages 247–250, December 2007.

[73] M. Moudgill, P. Bose, and J.H. Moreno. Validation of Turandot, a Fast Processor Model for Microarchitecture Exploration. *International Performance, Computing and Communications Conference*, pages 451–457, February 1999.

[74] T. Mudge. Power: A First-Class Architectural Design Constraint. *IEEE Computer*, 34(4):52–58, April 2001.

[75] Ramadass Nagarajan, Xia Chen, Robert G. McDonald, Doug Burger, and Stephen W. Keckler. Critical Path Analysis of the TRIPS Architecture. In *Proceedings of the 2006 International Symposium on Performance Analysis of Systems and Software*, pages 37–47, March 2006.

[76] Farid N. Najm. A Survey of Power Estimation Techniques in VLSI circuits. *IEEE Transactions on Very Large Scale Integrated Systems*, 2(4):446–455, December 1994.

[77] Karthik Natarajan, Heather Hanson, Stephen W. Keckler, Charles R. Moore, and Doug Burger. Microprocessor Pipeline Energy Analysis. In *Proceedings of the 2003 Annual International Symposium on Low power Electronics and Design*, pages 282–287, August 2003.

[78] J.S. Neely, H.H. Chen, S.G. Walker, J. Venuto, and T.J. Bucelot. CPAM: A Common Power Analysis Methodology for High-Rerformance VLSI Design. In *Proceedings of the Conference on Electrical Performance of Electronic Packaging*, pages 303–306, October 2000.

[79] M. Nemani and F.N. Najm. High-level Area and Power Estimation for VLSI Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(6):697–713, June 1999.

[80] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable Parallel Programming with CUDA. *ACM Queue*, 6(2):40–53, 2008.

[81] Asim Nisar, Mongkol Ekpanyapong, and Kuppuswamy Sivakumar. Original 45nm Intel Core^{TM} Microarchitecture. *Intel Technology Journal*, 12(3), October 2008.

[82] U.Y. Ogras, R. Marculescu, P. Choudhary, and D. Marculescu. Voltage-Frequency Island Partitioning for GALS-based Networks-on-Chip. In *Proceedings of the 46th Annual International Design Automation Conference*, pages 110–115, June 2007.

[83] The OpenMP API Specification for Parallel Programming. http://openmp.org/wp/.

[84] Dmitry Ponomarev, Gurhan Kucuk, and Kanad Ghose. Reducing Power Requirements of Instruction Scheduling through Dynamic Allocation of Multiple Datapath Resources. In *Proceedings of the 34th Annual International Symposium on Microarchitecture*, pages 90–101, 2001.

[85] M. Powell, Se-Hyun Yang, B. Falsafi, K. Roy, and T.N. Vijaykumar. Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories. In *Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, pages 90–95, July 2000.

[86] POSIX Threads Programming. https://computing.llnl.gov/tutorials/pthreads/.

[87] Nitya Ranganathan, Doug Burger, and Stephen W. Keckler. Analysis of the TRIPS Prototype Block Predictor. In *Proceedings of the 2009 International Symposium on Performance Analysis of Systems and Software*, pages 195–206, April 2009.

[88] Behnam Robatmili, Katherine E. Coons, Doug Burger, and Kathryn S. McKinley. Strategies for Mapping Dataflow Blocks to Distributed Hardware. In *Proceedings of the 41st Annual International Symposium on Microarchitecture*, pages 23–34, November 2008.

[89] Behnam Robatmili, Madhu Saravana Sibi Govindan, Doug Burger, and Steve Keckler. Reducing Performance Bottlenecks in Composable Multicore Proces-

sors. In *Submitted for review to International Symposium of Microarchitecture*, December 2010.

[90] Kurt W. Roth and Kurtis McKenney. Energy Consumption by Consumer Electronics in U.S. Residences. Technical Report D5525, Consumer Electronics Association, January 2007.

[91] Karthikeyan Sankaralingam, Ramadass Nagarajan, Haiming Liu, Changkyu Kim, Jaehyuk Huh, Doug Burger, Stephen W. Keckler, and Charles R. Moore. Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS Architecture. In *Proceedings of the 30th Annual International Symposium on Computer Architecture*, pages 422–433, June 2003.

[92] Karthikeyan Sankaralingam, Ramadass Nagarajan, Robert McDonald, Rajagopalan Desikan, Saurabh Drolia, Madhu Saravana Sibi Govindan, Paul Gratz, Divya Gulati, Heather Hanson, Changkyu Kim, Haiming Liu, Nitya Ranganathan, Simha Sethmadhavan, Sadia Sharif, Premkishore Shivakumar, Stephen W. Keckler, and Doug Burger. Distributed Microarchitectural Protocols in the TRIPS Prototype Processor. In *Proceedings of the 39th Annual International Symposium on Microarchitecture*, pages 480–491, December 2006.

[93] H. Saputra, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, J. S. Hu, C-H. Hsu, and U. Kremer. Energy-conscious Compilation based on Voltage Scaling. In *Proceedings of the Joint Conference on Languages, Compilers and Tools for Embedded Systems (LCTES/SCOPES)*, pages 2–11, June 2002.

[94] Greg Semeraro, Grigorios Magklis, Rajeev Balasubramonian, David H. Albonesi, Sandhya Dwarkadas, and Michael L. Scott. Energy-Efficient Processor Design Using Multiple Clock Domains with Dynamic Voltage and Frequency Scaling. In *Proceedings of the 8th Annual International Symposium on High-Performance Computer Architecture*, pages 29–40, February 2002.

[95] John S. Seng, Eric S. Tune, and Dean M. Tullsen. Reducing Power with Dynamic Critical Path Information. In *Proceedings of the 34th Annual International Symposium on Microarchitecture*, pages 114–123, December 2001.

[96] Simha Sethumadhavan, Rajagopalan Desikan, Doug Burger, Charles R. Moore, and Stephen W. Keckler. Scalable Memory Disambiguation for High ILP Processors. In *Proceedings of the 36th Annual International Symposium on Microarchitecture*, pages 399–410, December 2003.

[97] Simha Sethumadhavan, Franziska Roesner, Joel S. Emer, Doug Burger, and Stephen W. Keckler. Late-Binding: Enabling Unordered Load-Store Queues. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, pages 347–357, June 2007.

[98] Madhu S.Govindan, Doug Burger, Stephen W.Keckler, and the TRIPS Team. TRIPS: A Distributed Explicit Data Graph Execution Microprocessor. *Hot Chips 19: A Symposium of High Performance Chips*, August 2007.

[99] H. Shafi, P. J. Bohrer, J. Phelan, C. A. Rusu, and J. L. Peterson. Design and Validation of a Performance and Power Simulator for PowerPC Systems. *IBM Journal of Research and Development*, 47(5/6):641–651, 2003.

[100] Timothy Sherwood, Erez Perelman, and Brad Calder. Basic Block Distribution Analysis to Find Periodic Behavior and Simulation Points in Applications. In *Proceedings of the Internationl Symposium on Parallel Architectures and Compilation Techniques (PACT)*, pages 3–14, September 2001.

[101] Aaron Smith, Jim Burrill, Jon Gibson, Bertrand Maher, Nick Nethercote, Bill Yoder, Doug Burger, and Kathryn S. McKinley. Compiling for EDGE architectures. In *International Symposium on Code Generation and Optimization*, pages 185–195, March 2006.

[102] Aaron Smith, Ramadass Nagarajan, Karthikeyan Sankaralingam, Robert McDonald, Doug Burger, Stephen W. Keckler, and Kathryn S. McKinley. Dataflow Predication. In *Proceedings of the 39th Annual International Symposium on Microarchitecture*, pages 89–100, December 2006.

[103] Viji Srinivasan, David Brooks, Michael Gschwind, Pradip Bose, Victor Zyuban, Philip N. Strenski, and Philip G. Emma. Optimizing Pipelines for Power and Performance. In *Proceedings of the 35th Annual International Symposium on Microarchitecture*, pages 333–344, November 2002.

[104] Mike Stein. Crossing the Abyss: Asynchronous Signals in a Synchronous World. http://www.edn.com/index.asp?layout=article&articleid=CA310388.

[105] John E. Stone, David Gohara, and Guochun Shi. OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems. *Computing in Science and Engineering*, 12:66–73, 2010.

[106] D. Stroobandt and J. Van Campenhout. Accurate Interconnection Length Estimations for Predictions Early in the Design Cycle. In *VLSI Design, Special Issue on Physical Design in Deep Submicron*, volume 10, pages 1–20, 1999.

[107] Synopsys, Inc. PrimePower: Full-Chip Dynamic Power Analysis for Multi-million Gate Designs. www.synopsys.com/products/power/primepower_ds.pdf.

[108] Synopsys Incorporated. VCS: Comprehensive RTL Verification Solution. http://www.synopsys.com/products/simulation/simulation.html.

[109] D. Tarjan, M. Boyer, and K. Skadron. Federation: Out-of-Order Execution using Simple In-Order Cores. Technical Report CS-2007-11, University of Virginia, Department of Computer Science, August 2007.

[110] David Tarjan, Shyamkumar Thoziyoor, and Norman Jouppi. Cacti 4.0. HP Labs Technical Report, HPL-2006-86. http://www.hpl.hp.com/techreports/2006/HPL-2006-86.pdf.

[111] M. Taylor, W. Lee, S. Amarasinghe, and A. Agarwal. Scalar Operand Networks: On-chip Interconnect for ILP in Partitioned Architectures. In *Proceedings of the 9th Annual International Symposium on High Performance Computer Architecture*, pages 341–353, February 2003.

[112] The Intel Thread Building Blocks. http://www.threadingbuildingblocks.org/.

[113] William Thies, Michal Karczmarek, Michael Gordon, David Z. Maze, Jeremy Wong, Henry Hoffman, Matthew Brown, and Saman Amarasinghe. StreamIt: A Compiler for Streaming Applications. Technical Report MIT/LCS Technical Memo LCS-TM-622, Massachusetts Institute of Technology, Cambridge, MA, Dec 2001.

[114] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez. Reducing Power in High-Performance Microprocessors. In *Proceedings of the 1998 International Design Automation Conference*, pages 732–737, June 1998.

[115] UPC Consortium. UPC Language Specifications, v1.2. Technical Report LBNL-59208, Lawrence Berkeley National Lab, 2005.

[116] Madhavi Valluri and Lizy John. Is Compiling for Performance == Compiling for Power? In *Proceedings of the 2001 Workshop on Interaction Between Compilers and Computer Architectures (INTERACT-5)*, January 2001.

[117] Madhavi G. Valluri, Lizy K. John, and Kathryn S. McKinley. Low-power, Low-Complexity Instruction Issue using Compiler Assistance. In *Proceedings of the 19th Annual International Conference on Supercomputing*, pages 209–218, June 2005.

[118] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar. An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS. In *Proceedings of the 2007 International Solid-State Circuits Conference*, pages 98–589, February 2007.

[119] Vasanth Venkatachalam and Michael Franz. Power Reduction Techniques for Microprocessor Systems. *ACM Computing Survey*, 37(3):195–237, 2005.

[120] Elliot Waingold, Michael Taylor, Devabhaktuni Srikrishna, Vivek Sarkar, Walter Lee, Victor Lee, Jang Kim, Matthew Frank, Peter Finch, Rajeev Barua, Jonathan Babb, Saman Amarasinghe, and Anant Agarwal. Baring It All to Software: RAW Machines. *IEEE Computer*, 30(9):86–93, September 1997.

[121] Watanabe, Yasuko and Davis, John D. and Wood, David A. WiDGET: Wisconsin Decoupled Grid Execution Tiles. In *Proceedings of the 37th Annual International Symposium on Computer Architecture*, pages 2–13, June 2010.

[122] Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker. Scheduling for reduced CPU energy. In *Proceedings of the 1st Conference on Operating Systems Design and Implementation (OSDI)*, pages 13–23, November 1994.

[123] Steve Wilton and Norman Jouppi. An Enhanced Access and Cycle Time Model for On-Chip Caches. WRL Research Report 93/5. http://www.hpl.hp.com/techreports/Compaq-DEC/WRL-93-5.pdf.

[124] Wonyoung Kim, Meeta S. Gupta, Gu-Yeon Wei, and David Brooks. On-Chip VRM Design. Through Private Communication.

[125] Qiang Wu, Margaret Martonosi, Douglas W. Clark, V. J. Reddi, Dan Connors, Youfeng Wu, Jin L ee, and David Brooks. A Dynamic Compilation Framework for Controlling Microprocessor Energy and Performance. In *Proceedings of the*

38th Annual International Symposium on Microarchitecture, pages 271–282, November 2005.

[126] Yi-Ping You, Chingren Lee, and Jenq Kuen Lee. Compilers for Leakage Power Reduction. *ACM Transactions on Design Automation of Electronic Systems*, 11(1):147–164, 2006.

[127] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan. HotLeakage: A Temperature-Aware Model of Subthreshold and Gate Leakage for Architects. Technical Report CS-2003-05, University of Virginia, Department of Computer Science, March 2003.

[128] Wei Zhao and Yu Cao. New generation of predictive technology model for sub-45nm design exploration. In *Proceedings of the 7th International Symposium on Quality Electronic Design (ISQED)*, pages 585–590, 2006.

[129] Hongtao Zhong, Steven A. Lieberman, and Scott A. Mahlke. Extending multicore architectures to exploit hybrid parallelism in single-thread applications. In *Proceedings of the 13th Annual International Conference on High Performance Computer Architecture*, pages 25–36, February 2007.

[130] Victor Zyuban, David Brooks, Viji Srinivasan, Michael Gschwind, Pradip Bose, Philip N Strenski, and Philip G Emma. Integrated Analysis of Power and Performance of Pipelined Microprocessors. volume 53, pages 1004–1016, August 2004.

Vita

Madhu Saravana Sibi Govindan (official name: Madhu Sarava Govindan) was born as the only son to Govindan K. and Vasanthi Govindan S., on 10th September, 1980, in the town of Palayamkottai, in the state of Tamil Nadu, India. He attended high school in various places including Tirunelveli, Tuticorin, and Chennai. Later, he enrolled in the College of Engineering, Guindy, Anna University, for his undergraduate education in 1998. He graduated with a Bachelors degree in Computer Science and Engineering in 2002. After working in Bangalore for a company called Trilogy E-Business Software for about a year, he joined the Ph.D. program in the Computer Sciences Department at the University of Texas at Austin in 2003. While enrolled in the Ph.D. program he earned his Masters degree in Computer Science in 2006.

Permanent Address: New No. 12, Old No. 32, B Block
BBC Apartments, Vaithyram Street,
T. Nagar, Chennai - 600017, India.

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.