Near Threshold Computing: From Single Core to Many-Core Energy Efficient Architectures

by

Ronald Dreslinski, Jr.

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctorate of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2011

Doctoral Committee:

Professor Trevor N. Mudge, Chair
Professor David Blaauw
Professor Dennis Michael Sylvester
Assistant Professor Thomas F. Wenisch

ACKNOWLEDGEMENTS

I would like to start by thanking Steve Reinhardt, who in his wisdom took a chance on me as in incoming graduate student. As my advisor for my first several years here his advice was invaluable. When Steve left the University, it was Trevor Mudge who adopted Steve's students. I am thankful for the advice he gave me as I switched major research areas and began to focus on what would eventually become my thesis. His career and research advice were important to my development as a student and researcher. In addition I would like to thank Trevor for allowing me the opportunity to build my skill set by including me in countless research proposal writing sessions and allowing me a glimpse behind the curtain of being a faculty member.

In addition, I was closely advised by several other faculty members throughout my long career at the University of Michigan. I would like to acknowledge the outside advice from Chaitali Chakrabarti at Arizona State University as well as Krisztian Flautner from ARM for their thoughtful advice. Within the University of Michigan I also received weekly advice from intergroup meetings with David Blaauw, Dennis Sylvester, Thomas Wenisch, and Scott Mahlke. It was these collaborative relationships that yielded such a broad dissertation topic.

In my early years as a graduate student I was lucky to have several senior students as mentors. I would like to particularly thank Nate Binkert for my early experiences at preparing a research paper. In addition both Steve Raasch and Eric Hallnor provided useful guidance. In my early years as a graduate student I got involved designing a simulator (not something you should do if you want to graduate quickly). Through this experience I gained several important colleagues. For that reason I would like to thank those other M5 simulator developers that helped me understand how software should be written. Particularly I would like to thank Ali Saidi, Lisa Hsu, Kevin Lim, Gabe Black, and Korey Sewell in addition to those I have already mentioned earlier.

When I joined Trevor's research group I gained another set of colleagues. I wold like to particularly thank Geoff Blake for putting up with my demanding requests for perfection in every plot he made. Also I would like to thank Tae Ho Kgil, Sangwon Seo, Mark Woh, and Dave Roberts. The newer students who have worked tirelessly to help me run simulations

also deserve an acknowledgment, so thank you Joe Pusdesris, Tony Gutierrez, and Mike Cieslak.

I received a great deal of help from the circuits students working with David Blaauw and Dennis Sylvester. My closest and most rewarding work was done with Bo Zhai, Sudhir Satpathay, Dave Fick, Greg Chen, Michael Wieckowski, and Bharan Girdihar. In addition there were a whole host of other students who assisted in designing and taping out a massive 7-layer 3D stacked chip with 128 commercial cores on it (taping out a chip is the second thing I got involved with that you should not do if you want to graduate quickly), and for there help I am grateful.

Of course I needed some people to help keep my mind off of research every waking minute and for that I would like to thank my friends at the Blue Leprechaun, Brown Jug, and Good Time Charley's. In particular several friends helped me keep one foot in reality during all these years and those are Jeff Wegehaupt, Nenad Milosavljevic, Mike Gradillas, Jose Orozco, Matt Petry, and Thomas M. Dziuszko.

Lastly I would like to thank my family for helping me make it through all these years. In particular my parents Ronald and Margaret for there patience and monetary help. And my sisters Kathy and Laurie for being everything a brother needed and more.

TABLE OF CONTENTS

CHAPTER

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Near Threshold Computing: From Single Core to Many-Core Energy Efficient Architectures

by

Ronald Dreslinski, Jr

Chair: Trevor N. Mudge

Over the past four decades, the number of transistors on a chip has increased exponentially in accordance with Moore's law. This has led to progress in diversified computing applications, such as health care, education, security, and communications. A number of societal projections and industrial roadmaps are driven by the expectation that these rates of improvement will continue, but the impediments to growth are more formidable today than ever before. The largest of these barriers is related to energy and power dissipation, and it is not an exaggeration to state that developing energy-efficient solutions is critical to the survival of the semiconductor industry. Extensions of today's solutions can only go so far, and without improvements in energy efficiency they are in danger of running out of steam.

When examining the history of computers, a pattern emerges: successive generations of technologies, ranging from vacuum tubes to bipolar to NMOS-based technologies, were replaced when their energy overheads became prohibitive. However, there is no clear successor to todays technology, CMOS. The available alternatives are far from being commercially viable, and none has gained sufficient traction, or provided the economic justification for overthrowing the large investments made in today's CMOS-based infrastructure. Therefore, there is a strong case supporting the position that solutions to the power conundrum must come from enhanced devices, design styles and architectures, rather than a reliance on the promise of radically new technologies becoming commercially viable. This dissertation proposes that the solution to this energy crisis is the universal application of aggressive low voltage operation across all computation platforms. This can be accomplished by targeting so-called "near-threshold computing" (NTC) and by proposing novel methods to overcome the barriers that have historically relegated ultra-low voltage operation to niche markets. In particular, this dissertation explores the performance barrier that prevents the widespread adoption of NTC and provides architectures for single-core, multi-core, many-core, and digital signal processing systems. As a final proof of concept a 3D integrated prototype of 128 ARM Cortex-M3 cores and 256MB of DRAM is presented that shows a 6.4X improvement in energy-efficiency over the Intel Atom processor.

CHAPTER 1

Introduction

Over the past four decades, the number of transistors on a chip has increased exponentially in accordance with Moore's law [63]. Moore's seemingly innocuous observation made in 1965, characterized in precise technical terms by Dennard [24] in 1974, has continued to hold until the present day. This has fueled the computing growth that we have come to expect—improved performance, increased density, and reduced energy per device. The ramifications have been widespread for all segments of society, improving health care, education, security, communications, etc. Indeed devices almost unimaginable a generation ago are common place in todays society, such as cell phones, navigation systems, and cloud computing. If Moore's law continues to hold we can expect further exciting developments for all segments of society. For example it may enable handheld medical devices, such as diagnostic imaging devices, that presently are limited in use by their immense cost and size. Thus turning machines like ultrasound scanners into common day devices like the thermometer found in everyones medicine cabinet.

Todays current technology, complementary metal oxide semiconductors (CMOS), has continued to march in the direction of miniaturization per Moore's law [63]. New silicon-based technologies such as FinFET devices [37] and 3D integration [87] provide a path to increasing transistor counts in a given footprint. However, using device size as the metric of progress has become misleading since improvements in packing densities no longer translate into proportionate increases in performance or energy efficiency. Starting around the 65nm node, device scaling no longer delivers the energy gains that drove the semiconductor growth of the past several decades, as shown in Figure 1.1. The supply voltage has remained essentially constant since then and dynamic energy efficiency improvements have stagnated, while leakage currents continue to increase. Heat removal limits at the package level have further restricted more advanced integration. Together, such factors have created a curious design dilemma: more gates can now fit on a die, but a growing fraction cannot actually be used due to strict power limits.

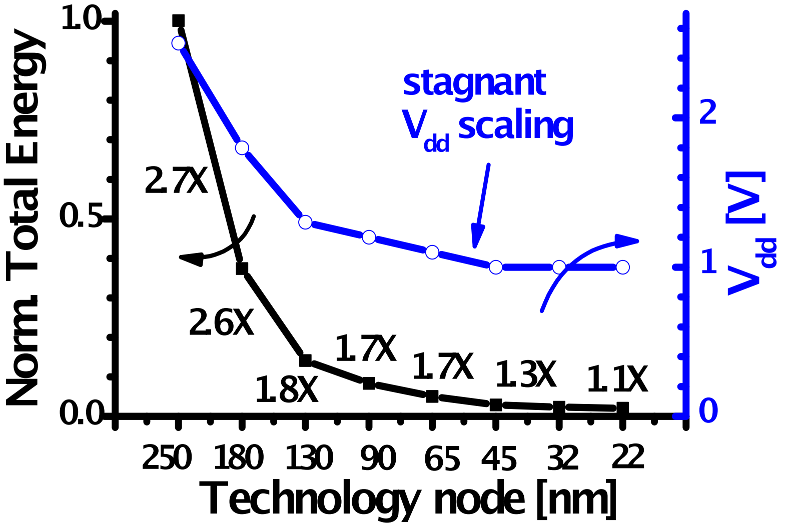


Figure 1.1: Technology scaling trends of supply voltage and energy.

At the same time, we are moving to a "more than Moore" world, with a wider diversity of applications than the microprocessor or ASICs of ten years ago. Tomorrow's design paradigm must enable designs catering to application spanning from high-performance processors and portable wireless applications, to sensor nodes and medical implants. Energy considerations are vital over this entire spectrum, including:

1) High-performance computers, targeted for use in data centers. These create large amounts of heat and require major investments in power and cooling infrastructure, resulting in major environmental and societal impact. In 2006 data centers consumed 1.5% of total US electricity, equal to the entire US transportation manufacturing industry [68], and alarmingly, data center power is projected to double every $\sim$5 years. These systems are the backbone of cloud computing.

2) Personal computers are becoming increasingly wireless and miniaturized, and are limited by tradeoffs between battery lifetimes (days) and computational requirements (e.g., high-definition video). Wireless applications increasingly rely on digital signal processing. While Moore's law enables greater transistor density, only a fraction may be used at a time due to power limitations and application performance is therefore muzzled by power limits, often in the 500mW-5W range.

3) Sensor-based computer systems critically depend on ultra-low power ($\leq\mu$W in standby) and reduced form-factor (mm^3). These promise to unlock new semiconductor applications, such as implanted monitoring and actuation of medical devices, as well as ubiquitous environmental monitoring, e.g., structural sensing within critical infrastructure elements such as bridges.

The aim of the designer in this era is to overcome the challenge of energy efficient computing and unleash performance from the reins of power to re-enable Moores law in the semiconductor industry. The proposed strategy is to provide 10 times (10X) or more improvements in energy efficiency at constant performance through widespread application of near-threshold computing (NTC), where devices are operated at or near their threshold voltage (V_{th}). By reducing supply voltage (V_{dd}) from a nominal 1.1V to 400-500mV, NTC obtains as much as 10X energy efficiency gains and represents the re-establishment of voltage scaling and its associated energy efficiency gains.

The use of ultra-low voltage operation, and in particular subthreshold operation ($V_{dd} < V_{th}$), was first proposed over three decades ago when the theoretical lower limit of V_{dd} was found to be 36mV [84]. However, the challenges that arise from operating in this regime have kept subthreshold operation confined to a handful of minor markets, such as wristwatches and hearing aids. To the mainstream designer, ultra-low voltage design has remained little more than a fascinating concept with no practical relevance. However, given the current energy crisis in the semiconductor industry and stagnated voltage scaling some foresee the need for a radical paradigm shift where ultra-low voltage operation is applied across application platforms and forms the basis for renewed energy efficiency.

NTC does not come without some barriers to widespread acceptance. In this dissertation three key challenges are identified that have been poorly addressed to date with respect to low voltage operation, specifically: 1) 10X or greater loss in performance, 2) 5X increase in performance variation, and 3) 5 orders of magnitude increase in functional failure rate of memory as well as increased logic failures. Overcoming these barriers is a formidable challenge requiring a synergistic approach combining methods from the algorithm and architecture levels to circuit and technology levels. Chapter 2 presents a top level view of these barriers and current research being conducted to overcome them. The next chapter, Chapter 3, presents a cache architecture that uses NTC to lower the power of single core systems where area is a major constraint. The remaining chapters will pick up the main theme of "Overcoming Performance Loss", which is the primary focus of this dissertation.

Solving the performance loss associated with NTC is critical in order to meet the stringent demands of the broad range of future applications, from tiny sensors to warehouse scale servers (such as Google and Facebook). There are a variety of code and architectures for this broad application space. For many, the use of parallelism—splitting the code up onto multiple cores—can be accompanied with running each core slower to consume less total power. The result can be used to improve the performance of the system while still retaining the energy gains of NTC operation. In Chapter 4 this dissertation explores the use of parallelism in multi-core processors for small scale systems, targeting handheld, laptop, and desktop computing. In such systems NTC operation creates different optimal operating points for logic and memory devices, as such, NTC offers a unique design tradeoff where several cores can be clustered together to connect to a single cache, reducing power. In Chapter 5, this work is expanded to cover digital signal processing (DSP) chips, where there are unique architectural constraints due to the highly parallel replication of logic, and more stringent memory constraints.

While parallelism can help overcome performance bottlenecks, not every piece of code or architectural design space allows for parallelism. For example, in small sensor processors, where form factor is essential, replicating a design requires to much overhead. Even for systems that can exploit parallelism, inherently sequential pieces of code/work can still form bottlenecks to scalability. To address this, in Chapter 6 the work on multi-core and many-core architectures is augmented by the use of voltage boosting—increasing the voltage for short periods of time to overcome serial bottlenecks in parallel applications.

Finally, Chapter 7 expands the multi-core research into systems with 100's of cores, currently referred to as many-core architectures. In particular the design of a 3D integrated chip—one where multiple silicon wafers are stacked together and connected with tiny through-silicon-via's—is explored. This type of 3D design provides an interesting synergy with NTC, particularly because the energy-efficiency of NTC designs reduces the thermal management constraints associated with large 3D stacks of silicon. In this Chapter a prototype design, Centip3De, is developed for fabrication and the energy savings of the design are analyzed. The Centip3De system connects 128 ARM Cortex-M3 processors and 256 MB of DRAM in a 7-layer 3D stack, and was sent for fabrication in a 130nm process. This design achieves an energy efficiency improvement of 6.4X compared to an Intel Atom processor implemented in 45nm technology.

Ultimately this dissertation shows through detailed architectural simulation and chip design/fabrication that Near Threshold Computing is a viable way to extend the lifetime of Moore's law. By extending its life we will enable the continued growth in computational power that designers and users have come to expect.

CHAPTER 2

Near Threshold Operation

Power has become the primary design constraint for chip designers today. While Moores law continues to provide additional transistors, power budgets have begun to prohibit those devices from actually being used. To reduce energy consumption, voltage scaling techniques have proved a popular technique with subthreshold design representing the extreme endpoint of voltage scaling. Although it is extremely energy efficient, subthreshold design has been relegated to niche markets due to its major performance penalties. This chapter defines and explores Near Threshold Computing (NTC), a design space where the supply voltage is approximately equal to the threshold voltage of the transistors. This region retains much of the energy savings of subthreshold operation with more favorable performance and variability characteristics. This makes it applicable to a broad range of power-constrained computing segments from sensors to high performance servers.

NTC does not come without some barriers to widespread acceptance. In this chapter the focus is on three key challenges that have been poorly addressed to date with respect to low voltage operation, specifically: 1) 10X or greater loss in performance, 2) 5X increase in performance variation, and 3) 5 orders of magnitude increase in functional failure rate of memory as well as increased logic failures. Overcoming these barriers is a formidable challenge requiring a synergistic approach combining methods from the algorithm and architecture levels to circuit and technology levels. The focus of this dissertation is on overcoming the performance loss. Other research is also being conducted to cover the remaining two barriers, which will be discussed briefly in this chapter. Much of the Introduction and this chapter was conducted with the collaboration of my colleague Michael Wieckowski. This work appears as a part of a journal article in The Proceedings of the IEEE [25].

2.1 Defining Operating Regions

Energy consumption in modern CMOS circuits largely results from the charging and discharging of internal node capacitances and can be reduced quadratically by lowering supply voltage (V_{dd}). As such, voltage scaling has become one of the more effective methods to reduce power consumption in commercial parts. It is well known that CMOS circuits function at very low voltages and remain functional even when V_{dd} drops below the threshold voltage (V_{th}). In 1972, Meindl et al. derived a theoretical lower limit on V_{dd} for functional operation, which has been approached in very simple test circuits [84, 33]. Since this time, there has been interest in subthreshold operation, initially for analog circuits [92, 60, 61] and more recently for digital processors [81, 69, 48, 93, 16, 102], demonstrating operation at V_{dd} below 200mV. However, the lower bound on V_{dd} in commercial applications is typically set to 70% of the nominal V_{dd} due to concerns about robustness and performance loss [88, 44, 39].

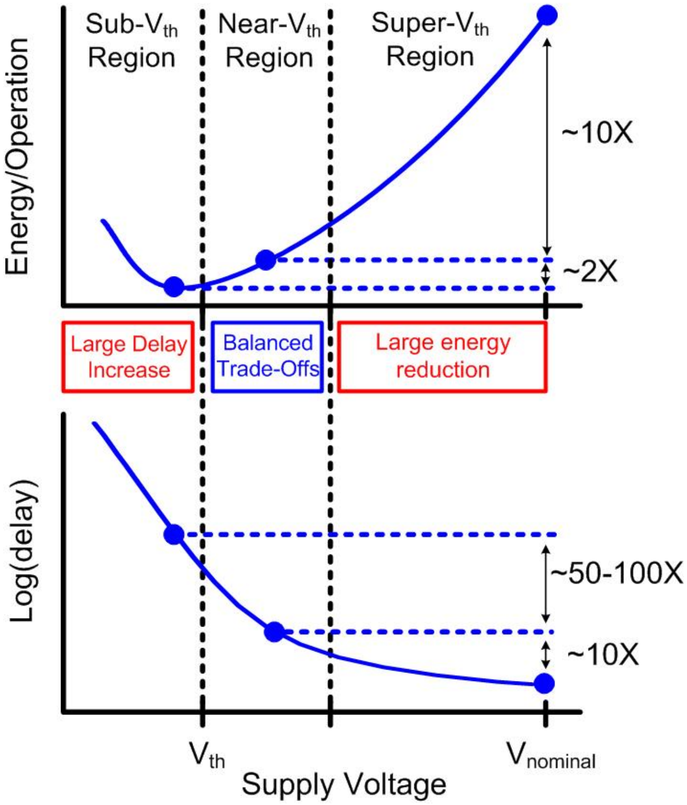


Figure 2.1: Energy and delay in different supply voltage operating regions.

Given such wide voltage scaling potential, it is important to determine the V_{dd} at which the energy per operation (or instruction) is optimal. In the superthreshold regime ($V_{dd} > V_{th}$), energy is highly sensitive to V_{dd} due to the quadratic scaling of switching energy with V_{dd}. Hence voltage scaling down to the near-threshold regime ($V_{dd} \sim V_{th}$) yields an energy reduction on the order of 10X at the expense of approximately 10X performance degradation, as seen in Figure 2.1 [98]. However, the dependence of energy on V_{dd} becomes more complex as voltage is scaled below V_{th}. In subthreshold ($V_{dd} < V_{th}$), circuit delay increases exponentially with V_{dd}, causing leakage energy (the product of leakage current, V_{dd}, and delay) to increase in a near-exponential fashion. This rise in leakage energy eventually dominates any reduction in switching energy, creating an energy minimum seen in Figure 2.1. The near threshold operating point is meant to be near the threshold voltage but the optimal location may vary depending on application and technology node.

The identification of an energy minimum led to interest in processors that operate at this energy optimal supply voltage [93, 102, 34] (referred to as V_{min} and typically 250mV-350mV). However, the energy minimum is relatively shallow. Energy typically reduces by only ~2X when V_{dd} is scaled from the near-threshold regime (400-500mV) to the subthreshold regime, though delay rises by 50-100X over the same region. While acceptable in ultra-low energy sensor-based systems, this delay penalty is not tolerable for a broader set of applications. Hence, although introduced roughly 30 years ago, ultra-low voltage design remains confined to a small set of markets with little or no impact on mainstream semiconductor products.

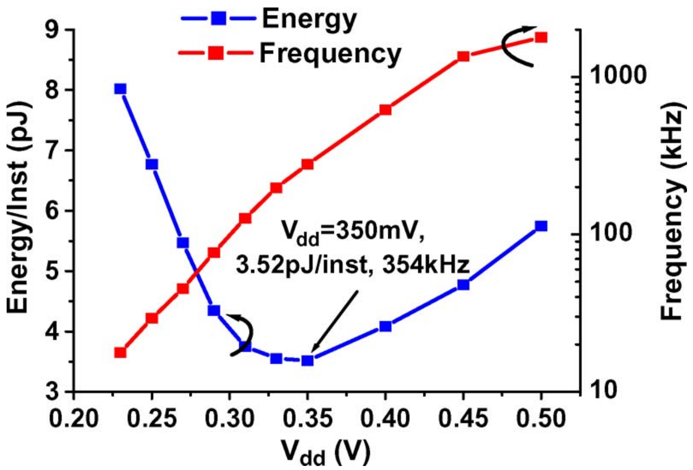


Figure 2.2: Subliminal processor frequency and energy breakdowns at various supply voltages.

2.2 NTC Analysis

Recent work at many leading institutions has produced working processors that operate at subthreshold voltages. For instance, the Subliminal [34] and Phoenix processors [80]

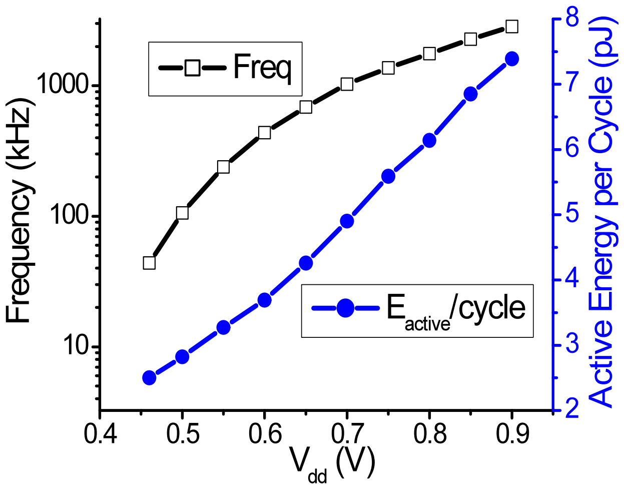


Figure 2.3: Phoenix frequency and energy breakdowns at various supply voltages.

designed by Hanson et al. provide the opportunity to experimentally quantify the NTC region and how it compares to the subthreshold region. Figure 2.2 and 2.3 present the energy breakdown of the two different designs as well as the clock frequency achieved across a range of voltages. As discussed in Section 2.1, there is a V_{min} operating point that occurs in the subthreshold region where energy usage is optimized, but clock frequencies are limited to sub-1MHz values (not pictured for Phoenix as testing was not conducted in subthreshold). On the other hand, only a modest increase in energy is seen operating at the NTC region (around 0.5V), while frequency characteristics at that point are significantly improved. For example, at nominal voltages, the Subliminal processor runs at 20.5 MHz and 33.1 pJ/inst, while at NTC voltages, a 6.6X reduction in energy and an 11.4X reduction in frequency are observed. For the Phoenix processor a nominal 9.13 MHz and 29.6 pJ/inst translate to a 9.8X reduction in energy and a 9.1X reduction in frequency. These tradeoffs are much more attractive than those seen in the subthreshold design space and open up a wide variety of new applications for NTC systems.

2.3 NTC in Different Computing Segments

2.3.1 NTC Integration in Ultra Energy-Efficient Servers

The exponential growth of the web has yielded a dramatic increase in the demand for server style computers with the installed base of servers expected to exceed 40 million by 2010 [40]. Server growth is accompanied by an equally rapid growth in the energy demand to power them. For example, it is estimated that the five largest internet sites consume at least 5MW each [47].

The tier 1 of a data center that serves web pages provides a perfect opportunity for NTC. The requests in these servers represent the bulk of requests to these data centers [82], consuming 75% of the overall energy. The workload is a stream of independent requests to render web pages that can be naturally executed in parallel. HTML is fetched from memory, subjected to relatively simple operations, and returned to memory without requiring extensive shared data. To achieve this, 10-100s of NTC cores on a single die can be used to obtain very high throughput with unprecedented energy efficiency.

2.3.2 NTC Integration in Personal Computing

The personal computing platform continues to evolve rapidly. *WiFi* is a standard on laptops but other mobile wireless communications are also starting to be supported. Future devices must be able to move seamlessly among communication alternatives. Such systems will combine a high level of processing power along with signal processing capabilities integrated into a much smaller form factor than todays laptops. Battery life is expected on the order of days, while functionality requirements are extreme and may include high-definition video, voice recognition, along with a range of wireless standards. The features of PC platforms that distinguish them from the two other systems are the dual needs to cope with variable workloads and energy efficient wireless communication.

In the PC platform space, cores may run at widely varying performance/energy points. The voltage and frequency of the cores and their supporting peripherals can be dynamically altered in real time to meet the constraints of performance and power consumption. This dynamic voltage and frequency scaling technique (DVFS) can be leveraged to enable adaptive NTC circuits in the personal computing space [71]. The scaling method may be driven by operating system commands and/or distributed sensors. Exploiting phase variations in workloads [11], efficient phase detection techniques need to be established for multicore multithreaded processors to enable power management schemes in achieving savings without significantly compromising performance.

2.3.3 NTC Integration in Sensor Networks

With advances in circuit and sensor design, pervasive sensor-based systems, from single to thousands of nodes, are quickly becoming a possibility. A single sensor node typically consists of a data processing and storage unit, off-chip communication, sensing elements, and a power source. They are often wirelessly networked and have potential applications in a wide range of industrial domains, from building automation to homeland security to biomedical implants. The versatility of a sensor is directly linked to its form factor—for a sensor to be truly useful in many new application areas, a form factor on the order of 1mm^3 is desirable while maintaining a lifetime of months or years.

To meet the above requirements, the key limiting constraint is energy. Both sensors and electronics are readily shrunk to $<$1mm^3 in modern technologies. However, current processors and communication systems require batteries that are many orders of magnitude larger than the electronics themselves (e.g., 50mm^3 processor die in a laptop vs. 167cm^3 4-cell lithium-ion battery). Hence, whether a sensor node is powered through batteries, harvesting, or both, power consumption will limit overall system size. To integrate a sensor node in $<$1mm^3, energy levels must be reduced by 4-7 orders of magnitude. Processing speed is not a major constraint in most sensor applications [67], easing the integration of NTC. Initial investigations showed simple sensor architectures coupled with NTC can obtain an active energy reduction of $\geq$100X [102].

2.4 NTC Barriers

Although NTC provides excellent energy-frequency tradeoffs, it brings its own set of complications. NTC faces three key barriers that must be overcome for widespread use; performance loss, performance variation, and functional failure. The following subsections discuss why each of these issues arises and why they pose problems to the widespread adoption of NTC. Section 2.5 then addresses the recent work related to each of these barriers. The focus of this dissertation will be on overcoming the performance loss, although brief discussions of the other barriers is addressed for completeness.

2.4.1 Performance Loss

The performance loss observed in NTC, while not as severe as that in subthreshold operation, poses one of the most formidable challenges for NTC viability. In an industrial 45nm technology the fanout-of-four inverter delay (FO4, a commonly used metric for the intrinsic speed of a semiconductor process technology) at an NTC supply of 400mV is 10X slower than at the nominal 1.1V. There have been several recent advances in architectural and circuit techniques that can regain some of this loss in performance. These techniques, described in detail in Section 2.5.1, center around aggressive parallelism with a novel NTC oriented memory/computation hierarchy. The increased communication needs in these architectures is supported by the application of 3D chip integration, as made feasible by the low power density of NTC circuits. In addition, new technology optimizations that opportunistically leverage the significantly improved silicon wearout characteristics (e.g., oxide breakdown) observed in low voltage NTC can be used to regain a substantial portion of the lost performance.

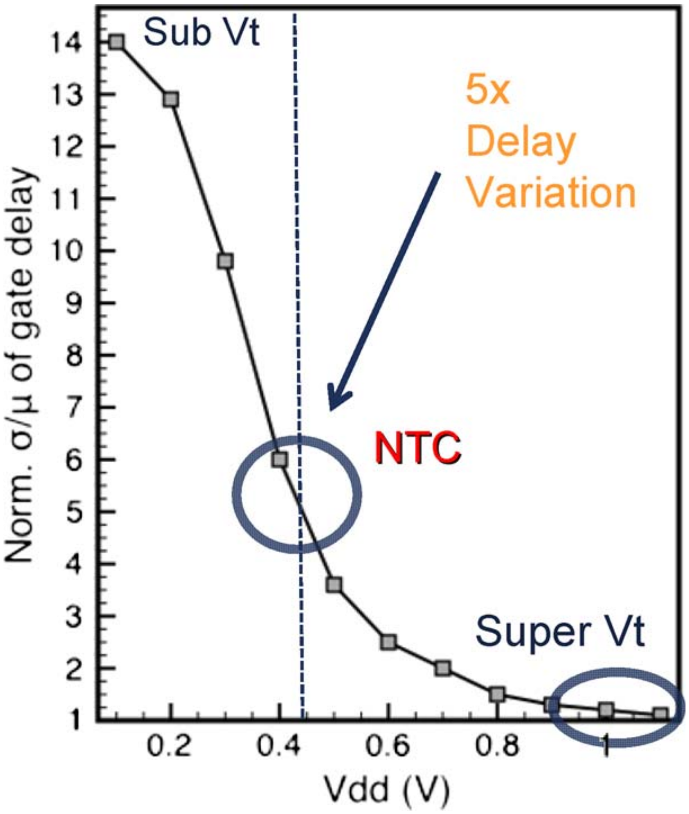


Figure 2.4: Impact of voltage scaling on gate delay variation.

2.4.2 Increased Performance Variation

In the near-threshold regime, the dependencies of MOSFET drive current on V_{th}, V_{dd}, and temperature approach exponential. As a result, NTC designs display a dramatic increase in performance uncertainty. Figure 2.4 shows that performance variation due to global process variation alone increases by approximately 5X from $\sim$30% (1.3X) [13] at nominal operating voltage to as much as 400%, (5X) at 400mV. Operating at this voltage also heightens sensitivity to temperature and supply ripple, each of which can add another factor of 2X to the performance variation resulting in a total performance uncertainty of 20X. Compared to a total performance uncertainty of $\sim$1.5X at nominal voltage, the increased performance uncertainty of NTC circuits looms as a daunting challenge that has caused most designers to pass over low voltage design entirely. Simply adding margin so that all chips will meet the needed performance specification in the worst-case is effective in nominal voltage design. In NTC design this approach results in some chips running at

1/10th their potential performance, which is wasteful both in performance and in energy due to leakage currents. Section 2.5.2 presents a new architectural approach to dynamically adapt the performance of a design to the intrinsic and environmental conditions of process, voltage, and temperature that is capable of tracking over the wide performance range observed in NTC operation. This method is complemented by circuit-level techniques for diminishing the variation of NTC circuits and for efficient adaptation of performance.

2.4.3 Increased Functional Failure

The increased sensitivity of NTC circuits to variations in process, temperature and voltage not only impacts performance but also circuit functionality. In particular, the mismatch in device strength due to local process variations from such phenomena as random dopant fluctuations (RDF) and line edge roughness (LER) can compromise state holding elements based on positive feedback loops. Mismatch in the loops elements will cause it to develop a natural inclination for one state over the other, a characteristic that can lead to hard functional failure or soft timing failure. This issue has been most pronounced in SRAM where high yield requirements and the use of aggressively sized devices result in prohibitive sensitivity to local variation.

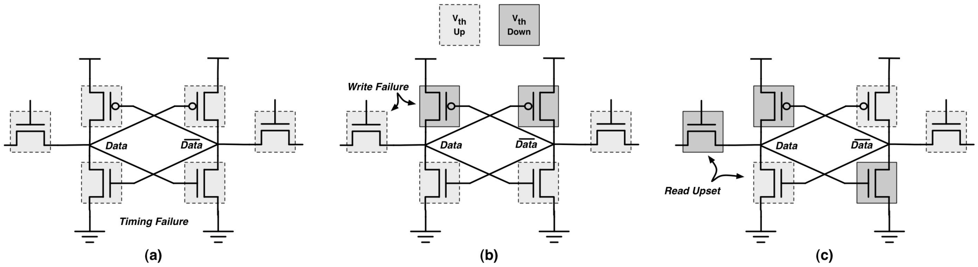


Figure 2.5: Effects of global and local variation on a standard 6T SRAM cell. (a) Global V_{th} reduction resulting in timing failure. (b) Global V_{th} P-N skew resulting in write failure. (c) Local V_{th} mismatch resulting in read upset.

Several variation scenarios for a standard 6T SRAM cell are shown in Figure 2.5. In (a), global process variation has resulted in both P and N devices being weakened by a V_{th} increase resulting in a potential timing failure during both reads and writes. In (b), a similar global effect has introduced skew between the P and N device strengths. This is particularly detrimental when the P is skewed stronger relative to the N resulting in a potential inability to write data into the cell. In (c), random local mismatch is considered and the worst case is shown for a read upset condition. The cell is effectively skewed to favor one state over another, and the weak pull-down on the left side cannot properly combat the strong access device at its drain. As such, the Data node is likely to flip to the

“1” state during normal read operations. While these examples are shown in isolation, a fabricated circuit will certainly experience all of them simultaneously to varying degrees across a die and with different sensitivities to changes in supply voltage and temperature. The resulting likelihood of failure is potentially very high, especially as supply voltage is reduced and feature sizes are shrunk.

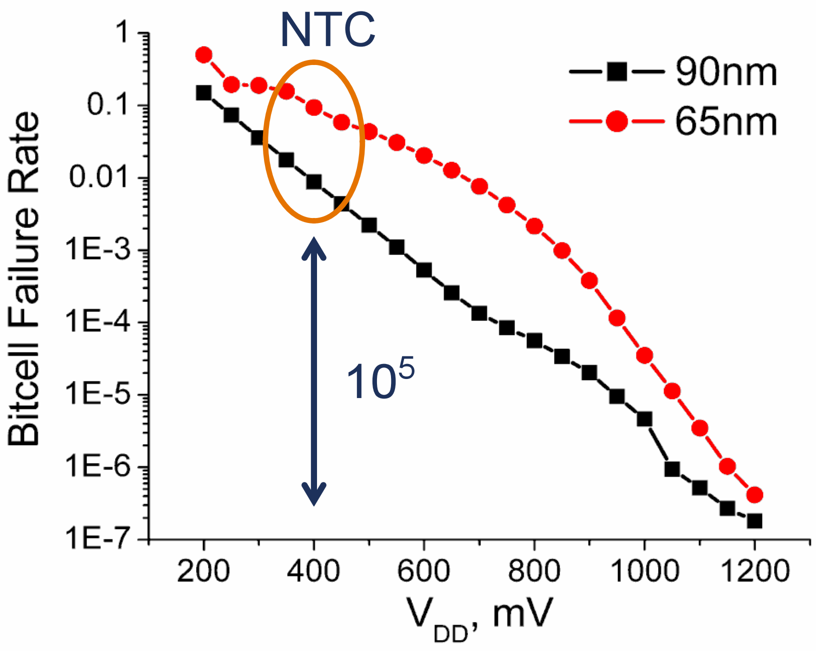


Figure 2.6: Impact of voltage scaling on SRAM failure rates.

For instance, a typical 65nm SRAM cell has a failure probability of $\sim 10^{-7}$ at nominal voltage, as shown in Figure 2.6. This low failure rate allows failing cells to be corrected for using parity checks or even swapped using redundant columns after fabrication. However, at an NTC voltage of 500mV, this failure rate increases by ∼5 orders of magnitude to approximately 4%. In this case, nearly every row and column will have at least one failing cell, and possibly multiple failures, rendering simple redundancy methods completely ineffective. Section 2.5.3 therefore presents novel approaches to robustness ranging from the architectural to circuit levels that address both memory failures and functional failure of flip-flops (FFs) and latches.

2.5 Addressing NTC Barriers

The following subsections will address some of the research being done to overcome each of the barriers presented in Section 2.4.

2.5.1 Addressing Performance Loss

Performance loss, as discussed in Section 2.4.1, can significantly hinder the adoption of NTC. The main focus of this dissertation is architectural techniques to overcome performance loss. The rest of the dissertation will present the parallel architectures to address this bottleneck, in addition to techniques for systems that can not employ parallelism because of either area or algorithmic concerns. These architectures span single-, multi-, and many-core architectures and include techniques for DSP and voltage boosting for serial bottlenecks.

2.5.2 Addressing Performance Variation

As noted in Section 2.4.2, the combined impact of intrinsic process variations and extrinsic variations, such as fluctuations in temperature and supply voltage, results in a spread in the statistical distribution of NTC circuit performance of $\sim$10X compared to designs at nominal supplies. Traditional methods to cope with this issue, which are largely centered on adding design margin, are inadequate and hugely wasteful when voltage is scaled, resulting in a substantial portion of the energy efficiency gain from NTC operation being lost. Hence, in this section architectural and circuit solutions to provide variation tolerance and adaptivity are discussed.

2.5.2.1 Soft Edge Clocking

The device variation inherent to semiconductor manufacturing continues to increase from such causes as dopant fluctuation and other random sources, limiting the performance and yield of ASIC designs. Traditionally, variation tolerant, two-phase, latch-based designs have been used as a solution to this issue. Alternatively, hard-edge data flip-flops (DFF) with intentional or useful skew can be used. Both of these techniques incur a significant penalty in design complexity and clocking overhead.

One potential solution to address timing variation while minimizing overhead is a type of soft-edge flip-flop (SFF) that maintains synchronization at a clock edge, but has a small transparency window, or softness. In one particular approach to soft-edge clocking, tunable inverters are used in a master-slave flip-flop to delay the incoming master clock edge with respect to the slave edge as shown in Figure 2.7.

As a result of this delay, a small window of transparency is generated in the edge-triggered register that accommodates paths in the preceding logic that were too slow for the nominal cycle time—in essence allowing time borrowing within an edge-triggered register environment. Hence, soft edge clocking results in a trade-off between short and long

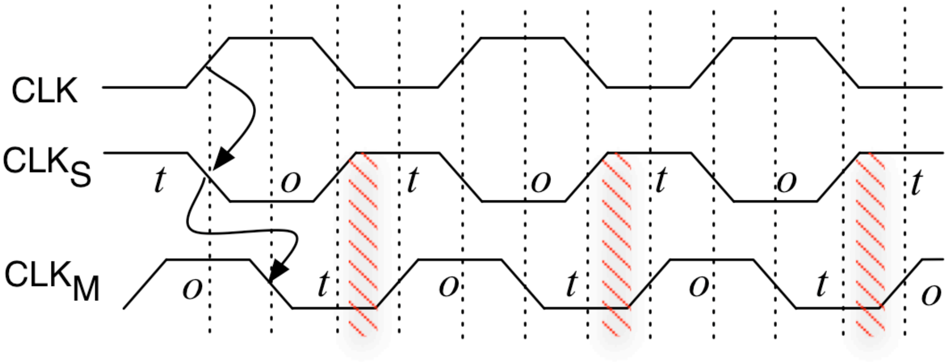


Figure 2.7: Delaying the master clock creates a window of transparency.

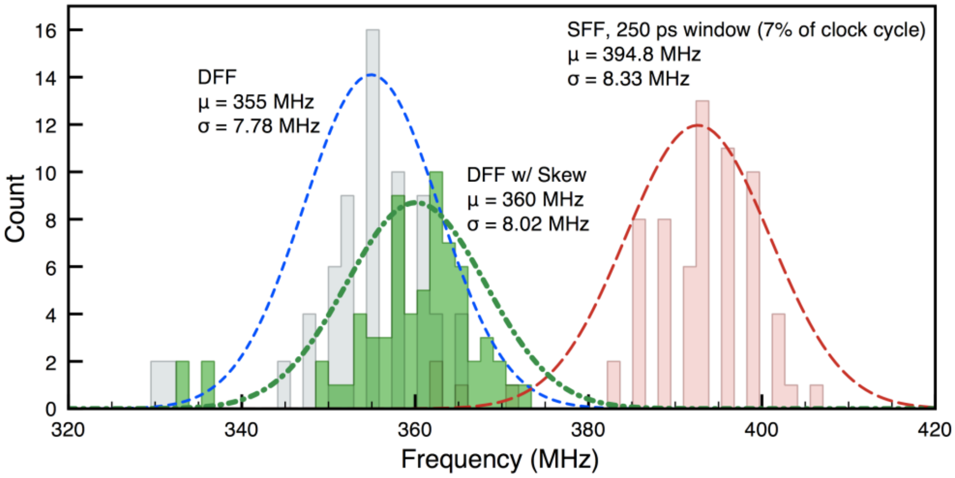


Figure 2.8: FIR filter with soft edge clocking compared to standard flip-flops (SFF); presented with and without useful skew.

paths and is effective at mitigating random, uncorrelated variations in delay, which are significant in NTC. In theoretical explorations at a nominal super-threshold supply voltage, it was shown that soft-edge clocking reduced the mean (standard deviation) clock period in benchmark circuits by up to 22% (25%). Joshi et al. [46] furthered this work by developing a library based on these soft flip-flops and providing a statistical algorithm for their assignment. In the work by Wieckowski et al. [95], this technique was employed in silicon to show that small amounts of softness in a FIR filter achieved improvements in performance of 11.7% over a standard DFF design and improvement of 9.2% compared to a DFF with useful skew. These increases in performance, shown in Figure 2.8, demonstrate a greater tolerance to intra-die variation that becomes even more important in the NTC operating region.

2.5.2.2 Body Biasing

At superthreshold supply voltages, body biasing (BB) is a well known technique for adapting performance and leakage to global variation of process, voltage, and temperature. Its use is becoming more widespread and was recently demonstrated in silicon within a communication processor application [30]. While effective in the superthreshold domain, the influence of body-biasing becomes particularly effectual in the NTC domain where device sensitivity to the threshold voltage increases exponentially. Body-biasing is therefore a strong lever for modulating the frequency and performance in NTC, and is ideally suited as a technique for addressing the increased detriments of process variation in NTC. Further, because P and N regions can be adapted separately using body biasing, and because the relative drive strength of P and N transistors can change dramatically from superthreshold to NTC, body biasing has the added advantage of allowing the P to N ratio of a design to be optimally adjusted.

Hanson et al. [33] show that the extreme sensitivity to process variation in NTC design tends to raise V_{min} and reduce energy efficiency. They explore the use of adaptive body-bias (ABB) techniques to compensate for this variation both locally and globally. Indeed, their later work on a subthreshold processor [34] implements these techniques in silicon and demonstrates their effectiveness. They further showed that the body bias voltages that tune the P to N ratio for optimal noise margin also minimizes energy. Hence, one tuning can be used to both increase robustness of the design as well as to reduce its energy consumption. They found that skewing P and N body biases in increments of 5mV to match strengths enabled them to improve the minimum functional voltage by 24%. For global performance they improved the variability for several target voltages, as seen in Figure 2.9. This directly demonstrates the effectiveness of ABB in dealing with variation, especially in low voltage

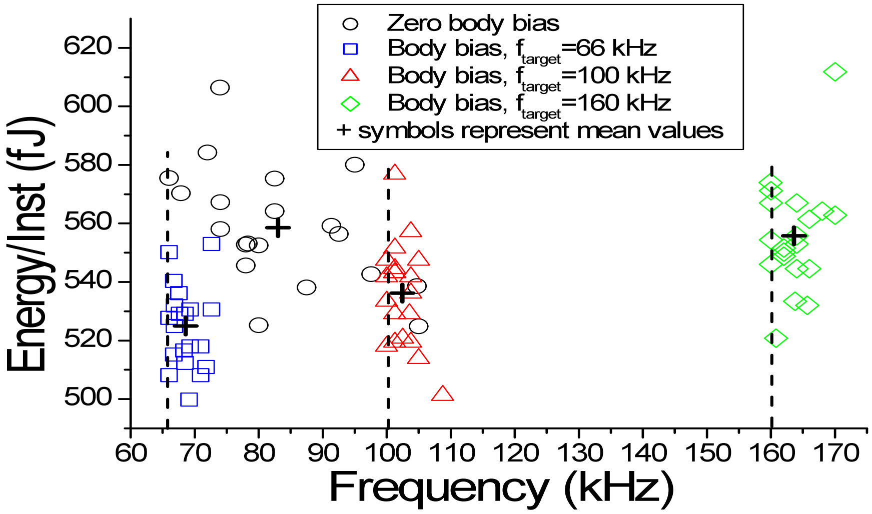


Figure 2.9: Body biasing techniques for three target frequencies.

designs, and is a technique that can be directly leveraged in NTC systems to cope with these same issues.

2.5.3 Addressing Functional Failure

The variations discussed in Section 2.4.3 not only impact design performance but also design functionality. In NTC the dramatically increased sensitivity to process, temperature and voltage variations lead to a precipitous rise in functional failure (the likelihood that a data bit will be flipped), particularly due to drive strength mismatch. In this section, architectural and circuit-level techniques for addressing SRAM robustness in NTC operation are discussed.

2.5.3.1 Alternative SRAM Cells

As mentioned previously, SRAM cells require special attention when considering cache optimization for the NTC design space. Even though it is clear that SRAM will generally exhibit a higher V_{min} than logic, it will still operate at supply level significant lower than the nominal case. This in turn reduces cell stability and heightens sensitivity to V_{th} variation, which is generally high in SRAM devices to begin with due to the particularly aggressive device sizing necessary for high density. This problem is fundamental to the standard 6T design, which is based on a carefully balancing act of relative device sizing to optimize read/write contention. The only solution to keeping SRAM viable for NTC applications is to trade-off area for improved low voltage performance. The question then becomes how best to do this resize and optimize the 6T devices, or abandon the 6T structure completely?

One example in which the basic 6T structure was maintained can be seen in the work by Zhai et al. [99]. The cell itself is optimized for single-ended read stability, and a supply modulation technique is used on a per column basis to improve writeability. Thus, the read and write operations are effectively decoupled by relying on extra complexity in the periphery of the core array. The result is a cell that is functional below 200 mV and that achieves relatively high energy efficiency.

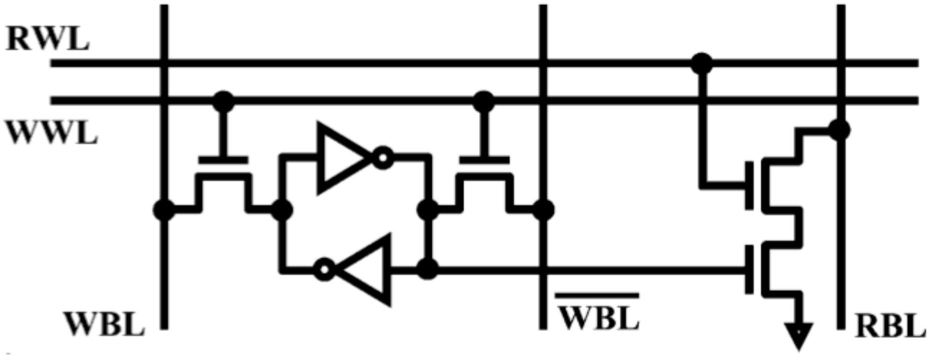


Figure 2.10: Alternative 8T SRAM cell, decoupling the read and write [18].

There have also been a number of alternative SRAM cells proposed that are particularly well suited for ultra-low voltage operation. For example, Chang et al. [18] developed an 8T design, in Figure 2.10, with the premise of decoupling the read and write operations of the 6T cell by adding an isolated read-out buffer, as shown in Figure 2.5. This effectively allows the designer to optimize the write operation sizing independently of the output buffer and without relying on supply modulation or wordline boosting. This greatly enhances cell stability, but incurs area overhead in the core array to accommodate the extra devices and irregular layout.

Similarly, Calhoun and Chandrakasan [16] developed a 10-transistor (10T) SRAM cell also based on decoupling read and write sizing and operation. The 10T cell is pictured in Figure 2.11 and offers even better low voltage operation due to the stacking of devices in the read port, though it suffers a commensurate area penalty. Such alternative SRAM cell designs successfully cope with the difficulty of maintaining proper operation at high yield constraints in the sub-threshold operating region, and offer promising characteristics for realizing reliable cache in NTC-based systems.

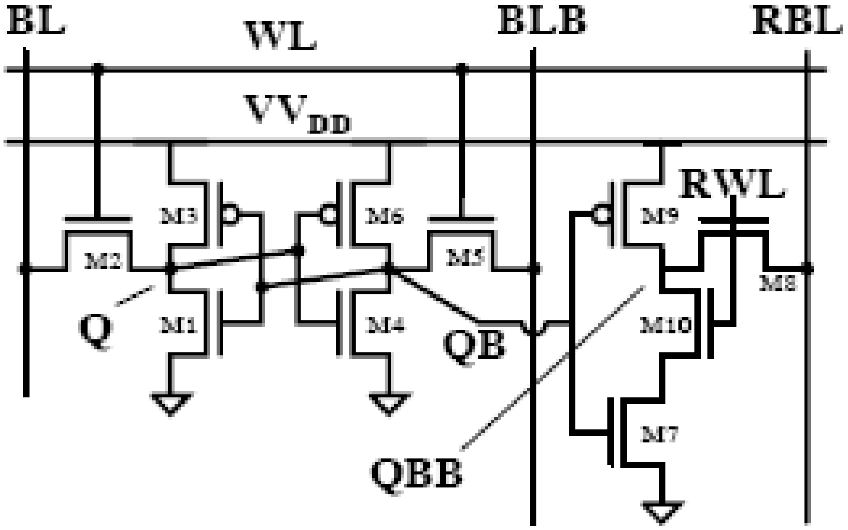


Figure 2.11: Alternative 10T SRAM cell [16].

2.5.3.2 SRAM Robustness Analysis Techniques

The importance of robustness for NTC systems means that credible analyses techniques must be available. This is particularly true for the case of large level-2 and level-3 (L2 and L3) caches where low bitcell failure rates are required to achieve high yield. Inaccurate estimates of robustness in such cases would lead to wasted die space, in the case of oversized cells, or large portions of unusable memory, when they are undersized.

Chen et al. [20] have developed a technique to determine proper cell sizing to maintain the same SRAM cell robustness at NTC voltages as traditional cells have at nominal. The technique they developed uses importance sampling as a means to determine the cell device sizes needed for a given robustness to variation. Using importance sampling for yield estimation, Chen compared a 6T, single-ended 6T with power rail drooping and an 8T bitcell at an iso-robustness and iso-delay condition. This condition requires that both cells be designed to tolerate the same level of process variation before functional failure while operating with the same nominal delay. The results for a 20 cycle latency in terms of SRAM bitcell area and energy consumption are presented in Figure 2.12 and 2.13. At higher V$_{dd}$, the differential 6T bitcell has the smallest area. The 8T bitcell becomes smaller below a V$_{dd}$ of 450mV and a twenty-cycle latency. As V$_{dd}$ approaches V$_{th}$, all bitcells must be sized greatly to maintain robustness unless delay is relaxed, making large arrays impractical. The differential 6T bitcell has the lowest dynamic energy consumption at most supply voltages. The single-ended 6T bitcell has the lowest leakage per cycle. V$_{min}$ increases

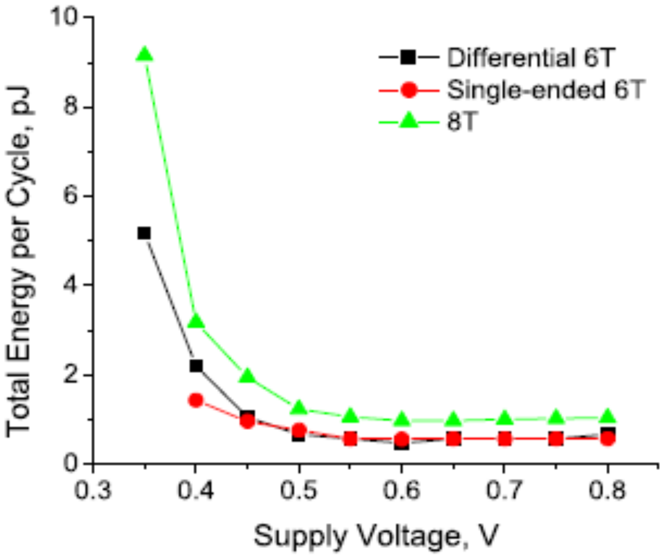


Figure 2.12: Energy of SRAM topologies for 20-cycle L2 cache across voltages.

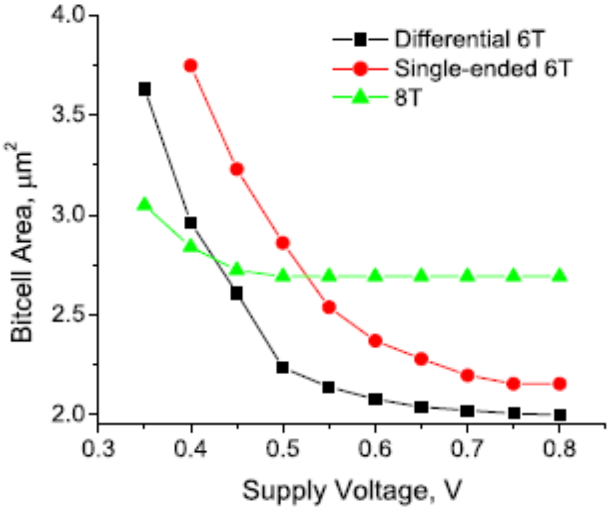


Figure 2.13: Size of SRAM topologies for 20-cycle L2 cache across voltages with iso-robustness.

with cache size and bank size, and decreases with associativity, activity factor and cache line length. For common cache configurations, V$_{min}$ may be near or even above V$_{th}$ and is significantly higher than reported in previously literature. By comparing SRAM bitcells at an iso-robustness and iso-delay condition, the best SRAM architecture and sizing for a design can be quickly and accurately chosen. This work will be valuable in assessing the viability of new SRAM designs, particularly in the NTC domain.

2.6 Conclusions

NTC operation promises to be an interesting solution to maintaing the continued growth in CMOS technology that has been the norm for the past 20 years. However there are three key bottlenecks to the widespread adoption of NTC: performance loss, variation, and functional failure. In this chapter NTC operation was defined, the barriers to adoption explained, and techniques to overcome them were explored. The rest of this dissertation focuses on overcoming the performance loss bottleneck for a wide range of architectures—single-core, multi-core, DSP, and many-core—to target an even wider range of applications—from tiny sensors to warehouse scale servers.

CHAPTER 3

Single Core Architectures

For modern sensor/embedded processors battery life and area are important design concerns. The use of NTC operation can help reduce power, however stringent area constraints and/or inherently serial applications prevent the use of parallel architectures to overcome performance bottlenecks. This chapter explores other uses of NTC operation for single-core architectures, particularly focusing on the memory system and area overheads.

Current commercial memory technologies have been limited in the degree of supply voltage scaling that can be performed if they are to meet yield and reliability constraints. This has limited designers from exploring the near threshold operating regions for embedded processors. Summarizing prior work this chapter shows how proper sizing of memory cells can guarantee that the memory cell reliability in the near threshold supply voltage region matches that of a standard memory cell. However, this robustness comes with a significant area cost. This chapter shows how to employ these cells to build cache architectures that greatly reduce energy consumption. The chapter then proposes an embedded processor based on these new cache architectures that operates in a low power mode, with minimal impact on full performance runtime. The proposed cache uses near threshold tolerant cache ways to reduce access energy combined with traditional cache ways to maintain performance. The access policy of the cache ways is then dynamically reconfigured to obtain energy efficient performance while minimally impacting the high performance mode runtime. Using near threshold cache architectures results show an energy reduction of 53% over a traditional filter cache. For the MIBench embedded benchmarks results show on average an 86% (7.3x) reduction in energy while in low power (10MHz) mode with only an average 2% increase in runtime in high performance (400MHz) mode. And for SpecInt applications analysis shows a 77% (4.4x) reduction in energy in low power mode with only an average 4.8% increase in runtime for high performance mode. In addition the chapter shows that these trends hold from 130nm to 45nm technology nodes. The work done in this chapter was conducted with the collaboration of my colleague Greg Chen. This work was published as an article at the 41st IEEE/ACM International Symposium on Microarchitecture (MICRO) [27].

3.1 Introduction

It was shown in Section 2.5.3 that proper sizing of memory cells can guarantee that the memory cell reliability in the near threshold supply voltage region can match that of a standard memory cell. However, this robustness comes with a significant area cost because the standard SRAM cell must be enlarged This Chapter shows how to employ these cells to build cache architectures that greatly reduce energy consumption. An embedded processor based on these new cache architectures that operates in a low power mode, with minimal impact on full performance runtime is presented.

Using the work done by Chen [19] as a starting point, memory hierarchies in which some or all of the memory cells in the system are designed to scale their supply voltages into the near threshold operating region are examined. This will allow for increased energy savings in low power mode. Then idea of a filter cache [52] is revisited in the context of near threshold operation and supply voltage scaling. It will be shown that a near threshold filter cache reduces the energy of a traditional filter cache by 36%. However, the filter cache has the potential to impact the overall performance of the system in the high performance mode by increasing the runtime by nearly 20%.

To overcome this a new cache architecture, the Reconfigurable Energy-efficient Near Threshold (RENT) cache, that reduces the runtime overhead in high performance mode while maintaining considerable energy savings in low power mode is proposed. The cache is composed of one near threshold tolerant cache way and several standard SRAM cache ways. The near threshold tolerant cache way is accessed in the first cycle and only on a miss are the other cache ways accessed. In this manner the near threshold tolerant cache way acts as a shield to the other cache ways. If the miss rate in the near threshold tolerant cache way exceeds a threshold, then the cache is dynamically reconfigured to access all the cache ways in parallel, providing a single cycle access to all of the cache space. This will help to almost eliminate any runtime increase in high performance mode. By using this technique combined with some advanced access policies the RENT cache shows a 53% reduction in energy over a traditional filter cache. This results in a system that provides a 86% (7.3x) energy savings in low power mode with only an average 2% increase in runtime while in high performance mode.

The rest of the Chapter is organized as follows. First, Section 3.2 will summarize the essential points necessary to characterize near threshold tolerant memories. Section 3.3

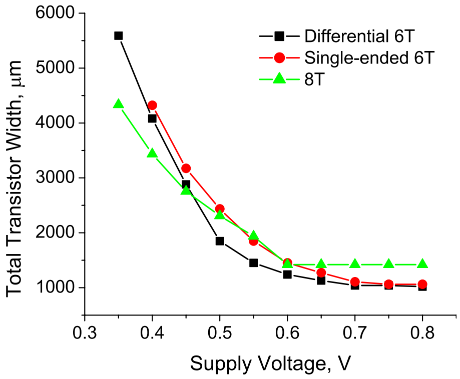


Figure 3.1: SRAM Cell area (measured in total transistor width) at different supply voltages to maintain ISO-Robustness.

will present the proposed architectural solutions that employ these near threshold tolerant techniques. Section 3.4 will describe the methodology, and Section 3.5 will present the results. Then the chapter is finished with a brief discussion of related work in Section 3.6, and provide concluding remarks in Section 3.7.

3.2 Low Voltage Tolerant SRAM cell design

Supply voltage scaling has been shown to be an effective way to handle lowering the energy consumption of logic circuits. There have been several cores built that show a quadratic savings in dynamic energy consumption with a delay degradation [102, 94]. These designs employ CMOS circuitry, which is quite resilient to supply voltage scaling. SRAM memory cells, on the other hand, do not voltage scale into the near and subthreshold regions as easily. Several ultra-low energy designs have been implemented [99, 16, 91, 50] but require increased cell size. This increased cell size reduces the energy gain from supply voltage scaling, because the larger cells consume more power than the smaller cells. In addition, when the die size is held constant, the increased cell size reduces the total overall size of the memory structure. This increased cell size is done in order to maintain reliability levels, but until recently there was no formal method to guarantee reliable operation in near threshold memory structures.

By modeling process variation, especially random dopant fluctuations (RDF), and employing a modified Monte Carlo analysis, Chen et al. showed how to calculate the robustness of an SRAM cell design at near and subthreshold supply voltages [19]. Because SRAM cells occur in large arrays, failure rates must be kept very low to ensure a reliable cache. For example, for 99% of 8kB caches to be reliable it would require a bitcell failure rate of 1.57x10^{-7}.

Among other things, the analysis calculates the necessary cell size to maintain the same failure rate as a standard SRAM cell, termed iso-robustness. In determining the iso-robust point the transistors are scaled individually to avoid read, write, and read-upset failures. The technology node for our study is 130nm and standard SRAM cells operate with acceptable reliably down to 800mV, later results for 90nm, 65nm, and 45nm are presented. For our robustness analysis cells that operate at supply voltages below 800mV are required to be sized to match the failure rate of a standard SRAM cell operating at 800mV. A plot of several different SRAM topologies is shown in Figure 3.1. The graph presents the necessary cell-area in terms of total transistor width to maintain iso-robustness for a given supply voltage. The three lines in the plot represent a standard 6T SRAM cell, a single-ended 6T SRAM cell [99] and a 8T SRAM cell [91]. It is possible to not only pick the proper supply voltage to meet the delay requirements, but to also pick the SRAM topology that provides the most area efficient solution.

This analysis has several ramifications for the design of a low-power embedded microprocessor. The SRAM in these systems must be designed for reliable operation. This analysis can is used to determine what size to make an SRAM cell in order to have it operate reliably down to a given supply voltage. The increased cell size will affect performance of a cache in two main ways. First at full supply voltage the SRAM cell will consume more energy and be slightly slower due to the increased size. Body-biasing techniques can be employed to regain speed, but at the cost of additional energy consumption. Secondly, there will be less total cache in the same die space. Given a design frequency to target, it is critical that proper analysis is performed to avoid oversizing the SRAM cell and reduce the performance of the chip in non-power saving mode. Conversely it is important not to undersize the cell and increase reliability problems in power savings mode.

3.3 Energy Efficient Cache Architectures

The ability to voltage and frequency scale cores has provided an order of magnitude savings in energy [101, 94, 102]. However, supply voltage scaling of the memory system has been limited by the tolerance of the SRAM cell. If the SRAM cells discussed in Section 3.2 are used, in combination with a voltage and frequency scaled core, even more energy savings could be possible. The goal of the design is to create a chip that can operate in two distinct modes. The first is a high performance mode where the system is fully powered and the core runs at a high frequency. The second mode is a power savings mode. In this mode the chip is frequency scaled to a slower performance target, and the supply

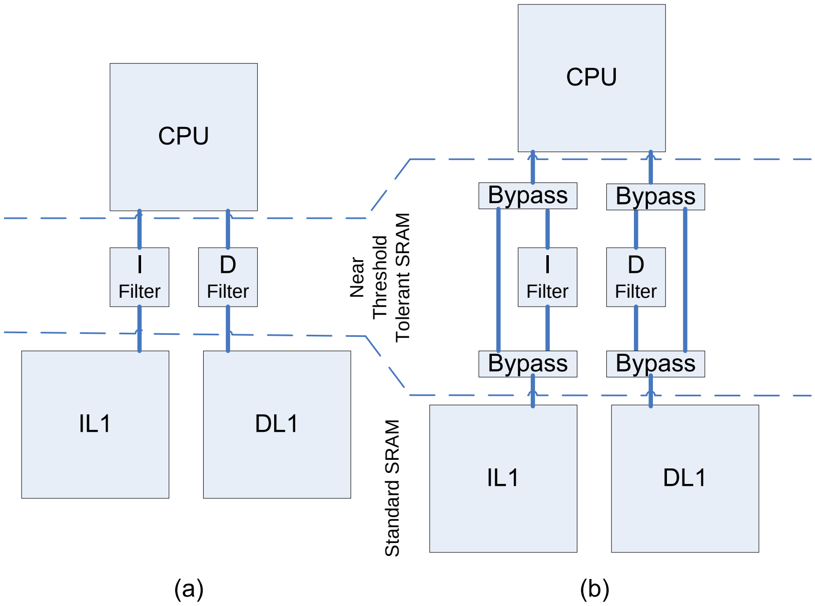


Figure 3.2: Near threshold filter cache architectures (a) with and (b) without bypass networks.

voltage can then be dropped to reduce energy consumption.

Due to the differing activity factors for logic and memory, the core and caches need to be operated in different voltage domains [26, 100]. In addition, providing two memory voltage domains will allow more complex architectures that involve both near threshold tolerant SRAM, and standard SRAM. The near threshold tolerant SRAM will be used to create energy efficiency, while the standard SRAM will help to provide larger capacity on chip since they will not need to have their size increased to maintain reliability. This requires the addition of at most two additional supply voltage domains to the chip and associated voltage level converters. A diagram of the regions in which a chip might be partitioned is shown in Figure 3.2(a and b). In the figure, the core is operated in one supply voltage domain, the filter cache in a separate near threshold tolerant SRAM cell domain, and the L1 in a third domain. Level converters are used to cross the dotted lines separating the voltage domains.

These changes to the chip and SRAM cell design provide opportunities to explore more energy efficient cache hierarchies. Simply using near threshold tolerant cells in place of current cells will allow the caches in a system to scale to much lower supply voltages, creating a dynamic energy savings at low frequencies. However the increase in the cell size will drastically reduce the total amount of cache available when the die size is held constant. This will negatively impact performance by increasing the number of off-chip accesses, which require large amounts of energy and have a long latency. This increase in latency will in turn slow the completion of the program, and prolong the leakage energy. At high frequencies, when the cache is operating at full supply voltage the energy per access of the cache will also be increased due to the larger cell sizes required to meet the iso-failure conditions set forth in Section 3.2 and will need to be operated at a higher supply voltage or body bias in order to operate at the same speed. In the following sub-sections three techniques will be discussed for better use of the near threshold tolerant SRAM cells to reduce energy consumption.

3.3.1 Near Threshold Filter Cache

Filter caches, a technique proposed in [52, 86], is a method to reduce cache energy. The idea behind filter caches is to place a small, low energy per access, cache in between the processor and the L1 cache. This cache then filters access to the larger, more energy hungry, L1 cache. Employing near threshold tolerant SRAM cells to design a filter cache would further reduce the energy even more. A diagram of what a near threshold filter cache architecture would look like is shown in Figure 3.2(a).

The main drawback to using filter caches is that if the memory access pattern creates a high miss rate in the filter cache, then the overall system energy goes back up due to the additional miss that causes an access to the L1 cache. The system performance is also degraded because all cache accesses that miss in the filter cache but hit in the L1 cache take two cycles instead of one. To overcome these drawbacks a simple bypass network could be employed to bypass the filter cache when the miss rate in the filter cache is too high. Figure 3.2(b) provides an illustration of what a bypass filter cache would look like. This bypass filter cache also presents some drawbacks. First, in order to save energy the filter cache needs to operate with a writeback policy instead of a writethrough. This means when the filter cache is bypassed, the entire filter cache must be flushed and written back. In addition, it is difficult to track when the bypass network should be turned off, since it is difficult to approximate performance of the filter cache. These two complications prevent more aggressive dynamic adjustment of the use of the bypass network.

3.3.2 RENT Caches

Although naively employing a near threshold filter cache provides great energy savings, there are new architectures that can provide even further savings. An additional drawback of the bypass filter cache design is that when the filter cache is being bypassed, a portion of the cache space is not being used. That means there is effectively less total cache space on

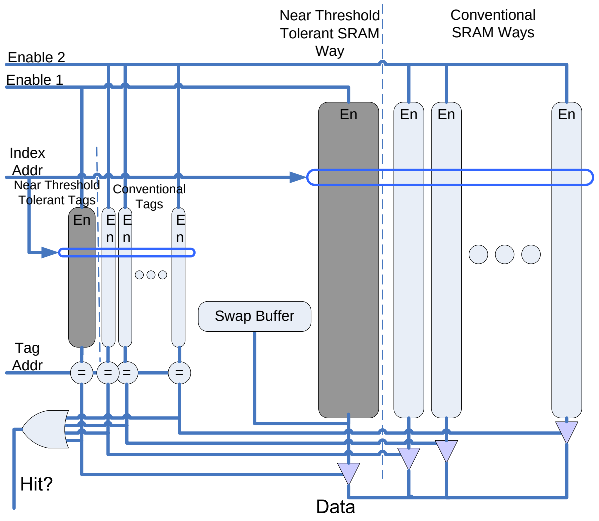
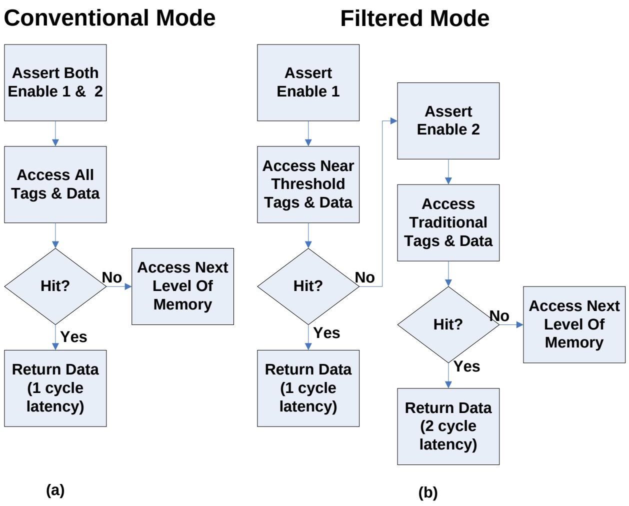


Figure 3.3: Cache architecture of the RENT cache.

the chip, leading to a larger number of off-chip accesses. These off-chip accesses require large amounts of energy and have a long latency. These longer latencies lead to an increase in runtime for the program, prolonging leakage energy and degrading performance. In order to minimize the amount of cache space lost and the performance degradation that occurs from using near threshold tolerant SRAM cells, while still maintaining the energy efficiency of the filter cache, a new cache architecture is proposed based on the work done in [42, 73, 2].

The proposed architecture is called the Reconfigurable Energy-efficient Near Threshold (RENT) cache. A diagram of the cache structure is presented in Figure 3.3. The basic premise behind the cache is as follows: There is one way of the cache and tags that is implemented with near threshold tolerant SRAM cells. The other ways of the cache and tags are implemented with standard SRAM cells. The cache will be operated in two distinct modes. We call these modes conventional and filtered. In the conventional mode the cache is accessed in the regular manner. That is, the index portion of the address reads out all the cache tags and data from each way in parallel. The tags are then matched with the incoming address and the appropriate way of the cache is enabled to the output data bus. The new mode of operation, filtered mode, is designed to act like a filter cache. When filtered mode is in use only the first way, the near threshold tolerant way, of the cache and tags are accessed on the first cycle, via the enable 1 signal. If there is a hit, the cache will return the data and energy will be saved by only accessing this one way of the cache. If there is a miss on this first cycle access, then the data from the near threshold tolerant way is stored in a swap buffer. The enable 2 signal is then used and the rest of the cache is



Conventional Mode
Filtered Mode
Assert Both Enable 1 & 2
Access All Tags & Data
Hit?
No
Access Next Level Of Memory
Yes
Return Data (1 cycle latency)
Assert Enable 1
Access Near Threshold Tags & Data
Hit?
No
Yes
Return Data (1 cycle latency)
Assert Enable 2
Access Traditional Tags & Data
Hit?
No
Access Next Level Of Memory
Yes
Return Data (2 cycle latency)
(a)
(b)

Figure 3.4: RENT cache access policy flow charts for (a) conventional mode and (b) filtered mode.

checked for a hit. When a hit is found it is both enabled onto the data bus and written to the near threshold tolerant way. On a subsequent cycle the data stored in the swap buffer is then written into the cache way where the hit occurred. This action essentially swaps the value in the near threshold tolerant way and the way in which there was a hit. This ensures that the most recently used (MRU) block is in the near threshold tolerant cache way. This swapping action is similar to the writeback buffer in the filter cache, and the design is no more complicated than creating an exclusive filter cache. Figure 3.4 shows a flowchart of the two access modes. This addresses two of the drawbacks of the bypass filter cache. First, it utilises the low voltage tolerant SRAM cells, even when in high performance mode leading to more on-chip cache. Second, it prevents the flushing action required when switching modes, since the data in the cache ways is still available in high performance mode, unlike the filter cache when it is bypassed. It does however present to tradeoffs. First, the near-threshold tolerant cache ways will consume more energy at full voltage because they are larger with more capacitance and need to be operated with body-biasing to achieve the same speed. Second, the swap mechanism requires an additional cycle to complete some cache accesses, prolonging execution time in filtered mode.

Changing from conventional mode to filtered mode could be identified explicitly by the programmer, but greater savings could be achieved if the cache was allowed to choose operating modes dynamically based on performance. This would allow the cache to adapt to different program phases. It is relatively easy to decide when to switch from filtered mode to conventional mode. A counter can be used to monitor the hit rate of the near

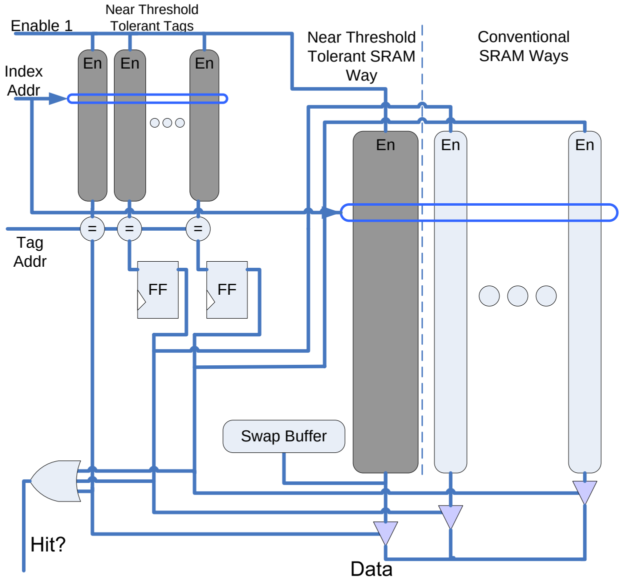


Figure 3.5: Architecture of the alternative RENT cache.

threshold tolerant way, when the hit rate drops below a predefined threshold the cache changes modes. It is harder to decide when to transition back to the filtered mode of operation. In order to determine when to switch, we make use of some information from the replacement policy. If the cache is using a pseudo least recently used (LRU) replacement policy than the MRU block can be easily determined. This is due to the fact that the pseudo-LRU policy always identifies the MRU block in the cache tags. If the system tracks the number of times the cache hits on an MRU block and compare it to the number of times the cache hits on any non-MRU block the system can calculate what the miss rate would have been for the near threshold tolerant cache way. This follows because the MRU block would be in the near threshold tolerant cache way in filtered mode. When this hit rate exceeds some threshold, the system can switch back to filtered operation. In both cases, after a switch has occurred a suitable number of accesses must be completed to allow the cache to reach a steady state before changing modes again. These methods require a very small overhead of two counters to track accesses and MRU/filter hits. This addresses the final drawback of the bypass filter cache, by allowing more aggressive mode transitions due to the ability to accurately track filter cache performance in both modes of operation.

3.3.3 Alternative RENT Caches

Further possible improvements can be made to the RENT cache to reduce energy consumption. Notice that in filtered mode the system access the additional ways of the cache in the second cycle. If instead the system had used the first cycle to check not only the first set of tags, but all the other cache tags in the set as well, the system could know by the second cycle the way in which the data we seek resides. The modified architecture is presented in Figure 3.5. In this case the system makes all the tags using near threshold tolerant SRAM cells and accesses them all on the first cycle. In parallel with the tag check the system accesses the near threshold tolerant way of the data cache. If there is a hit in the near threshold tolerant way the data can be provided in a single cycle, as before. If there is a miss, the tag check will provide which conventional way of the cache the system should access on the second cycle. This reduces the energy consumed because on the second cycle only one way of the conventional cache is accessed. The flip-flops shown in Figure 3.5 are used to delay the enable of the conventional ways until the next cycle. There is also the added benefit of knowing if the cache will miss after the first cycle. In this case an access to the next level of memory can be initiated one cycle earlier, reducing the off-chip access time by one cycle. Of course the system is trading off the energy reduction of accessing all the additional ways of the cache with the increase in energy for the tag look up. If the conventional ways of the cache are rarely accessed then the system may consume more energy with this design.

3.4 Methodology

A system with two different operating modes was created. The first was a 400 MHz full power mode. The frequency was determined by the ARM9 CPU model used for energy analysis, the details of which were obtained from consultation with the chip designers. The second mode, a low power mode, was chosen to operate at a 10 MHz clock frequency. This operating frequency was picked to provide a significant energy savings while still being fast enough to handle a wide range of simpler applications. A summary of the resultant design parameters for the system is in Table 3.1. For this study a system with a split instruction and data cache is used, but to keep the analysis space small the same sizes and types of caches for both the instruction and data memory hierarchies is used. For all comparisons the die size was held constant.

3.4.1 Simulation Environment

For simulation a modified version of the M5 simulator [9] is used. The simulator was modified to incorporate the dynamic and leakage energy modeling necessary to determine the energy of the overall system.

Simulation Parameters

			Traditional		Near Threshold	
	Baseline		Filter Cache		Filter Cache	
	10Mhz	400M	10MHz	400M	10MHz	400M
Core Voltage	450mV	1.2V	450mV	1.2V	450mV	1.2V
Filter Cache Voltage	N/A		800mV	1.2V*	500mV	1.2V*
Filter Cache Size	N/A		1 kB		512 B	
L1 Cache Voltage	800mV	1.1V	800mV	1.1V	800mV	1.1V
L1 Cache Size	8 kB		7 kB		7 kB	
L1 Cache Ways	8		7		7	

Table 3.1: Simulated System Parameters. *Body-Biasing used to meet timing constraints.

3.4.2 Benchmarks

For the simulation the MiBench benchmarks [32] were used. These benchmarks are small embedded system benchmarks and represent a range of applications, from automotive to encryption. A subset of benchmarks that compiled to the target architecture were run to completion for all test cases using the reduced input sets. Further analysis was done using the complete SpecInt 2000 benchmarks [36]. This analysis shows similar results as the MiBench studies and is briefly presented at the end of the results section.

3.4.3 Energy Models

We create accurate energy consumption models for the system using SPICE analysis coupled with iso-robustness analysis. The analysis accounts for CPU, cache, bus, level converter, and off-chip access energy. All data presented is using a commercial 130nm process, some additional analysis was done to verify that the trends still hold in both 90nm, 65nm, and 45nm. The results of that comparison are presented in Section 3.5.6.

3.4.3.1 Cache Model

We use the iso-robustness analysis method discussed in Section 2, assuming that standard SRAM cells can operate reliably down to 800mV, to determine the appropriate cell size to meet the delay of the core and iso-robustness reliability constraint for any near threshold SRAM cells. SPICE modeling is used on the SRAM cell to determine the dynamic and leakage energy consumption of the cell at different operating points. The energy in the data and tag arrays of the cache as well as any decoders associated with the cache are accounted. However the cache controller energy is not modeled. This will result in energy figures being slightly lower than that of a real system. The analysis has simulated and accounted for the energy and additional cycle latency of the swap buffer. The use of body-bias techniques is assumed in our experiments to reduce the cache leakage energy, or to meet timing constraints.

3.4.3.2 CPU Model

For the CPU model, the analysis bases the core energy consumption numbers from a cacheless 400MHz ARM9 core. SPICE simulation of some representative smaller CMOS circuitry is used in order to determine the impact of voltage and frequency scaling on the core for both dynamic and leakage energy.

3.4.3.3 Off-Chip Energy

The off-chip latency was derived for a memory system composed of SRAM for data, and ROM for instructions. This is a typical setup for an embedded system, and the latency was 20 cycles at 400 MHz. The energy consumed in the package and a reasonable amount of off-chip routing is accounted for in all the measurements. However, the energy of the off-chip memory itself is not accounted for in the simulation as this could vary widely depending on the application. A designer could use our figures as part of their total power/performance calculations.

3.5 Results

3.5.1 Filter Cache

The first analysis to be performed was on the simple near threshold filter cache. The first step was to determine the L1 size for the baseline system without a filter cache that provided the lowest energy solution. A sweep of L1 sizes and associativities yielded an optimal size of an 8kB, 8-way cache for the MiBench benchmarks. Then, while holding the die size constant the lowest energy system with a near threshold filter cache size was determined. The analysis was done keeping in mind that the size of the near threshold filter cache SRAM cells would be larger in size than the standard ones used in the L1 cache. Across the benchmarks the optimal size was a filter cache of either 512 B or 1 kB. For our studies a 512 B filter cache and a 7kB, 7-way L1 cache are chosen. For comparison a filter cache designed with standard SRAM cells, which do not support voltage scaling below 800mV, is also evaluated. This configuration is termed as a traditional filter cache. For this case a configuration of a 1 kB filter cache and a 7kB, 7-way L1 cache is chosen. A larger filter cache is possible in this case because the SRAM cells do not have to be sized to operate at lower supply voltages. A summary of the baseline and filter cache systems can be found in Table 3.1. For the initial results the bypass method described in Section 3.3.1 is not used, the impact of using a bypass on the filter cache will be evaluated in Section 3.5.1.2.

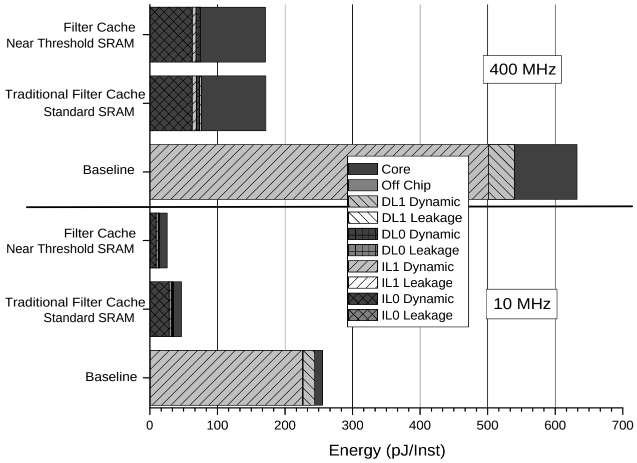


Figure 3.6: Filter cache energy breakdown for BitCount benchmark.

3.5.1.1 Without Bypass

The results of the analysis are presented in Figure 3.6. In the figure there are 6 bars, the first 3 bars are for the system at 400 MHz and the second 3 are for the system at 10 MHz. The bars present the total energy of the system divided by the number of instructions completed. To illustrate the analysis the BitCount benchmark is used. The first thing to note is that by simply using voltage scaling on the baseline system it can reduce the energy consumption to complete this benchmark by 60% (3rd bar vs. 6th bar). It can also be seen that, because the system is able to more aggressively voltage scale the core, the cache quickly becomes the dominant energy source at low frequencies. From the figure it can be seen that the IL1 consumes most of the energy in the memory system for BitCount. Using a filter cache dramatically reduces this IL1 dynamic energy by shielding the IL1 with a small filter cache. This results in a 73% (2nd and 3rd bars) reduction in energy at 400MHz and a 82% reduction in energy at 10 MHz (5th and 6th bars) over their equivalent speed baseline counterparts.

Figure 3.7 presents a zoomed in portion of Figure 3.6 to help better see the shorter bars. The addition of near threshold supply voltage scaling capabilities on the filter cache does two things. First, it reduces the energy consumption at 10 MHz a further 45% over

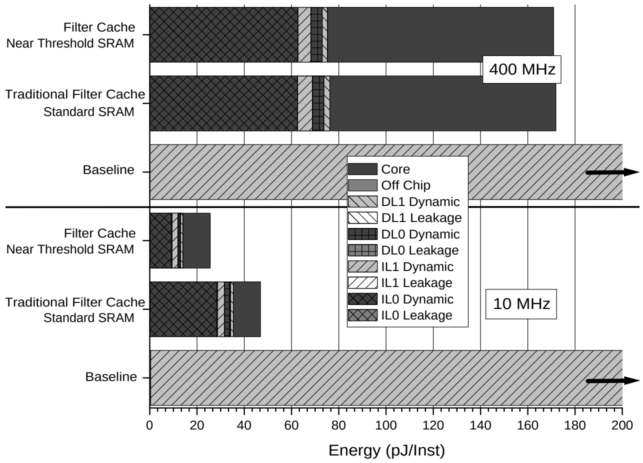


Figure 3.7: Filter cache energy for BitCount benchmark (Zoomed In).

the traditional filter cache. Second, due to the larger cell sizes, the energy at 400 MHz is increased by around 1%. This increase is mitigated by the fact that although the near threshold filter cache has larger cell sizes, there are only half as many cells as the traditional filter cache (see Table 3.1).

Figure 3.8 shows additional benchmarks from the MiBench suite and the performance of the near threshold filter cache. Figure 3.9 shows how the near threshold filter cache's energy consumption compares to the traditional filter cache and the baseline at 10 MHz. On average the near threshold voltage scaled filter cache shows an 84% reduction in energy over the baseline at 10 MHz, and a 36% reduction over a traditional filter cache.

3.5.1.2 With Bypass

The filter cache does present some drawbacks. As mentioned in Section 3.3.1 the existence of the filter cache can degrade performance when the miss rate in the filter cache is high. Figure 3.10 presents the increase in runtime that occurs over the baseline system when a filter cache is used. Note that for this analysis a filter cache size of 1kB and a 6kB, 6-way L1 is used to compare to equal die size RENT caches in Section 3.5.2. On average a 17% increase in runtime occurs with a standard deviation of 9%. The worst case was a 29% increase in runtime for the Patricia benchmark. In order to reduce the runtime increase a bypass network can be employed which is enabled when the miss rate is high. The operation of the bypass filter cache is described in Section 3.3.1. Figure 3.10 also shows the resultant runtime when the bypass network is used. Notice that now the average runtime increase is only 3.3% with a 5.6% standard deviation. The benchmarks that still have

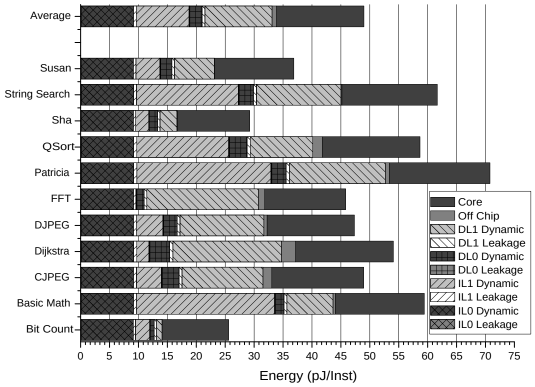


Figure 3.8: NT filter cache energy breakdowns for all Benchmarks at 10MHz.

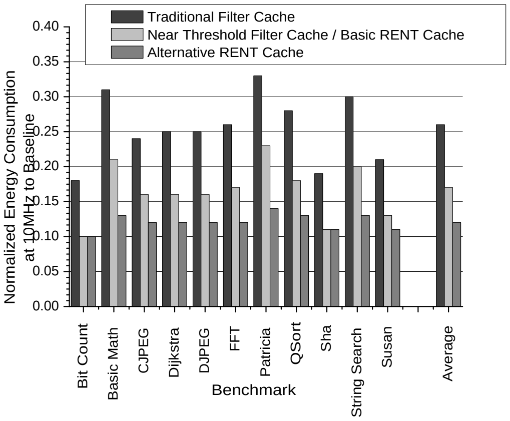


Figure 3.9: Cache energy comparisons for all benchmarks normalized to the baseline at 10MHz.

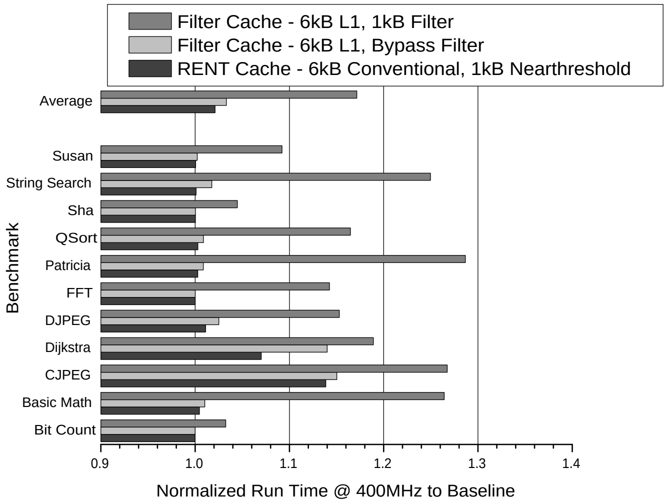


Figure 3.10: Normalized runtimes at full voltage to baseline machine.

a significant increase in runtime happen as a result of having a working set that is larger than the L1 cache. This leads to additional off-chip accesses which are long latency. Those benchmarks would benefit from being able to utilize the cache space that is being disabled by the bypass network.

3.5.2 RENT Cache

Using the RENT cache the system is able to increase the cache capacity on chip when running in the conventional mode. The simulation configuration is slightly different because all ways in the cache should be the same size. The resultant cache sizes are presented in Table 3.2, notice that the near threshold tolerant cache way requires about the same space as two conventional cache ways and the total cache size is decreased. This is better than the bypass cache because it allows the benchmark to utilize more of the cache on chip in conventional mode, 7kB. In addition the cache is also able to adapt dynamically to different phases of program behavior. The third bar in Figure 3.10 shows that with the use of the RENT cache the system can reduce the runtime overhead even further. The average increase is now just 2.1% on average with a 4.4% standard deviation. The use of the RENT cache has also kept the energy consumption at the same level in the low power mode as the bypass filter cache.

	Baseline	RENT Cache
# of Near Threshold Cache 1kB Ways	0	1
# of Conventional Cache 1kB Ways	8	6

Table 3.2: Simulated System Parameters for RENT Cache

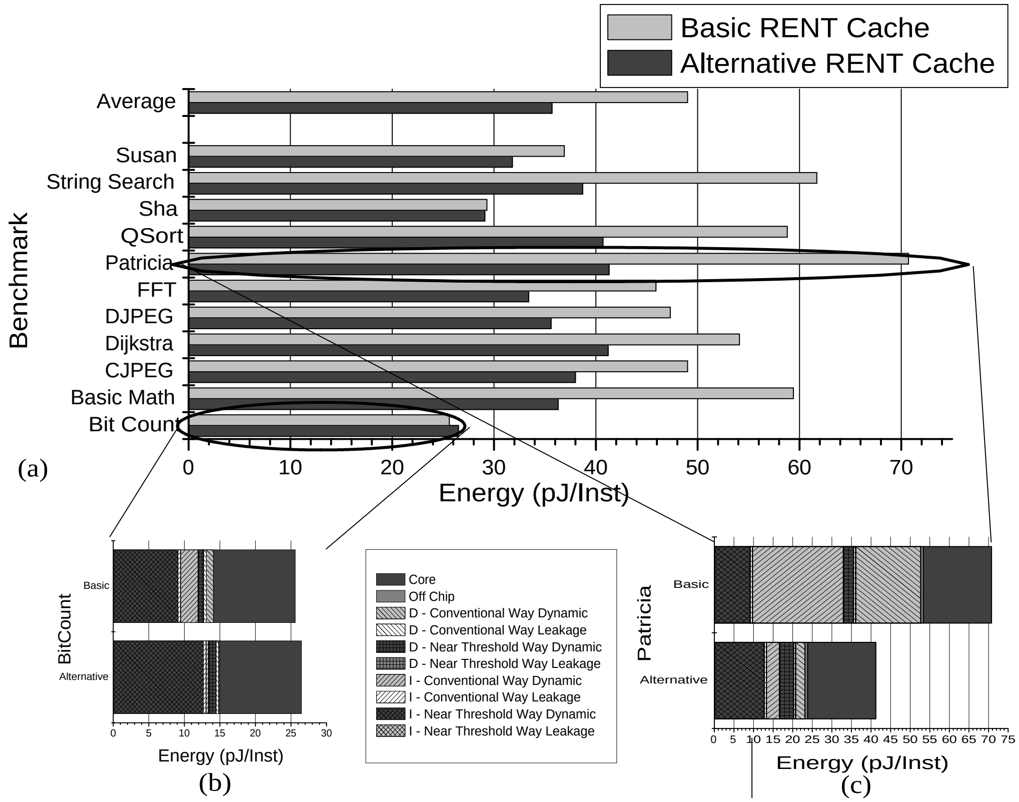


Figure 3.11: Comparison of energy breakdowns for RENT cache policies (a) for all benchmarks at 10MHz. With a breakdown of energy for (b) Bitcount and (c) Patricia

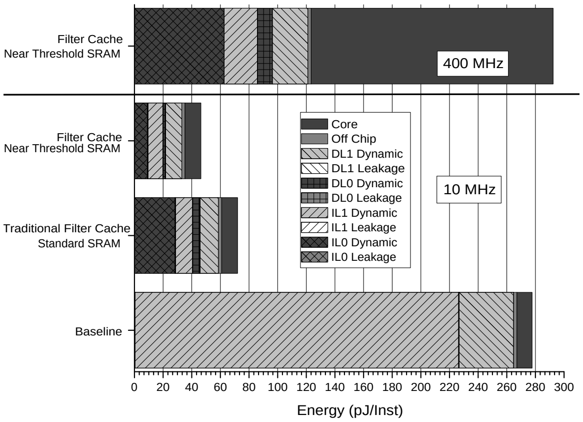


Figure 3.12: Energy breakdowns for the GZIP benchmark.

3.5.3 Alternative RENT Cache

Using the alternative RENT cache the energy savings can be even greater. The die size is slightly larger to accommodate the near threshold tags, but was not significant enough to justify the removal of another full data cache way. In Figure 3.11(a) the basic and alternative version of the RENT cache are compared across the MiBench benchmarks. For almost all the cases there is a decrease in total energy. A blown-up detailed view of the Patricia benchmark is presented in Figure 3.11(c). It clearly shows a reduction in the energy of the conventional ways of the cache. The dynamic energy to the near threshold portion of the cache is increased due to the additional tag checks being done. In this case the decrease in the conventional access energy outweighed the increase in the tags and the alternative method was better. Figure 3.11(b) presents a case in which the basic RENT cache outperforms the alternative version, the BitCount benchmark. In this case there was already very little conventional way dynamic energy and the increase in the filter way dynamic energy from the additional tag accesses was too large for the alternative method to be more energy efficient. Note that in all cases there is a slight reduction in runtime and therefore a small reduction in the leakage energy for all the alternative RENT cache methods. This reduction in runtime comes from being able to issue off-chip requests after the first cycle instead of the second cycle, thus reducing the total runtime by one cycle per off-chip access. It is hard to see this decrease in the graph, but the data confirms this expected result. From Figure 3.9, on average the alternative RENT cache policy provides an additional 27% average energy reduction over the basic RENT cache, 54% over the traditional filter cache, and an 88% over the baseline at 10MHz.

3.5.4 Spec2000 Analysis

Analysis was also done using the SPECINT benchmarks. Figure 3.12 shows the energy breakdown of the GZIP benchmark for the traditional and near threshold filter caches. Notice that even for these larger benchmarks the system is still seeing a 34% reduction in energy using the near threshold SRAM cells. Figure 3.13 and Figure 3.14 show the energy breakdowns of the 10MHz basic and alternative RENT caches. The system still show significant energy reductions using the alternative policy for the SpecInt benchmarks. Overall the system sees a 57% reduction on average using the alternative RENT cache over a traditional filter cache at 10MHz.

3.5.5 Power Savings Mode

The power savings figures are calculated by taking the energy consumption of the alternative rent cache in the low power mode, 10MHz, and comparing that to the high performance mode, 400MHz. For the MiBench benchmarks on average the system shows an 86% (7.3x) reduction in energy when operating in low power mode, with only a 2% increase in runtime in high performance mode. For the SpecInt benchmarks the system shows on average a 77% (4.4x) reduction in energy in low power mode with only an average 4.8% increase in runtime while in the high performance mode.

3.5.6 Technology Node Comparison

Additionally it was verified that the trends still hold in smaller technology nodes. Figure 3.15 shows the average MIBench performance at different technology nodes for both the RENT and alternative RENT cache. The bars are normalized to the performance of the baseline in the low power mode at the same technology node (i.e. 45nm RENT cache is normalized to 45nm Baseline). From the graphs you can see that the systems are still seeing a 88-90% reduction in cache energy for the basic RENT cache across all nodes. The breakdown of where the energy goes shifts to leakage at smaller nodes, but the same overall trends hold.

Each technology node has a different optimal operating voltage and frequency for both the low power mode and the high performance mode, for brevity they are not presented here.

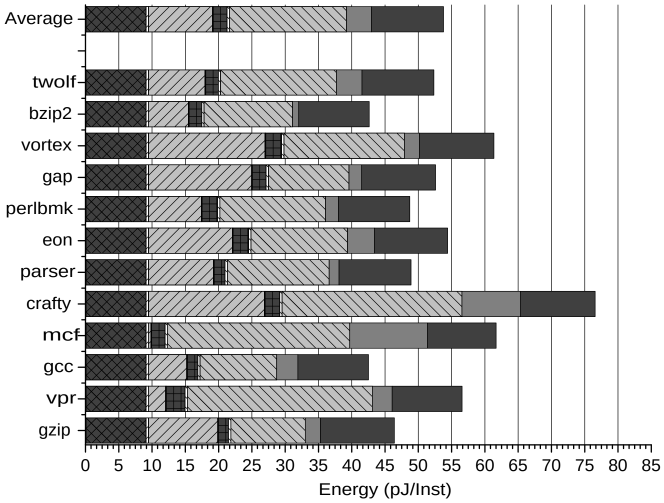


Figure 3.13: Energy breakdown for SpecInt 2000 benchmarks with the basic RENT cache at 10MHz.

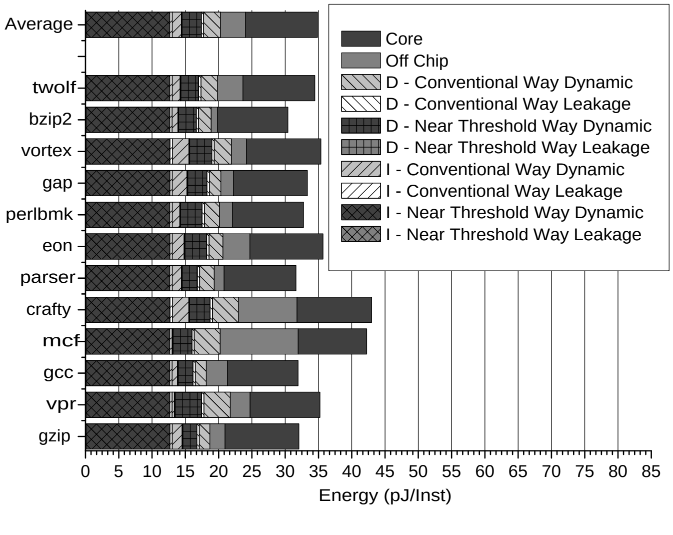


Figure 3.14: Alternative RENT.

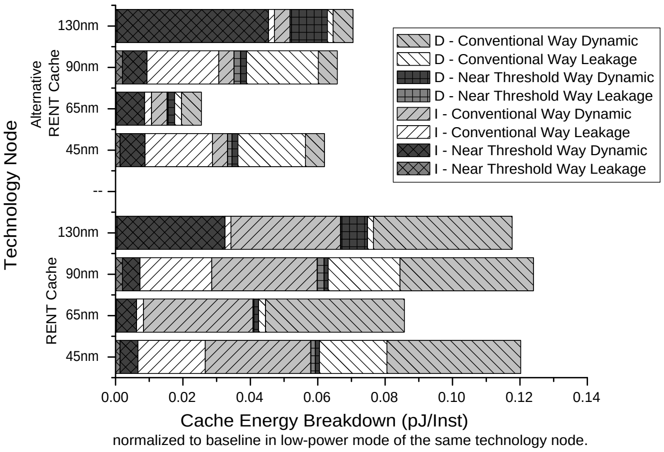


Figure 3.15: Normalized Cache Energy for Average MIBench Performance at Different Technology Nodes.

3.5.7 Additional Analysis

A number of additional experiments were run, but for brevity the results are not presented here. These experiments included sensitivity analysis for the off-chip latency and energy consumption numbers. Eventually, if the off-chip energy becomes very large, focusing on the core and cache is less important. Such systems will require a corresponding research effort to reduce main memory power, but off-chip energy savings is left to future research. Additional analysis also included threshold voltage techniques to decrease leakage energy as well as drowsy cache techniques [49]. The results presented in this chapter utilize the body bias techniques and without body-bias techniques, the configurations consumed anywhere from 5-12% more energy. The drowsy cache analysis also showed further energy savings of around 4% depending on the policy used. Analysis was also done for a 1MHz low power target, where the core was scaled to 300mV, but in that case the leakage energy of the caches began to dominate and only provided marginal savings over the 10MHz system. This was the subthreshold region mentioned in Section 3.1.

3.6 Related Work

There has been a significant amount of work done in the area of energy efficient cache architectures, particularly for embedded applications. Our work differs from the previous work in that we specifically use a near threshold tolerant SRAM design to explore cache architectures that can scale into the near threshold operating region. This results in significant energy savings of 36% over a traditional filter cache. The original work on filter caches was presented by Kin et al. [52] and Tang et al. [86] expanded on that work by creating a prediction network that would allow the instruction filter cache to be bypassed.

Further work in the reduction of cache energy was done by Albonesi [2]. That work proposes a cache that reconfigures itself to be optimal in size. This is achieved by enabling and disabling ways of the cache. This work is orthogonal to the work presented in this chapter, and could be used to further improve energy performance by disabling cache ways for applications with small working sets. Inoue et al. investigated the idea of way prediction [42] and Powell et al. [73] expanded on it. In their work only the predicted way is accessed on the first cycle, after which the other ways are accessed. This is similar to the RENT cache in that RENT is only accessing one way of the cache on the first cycle. This chapters work differs because it always starts with the near threshold tolerant way of the cache and swap the MRU block into that way of the cache. This helps to leverage the low energy of accessing this way of the cache. Zhang [103] proposes a multi-column cache to reduce energy, this work expands on the idea of way-prediction focused mainly on the MRU way of the cache. Zhang performs swapping of the MRU block into the next predicted way, although this is not guaranteed to be the first way. Thus, preventing it from being used with the near-threshold SRAM cells. Zhu's [105] work most closely resembles our work in that it focuses on alternating access patterns to the cache based on performance to achieve energy optimality. However Zhu's work builds on that of Zhang and again because the predicted way is not always in the same place near-threshold techniques do not work. And lastly, Zheng [104] proposes a cache that uses way concatenation to help reconfigure the cache into the optimal energy configuration.

In addition there have been several studies investigating the trade-offs of swapping cache locations. Batson [7] did an investigation of such techniques and found the energy performed from swapping outweighed the gains of hitting on a prediction. However this pertained to standard SRAM cells, if the energy savings of the prediction is increased using near-threshold SRAM, this trade-off favors the use of swapping as was shown by our results.

Additional work was done looking at caches with cache ways of different latencies. Fujii [29] and Balasurbramonian[6] both look at techniques where low energy cache ways are used and operated at slower speeds. And faster, high energy cache ways are accessed first or by hot addresses streams. Their focus is on much larger caches and the reduction of leakage energy. Their work differs in that they focus on using low energy cache ways second after the access to the high energy way or place hot, frequent accesses in the high energy way and cold, infrequent accesses in the low energy way. Whereas RENT cache performs the lookup in the opposite order.

There has also been a large number of studies on subthreshold and near-threshold systems[100, 102, 94, 26]. These studies, however, focus on small performance targets or chip multiprocessing, unlike our system which targets a single general purpose core that can operate in both low power mode and high performance mode. There has also been substantial work on subthreshold and near threshold SRAM design [99, 16, 91, 50], but none of these considers potential cache architectures for embedded applications, or properly address yield and robustness constraints.

3.7 Conclusion

Embedded processors, particularly for mobile applications, are requiring ever increasing performance but still have battery lifetime and area as a critical design parameters. In this chapter an embedded processor with a high performance mode to handle time sensitive and compute intensive work, and a low power mode which can complete non-critical tasks in an energy efficient manner is presented. To achieve this a near threshold tolerant memory structures coupled with voltage and frequency scaling is investigated.

This chapter proposes the RENT cache to provide both a significant reduction in energy in low power mode and a minimal increase in runtime in high performance mode. The cache is designed with one near threshold voltage tolerant cache way to filter accesses to the rest of the cache. This cache way is accessed first, and only on a miss are the other cache ways accessed. If the miss rate in the near threshold cache way becomes to large and degrades performance the cache is dynamically reconfigured to act like a conventional cache, where all the cache ways are accessed in parallel. This changes the cache to have a uniform, single cycle hit latency. Using this technique we show a 53% reduction in energy over a traditional filter cache. This leads to a system that provides a 86% (7.3x) reduction in energy while in low power mode with only a 2% increase in runtime in high performance mode.

CHAPTER 4

Multi-Core Architectures

This chapter presents the use of NTC operation to target parallelizable embedded applications requiring higher performance, but where battery life is important. In particular this chapter shows that logic and memory cells have different optimal supply and threshold voltages, and therefore proposes to allow the cores and memory to operate in different voltage regions. With the memory operating at a different voltage, this chapter then explores the design space in which several slower cores clustered together share a faster L1 cache. Results show that an architecture such as this is optimal for energy efficiency. In particular, SPLASH2 benchmarks show a 53% energy reduction over the conventional CMP approach (70% energy reduction over a single core machine). In addition this chapter explores the design trade-offs that occur in systems with a separate instruction and data cache. Results show that some applications prefer the data cache to be clustered while the instruction cache is kept private to the core allowing further energy savings of a 77% reduction over a single core machine. The work in this chapter was completed with the collaboration of my colleague Bo Zhai. Some of this chapter appeared in both an article presented at the 16th International Conference on Parallel Architecture and Compilation Techniques (PACT'07) [26] and an article presented at the 13th ACM/IEEE International Symposium on Low Power Electronic Design (ISLPED) [100].

4.1 Introduction

Previous work by Zhai et al. [98] and Calhoun [15] has shown that for a CMOS digital circuit there exists an energy optimal supply voltage (V_{min}) below which energy consumption increases because of exponentially increased propagation delay and leakage energy. V_{min} usually occurs in the subthreshold region. However, these papers did not address how to choose the threshold voltage (V_{th}) to further improve energy savings. Zhai et al. [100]

showed that by properly controlling V_{th} to reach the energy optimal voltage a greater savings can be achieved. In this chapter that work is extended to investigate, in detail, architectural choices that can be used to exploit this property.

Zhai et al. [102] shows that, due to the different activity factors and leakage rates for memory cells and logic, the V_{min} of the processor and memory are usually different. As a result, operating the entire system under a single V_{dd} leads to sub-optimal energy efficiency. Therefore in this chapter the use of a chip design that has two separate V_{dd}/V_{th} domains for the core and memory is proposed.

However only changing the V_{th} does not solve the issue of performance loss from aggressive voltage scaling. So in addition, this chapter proposes to employ multiple cores in the near threshold regime, where the best performance for energy trade-off is encountered. This chapter further explores separate control of V_{dd} and V_{th} for the core and the memory, where memory is allowed to operate at speeds both slower and faster than the core it is attached to. In particular, a cluster that has several slower cores connected to the same faster cache is presented.

Clusters have advantages compared to the conventional CMP approach. Applications that have high communication to computation ratios can share data with other cores in the cluster without the coherence overhead of communicating over the bus that connects the different L1's. However, the cores are now contending for the same cache space, which may result in effectively smaller cache sizes due to conflict and capacity misses generated by the other cores within the cluster. L1 sharing also requires a bus between the cores and the L1. Having more cores within a cluster increases the size and capacitance of this bus. This chapter investigates all the major factors that could affect the system energy efficiency, such as L1 cache size, the cluster size, total number of clusters and the selection of V_{dd} and V_{th} within a cluster. Architectural simulation together with circuit-level modeling shows that for most of the SPLASH2 [96] benchmarks the energy optimal point is 2 cores in a cluster. This configuration provides about a 53% increase in energy efficiency over the conventional CMP design.

In addition the possibility of separately clustering the instruction and data caches the impact this has on different applications is explored. Simulations show that due to the low miss rate and high access rate of the instruction cache, further energy savings can be found by keeping small private instruction caches for each core and a clustered data cache. Using this technique a system can increase energy efficiency further to about a 59% reduction over the conventional CMP design.

This chapter is organized as follows: In Section 4.2 the proposed architecture is introduced. Section 4.3 lays out benchmark selection for evaluation of the architecture. Then

Section 4.4 details the energy modeling of a complete system and Section 4.5 details the architectural simulation results of the SPLASH2 benchmarks. In Section 4.6 the impact of splitting the L1 cache into instruction and data caches and hit under miss policies is presented. Finally, Section 4.7 and Section 4.8 present related work and concluding remarks. All technology numbers for this Chapter are drawn from an industrial 0.13um CMOS technology.

4.2 Proposed Near Threshold Architecture

This section explores the unique architectural opportunities of clustering cores to a single cache that is afforded by NTC operation.

4.2.1 New Memory Architecture

Traditional computer design is usually limited by the SRAM/cache speed leading to conventional designs in which the caches run at the same speed or slower speeds than the core. However, the picture changes in the near threshold domain because a system can now take advantage of the V_{dd} and V_{th} knobs, as discussed in Section 2, and have the memory module operate at a speed faster than the core. In the subthreshold regime, a 100mV boost in V_{dd} from 300mV to 400mV can bring almost 10X performance speedup according to silicon measurements in [102]. The V_{th} can be similarly tuned to adjust speed. However, this is not feasible in the superthreshold regime because driving current depends on V_{dd}/V_{th} following the a-power law [76].

Further investigation is performed on the use of multiprocessing techniques in conjunction with voltage scaling techniques to reduce energy consumption. The conventional way that multiprocessing is implemented is illustrated in Figure 4.1(a) where there is one cache per core. In this Chapter a new micro-architecture shown in Figure 4.1(b) is proposed, where there are several cores sharing one local cache forming a ”cluster”. The local cache serves k cores by running k times faster than each individual core. This is achieved by assigning proper V_{dd} and V_{th} to the cores and the SRAM/caches. To simplify the problem, it is assumed that all the cores are running at the same speed and same V_{dd} and V_{th}. But the V_{dd}/V_{th} between the cores and the SRAM/cache could be different. The clusters are connected to the same next level memory, which could be an on-chip cache or an off-chip DRAM.

Because the cores and the memories could potentially operate at different supply voltages, level converters will be needed in between the cores and memories. Subthreshold level converters have been shown feasible in [102]. Considering the V_{dd} space being ex-

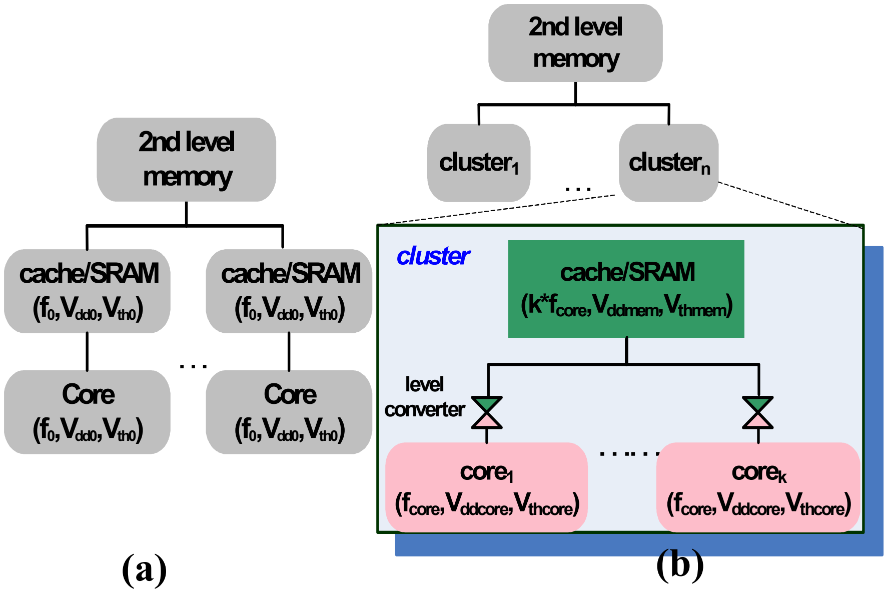

2nd level memory
cache/SRAM
(f_0, V_{dd0}, V_{th0})
cache/SRAM
(f_0, V_{dd0}, V_{th0})
Core
(f_0, V_{dd0}, V_{th0})
Core
(f_0, V_{dd0}, V_{th0})
(a)
2nd level memory
$cluster_1$
$cluster_n$
cluster
cache/SRAM
$(k^*f_{core}, V_{ddmem}, V_{thmem})$
level converter
$core_1$
$(f_{core}, V_{ddcore}, V_{thcore})$
$core_k$
$(f_{core}, V_{ddcore}, V_{thcore})$
(b)

Figure 4.1: Conventional (a) and proposed (b) multi-processor architectures.

plored, the delay of the level converter is minimal compared to the critical delay of the core and the memory. In addition, since multiple cores are connected to a single L1, the tightly coupled nature of the L1 will be removed and a bus will be needed to connect the cores and the cache. This bus will become larger and expend more energy as the number of cores within the cluster is increased.

4.2.2 Architectural Trade-offs

This proposed architecture takes advantage of several different trade-offs within the design. The next few sub sections break down each of the architectural parameters that are able to be varied and describes the trade-offs being made. Each of the different trade-offs yield an optimal energy point for a given benchmark. In Section 6 a complete analysis of the parameters and their interactions for several different benchmarks to find the energy optimal point for each is performed.

4.2.2.1 L1 Cache Size

The size of the L1 is an important factor in controlling the overall system energy. Traditionally a system prefers a larger L1 cache because it results in more hits and therefore a shorter average memory access time. When considering energy consumption, this shorter average access time means that the CPU itself can complete the work in fewer total cycles. Since it takes fewer total cycles the system can slow down and still meet the timing deadline with less energy expended. In addition the higher hit rate also means fewer accesses to the L2. Since the L2 is running at nominal voltage it has a high energy cost per access. Having fewer accesses results in less energy and greater opportunity for the L2 to take advantage of techniques like drowsy caches [49].

On the other hand increasing the size of the L1 also increases the energy per access. This happens because the tag lookup requires a larger structure, and the busses that run through the cache become larger with more capacitance. A muxed based memory is used to reduce the increase in access energy by banking memory to reduce the size of the internal buses. This adds complexity to the cache because muxes are inserted, but it reduces the energy increase compared to a single large cache bank. Even though these techniques are used to limit the increase in energy, there is still additional energy consumed per access. This additional energy per access will result in more energy being consumed for larger caches.

The trade-off here is really to balance the L1 access energy with both the decrease in L2 energy and the faster completion time. This parameter is highly dependent on the access pattern and data set size of the application. Applications with small working sets tend to prefer smaller caches because there tends to be fewer L1 misses with smaller working sets.

4.2.2.2 Cluster Size

The size of the cluster is also an important factor, and highly depends on the benchmark. The biggest benefit of increasing the cluster size can occur in applications with high communication to computation ratios. Clustering the CPUs together allows cores within a cluster to communicate through shared memory without having to traverse the bus connecting the L1's. As the number of cores in a cluster is increased a given core can reach more cores in a single cycle. The other benefit is similar if the cores are sharing data. The sharing pattern doesn't matter as much as the amount of shared data. Similar to the communication latency being shorter, data can be shared between cores within a cluster without the overhead of the coherence protocol. This way if a block was being ping-ponged between two cores, by putting them in the same cluster the system lowers the average access latency. This decrease in both the average communication latency and shared memory latency results in lower power in two main ways. First the average number of cycles required to complete the program is lower, and thus the cpu and system can be clocked at a slower rate and lower voltage and still meet the required execution time. This will result in lower overall energy consumption. Second the number of Snooping access that occur to other L1's is decreased. This means fewer tag lookups in all the L1 caches as well as the snoop request on the bus between the L1's.

On the other hand, as the number of cores in a cluster is increased the cores begin to contend for the same cache resources. This will result in more conflict and capacity misses unless the size and/or associativity of the cache size is increased. The trade-offs of an increased cache can be found in Section 3.2.1. Given a fixed cache size, the increased number cores will generate more L2 accesses, which are costly in terms of energy because the L2 is operating at a higher voltage. In addition the cache must be clocked at a higher rate to satisfy the requests. This higher frequency will result in the cache needing a higher voltage, thus increasing the energy required per access. Also, as the number of cores increases within the cluster they cores must be physically connected with a bus. Traditionally the L1 cache can be tightly coupled with the core creating a fast interface with low capacitance. With a clustered architecture a bus will need to be added to connect the cores. This bus will slow memory accesses, and the capacitance will increase with the number of cores. Driving that larger capacitance will result in increased energy consumption.

The trade-off here is the speeding up that occurs due to the sharing between cores at the cost of increasing cache contention, access energy, and bus size. This is highly dependent on the application. For example an application that has a high communication to computation ratio and a large amount of shared data that is frequently accessed by many cores will prefer a large cluster size. While independent workloads (several independent threads) that have little to no communication or shared data will prefer smaller cluster sizes.

4.2.2.3 Number of Clusters

The number of clusters in a system can also be classified as the number of cores in the system when the cluster size is held constant. The benefit of having more clusters is that more parallelism can be exposed. This increased parallelism means that the task can be divided among more cores and each of the cores can be clocked at a slower frequency and lower voltage. The lower voltage and frequency will lead to energy savings.

On the other hand with increased parallelism more communication overhead is introduced between the cores and additional code to parallelize the application is required. This overhead results in slightly larger instruction counts and more communication across the snooping bus. In addition Amdahl's law applies to the amount of parallelism we will be able to extract in comparison to the inherently serial portion of the code. This of course means that the number of clusters is also dependent on the application that is being run in terms of the amount of parallelism that can be extracted and the amount of code overhead and communication that occurs when the application is parallelized.

4.3 Benchmarks for Evaluating the Design

For our analysis we have chosen to use the SPLASH2 benchmark suite [96]. These benchmarks represent several different forms of parallel applications. Although we are targeting less computationally intensive workloads, i.e. MPEG decoding, we feel that the SPLASH2 benchmarks offer a wide variety of parallel execution models. The important factors to consider for any application are the amount of parallelism that is present in the application, the data set sizes, the communication to computation ratios, and the amount of data sharing that occurs. These factors play the most important role in the performance of the machine. We used the smaller SPLASH2 input sets for faster simulation time and to represent the smaller nature of the problems that we are targeting. The applications we chose to run were Cholesky (cho), FFT, FMM, LU non-contiguous blocks (lun), LU-contiguous blocks (luc), Radix (rad), and Raytrace (ray).

Although these applications are not exactly what we are targeting, we can use them to determine if clustering is beneficial and determine the performance range at which we get the best energy efficiency.

4.4 Simulation Methodology

Our simulation was done with the M5 simulator [9]. It was modified to allow clustering support of the L1 caches. To provide the ability for multiple cores to access the cache in the same cycle, the cache is operated at a higher frequency and the cores within the cluster are run in different phases of the cache clock. For example if there were 4 cores per cluster the cache would be clocked 4X the speed of the cores. The first core would have a rising edge of it's clock on the 1st, 5th, 9th,... clock edge of the cache. The second core would have a rising edge on the 2nd, 6th, 10th,... clock edge of the cache and so forth. This means that each core will see a single cpu cycle latency to the L1 without the need for an arbiter on the bus, or multiple ports to the cache.

4.4.1 Power Models

In order to properly attain energy numbers from our architectural simulation, we needed to determine a power model for all the components in the system. The following subsections will describe how we arrived at the model for each component in the system.

4.4.1.1 Core and L1

The core energy estimations are based upon the processor core frequency and power consumption numbers from ARM946 in [3]. Then we used the same fitted model as in Section 2 to capture the V$_{dd}$ and V$_{th}$ dependency of delay and leakage. The processor without caches consumes 86mW while running at 233MHz with a nominal V$_{dd}$ of 1.2V. The cache power/energy numbers of different sizes were extrapolated from a memory compiler in 0.13um technology. The baseline machine is assumed to have the parameters in Table 4.1. 64kB of unified data and instruction L1 cache is chosen as a reasonable number considering the problem size of the SPLASH2 benchmark applications [96].

Parameter	Value
CPU	233MHZ, in-order functional mode
Unified D/I L1 Cache	64kB, 2-way, 64B block size
L2	2MB, 8-way, 10-cycle Latency

Table 4.1: Baseline Architecture.

Previous work [101] has shown that current sensitivity to subthreshold variation increases dramatically with reduced V$_{dd}$. Previous work [93, 14, 91, 50, 101] has illustrated successful SRAM designs that operate in subthreshold regime. However, all these works result in an area overhead. Part of the reason for the area overhead is that larger channel area helps suppress random dopant fluctuations (RDF), the dominating factor in the subthreshold regime [70].

In order to factor in the design area overhead of voltage scalability for the SRAM, we carried out Monte Carlo simulations and determined the amount of up-sizing needed for the memory cells under a certain yield constraint. A exponential function is then fitted to the results and used in the rest of the Chapter. This up-sizing increases the physical size of the cache, which, in turn, increases both the access energy and the bus length to connect the cache. These additional increases were modeled in our evaluations.

4.4.1.2 DRAM and L2 Design

Off-chip access to DRAM is power-hungry because of the high capacitance in the chip package and off-chip wires. Therefore, a energy-oriented design needs to have a big enough L2 cache so that it can shield off the majority of the conflicting L1 misses from accessing the off-chip memory. For our analysis we have chosen a 2MB on-chip L2 cache. Considering the size of this L2, we decide to fix its operating voltage at nominal (1.2V) instead of voltage scaling to avoid over design. As aforementioned, designing a large L2 for voltage scalability with high yield implies significant area overhead and therefore results in high switching and leakage energy.

From simulation we found that the L2 is not heavily accessed and has a much lower activity rate compared to the L1. Therefore we will design the L2 cache by using low standby leakage techniques such as drowsy cache [49]. The standby leakage is assumed to be 1/20 of that during active mode. The same argument does not hold for the L1 because the L1 is accessed considerably more frequently than the L2. The L1 is also smaller in size than the L2 cache and we are able to trade off some area for energy efficiency and voltage scalability.

4.4.1.3 Bus Modeling

The physical parameters are drawn from a 0.13um technology doing detailed analysis of the repeaters and sizes necessary for proper operation. Using this model we can include bus energy for both the CPU-L1 bus and the L1-L2 bus, which consists of data, address, and command fields. The length of the CPU-L1 buses scales linearly with cluster size and the length of the L1-L2 buses linearly with cluster number. Basic footprint size is taken from a commercial processor core [3]. No breakdown of bus energy is presented in the graphs, but all graphs and data referring to total energy include the bus energy numbers.

4.4.2 Baseline Machine

Our baseline machine is summarized in Table 4.1. It is a simple in-order CPU running at 233 MHz with a 64kB unified cache. For all configurations we hold the runtime to be the same as that of the baseline machine. In Section 6.6 we will look at the optimal choices at different baseline performance numbers (CPU Frequencies).

4.4.3 Simulation Configurations

We simulate the system by varying the cache size from 4-128kB, the number of clusters from 1-16, and the number of cores per cluster from 1-8 for each SPLASH2 benchmark. We then do a power analysis with different voltage scaling techniques. One in which we do the same V_{dd} scaling on both the L1 and core, another where we scale V_{dd} separately for the L1 and core, and the third where we scale V_{th} separately as well.

4.5 Results

In order to analyze the impact of the proposed architectures in energy efficient processor design, we must quantify the system performance energy trade-off using architectural simulations. We performed the simulation using the M5 Simulator [10] to determine the energy consumption for different configurations. During the cluster mode, we assume that the L1's are k times faster than the core, where k is the number of cores per cluster.

The relative latency of the components need to be scaled so as to capture the voltage scaling effect. For instance, when we have a multicore system, each core and L1 inside the system can be accordingly slowed down compared to the single-core baseline machine, since our constraint is to keep the runtime constant. In architectural simulation, this is equivalent to having a faster L2 and DRAM.

4.5.1 Optimal L1 Size

It is known that the size of the L1 affects the system performance in terms of average access latency, but more importantly we found that it also affects the energy efficiency of the system. In order to understand how the choice of L1 size impacted the energy performance of a system, we first performed an analysis on a supply voltage scaled uniprocessor system. With the number of parameters we intended to vary for the complete study, this was a simple way to do a first pass analysis and present some interesting results while only varying one parameter. We present the results while allowing the die size to increase as we increase the L1. A similar study was done where the die size was held constant and the L2 was made smaller to compensate the increase in the L1 and the results were similar.

With a single processor and single L1 system, we varied L1 sizes and optimized the energy consumption for each size by tuning the V_{dd} and V_{th} of each component. Figure 4.2 shows the energy consumption for Cholesky. We found similar trends for all the SPLASH2 benchmarks, although the optimal cache size varied across the applications. The energy consumption for the processor core, L1, and L2 are also shown. As L1 size increases from 4kB to 128kB, the number of accesses to the L1 remains constant because the same instruction stream and data pattern come from the processor core. However, L1 energy consumption increases with larger L1 sizes due to larger array size and larger capacitance. The other implication of various L1 sizes is the L1 performance. The L1 miss rate reduces significantly with larger L1s, which also results in a lower number of access to the L2 due to fewer L1 misses and writebacks. Therefore the L2 energy reduces with larger L1 sizes. The core's energy consumption also reduces with larger L1 sizes because of the reduced number of clock cycles that results from a higher L1 hit rate (lower average access latency).

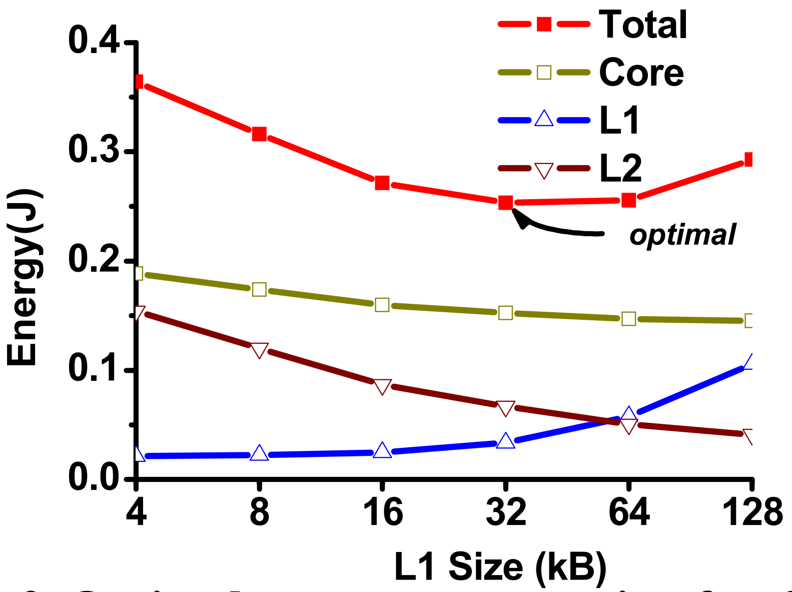


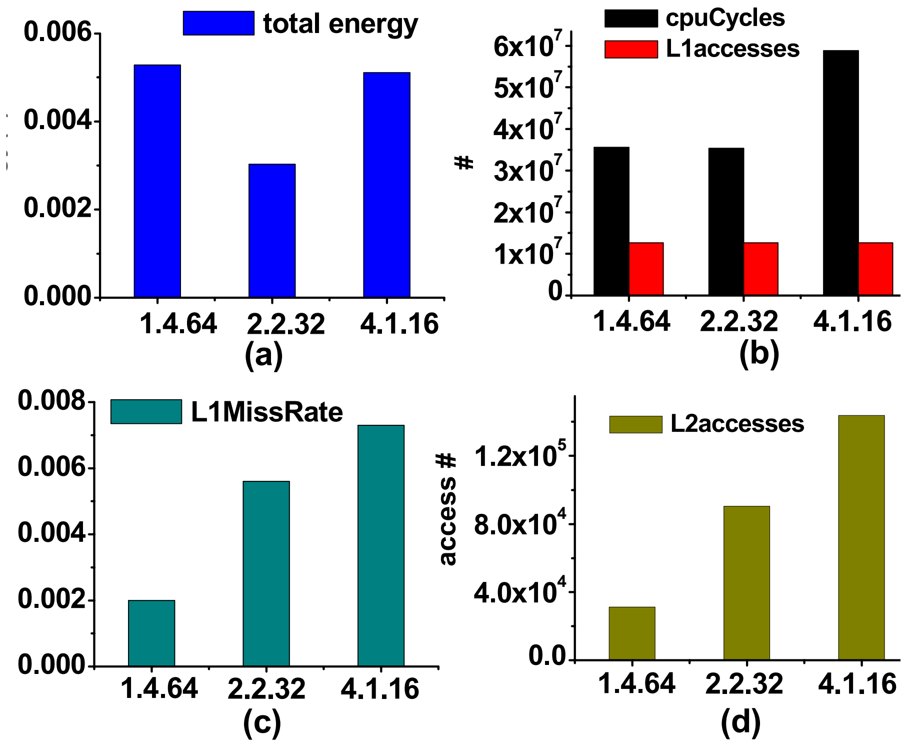
Figure 4.2: Optimal energy consumption for *Cholesky* when using one core and one L1.

4.5.2 Optimal Cluster Size

After having an understanding of how the L1 size impacted energy performance we move on to studying the impact of multiprocessing and clustering on the energy consumption. The setup for this study is described in Section 5.3. The analysis was done on the 240 configurations for each benchmark.

In order to fully understand the benefit of running in cluster mode Figure 4.1(b) vs. conventional connection Figure 4.1(a) we specifically compare three cases in which the overall die size is held constant and present the results in Figure 4.3. Figure 4.3(a) shows the energy consumption of the three candidates for Cholesky in the SPLASH2 benchmark suite. The best configuration for energy efficiency is 2 cores per cluster, with 2 clusters.

To help illustrate the reason 2 cores per cluster was optimal we break down the results in more detail in Figure 4.3(b-d). At 4 clusters with 1 core per cluster, a conventional CMP, the number of CPU cycles is larger than both of the clustering cases. This occurs because the average memory access latency is longer because the L1 miss rate is high due to the fact that shared memory accesses are forced to miss in the local L1 cache and snoop neighboring L1's to get the data. Now as the number of cores in a cluster is increased to 2 some of the shared memory of other cores is visible within the cluster's L1 to both cores without having to access another L1. This significantly reduces the average memory access time and results in a reduced number of CPU cycles. Meanwhile the larger shared cache within the cluster results in a higher hit rate and reduced accesses to the L2. Reducing the number of access to the L2 reduces the energy consumed by the L2 as was shown in



Notation: (cluster #).(cores per cluster).(L1 size per cluster in KB

Figure 4.3: Three configuration comparison.

Section 6.1.

As we scale to 4 cores per cluster we now have reduced the L1 miss rate even further, but the energy per access of the L1 and the energy to operate the large bus to connect the cores within the cluster begin to outweigh the gains we see in reduced L2 traffic. Also, you can see that there is not a large reduction in CPU cycles. This happens because the first set of clustering encapsulated most of the memory sharing and synchronization that took place in this application.

For this study it is important to note that we chose three points at which the die size was nearly the same. Remember that there are some interconnect and memory scaling issues, but the sizes are still close. In the later studies we will present the global optimal configuration without regard to total cache or die size. We wanted to show in this study that given a fixed die size, clustering is the optimal choice for this benchmark.

4.5.3 Various Energy Savings Modes

Now that we have shown for a fixed die size that clustering can outperform conventional CMP designs, we explore the entire design space and analyze the impact of using the V_{th} to further improve performance. We again use the same 240 configurations per benchmark as in Section 6.2. The results presented in this section are for the Cholesky benchmark, results for other benchmarks are summarized in Section 6.5. In this study we evaluate the impact of different scaling approaches. Specifically, we assume the baseline single-core single-cache machine does not do voltage scaling and runs constantly at 1.2V. The three different scaling approaches are: 1) traditional V_{dd} scaling using MP while maintaining one core per L1 cache (Figure 4.1 (a)), 2) V_{dd} scaling using MP but with cluster mode configuration and 3) V_{dd} and V_{th} scaling with cluster mode configuration. The runtime of all three systems is set to match that of the baseline machine.

The results are presented in Figure 4.4. Conventional CMP V_{dd} scaling brings us 38.1% savings over the baseline machine, but clustering cores with V_{dd} scaling results in a 18.9% improvement over traditional scaling techniques and 49.9% improvement over the baseline. Finally we show that clustered cores with both scaled V_{th} as well as V_{dd} yields a global optimal energy, and about a 53% percent improvement over using traditional scaling techniques on a conventional CMP.

The V_{dd} scaling technique with conventional CMP finds an optimum with 4 cores and 4 L1 caches at a V_{dd} of 0.8V. With clustering, the system finds an optimal with 2 cores per cluster at 3 clusters. The optimal L1 size for all 3 scaling approaches is 64kB for this benchmark. The energy optimal point was chosen without regard to die size. If given a die size constraint the energy optimal point can be found within the configurations that meet

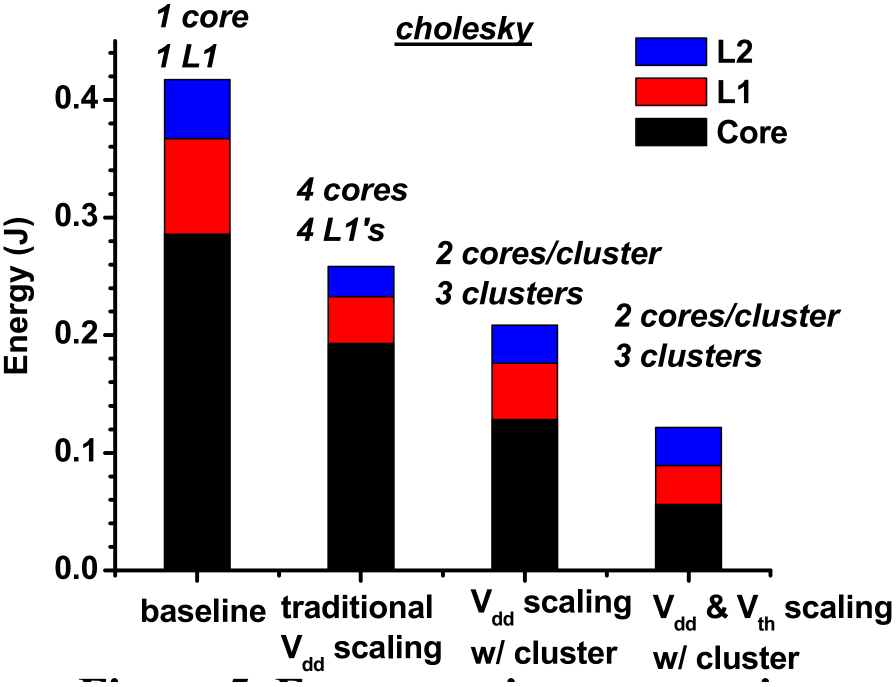


Figure 4.4: Energy savings comparison using various scaling methods.

that die constraint. Section 6.2 provides an example of fixed die size analysis where three configurations with the same die size are compared.

4.5.4 Multi-Dimensional Analysis

To understand the entire design space further we present Figures 4.5, 4.6, 4.7, and 4.8. These figures breakdown the entire design space for a fixed total L1 size, 128kB, for the Cholesky benchmark using both V$_{dd}$ and V$_{th}$ scaling. In Figure 4.5 we present the total energy of the system, you can see that there is a well along the 2 cluster axis. This minimum represents the fact that the optimal number of clusters is 2. Additionally there is a dip along the 2 cores/cluster line, resulting in a optimal location of 2 clusters with 2 cores per cluster. To understand this better we break down the core energy in Figure 4.6, the L1 energy in Figure 4.7, and the L2 energy in Figure 4.8. Note the cluster number axis is in the opposite direction for Figure 4.7.

The core energy looks like a surface that decreases from left to right and from back to front. As you go from front to back or from left to right there is an increasing total number of cores. As we increase the number of cores, we can run each of them at a lower voltage and still maintain the same performance constraint. Since there is a quadratic relationship in voltage and a near-linear relationship in computational power as we increase the number

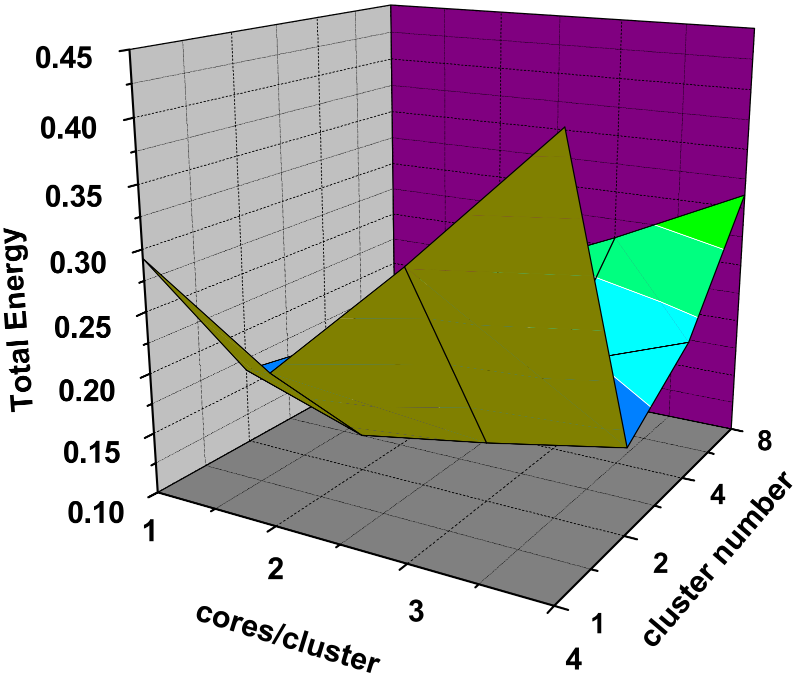


Figure 6. Total Energy for *Cholesky*

Figure 4.5: Total Energy for *Cholesky*.

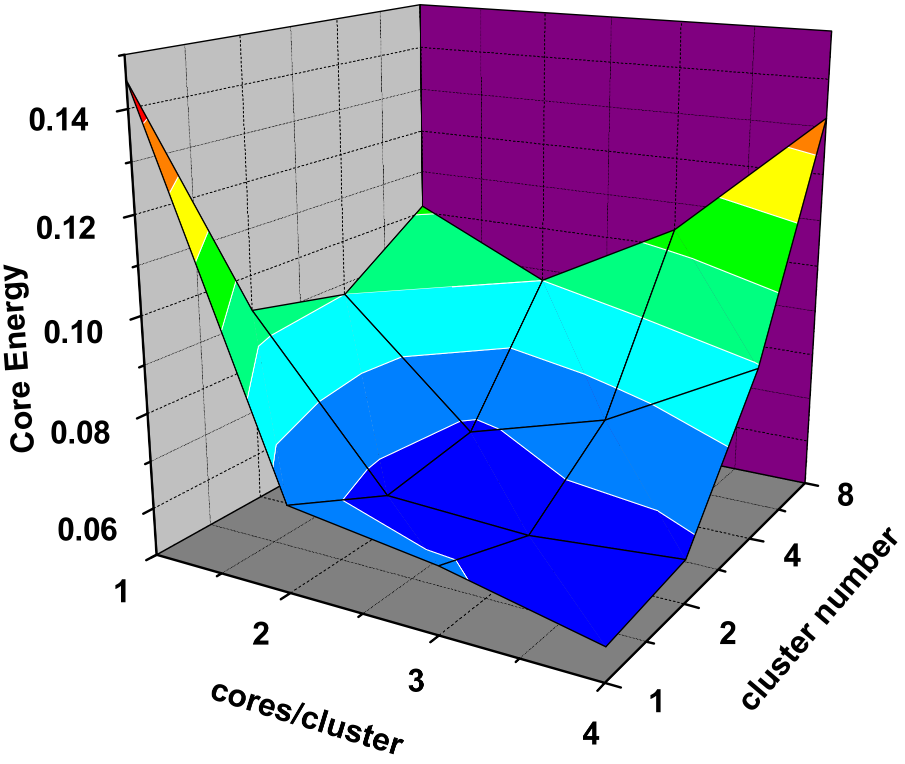


Figure 4.6: Core Energy for *Cholesky*.

of cores, the overall energy consumed reduces until the core reaches the near threshold regime. If we lower the voltage further we see less energy gain for a similar decrease in voltage. In this graph the point at which the cores begin running in the near threshold regime is around 4-6 cores. Beyond that point the improvement form lowering the voltage is less than the overhead of making the application more parallel and the total increase in number of cores at a lower voltage.

The L1 energy increases from front to back, decreasing number of clusters. This is because the fewer clusters that are present in the system the larger the L1 caches will become. This occurs because we are holding the total cache size on the chip constant. As we increase the cache size, the energy per access is increased, and therefore the overall L1 energy increases. As we go from left to right, increasing the number of cores per cluster, the energy increases. As we increase the cluster size, we require that the L1 operate at a higher frequency to satisfy the requests. This increase in frequency requires us to increase the voltage of the L1 and the energy savings from the faster communication amongst the cores is outweighed by the increase in the energy per access at the higher voltage.

In Figure 4.8 we present the L2 cache energy. It is increasing from left to right and from front to back. This increase is related to the larger number of cores. In particular there are

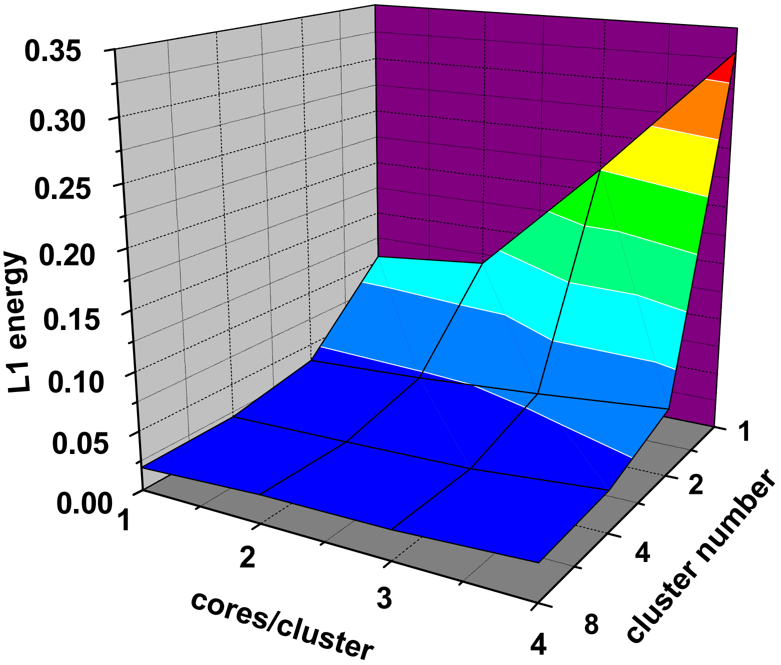


Figure 8. L1 Energy for *Cholesky*

Figure 4.7: L1 Energy for *Cholesky*.

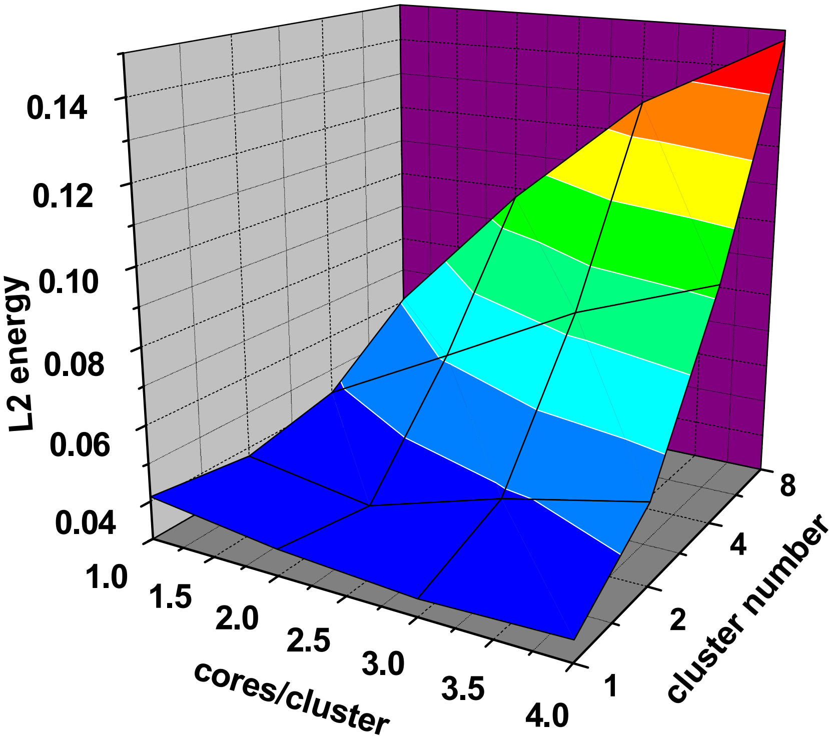


Figure 4.8: L2 Energy for *Cholesky*.

more cores sharing the same total L1 space, because we held it constant. This results in more contention and a higher miss rate among the L1's. With more L1 misses there will be more L2 accesses and the energy consumed by the L2 will increase. This increase is highly dependent on how much contention and sharing there is among the different applications.

When the components are summed for the different figures you result in wanting few clusters to reduce the L2 traffic, but more clusters to keep the L1 sizes smaller and less energy hungry. This summation leads to an optimal point of around 2 clusters. The core energy dictates that the optimal total number of cores is somewhere around 4-6 cores. Divided into 2 clusters that means either 2 or 3 cores per cluster. Thus resulting in the energy optimal point of 2 cores per cluster and 2 clusters for this benchmark with 64kB of L1 cache per cluster. This same analysis was done holding either the cluster size or number of clusters constant to understand the effects of the L1 size on the optimal solution. The graph is consistent with the points presented in Section 3.2 and was not presented here for brevity.

4.5.5 Global Optimal Solutions

Now that we showed the energy optimal point for the Cholesky benchmark we ran the same analysis for additional SPLASH2 benchmarks and the results are presented in Table 4.2. The table shows the global energy optima for different applications in the SPLASH2 benchmark suite, Clustering is optimal for 6 applications with the optimal cluster size of 2 cores.

	n_c	k	L1 size (kB)*	*energy savings*
cho	3	2	64	70.8%
fft	2	2	32	72.6%
fmm	8	2	128	79.7%
luc	3	2	32	77.8%
lun	2	2	64	68.4%
rad	16	1	128	84.2%
ray	3	2	128	65.1%

Table 4.2: Each Benchmarks Optimal Configuration. k: number of cores per cluster; n_c: number of clusters; *L1 size is per cluster. Energy Savings is relative to baseline uniprocessor machine.

4.5.6 Optimality under Different Performances

So far we have compared the energy savings under constant baseline performance, 233MHz. The optimal V_{dd}/V_{th} configuration changes with target performance, therefore we present the optimal energy consumption for Cholesky under different performance requirements Figure 4.9(b). The MP scaling uses clustering and optimal V_{dd}/V_{th} selection. Performance on the x-axis refers to the frequency of the baseline single-core single-L1 machine. With reduced target performance, from right to left in Figure 4.9(b), the energy savings increases first because of relaxed frequency constraints on each core and L1 cache, both of which operate in the near threshold regime. Then energy consumption increases because the cores begin to scale out of the near threshold and into the subthreshold regime causing considerable performance loss in comparison to the power gain. Also the lengthened execution time prolongs L2 leakage energy.

The V_{dd} and V_{th} settings with different performance targets are shown in Figure 4.9(a). The V_{dd} for the L1 slightly reduces from 233MHz to 100MHz due to relaxed frequency. However it holds around 600mV because lower V_{dd} implies significant up sizing as described in Section 3. And this area increase from SRAM redesigning nullifies the savings from reduced V_{dd}. The V_{th} increases with lowered performance requirements to balance

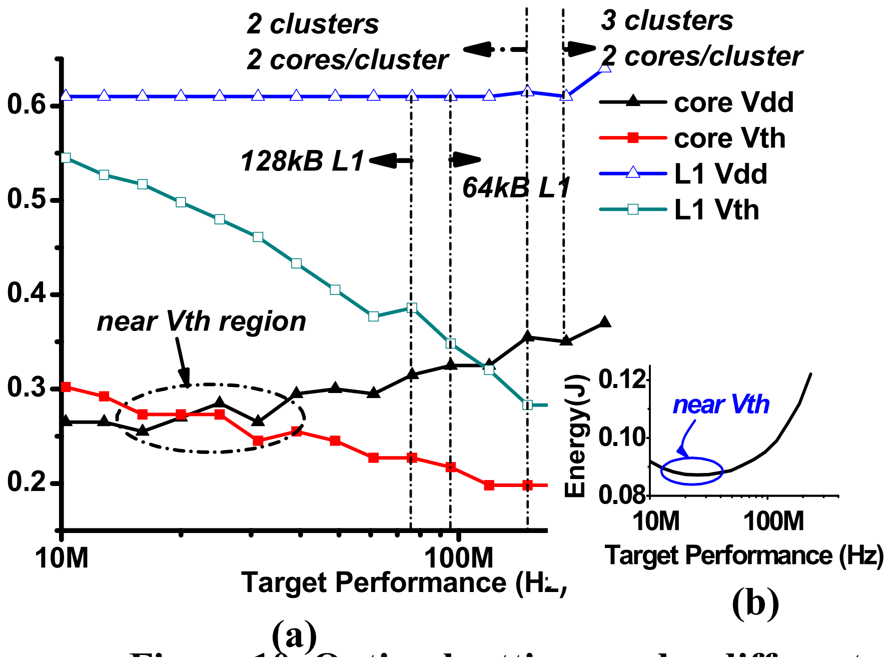


Figure 4.9: Optimal settings under different target performances.

switching energy and leakage energy. The cores operate at a significantly lower V_{dd} and V_{th} than the L1s for all of the swept performance range because it is running at half the speed of the L1. In addition logic gates are more robust than SRAM and voltage scale without area overhead.

In Figure 4.9(a) the optimal number of clusters increases from 2 to 3 for targets above 150MHz. This can be attributed to the fact that in order to meet the higher performance constraint we need more computational power. The increase in the number of clusters, and thus total cores, requires less energy than voltage scaling the smaller number of cores to meet the same constraint. Figure 4.9(a) also shows that the optimal L1 size changes from 64kB to 128kB for targets below 76MHz because the relative contribution of L2 energy consumption starts to increase. A larger L1 helps to suppress L2 accesses. Although the larger L1 incurs more energy per access, the amount of energy saved from reducing the L2 accesses outweighs any increase in the L1.

Finally, we have highlighted the near V_{th} region on both plots in Figure 4.9. Optimal energy consumption is achieved at this voltage regime and at a target frequency of tens of megahertz (15MHz-50MHz). These operating frequencies are much higher than those shown in other subthreshold work [102] and are well suited for parallelizable embedded applications requiring higher performance, but where battery life is important. Such applications might include MPEG and MP3 decoding.

4.6 Split L1 Cache

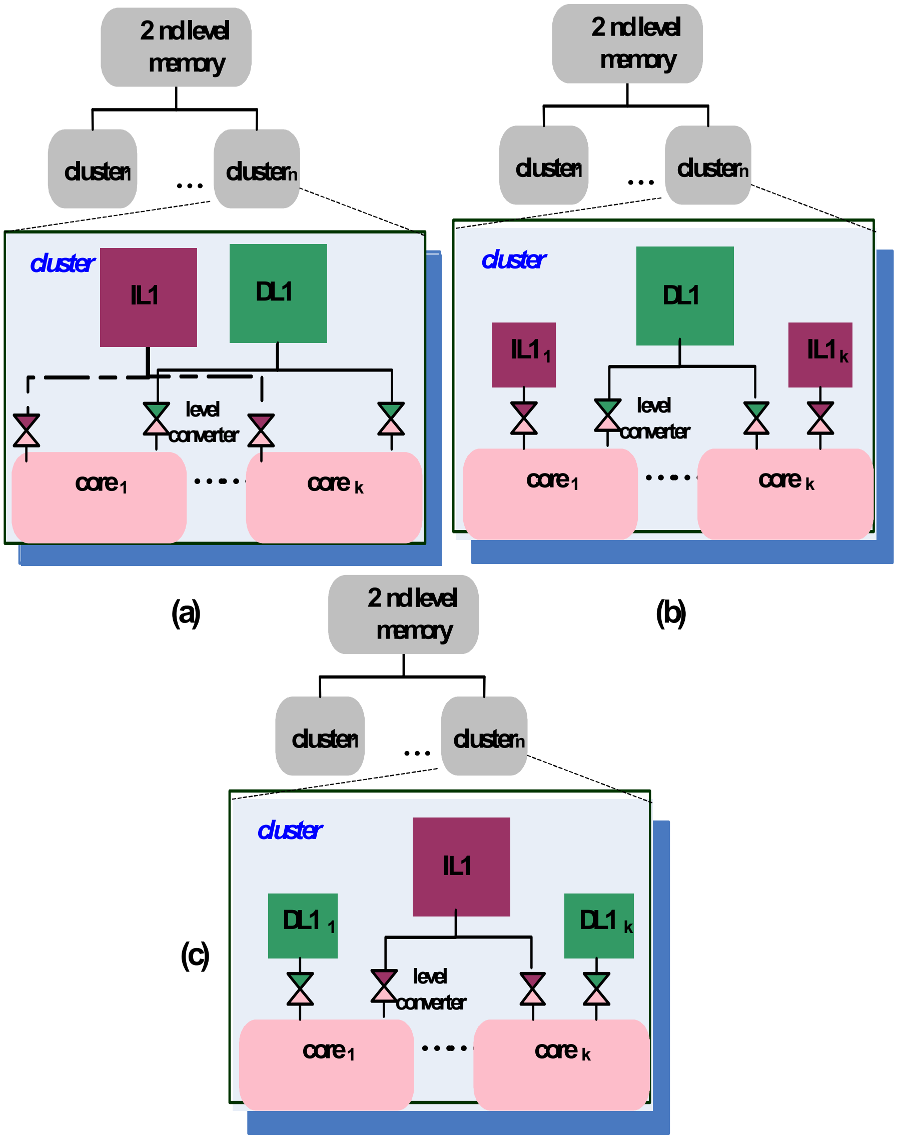


Figure 4.10: (a) Clustered I and D, (b) Clustered D and (c) Clustered I architectures.

We further investigated the impact of split instruction (IL1) and data (DL1) caches. By splitting the unified L1 cache we present a few more architectural choices. First we could cluster both the IL1 and DL1 for each cluster as in Figure 4.10(a). Second we could explore other architectures where we only cluster the IL1 and leave each core a private DL1 as in Figure 4.10(b), or vice versa as in Figure 4.10(c). In this section we will explore those three different design choices and discuss their impact on different programing models.

4.6.1 Architectural Trade-offs

There are several architectural trade-offs that occur when we consider splitting the unified cache. By splitting the L1 cache we remove any contention that was occurring between the instruction and data streams. In some benchmarks the data stream constantly evicted instructions from the cache do to conflicts and the overall number of cycles to complete the program was increased. By removing these conflicts we can decrease the number of cycles improving energy efficiency. In the next two subsections we break down the benefits of clustered versus private versions of both the IL1 and DL1.

4.6.1.1 Clustered IL1 vs. Private IL1

At first glance clustering the instruction cache seems to make sense for applications where each thread is executing the same instructions, i.e. SIMD. The reasoning is that by sharing the cache you can reduce the die size and improve performance because only one thread will incur a miss, and others will hit the line in the shared cache reducing execution time and thus energy. But if the instruction stream is small, the instruction cache can end up with a low miss rate and a high access rate. Since we are limited by Amdahl's law in only improving performance for misses, the gains we see in this case are marginal. Further, by clustering the instruction cache we require the cache to operate at a higher frequency, and therefore a higher energy per access. However, if the instruction miss rate is large enough, then it begins to make sense to share the instruction cache to improve the access latencies. In applications that run different sets of instructions on each thread, i.e. MIMD or mutliprogrammed workloads, it does not makes sense to cluster the instruction cache.

4.6.1.2 Clustered DL1 vs. Private DL1

Unlike the instruction cache, the data cache is only accessed by around 30% of the instructions in the SPLASH2 benchmark suite [96]. With the lower access rate, higher miss rate, and larger data sharing, the data cache is suited well for clustering. If the application does large amounts of communication through shared memory or shares large amounts of data, clustering the data cache will result in shorter synchronization delays and access times. This results in the program completing in fewer cycles leading to energy savings. On the other hand, clustering the data cache does not make sense if the applications do not share data or synchronization variables because it will lead to contention in the cache and no improvement for the core in terms of cycles to completion.

4.6.2 Evaluation Methodology

We chose three different SPLASH2 benchmarks to analyze on the three different architectures. We wanted to get a variety of different synchronization schemes. We picked Cholesky because of the high use of locks and pauses but low use of barriers. We chose LU because it uses many barriers and no locks or pauses. And finally we chose FFT which uses very few barriers, locks, or pauses.

We simulated the benchmarks with the same configuration choices as before, but now we allowed the L1 cache to be split. We varied the IL1 and DL1 sizes from 4-128kB independently while exploring all 3 configurations in Figure 4.10.

4.6.3 Results

The results of the optimal configuration for each of the three architectures on the three benchmarks is presented in Table 4.3. The energy savings is presented in percentage reduction over the baseline single core machine with 64kB split IL1 and DL1s. You can see that the benchmarks prefer to run with a clustered data cache and a private instruction cache for the best overall energy savings.

	IL1	DL1	n_c	k	IL1 size*	DL1 size*	*energy savings*
cho	Unified		3	2	64kB Unified		70.8%
	Clustered	Private	3	2	8kB	128kB	73.0%
	Private	Clustered	2	2	16kB	64kB	76.5%
	Clustered	Clustered	3	2	16kB	64kB	71.1%
lun	Unified		2	2	64kB Unified		68.4%
	Clustered	Private	2	2	8kB	32kB	64.2%
	Private	Clustered	1	2	16kB	32kB	72.9%
	Clustered	Clustered	2	2	8kB	64kB	66.4%
fft	Unified		2	2	32kB Unified		72.6%
	Clustered	Private	1	2	4kB	64kB	72.1%
	Private	Clustered	1	2	8kB	64kB	75.0%
	Clustered	Clustered	2	2	4kB	32kB	69.7%

Table 4.3: Each Benchmarks Optimal Configuration for split L1. k: number of cores per cluster; n_c: number of clusters; *L1 size is per cluster. Energy Savings is relative to baseline uniprocessor machine.

The SPLASH2 benchmarks all exhibited low miss rates for small instruction caches (4-16kB) so it did not make sense to cluster the instruction cache as described in Section 7.1.1. This is easily seen in both the FFT and LU benchmarks where each instance of clustering the IL1 results in worse performance than the unified version. In the case of Cholesky, the increase seen over the unified version when the IL1 is clustered is a result of less contention on cache lines between the instruction and data caches. With less contention we reduce the number of cycles and gain more energy savings. Even though for Cholesky a clustered IL1 outperforms the unified cache case, a clustered DL1 and private IL1 is still the overall optimal choice. This happens for the same reasons as the FFT and LU benchmarks.

4.7 Related Work

There has been a myriad of different work done in both the fields of subthreshold design and parallel architectures for low power. The most recent work done for power aware parallel architectures was by Li et al. [57]. Li proposes a dynamic runtime method to use dynamic voltage and frequency scaling available on cores to optimize power given a performance constraint on parallel applications. Our work differs in that we focus on near threshold operation and the additional architectural choices for clustering that it allows. Li's work could be extended to work in combination with our design to dynamically put nodes to sleep to find the most optimal power configuration available. It could also be used to allow only one core per cluster to operate when applications require more cache space, or to dynamically reassign threads to cores within clusters that communicate more often.

Other power aware parallel CMP and simultaneous multithreading (SMT) designs have been proposed by [53, 55, 54, 58, 78]. This work differs from both Li's [57] and our work in that it focuses on mutliprogrammed workloads, whereas our work focuses on parallel applications.

Huh et al. [38] does an in depth study of the design space of CMP's. However they do not evaluate either power considerations or the possibility of clustering multiple cores to the same L1 cache. Other work exploring the design space of CMP's that considers power budgets was done by Ekman and Stenstrom [28] although their work focuses on the types of cores that should be used in a CMP given the amount of parallelism applications present. Our work differs in that we assume a fixed core design and chose the optimal number of cores and the amount of clustering that occurs.

There is also an extensive body of work on reducing energy overheads in bus based CMP's [64, 65]. Moshovos [64] and others provide different techniques to reduce the number of snoops into L1 caches by filtering out requests that will not find the block in that cache. This technique could be used to lower the energy consumption of the snoops in our system removing one of the performance benefits of clustering. However, clustering not only eliminates snoop energy but also helps reduce the latency to shared data structures in the cache within a cluster. This reduction in latency means the system can be run at a slower frequency and still complete in time. Snoop filtering techniques help to reduce the overall system energy further but when removing the energy consumed by snoops in our simulations we still find clustering to be the optimal choice.

4.8 Conclusions

In this Chapter we have investigated the optimal threshold voltage selection for energy efficient near threshold design. By separately controlling the supply voltage and the threshold voltage of the core and the memory, we can achieve better energy efficiency. We have combined these techniques with a novel multiprocessor architecture where multiple cores share one faster L1 cache in a cluster to further improve energy savings. We found that for a typical SPLASH2 application the proposed architecture can provide about 70% energy savings over a uniprocessor system and about 53% over conventional multiprocessor scaling. The optimal cluster size is 2 cores for most of the SPLASH2 benchmarks that we investigated, and the system achieves best energy efficiency when operating in the near threshold voltage regime. The optimal target frequency was that of a single 10-50MHz core, showing that this technique is well suited for parallelizable embedded applications requiring higher performance, but where battery life is important such as MPEG decoding. We further studied the impact of splitting the instruction and data cache. We found that clustering the data cache while keeping a separate private instruction cache per core performed the best for the SPLASH2 applications providing an energy savings of up to a 77% reduction over that of a single core machine.

CHAPTER 5

Signal Processing Architectures

In addition to conventional multiprocessor systems, power has become a critical design constraint for embedded handheld devices. This chapter proposes a power-efficient single instruction multiple data (SIMD) architecture, referred to as Diet SODA, for digital signal processing (DSP) applications. The key design idea is to apply NTC operation on SIMD architectures to significantly lower the power consumption. The major features of Diet SODA are very wide SIMD width, scatter/gather data prefetcher, and dual mode operation. A case study was performed on digital still camera (DSC) applications; the results show that Diet SODA achieves $\sim$130x better performance and $\sim$500X better energy efficiency than a conventional DSP solution. The work in this chapter was conducted with the collaboration of my colleagues Sangwon Seo and Mark Woh. Some of the results presented here appear as part of an article in the Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design [79].

5.1 Motivation

Mobile devices have rapidly proliferated and the deployment of the handheld devices will continue to increase at a spectacular rate. As today's devices not only support advanced signal processing of wireless communication data but also provide for richer sets of various applications, power dissipation has become a more important design constraint. Increasing power consumption leads to increasing energy costs as well as impacts chip reliabilities. Therefore, more power-efficient processors for embedded DSP applications are highly required.

Among many DSP applications, high resolution cameras have become an integral part of most cell phone designs. As a result, the market for these cameras has mirrored the spectacular growth in mobile phones [22]. Furthermore, the expectation is that these mobile cameras produce an image whose quality should approach that of high quality digital still cameras (DSCs). Therefore, a DSC processor needs to be of high-performance to support a large amount of image data and perform the DSC image processing tasks in a highly energy-efficient manner in order to conserve critical battery life for other phone applications.

Traditionally, the DSC image signal processing pipeline is implemented in digital signal processors (DSPs) or application specific integrated circuits (ASICs). DSP-based solutions [41] support high flexibility and handle various DSC algorithms, but they suffer from lower performance and higher energy consumption than ASIC solutions. ASIC-based solutions [97, 66] are highly specialized and optimized for the DSC image signal processing pipeline, but such designs lack flexibility and require longer design time. Therefore, to achieve high processing performance efficiency at low cost while maintaining programmability, hybrid architectures [85] are employed. In these designs, ASICs are typically used for a preview mode, where high processing capabilities are desired, and DSPs are adopted for picture-taking and post-processing modes, where flexibility is more important. However, such a heterogeneous solution is inefficient to build and maintain.

To address these challenges, this Chapter presents a power-efficient programmable architecture, Diet SODA, that has been optimized for DSC image signal processing. Diet SODA exploits near-threshold operation [100] on a wide-SIMD architecture — SODA [59]. In the near-threshold operation, circuits operate at lower than normal supply voltages, reducing power consumption by $\sim$100x. Near-threshold operation offers a new opportunity for mobile applications such as DSCs to reduce power consumption. However, the reduction in power consumption comes at a cost of a $\sim$10X performance degradation. Diet SODA overcomes these hurdles by exploiting architectural features specific to near threshold operation. The key features of Diet SODA are 1) very wide SIMD width to exploit the significant amount of data level parallelism (DLP) inherent in DSC applications that helps overcome the frequency loss from operating in the near-threshold region; 2) scatter-gather data prefetcher to support 2D memory access enabled by the latency difference between the full voltage SIMD memory and SIMD data engine operating at near-threshold voltage; and 3) dual voltage modes where the SIMD data engine operates at either full or near-threshold voltage based on processing demands. The customized architecture was implemented in Verilog and synthesized in IBM 90nm technology using Synopsys physical compiler.

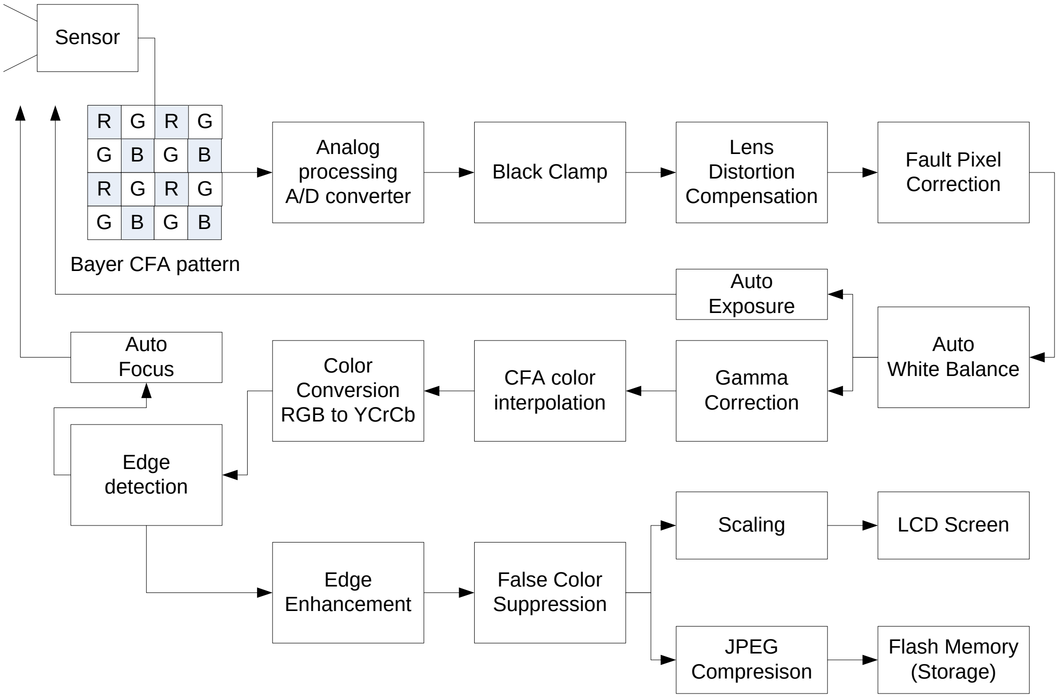


Figure 5.1: A typical DSC image signal processing pipeline [21], [41]

5.2 DSC Algorithm Analysis

5.2.1 DSC Signal Processing Pipeline

Figure 5.1 shows a typical DSC image signal processing pipeline [41], which performs multiple processing steps to generate a high-quality image. The image is first captured by a CCD or CMOS sensor using a Bayer-pattern [8] color filter array (CFA). Then, the image is digitized with a 10- or 12-bit A/D converter. The *Black Clamp* adjusts the pixel values by subtracting a black offset value from all pixel values. The *Lens Distortion Compensation* adjusts the brightness of pixels depending on the spatial locations and the *Fault Pixel Correction* interpolates defective pixels with neighboring pixels. After all these pre-processing steps, *Auto White Balance* computes the average brightness of each color component and balances the energy of the colors. Based on the brightness information, *Auto Exposure* appropriately adjusts the CCD or CMOS exposure time and gain. After the white balanced image pixels are compensated by *Gamma Correction*, *CFA color interpolation* uses the one-color-per-pixel Bayer-pattern image to interpolate and generate the full color (R, G, and B) resolution for each pixel. The RGB color pixels are filtered by *De-Noise* and scaled down and sent to the LCD screen in preview mode. In picture-taking mode, the noise-filtered images are transformed to the YCrCb color domain. *Edge Detection* detects edges to help *Auto Focus*, and *Edge Enhancement* is performed. Next, *False Color Suppres-*

sion occurs, and finally the image is compressed by using *JPEG Compression* and stored in flash memory. Later, post-processing tasks such as *Histogram Calculation*, *Histogram Equalization*, and *Spatial Frequency Filtering* are used to enhance the quality of the stored images.

5.2.2 Characteristics of DSC Algorithms

In this section, we analyze the key algorithms in the two modes (preview and picture-taking) of DSC signal processing pipeline and post-processing tasks to find opportunities for improving the processing performance and energy efficiency.

Mode	Task	SIMD	Scalar	Overhead
Preview Picture-Taking	Black Clamp	100%	0%	0%
	White Balance	100%	0%	0%
	Auto Focus	71%	14%	14%
	Gamma Correction	0%	100%	0%
	CFA Interpolation	87%	0%	13%
	Auto Exposure	95%	2%	2%
	Color Conversion	100%	0%	0%
	Edge Detect/Enhance	81%	0%	19%
Post-Processing	Histogram Equalize	38%	44%	18%
	Spatial Freq. Filter	77%	3%	20%

Table 5.1: Data level parallelism analysis for DSC image signal processing algorithms. Instructions are categorized into three groups: SIMD, scalar, and overhead instructions.

Table 5.1 presents the data level parallelism (DLP) analysis of the DSC signal processing algorithms. Instructions are broken down into three categories: SIMD, scalar, and overhead workloads. The SIMD workload consists of traditional arithmetic/logical functional operations and load/store operations that can be executed in SIMD-fashion. The scalar workload consists of instructions running only on the scalar datapath such as control instructions and address generations for local SIMD and scalar memories. The overhead workload consists of instructions to assist SIMD computations and scalar computations such as shuffle operations, predication operations, and data movements between the SIMD datapath and scalar datapath. The workloads of each category are calculated based on hand-written assembly codes and are weighted by dynamic execution frequency.

As can be seen in Table 5.1, most of the DSC signal processing algorithms have significant DLP. Exceptions are *Gamma Correction* and *Histogram Equalization*, where memory access patterns inhibit parallelization. The remaining DSC signal processing algorithms can be grouped into three categories.

(1) Pixel Independent Kernels: In this set of kernels, some basic arithmetic/logical and multiply-and-accumulate (MAC) operations are applied on every pixel independently. Therefore, these kernels can easily be mapped onto a SIMD architecture. *Black Clamp*, *Color Space Conversion*, *Brightness/Contrast Enhancement*, and *Hue/Saturation Enhancements* fall into this category. Although *Gamma Correction* is also a pixel-independent operation, this kernel cannot be easily parallelized on a SIMD architecture because each SIMD lane has to access different memory locations at the same time.

(2) Pixel Dependent Kernels: This set of kernels includes *CFA Interpolation*, *Edge Detection/Enhancement*, and *Spatial Frequency Filtering* that operate on pixels in a 2D neighborhood. The size of the 2D neighborhood is typically 3x3, though 5x5 or 7x7 sizes are also used. Traditional processor architectures spend more than half of the total instructions aligning the 2D data [21]. Therefore, for these kernels, 2D data access must be supported. This is done by a combination of multi-bank memory organization and a SIMD shuffle network.

(3) Statistics Gathering Kernels: The statistical information of the whole or partial frame is gathered for *White Balance*, *Auto Exposure*, *Histogram Calculation* and *Histogram Equalization*. Some of these kernels can be supported by a SIMD adder tree. *Histogram Calculation* is another kernel where memory access patterns inhibit parallelization for a SIMD architecture.

5.3 Diet SODA Architecture

In this section, we propose a power-efficient architecture, referred to as Diet SODA, for DSC processors. Diet SODA exploits key characteristics of the DSC image processing algorithms described in Section 5.2. This architecture operates in two modes: 1) dual voltage (DV) mode to handle low power applications such as the DSC image processing pipeline, and 2) the full voltage (FV) mode to handle advanced wireless communications. In DV mode, the memory operates at full voltage but the SIMD pipelines operate at near-threshold voltage. In FV mode, the SIMD data engines operate at full voltage as well.

5.3.1 Diet SODA PE Design

Figure 5.2 shows the architectural details of a single processing element (PE) of Diet SODA. The PE contains two different voltage domains: full voltage (FV) and dual voltage (DV). DV domain operates at either full or near-threshold supply voltage. The PE consists of: 1) multi-bank SIMD memory; 2) scalar memory; 3) SIMD data prefetcher; 4) SIMD pipeline; 5) scalar pipeline; and 6) 8-wide address generation unit (AGU) pipeline.

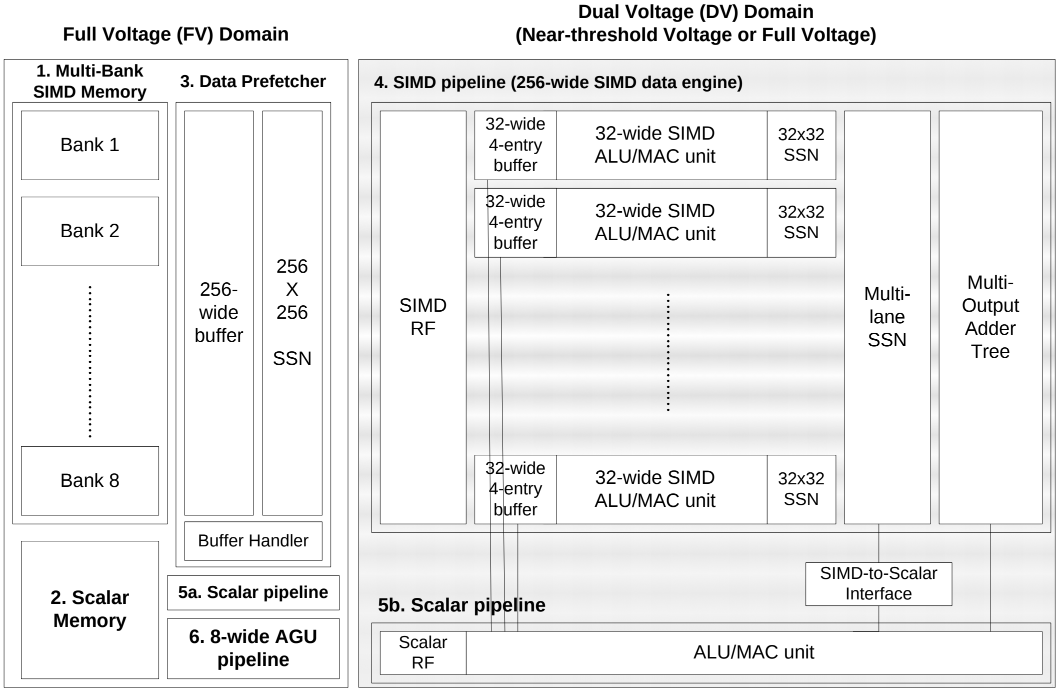


Figure 5.2: Diet SODA processing element (PE) for DSCs. The PE contains two different voltage domains: full voltage (FV) and dual voltage (DV). DV domain operates at either full or near-threshold supply voltage. The PE consists of: 1) multi-banked SIMD memory; 2) scalar memory; 3) SIMD data prefetcher; 4) SIMD pipeline; 5a) scalar pipeline in full voltage domain; 5b) scalar pipeline in dual voltage domain; and 6) 8-wide address generation unit (AGU) pipeline.

The SIMD pipeline consists of a 256-wide 16-bit datapath with a SIMD register file (RF), 256 functional units, 256 4-entry buffers used for intermediate data, a two stage SIMD shuffle network (SSN), and a multi-output adder tree. The SIMD datapath consists of eight groups of 32-wide SIMD units. With the support of a multi-banked SIMD memory and an 8-wide AGU pipeline, these groups of SIMD partitions can work on eight different memory sections concurrently. There are two scalar pipelines, one in each voltage domain; both pipelines consist of one 16-bit datapath and are used to perform sequential algorithms in addition to coordinating the SIMD units. The 8-wide AGU pipeline handles memory address calculation for the 8-bank SIMD memory and the data prefetcher.

5.3.2 SIMD Pipeline Width

Although near-threshold operation allows circuits to consume significantly less power, the processing performance also degrades. To compensate for the degraded performance, the number of SIMD lanes is increased.

The DSC signal processing pipeline for a VGA-size (640x480) image and a full-HD

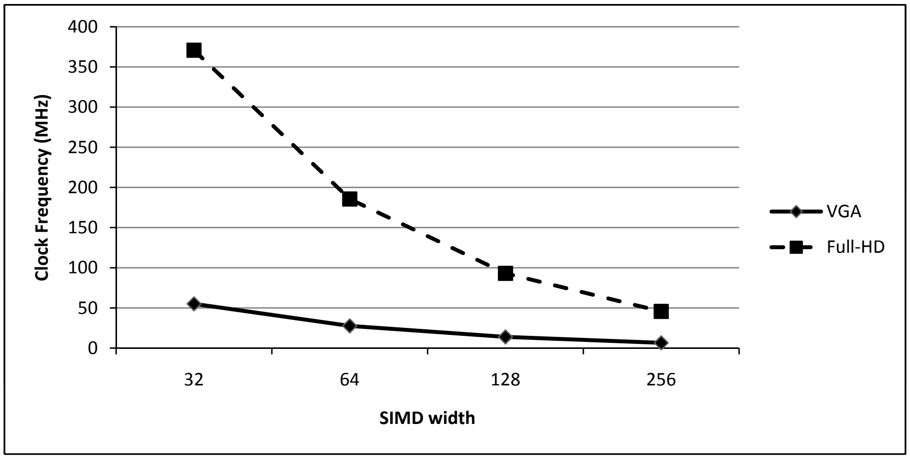


Figure 5.3: Minimum clock frequencies based on different SIMD width configurations to run the preview mode of DSC signal processing pipeline shown in Figure 5.1.

(1920x1080) image at 30 fps is used as the evaluation point to decide the number of SIMD lanes. Figure 5.3 shows the minimum clock frequency required for VGA and full-HD for different SIMD width configurations — 32, 64, 128, and 256. Thus, to process full-HD images at 30 fps, a 32-wide SIMD pipeline needs to operate at more than 370MHz, while a 256-wide SIMD pipeline needs to operate at around 50 MHz.

To investigate how much voltage/frequency scaling can be achieved while still meeting the performance requirements, the power consumption for each SIMD width configuration was measured. First, a representative test circuit was laid out in IBM 90nm technology, parasitic extraction was performed and annotated. Then, SPICE simulations were done to determine the voltage, frequency, and power characteristics at different supply voltages. To obtain power numbers, the PE logic was then synthesized with Synopsys Physical Compiler and scaled to match the representative test circuit. Figure 5.4 shows power consumption and achievable clock frequencies depending on the corresponding supply voltage for each candidate SIMD width. The solid vertical lines provide guidelines on what the minimum supply voltage is required to process VGA and full-HD images at 30 fps. The results show that although a 32-wide SIMD data engine is capable of handling VGA processing requirements, to support full-HD images, a SIMD width of greater than 32 is required. On the other hand, wider SIMD widths do not always guarantee better energy efficiency due to the additional hardware and critical path delay increases, resulting in a higher minimum clock frequency. In this paper, a 256-wide SIMD configuration is chosen to maximize the benefit of using near-threshold operation while maintaining the real time processing constraints of both VGA and full-HD. With this configuration, the supply voltage needs to be 600mV using a clock frequency of 50MHz.

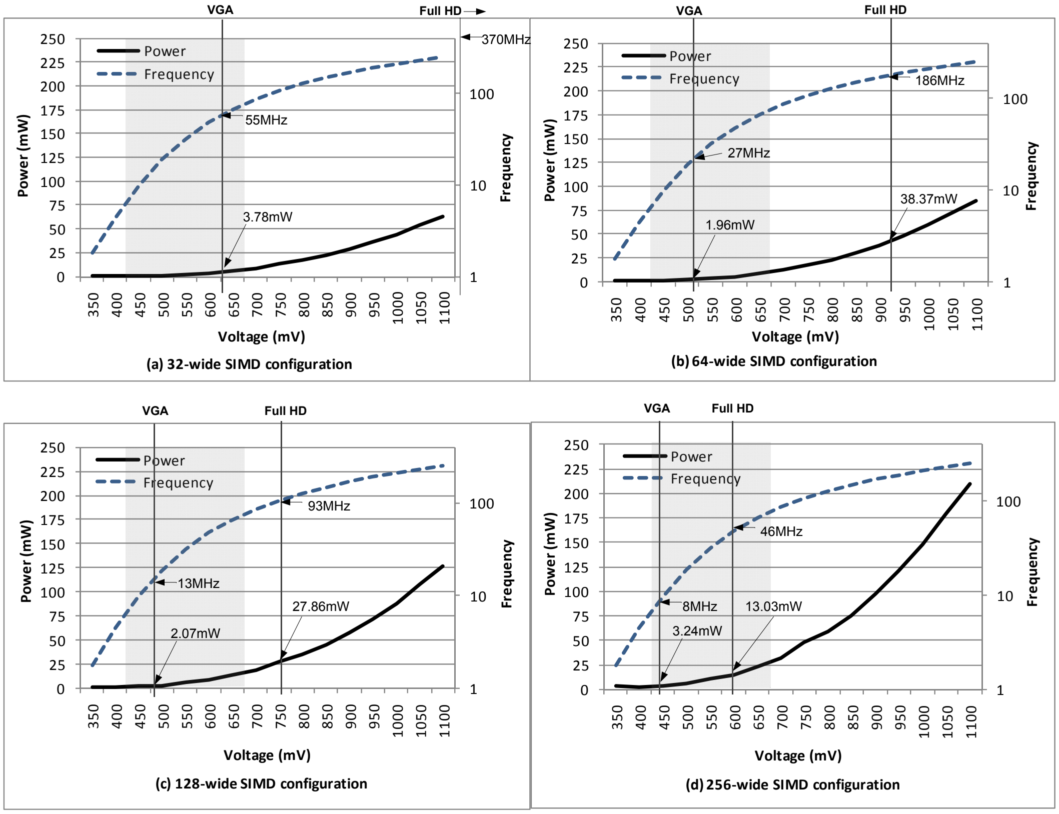


Figure 5.4: Near-threshold operation is applied to four different SIMD width configurations: 32, 64, 128, and 256. Solid vertical lines provide guidelines for the minimum supply voltage necessary to meet VGA and full-HD processing demands. Gray boxes represent the near-threshold regions.

5.3.3 Scatter-Gather Data Prefetcher

While operating in the DV mode, the SIMD memory operates significantly faster than the SIMD pipeline. Therefore, two-dimensional (2D) data accesses can be achieved by performing multiple memory accesses to the same memory banks in a single cycle of the SIMD pipeline. There is also sufficient time to perform complicated shuffling operations before delivering the data. Because of these non-traditional memory access patterns and additional shuffling capabilities, a significant reduction in the required number of SIMD instructions can be obtained.

Figure 5.5 shows the process of data alignment using the SIMD data prefetcher. First, the required data is read from a multi-bank memory. Then the data prefetcher stacks the data in the location indicated by the data prefetcher pointer. The pointer then advances to the next data section and repeats the process for the next load operation. In addition, with the support of SSN, more complex shuffle operations can be implemented.

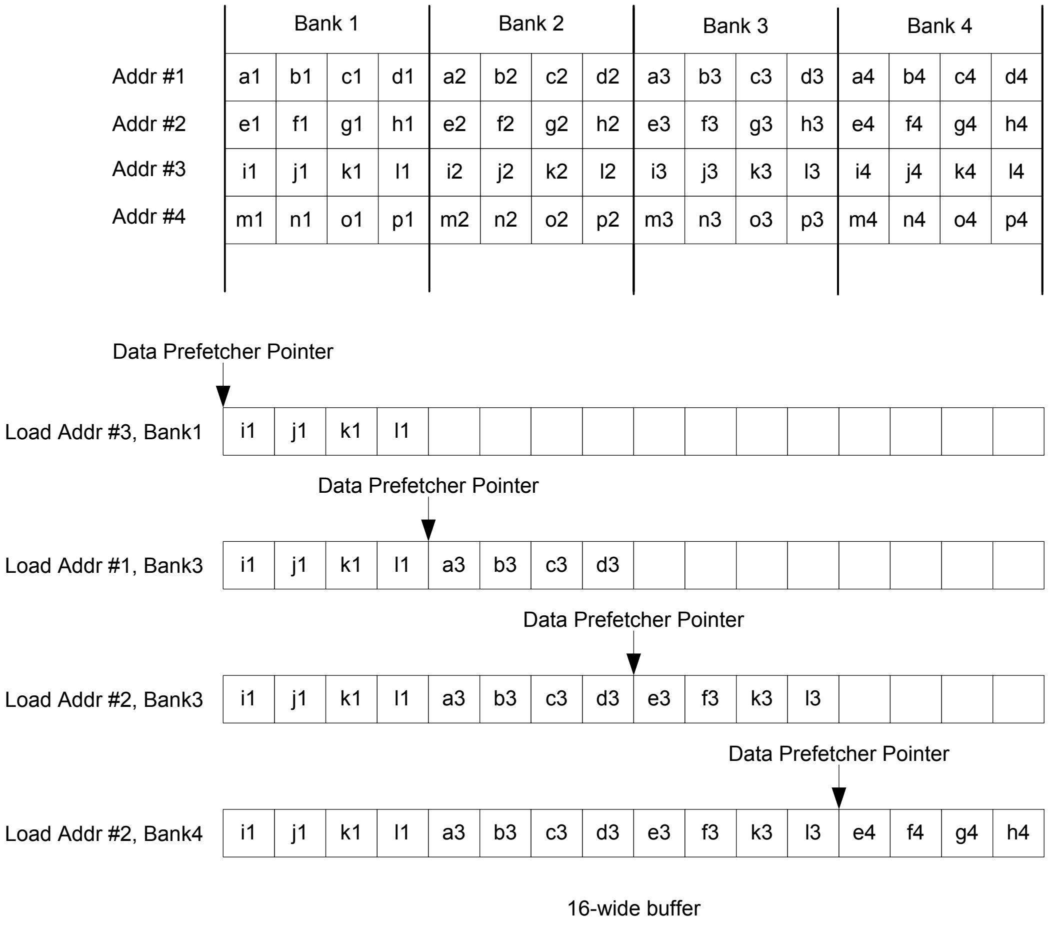

Bank 1
Bank 2
Bank 3
Bank 4
Addr #1 a1 b1 c1 d1 a2 b2 c2 d2 a3 b3 c3 d3 a4 b4 c4 d4
Addr #2 e1 f1 g1 h1 e2 f2 g2 h2 e3 f3 g3 h3 e4 f4 g4 h4
Addr #3 i1 j1 k1 l1 i2 j2 k2 l2 i3 j3 k3 l3 i4 j4 k4 l4
Addr #4 m1 n1 o1 p1 m2 n2 o2 p2 m3 n3 o3 p3 m4 n4 o4 p4
Data Prefetcher Pointer
Load Addr #3, Bank1 i1 j1 k1 l1
Data Prefetcher Pointer
Load Addr #1, Bank3 i1 j1 k1 l1 a3 b3 c3 d3
Data Prefetcher Pointer
Load Addr #2, Bank3 i1 j1 k1 l1 a3 b3 c3 d3 e3 f3 k3 l3
Data Prefetcher Pointer
Load Addr #2, Bank4 i1 j1 k1 l1 a3 b3 c3 d3 e3 f3 k3 l3 e4 f4 g4 h4
16-wide buffer

Figure 5.5: Example of complex data shuffling with 4-bank 4-wide SIMD memory, SIMD data prefetcher, and 16-wide buffer.

5.3.4 Operating Modes

In this section, dual voltage (DV) and full voltage (FV) modes in Diet SODA are described. Table 5.2 provides the configuration of each component of Diet SODA PE for each mode.

5.3.4.1 DV Mode

In the DSC signal processing pipeline, preview and picture-taking tasks are performed in DV mode because these tasks do not require very high data processing rates. Consequently, the supply voltage of the SIMD data engine is operated at near-threshold voltage to significantly lower energy consumption. More specifically, the SIMD pipeline and scalar pipeline (5b in Figure 5.2) in the DV domain operate at near-threshold voltage, while the SIMD memory, scalar memory, SIMD data prefetcher, and 8-wide AGU pipeline operate

	Components	DV Mode	FV Mode
PE	1. Multi-Bank SIMD Memory	on	on
	2. Scalar Memory	on	on
	3. Data Prefetcher – Buffer & Buffer Handler	on	off
	3. Data Prefetcher - SSN	on	on
	4. SIMD pipeline – SIMD RF	off	on
	4. SIMD pipeline – Other modules except SIMD RF	on @ NTV	on
	5a. Scalar pipeline	on	off
	5b. Scalar pipeline	on @ NTV	on
	6. 8-wide AGU pipeline	on	on

Table 5.2: Architectural modules that are turned on and off for dual voltage (DV) and full voltage (FV) modes.

at full voltage. As can be seen in Table 5.2, the SIMD RF is switched off because the latency of the SIMD memory is much lower than that of SIMD data engine so the SIMD pipeline is capable of directly handling data from the SIMD memory. This results in a reduction of energy consumption by eliminating SIMD RF accesses. The 4-entry buffer in each SIMD lane operates as a small RF to hold recently produced values for consumption by subsequent instructions.

5.3.4.2 FV Mode

Recent DSCs support video recording at full-HD (1920x1080) resolution. This necessitates additional processing capability, and therefore requires Diet SODA to operate in FV mode. In this mode, the SIMD pipeline operates at full voltage along with the SIMD memory and 8-wide AGU pipeline. On the other hand, the SIMD data prefetcher is turned off because there is no time slack between the SIMD memory and the SIMD data engine to prefetch data in advance. Also, the scalar pipeline (5a in Figure 5.2) in the FV domain is turned off and another scalar pipeline (5b in Figure 5.2) in the DV domain works for the overall system. In this mode, the SIMD RF is switched on so that faster operations are supported.

5.3.5 Mapping Example: CFA Interpolation

This section describes how one of the key DSC algorithms, *CFA interpolation*, is mapped onto Diet SODA. The edge-directed CFA interpolation compares horizontal gradients and vertical gradients, and the interpolation method is chosen as described in Figure 5.6-(a). In Diet SODA, the data prefetcher loads Addr#1 and Addr#3 from SIMD memory, and shuffles them to obtain *v*0 by using the data prefetcher and SSN, Figure 5.6-

(b).

The aligned $v0$ is then used to calculate the vertical gradients ($v2$), Figure 5.6-(c). Similarly, Addr#2 is loaded and shuffled to obtain $v1$ which is used to calculate the horizontal gradients ($v3$). With these SIMD registers ($v2$ and $v3$), the edge-directed interpolation is easily implemented. For example, based on the comparison of the first elements of the two vectors ($b1 - j1$=ΔH and $e1 - g1$=ΔV), the second elements of the vectors are shifted ($(b1 + j1) >> 1$, $(e1 + g1) >> 1$) or add-shifted ($((b1 + j1) + (e1 + g1)) >> 2$). This series of operations rely upon the data prefetcher and SSN in FV domain and the SIMD pipeline in DV domain.

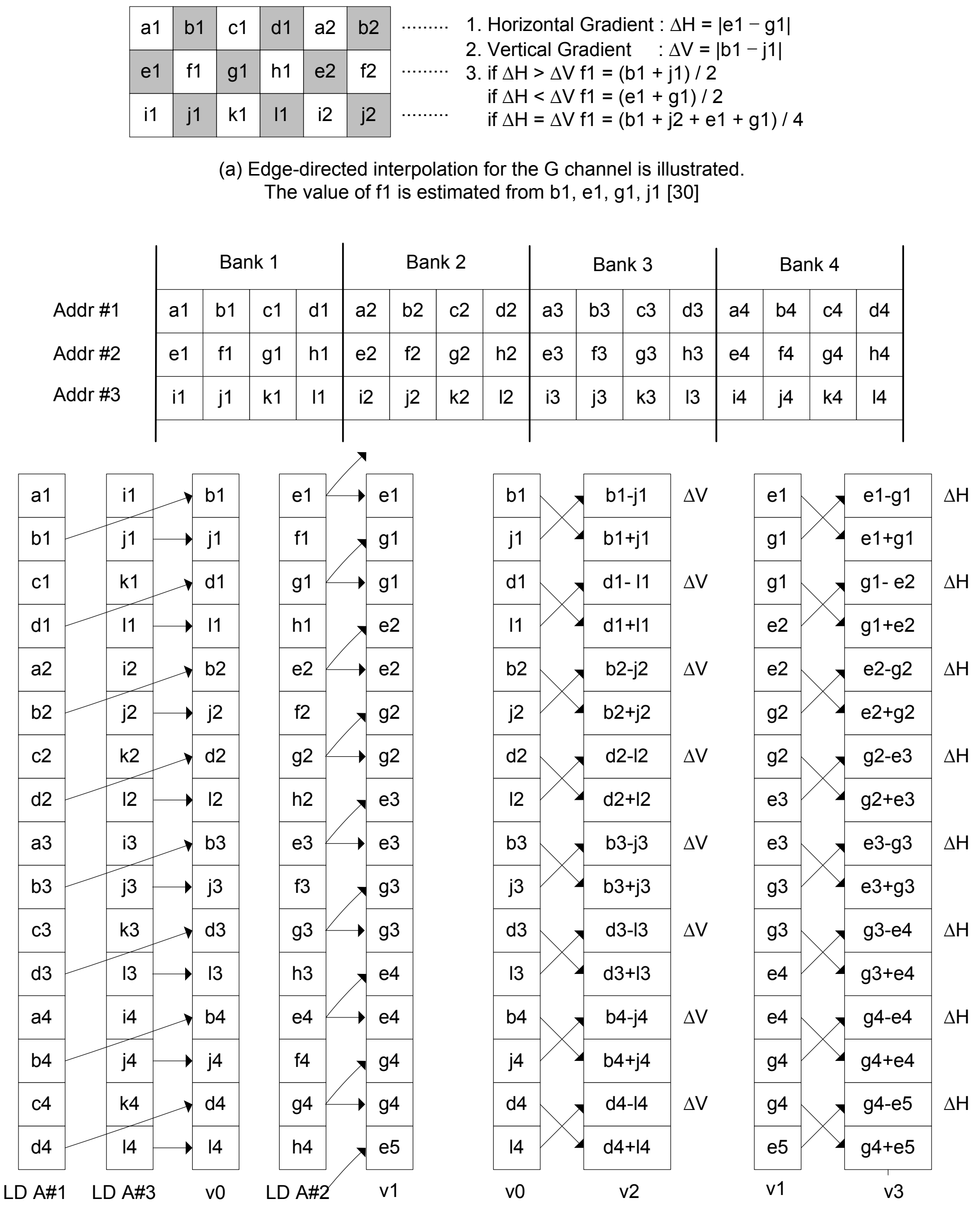
Thus, the data prefetcher module aligns data for the SIMD data-path. Also the vector for the horizontal gradients ($v3$) are prepared with $v1$, which are loaded from Addr#2 and shuffled. Therefore, this CFA interpolation process is expedited by data prefetcher module to align data for the SIMD datapath.

5.3.5.1 3x3 Convolution

A 3x3 convolution (5.1) is one of the commonly used algorithm in DSC signal processing pipeline such as edge detection and edge enhancement. There are many ways to program this algorithm on Diet SODA and here, Figure 5.7 shows one of the methods how to map 3x3 convolution on Diet SODA.

$$\begin{bmatrix} a_1 & a_4 & a_7 \\ a_2 & a_5 & a_8 \\ a_3 & a_6 & a_9 \end{bmatrix} * \begin{bmatrix} b_1 & b_4 & b_7 \\ b_2 & b_5 & b_8 \\ b_3 & b_6 & b_9 \end{bmatrix} = \sum_{k=1}^{9} a_k * b_k \tag{5.1}$$

The three pixel rows are loaded into three SIMD registers: $v1$, $v2$, and $v3$. The 3x3 mask values are loaded into nine scalar registers. The $v1$ is multiplied by c_{11}, and then $v1$ is shuffle down by 1 and multiplied by c_{12}, and $v1$ is shuffled down by 1 again and multiplied by c_{13}. These three multiplied SIMD registers are added, then $v4$ contains inner-product of the first rows. The $v2$ and $v3$ are process in the same way to generate $v5$ and $v6$. The SIMD addition of $v4$, $v5$, and $v6$ generate the final convolution values per each lane. For example, the first element is $\sum_{k=1}^{9} a_k * b_k$, and the second element is $\sum_{k=4}^{12} a_k * b_k$. In this way, 3x3 convolution process is parallelizable.


1. Horizontal Gradient : ΔH = |e1 − g1|
2. Vertical Gradient : ΔV = |b1 − j1|
3. if ΔH > ΔV f1 = (b1 + j1) / 2
if ΔH < ΔV f1 = (e1 + g1) / 2
if ΔH = ΔV f1 = (b1 + j2 + e1 + g1) / 4
Bank 1
Bank 2
Bank 3
Bank 4
Addr #1
Addr #2
Addr #3
LD A#1
LD A#3
v0
LD A#2
v1
v0
v2
v1
v3
ΔV
ΔH

(a) Edge-directed interpolation for the G channel is illustrated.
The value of f1 is estimated from b1, e1, g1, j1 [30]

(b) Data alignment by data prefetcher and SSN

(c) SIMD ALU operations in SIMD datapath

Figure 5.6: An Edge-directed CFA interpolation mapped on Diet SODA.

5.4 Results and Analysis

5.4.1 Methodology

The DSC image signal processing pipeline algorithms are implemented in C to evaluate system performance, memory requirements, and non-parallelizable bottlenecks. Next, the C benchmark codes [72] are transformed to assembly codes for Diet SODA. The Diet SODA processor is implemented as an RTL Verilog model and synthesized for IBM's 90nm

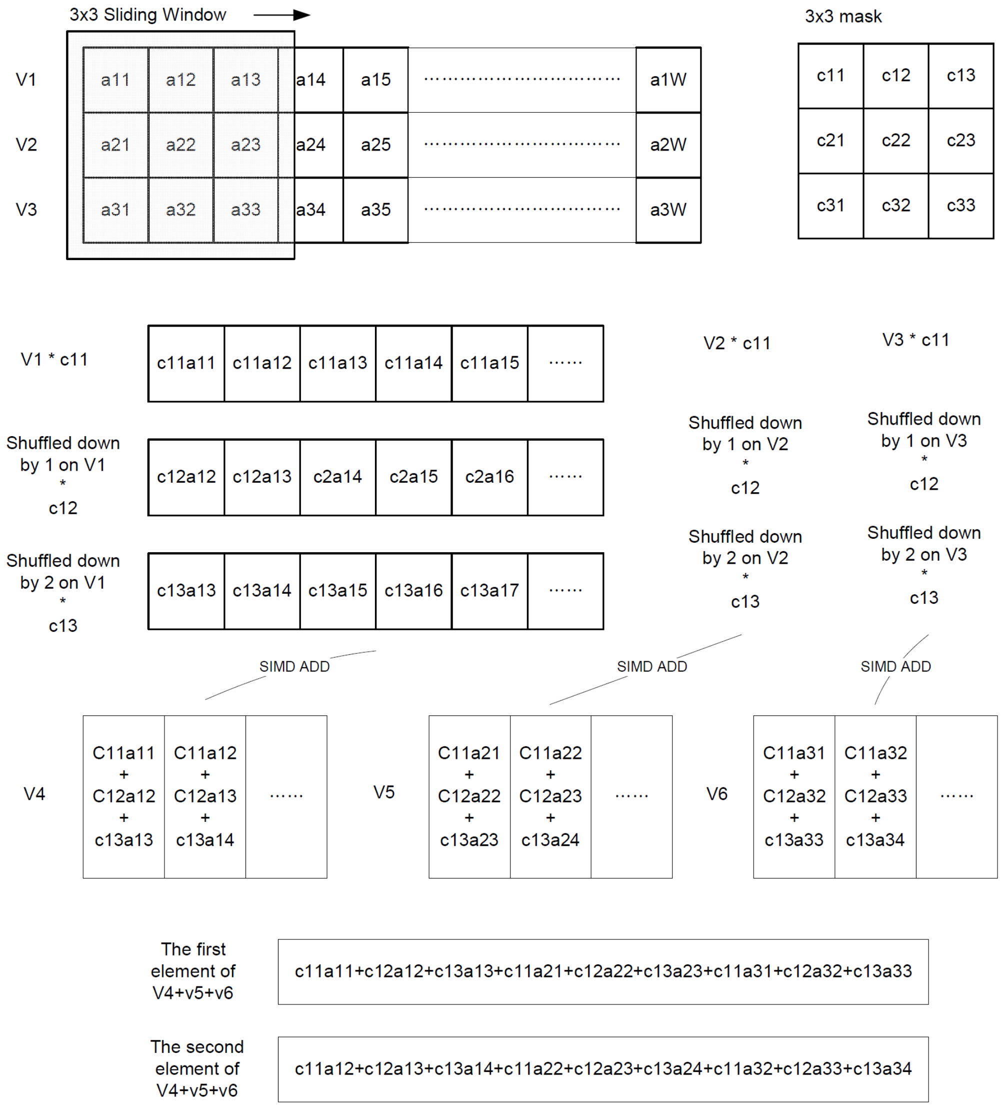

3x3 Sliding Window
V1
V2
V3
a11 a12 a13 a14 a15 .... a1W
a21 a22 a23 a24 a25 .... a2W
a31 a32 a33 a34 a35 .... a3W
3x3 mask
c11 c12 c13
c21 c22 c23
c31 c32 c33
V1 * c11
c11a11 c11a12 c11a13 c11a14 c11a15 ......
Shuffled down by 1 on V1 * c12
c12a12 c12a13 c2a14 c2a15 c2a16 ......
Shuffled down by 2 on V1 * c13
c13a13 c13a14 c13a15 c13a16 c13a17 ......
V2 * c11
Shuffled down by 1 on V2 * c12
Shuffled down by 2 on V2 * c13
V3 * c11
Shuffled down by 1 on V3 * c12
Shuffled down by 2 on V3 * c13
SIMD ADD
SIMD ADD
SIMD ADD
V4
C11a11 + C12a12 + c13a13
C11a12 + C12a13 + c13a14
......
V5
C11a21 + C12a22 + c13a23
C11a22 + C12a23 + c13a24
......
V6
C11a31 + C12a32 + c13a33
C11a32 + C12a33 + c13a34
......
The first element of V4+v5+v6
c11a11+c12a12+c13a13+c11a21+c12a22+c13a23+c11a31+c12a32+c13a33
The second element of V4+v5+v6
c11a12+c12a13+c13a14+c11a22+c12a23+c13a24+c11a32+c12a33+c13a34

Figure 5.7: A 3x3 Convolution operation mapped on Diet SODA. A 3x3 convolution mask is applied to 3x3 pixels.

technology using the Synopsys Physical Compiler. The clock frequency is targeted for 400MHz @ 1V, and the power numbers for the SIMD data engine are scaled down for 50MHz @ 600mV by the process shown in Section 5.3.2.

5.4.2 Area and Power

The area and power breakdown of this processor are presented in Table 5.3. The preview mode of full-HD images at 30 fps consumes about 70mW and 246mW in DV mode and FV mode, respectively. Therefore, the DV mode offers about 3.5X better power efficiency.

				Dual Voltage Mode		Full Voltage Mode	
	Components	Area mm^2	Area %	Power mW	Power %	Power mW	Power %
PE	SIMD Data eDRAM (128KB)	1.04	8 %	32	46 %	17	7 %
	SIMD Register File (8KB)	3.17	23 %	-	-	54	22 %
	SIMD ALUs, Multipliers, and SSN	4.50	32 %	5	7 %	81	33 %
	SIMD Pipeline+Clock+Routing	1.42	10 %	2	3 %	42	17 %
	SIMD Buffer (2KB)	0.58	4 %	4	6 %	27	11 %
	SIMD Adder Tree	0.49	4 %	< 1	< 1 %	1	< 1 %
	Data Prefetcher	1.13	8 %	3	4 %	-	-
	Scalar/AGU Pipeline & Misc.	1.53	11 %	24	34 %	24	10 %
Total	90nm (1V@400MHz, 600mV@50MHz)	**13.85**	100 %	**70**	100 %	**246**	100 %

Table 5.3: Area and Power Summary of Diet SODA for Preview Mode of Full-HD Images at 30 fps. For comparison, the results of both DV mode and FV mode are presented. The eDRAM proposed in [74] are used for SIMD local memory.

About 80% of the total power dissipation in DV mode is consumed by SIMD memory and the scalar/AGU pipeline operating at full voltage. In particular, the SIMD memory consumes a large part of the power because the number of SIMD memory accesses is increased due to the SIMD RF being switched off. However, the SIMD data engine operating in DV mode consumes 18X less power than the SIMD datapath operating in FV mode, which offsets the increased SIMD memory power and highlights the advantage of using near-threshold operation.

5.4.3 Performance

Table 5.4 presents the latencies of DSC processing algorithms - preprocessing (*Black Clamping*, *Lens Distortion Compensation*, *Fault Pixel Correction*, *White Balance*, *Gamma Correction*), *CFA Interpolation*, *Color Space Conversion*, *Edge Detection/Enhancement*, *Scaling*, and *JPEG Compression*. As can be seen in Table 5.4, the preview modes of both VGA and Full-HD images are processed within a time constraint of 33 ms thus meeting the 30 fps requirement.

CFA Interpolation and *Edge Detection/Enhancement* are the most demanding workloads taking about 60% of the processing time. While most algorithms deals with only one color component (R, G, or B) per each pixel location, *CFA interpolation* generates all of three components for each pixel locations. Therefore, the memory size and workload for this interpolation algorithm are increased. *Edge Detection/Enhancement* works with only one component per each pixel location, but 3x3 matrix convolutions in this task require significant processing time and shuffling for MAC calculations and realignments.

Task	**Latency (VGA)**	**Latency (Full-HD)**
Black Clamp, Distortion Compensation, Fault Pixel Correction, White Balance, Gamma Correction	0.57 ms	3.89 ms
CFA Interpolation	1.02 ms	6.67 ms
Color Conversion	0.36 ms	2.43 ms
Edge Detection Edge Enhancement	0.82 ms	5.29 ms
False Color Suppression Scaling	0.31 ms	2.11 ms
Total	**3.08 ms**	**20.38 ms**

Table 5.4: The Latencies of DSC signal processing pipeline algorithms for the preview mode of a VGA image and a Full-HD image.

5.4.4 Comparison With Other Solutions

The DSC image signal processing pipeline in Figure 5.8 [21] is used to compare the performance of Diet SODA with one high-end commercial DSP and one coarse-grained reconfigurable image stream processor — TI TMS320C64x [1] and CRISP [21]. The pipeline is divided into three task groups: 1) color gain adjustment, gamma correction and CFA interpolation; 2) noise reduction and smooth filter; and 3) color space conversion and edge enhancement.

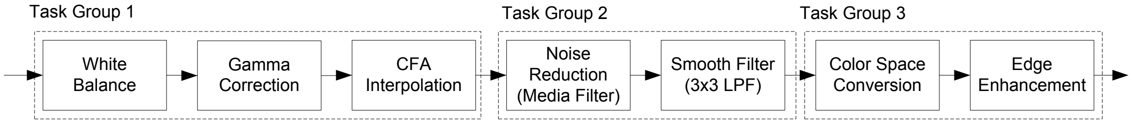


Figure 5.8: A Test DSC image signal pipeline [21]

Table 5.5 shows the execution time comparison with TMS320C64x, CRISP, and Diet SODA for a 4072x2720 image. Results show that Diet SODA is approximately 140X and 1.6X faster than TMS320C64x and CRISP, respectively. The wide SIMD datapath allows the DSC image signal processing algorithms to operate on many pixels at the same time. In addition, scatter-gather data prefetcher helps data alignment issues.

Table 5.6 shows comparisons of technology, area, power consumption, and normalized energy with TMS320C64x and CRISP. Normalized power and area results for 90nm technology are estimated using a quadratic scaling factor based on Predictive Technology Model [43]. The results show that the energy efficiency of Diet SODA is more than 500X better than that of TMS320C64x and also comparable to that of the reconfigurable image

	TMS320C64x [1]	**CRISP** [21]	***Diet SODA PE**
Task Group 1	6440 ms	220 ms	110 ms
Task Group 2	20550 ms	110 ms	80 ms
Task Group 3	9690 ms	110 ms	80 ms
Total	36680 ms	440 ms	270 ms

Table 5.5: Execution Time Comparison with TI TMS320C64x, CRISP, and Diet SODA. Task Group 1 - White Balance, Gamma Correction, CFA Interpolation; Task Group 2 - Noise Reduction, Smooth Filter; Task Group 3 - Color Space Conversion, Edge Enhancement. *Diet SODA operates in DV mode.

stream processor, CRISP [21]. Even though we show comparable energy and only slightly improved performance numbers over the CRISP design, our design is more flexible and maintainable because the reconfigurable interconnection and heterogeneous processing elements of CRISP must be manually designed before fabrication.

	TMS320C64x [1]	**CRISP** [21]	**Diet SODA PE**
Tech.	0.13um	0.18um	90nm
Freq.	600MHz	115MHz	400MHz,50MHz
Power	718mW@1.2V	218mW@1.8V	69.6mW@DV**
Area*	34.5mm^2	1.9mm^2	10.4mm^2
Energy*	11k	9.3	18.8

Table 5.6: Chip Statistics and Energy Comparison with TI TMS320C64x, CRISP and Diet SODA. *Area and energy are normalized to 90nm technology. **Diet SODA operates in DV mode - 1V and 600mV.

5.5 Conclusion

In this chapter, a programmable substrate for an ultra-low power signal processor using near-threshold operation was proposed. Near-threshold operation reduces energy but suffers from degraded performance, but this can be overcome by using parallelism. DSC algorithms on SIMD architectures offer an abundant amount of data level parallelism, forming a natural synergy with near-threshold operation. In addition, because memory systems operate at faster rates than SIMD data engines in near-threshold operation mode, scatter-gather prefetcher was introduced to exploit latency difference and lower instruction counts. Diet SODA also uses a dual voltage mode to increase performance for kernels that requires high processing power. Our results show that Diet SODA with a 256-lane SIMD unit operating at 600mV and 50MHz in an IBM 90nm technology can meet the processing requirements of full HD resolution at 30 fps while consuming only 70mW. This is on the order of 130X better performance and approximately 500X better energy efficiency over a DSP solution, and provides a more flexible solution than equivalently powered ASIC designs.

CHAPTER 6

Voltage Boosting

While traditional designs can reduce voltage to save power, their scaling range is limited because they are optimized to maximize performance at high nominal voltages. On the other hand, parallel systems optimized for NTC operation cannot achieve the same single thread performance as traditional designs. Analysis shows that systems designed for NTC perform, on average, with 40% better energy-efficiency. In this chapter, we explore low-voltage system designs that use voltage boosting—raising the voltage above nominal—to provide greater single-thread performance when it is needed. Specifically, the use of dual-V_{dd} power distribution is explored to enable fast boosting—switching in 10s of cycles—to overcome serial bottlenecks. Furthermore, dynamic frequency and voltage scaling can be leveraged to adjust the nominal/boosted voltage ratio to adapt to long-term variations in program behavior over 10,000s of cycles.

Through two case studies, it is demonstrated that boosting helps overcome parallelization limitations in transactional memory programming and throughput computing. In transactional memory, boosting mitigates penalties from high conflict rates, achieving up to a 50% performance improvement (12% on average) over a conventional system with the same thermal design power. In throughput computing, results show that a near threshold boosted system can match the 99^{th}-percentile request latencies of a full-voltage non-boosted system while operating at a at 70% lower power.

The work in this chapter was done with the collaboration of my colleagues Geoffrey Blake, David Meisner, and Dave Fick.

6.1 Introduction

Typically, designers of an energy-efficient system begin by selecting a design optimized for high-performance operation and reduce power consumption by lowering its voltage—traditional dynamic frequency/voltage scaling (DVFS). This design approach enables the best performance at high voltage; however, this approach limits how far operating voltage can be lowered due to leakage energy and increased variability, providing only marginal energy savings opportunity.

Instead, we advocate a design approach where a system is optimized primarily for low voltage operation. Under this approach, leakage and variability become key design constraints, and can be addressed through transistor sizing and other circuit techniques. Chapter 4 argues that multicore parallelism might be used to overcome the degraded performance of low voltage operation, as the lower supply voltage allows many more cores within a fixed thermal design power (TDP) budget—the maximum power the cooling system is designed to dissipate—while achieving far greater energy-efficiency per core. This chapter will show that designs targeted for low-voltage operation are, on average, 40% more energy efficient than conventional designs. Unfortunately, this approach comes at the expense of the high single-thread performance achievable by conventional high-voltage designs.

The underlying premise of the NTC approach is that workloads can be effectively parallelized. However, in some cases, single thread performance may be needed to overcome "Amdahl bottlenecks" in applications. These bottlenecks could arise from inherently serial code regions, lock contention, and/or communication overheads. Single-thread performance might also be needed to meet latency targets for tasks that fall in the long tail characteristic of many server task-size distributions. To mitigate these Amdahl bottlenecks in NTC architectures, this chapter introduce a design for voltage boosting that exploits the use of dual V_{dd} rails to allow cores to switch frequencies in 10's of cycles—significantly less than the 10,000's of cycles required for traditional DVFS—in order to respond to short-lived bottlenecks. Our key contributions are:

- *How to boost*
  - Circuit designs for voltage boosting
  - Analysis of voltage boosting architectures
- *When to boost*
  - Overcoming transactional memory bottlenecks
  - Reducing tail response time in server workloads

Our results show that boosting can overcome bottlenecks introduced by conflicts in transactional memory, and can achieve up to a 50% performance improvement, 12% on average, over a non-boosting system. A second analysis shows that in web workloads, where request latency is critical, the 99^{th}-percentile response time can be reduced by 35%

with the use of voltage boosting, which has a TDP of 70% less than a traditional full-voltage system with a matching 99^{th}-percentile latency.

6.2 NTC Impact on Single Thread Performance

This section gives detail on reasons why single thread performance is still important in NTC systems with high degrees of parallelism, like those discussed in Chapter 4.

6.2.1 Parallelization Complications

The NTC approach assumes that the application can be efficiently parallelized. This assumption has many complications, even for systems like web servers where abundant parallel work exists. This chapter explores three main problems: (1) Amdahl bottlenecks, or serial/communication portions of code that limit scalability of parallel systems; (2) the difficulty of programming parallel applications; and (3) the increase in latency created by parallelization.

6.2.1.1 Amdahl Bottlenecks

When code is parallelized, there are often regions that are inherently serial. During these regions, many cores may be stalled waiting for another core. For example, cores may stall at a barrier waiting for the last-arriving core. Another example is in producer-consumer relationships, where the consumer may stall until the producer supplies data. A third example arises in lock-based code, where several cores may contend for the same lock. These will be referred to as “Amdahl bottlenecks”.

Figure 6.1(a) shows a situation where *CPU 0* creates an Amdahl bottleneck. In this case *CPU 0* has a critical section (or lock) that is shown in pink. As additional cores *CPU 1-4* run out of parallel work (or stall on the same lock) they idle waiting for *CPU 0* to complete. The *Cn* measurement at the top shows how many cores are stalled due to the critical section as time progresses. Figure 6.1(b) shows that if the critical section is boosted to run faster, it can reduce the latency of the other cores and the total idle time of the system. In Section 6.5 a transactional memory system with such bottlenecks will be analyzed, and results will show that boosting improves performance by as much as 50% in workloads with excessive bottlenecks.

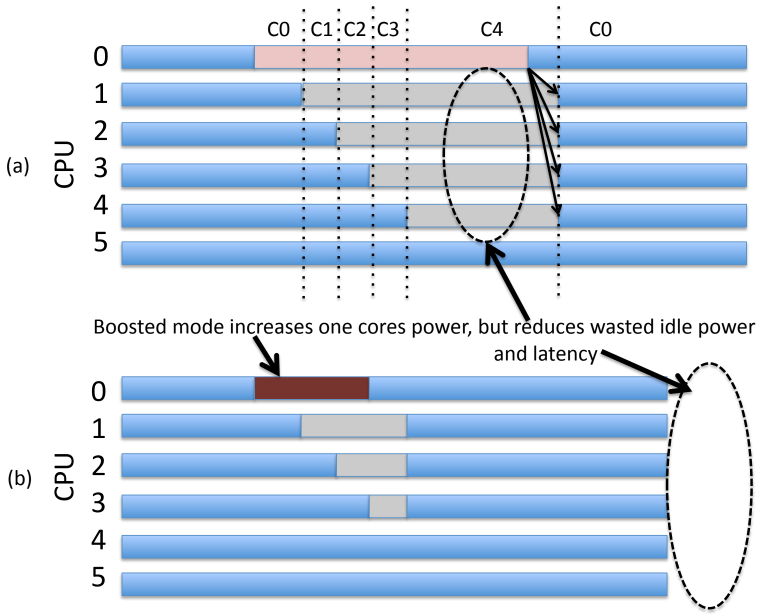

C0
C1
C2
C3
C4
C0
0
1
2
3
4
5
(a)
CPU
Boosted mode increases one cores power, but reduces wasted idle power and latency
0
1
2
3
4
5
(b)
CPU

Figure 6.1: (a) Amdahl bottleneck in a system and how boosting (b) can be used to improve the system.

6.2.1.2 Programability

Another major problem with parallel code is that it is difficult to write and verify. In lock-based systems, critical sections are protected using lock variables. Careful ordering is needed in acquiring these locks to avoid deadlock. In addition, programmers are required to use barrier and fence operations in weakly consistent memory systems to ensure proper code behavior. Since errors may only appear in certain interleavings of the parallel code, it is difficult to verify such systems. Recent work in the area of transactional memory [75] has shown that by using transactions, students can create code with less errors. However, these novice transactional programmers may create large transactions with a high probability of conflicts with other transactions. In the cases where contention is high, parallelism is hindered as transactions continually abort and restart. The net result is a system that consumes more energy, and achieves only moderate gains in performance even with substantially more cores. In Section 6.5 a study will show how voltage boosting can overcome these high conflict workloads and improve performance, while providing the programmability benefits of transactional memory.

6.2.1.3 Latency and Response Time

Even systems with inherently large amounts of parallelism still may be hindered by parallelization. For example request-parallel systems, such as web servers, serve multiple independent requests concurrently. Figure 6.2 shows the impact of parallelism on throughput and latency (response time) in SpecWeb99.

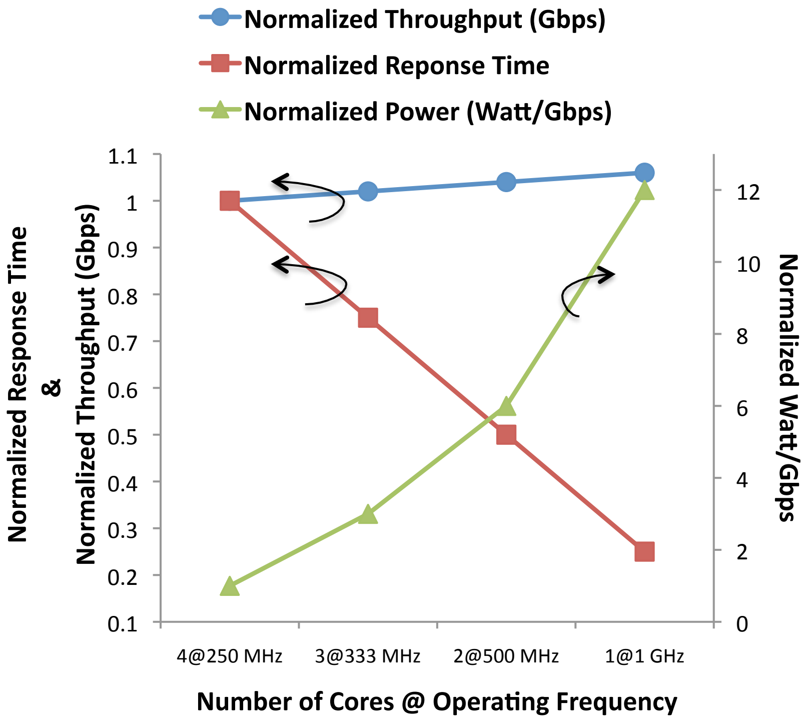


Figure 6.2: SpecWeb99 latency, throughput, and power for various parallel operating points.

As shown, for small numbers of cores the system has a low latency, but the cores are clocked at high frequency consuming larger amounts of power. For systems with higher degrees of parallelism, the same throughput can be achieved at lower power levels, however the latency of a single request becomes larger. If the system is subject to a per-request response time target, performance can be reduced only until the deadlines are just met. In Section 6.6, a web server workload will be analyzed to show that system TDP can be reduced and boosting can help reign in the 99^{th}-percentile latencies, reducing variation in the response time.

6.3 Circuit Techniques

Voltage boosting—raising the voltage above nominal—can allow systems designed for energy-efficient operation to operate at higher frequency to overcome Amdahl bottlenecks or reduce latency. For the system to support boosting, there are several design considerations. First, when boosted, the system can not exceed TDP. Second, the boosting capability should minimally impact the performance of the system at normal operation. And finally, the area overhead on the system should be kept low.

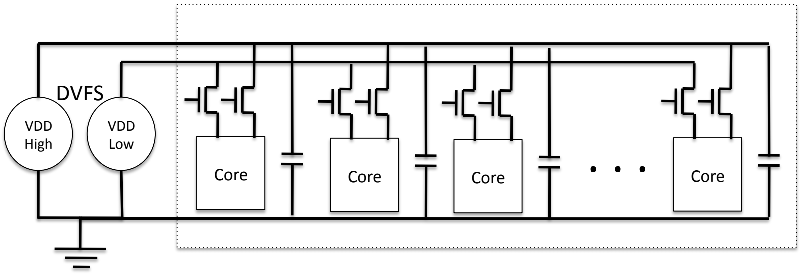


Figure 6.3: Dual-V$_{dd}$ chip. Each core provides decap on high voltage rail to allow a single core to boost in 10's of cycles. DVFS can be used on external power supplies to adjust degree of boosting over longer time frames. Area overhead of decap and power transistors is 5-10% for 16 cores. Either an additional metal layer is needed to route the second power grid, or an additional 10% area overhead[10,22].

6.3.1 Boosting via Dual-V$_{dd}$

To meet these objectives, this chapter proposes implementing voltage boosting via dual power supplies, as illustrated in Figure 6.3. A dual-V$_{dd}$ system has two distribution networks for V$_{dd}$ supplied by two independent external voltage sources, high and low. Each core in the system is connected via multiple power transistors (shown as a single transistor in the diagram) to each of the voltage rails. In addition, decoupling capacitors connect in parallel with each core on the high voltage network, improving the stability of the network during voltage switches. Dual-V$_{dd}$ systems have been proposed in the past [89, 45, 56], but differ from this approach in the scope and speed of switching as well as the use of decoupling capacitors.

In normal operation all the cores are connected to the low voltage rail. To boost performance, a single core is switched from the low rail to the high rail. Only a single core may switch at a time to limit voltage droop on the high rail. The key insight is that the total decoupling capacitance presented by the cores is much larger than that of a single core.

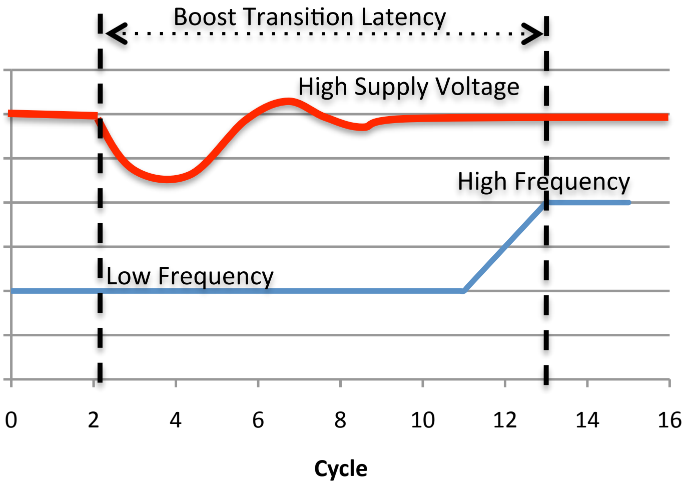


Figure 6.4: Boost transition. When boosting occurs, voltage droops on the high supply, core remains at low frequency. Once stable the frequency changes.

Thus, if only one core switches, then the voltage droop on the supply is small and recovery is fast. The number of cores that may be powered off the high rail is limited either by the TDP of the chip, or the current delivery capacity of the high-voltage rail.

Durring transitions, the high voltage net will droop (see Figure 6.4). During the droop, all cores on the high rail must operate at the low frequency to avoid setup and hold time violations. Once the voltage stabilizes, cores can begin operation at a higher frequency. To react rapidly to short-lived bottlenecks, this transition time should be kept short. Based on calculations for a ARM Cortex-M3 that has been synthesized, routed, and back-annotated for capacitance, our estimates show that, in a 16-core system, the area overhead of the decoupling capacitance is less than 5% to achieve a transition time of less than 20 cycles. In addition to the 5% area overhead for decoupling capacitance, either an additional metal layer is needed to route the high voltage power grid, or an additional 10% area overhead [45, 89]. For larger systems consisting of more cores, the area overhead is reduced because the extra capacitance can be distributed amongst the cores. The key point is that using a dual-V_{dd} system, a core can boost in 10's of cycles with less than a 5% area overhead. Section 6.5 will provide a sensitivity analysis of the boosting latency on performance.

6.3.2 Boosting Degree via DVFS

When systems are deployed they carry a maximum thermal power budget. This allows package engineers and system designers the ability to guarantee they are able to provide both enough energy and cooling for the system. Because the system needs to maintain a strict thermal power budget, if one core needs to boost then the other cores must be lowered to compensate. For example, if the thermal budget allows all 16 cores to run at 320MHz and a single core is boosted to 1GHz, the remaining cores need to be reduced to 250MHz to stay within the thermal design budget. Since it takes long periods of time to transition cores to a lower voltage via the off-chip regulator, the low voltage rail must be already at the low operating point before the boost happens. This means that if there is no suitable candidate for boosting, i.e. the work is perfectly parallel, than this lower voltage on the rail degrades overall performance. Therefore, there is a optimal ratio of boosting based on the amount of bottlenecks in the system.

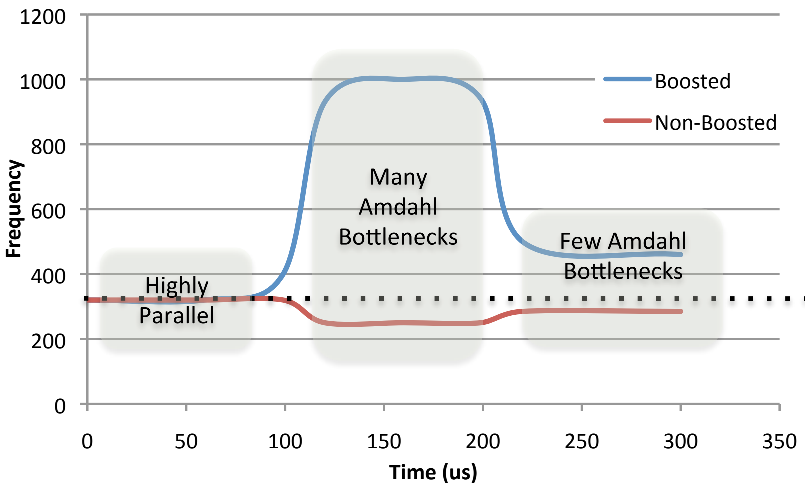


Figure 6.5: DVFS Techniques to adjust boosting degree. DVFS operates over 10's of microseconds and responds to changes in workload characteristics. The more bottlenecks in the system, the greater the boosting difference.

Using DVFS the system can adapt the difference between the high and low voltage rails to match the amount of bottlenecks present in the system. If the system has many bottlenecks, then the boosting differential should be larger. Since DVFS takes many thousands of cycles to adjust voltage, it should be used to adjust the voltage differential over phases of program behavior. Figure 6.5 shows how DVFS can be applied on the supply voltages to adapt the degree of boosting to match the system workloads. Note that the timescale for adjusting such voltages is in microseconds.

Combining these two techniques, voltage boosting can help overcome bottlenecks by boosting in 10's of cycles, and the system can adapt the amount of boosting based on the amount of bottlenecks present in the code over longer phases of execution.

6.4 Boosting Architectures

6.4.1 Designs and Tradeoffs

There are many ways to configure a system that supports voltage boosting. Rather than lower the voltage of all the cores in the system as described in Section 6.3 to accommodate a thermal budget a system could temporarily clock gate several cores to make room in the thermal budget. The following subsections will describe 3 such systems and describe the tradeoffs associated with each. This section concludes with simulation results of these systems on scientific workloads.

6.4.1.1 Traditional CMP Designs

Figure 6.6(a) presents a traditional chip multi-processor (CMP) architecture. In this system each core has a private L1 and connects to a shared L2 through a bus. There are several options for boosting. The first approach is simply to turn off some cores and their associated L1 and boost the remaining cores, Figure 6.6(b). If the system has writeback L1 caches, when boosting occurs the system must stall as the dirty lines in each L1 that is turned off is written back to the L2. If on the other hand the L1's were to be write-through, the bandwidth on the bus would be extremely high in the non-boosted mode. This approach allows the cores to scale to the highest voltage, because they can scale up by the power saved from both the L1 and cores that are turned off.

The second approach is presented in Figure 6.6(c), where just the cores are turned off. This allows the system to boost immediately, but because the L1's remain on the boosted cores can not boost to as high of a voltage. The caches can also improve performance by continuing to snoop and reply to requests for dirty lines in the cache. However, the cache space is ultimately wasted, as there is no way for the boosted cores to put data in them that wasn't already present when the system went to boosted mode.

6.4.1.2 NTC CMP Designs

The next set of designs seeks to leverage some interesting findings from NTC operation and to provide fully utilizable cache space in boosted mode. The first observation is that due to the differing activity factors for SRAM and logic, the optimal operating point for caches and cores is different. Chapter 4 showed that SRAM performs better at slightly higher voltages and speeds, as can be seen in Figure 6.7.

This led to a proposed architecture where the cache is run at a faster rate than the cores, and multiple cores shared a L1, see Figure 6.8(a). In this system the cache is run at 4X the

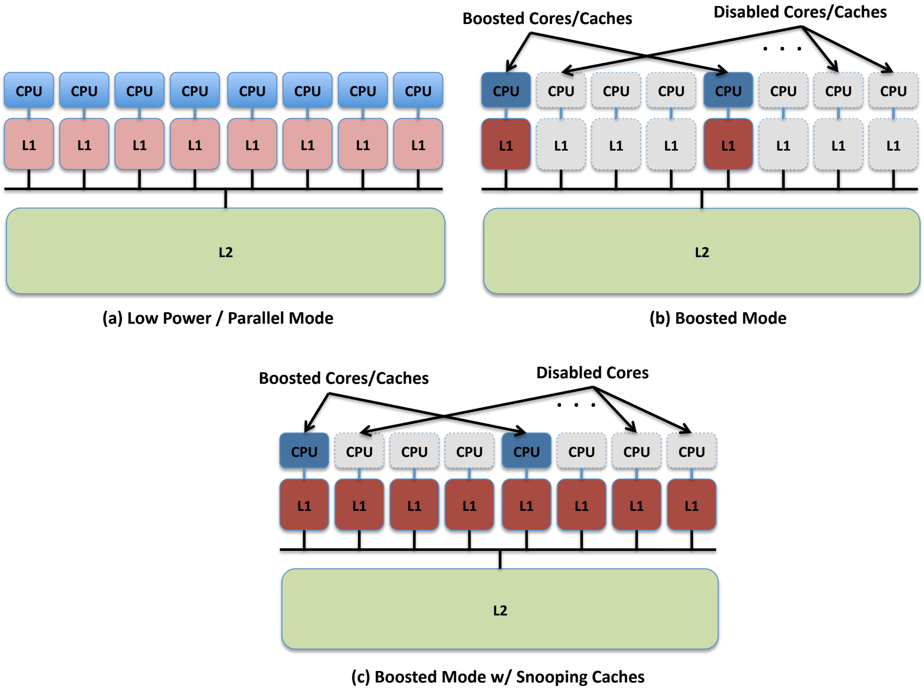


Figure 6.6: Traditional CMP Architectures and Boosting Options (a) Non-Boosted, (b) Boosted, (c) Boosted w/ Snooping.

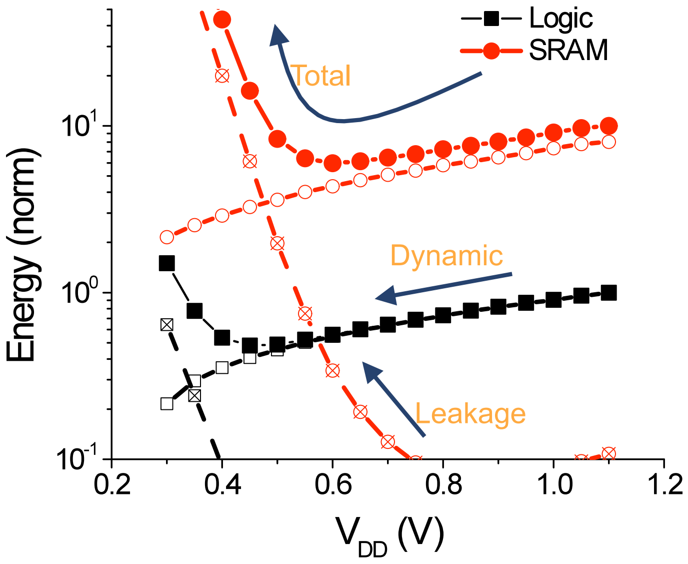


Figure 6.7: Difference in energy optimal SRAM and Logic. Note SRAM has an optimal at higher voltages, enabling NTC architectures.

frequency of the cores. The cache is than time multiplexed to provide each core a response in a single cycle. This allows cores to communicate data within the L1 without having to snoop on the bus, thus reducing energy. The total L1 size is less than 4X the size in the traditional design space, because the cores can achieve the same performance with less total cache do to cache sharing and reduced runtime.

It is important here as well, because the system can be boosted with less impact. First, the cache doesn't need to change frequency only the cores. The system simply disables 3 of the cores and speeds up the remaining core to match the frequency of the cache. This has two main benefits. First, the core still sees the full cache space. This means that the core has more space and will take fewer misses. Second, the latency of the core to main memory increases and the additional cache space helps to hide that latency.

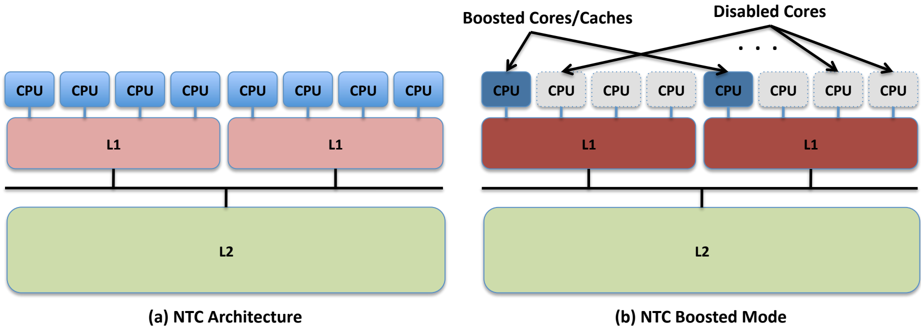


Figure 6.8: NTC Architecture (a) Non-Boosted, (b) Boosted

6.4.2 Tradeoff Analysis

Table 6.1 shows the tradeoffs of the 4 different systems. Time to boost is best for all systems which can do so instantaneously, no need for writeback. Unboosted bus bandwidth is best for the NTC system because it doesn't need to snoop as often and the cores share an L1, it is worst for the writethrough (WT) case, where the constant writethrough traffic occurs. Traditional systems can boost to the highest frequency because they disable the cores and L1's. The snooping design boosts to the lowest frequency because it leaves all the L1's on. The NTC design receives a 0 for boost frequency, because it can't boost as high as the traditional designs but can achieve higher than the snoop based because the total cache space is smaller due to sharing effects in non-boosted mode. Total cache space is best for NTC where the full cache is available, and marginally better for the snoop case due to limited snoop hits within the caches still powered on.

Config.	**Time to Boost**	**Bus Bandwidth**	**Boost Freq.**	**Cache Space**
Trad. WB	-	0	+	-
Trad. WT	+	-	+	-
Trad. Snoop	+	0	-	0
NTC	+	+	0	+

Table 6.1: Summary of the tradeoffs of each boosting architecture. + Good, - Bad, 0 Neutral

6.4.3 Methodology

Power analysis for the cores was done using synthesized ARM Cortex-M3 cores post place and route (APR). Caches for NTC operation were designed using the technique proposed by Chen et al. [20]. Power was calculated for caches post APR for both traditional 6T and NTC enhanced 6T designs. All systems were compared with the same thermal design budget (Peak Power Draw). The analysis was done to examine which of the approaches provides the best boosted performance and is therefore a static run of the benchmark in the boosted state.

6.4.4 Simulator and Benchmarks

For this study the M5 simulator [9] was used. It was modified to support the cache sharing required by NTC architectures. The benchmarks were from the SPLASH-2 [96] scientific workload suite. The baseline traditional system was 16 cores and 8kB of instruction and data cache per core. The equivalent performing system in NTC was 16 cores in clusters of 4 cores sharing a 24kB instruction and data cache. Snooping architecture results are not included here because the snooping based approach always loses in static analysis. The advantages of the snooping based approach occur when boosting needs to be quick and when dirty blocks reside in the other L1's, both conditions that do not occur in a static analysis.

6.4.5 Results

Figure 6.9 presents the key findings. In the analysis two comparisons are made. First, the system performance in boosted mode is compared. Second, for a fixed workload completion deadline the total energy of each system is compared.

In boosted mode the additional cache space provided by the NTC approach helps to reduce the L1 cache miss rate and provide faster completion. However, within the same

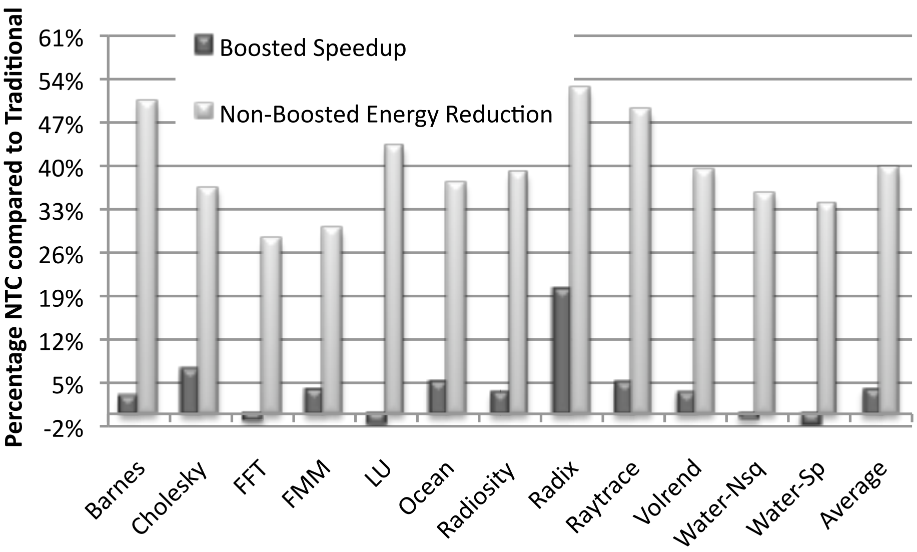


Figure 6.9: Comparison of NTC against Traditional Architectures. NTC is 40% more energy efficient in non-boosted mode, and runs 4% faster in boosted mode on average.

thermal budget the cores/cache must run slower than the traditional approach since more cache is powered on. This results in a tradeoff. For some benchmarks, such as *Radix* and *Cholesky*, the decrease in cache miss rates dominates and the performance is improved. However, for some benchmarks, such as *LU* and *Water*, there is not significant reduction in cache miss rates and from the larger cache, and the degraded processor performance dominates. Overall, most benchmarks see an increase in performance in NTC boosted mode. The average performance improvement is 4%, while some benchmarks see as high as 21%.

In the non-boosted mode thermal design consideration is removed and instead the analysis will compare optimal energy performance. For this experiment the total workload completion time is fixed, and the processors and caches are slowed down to meet the deadline. For the NTC workloads, the shared cache provides single cycle sharing of data between cores in the same cluster. It also reduces snoop traffic on the bus between L1's. This coupled with the ability to share instruction cache space helps to reduce the total number of cycles the NTC machine must run. Since it takes less cycles, the core speed can be reduced and still meet the deadline. Figure 6.9 shows that for all the benchmarks, the NTC design consumes less energy than the traditional CMP approach, by 40% on average.

Overall the NTC design provides a more energy efficient solution in non-boosted mode, and when boosted can outperform a traditional design on benchmarks where cache capacity is critical. The results show a system that is, on average, 40% more energy efficient in non-boosted mode, and 4% faster in boosted mode.

6.5 Transactional Bottlenecks

This study helps to illustrate how boosting helps to overcome two of the parallelization complications mentioned in Section 6.2. First, in Section 6.2.1.1 Amdahl bottlenecks limit the amount of parallelism that can be extracted from parallelization. With a transactional memory system, conflict information can be used to boost cores that present such bottlenecks. Second, Section 6.2.1.2 discussed the difficulties associated with parallel programming. Transactional memory has been proposed to ease the programming effort by making it simpler to write parallel code. However, naive coders may still create systems with large transactions that have high conflict rates, further exacerbating the Amdahl bottleneck problem. Blake et al. [12] showed that conflict rates can exceed 90% for some of these workloads, and provide an interesting problem for parallel scalability.

Feature	Description
Processors	16 Alpha ISA, 1 IPC @ 200-1400 MHz
L1 Caches	Private Data and Instruction Caches, 64kB, 2-way associative, 64-byte line size, 1 cycle latency
Interconnect	Shared bus at core speed
L2 Cache	Shared 32MB, 16-way associative, 64-byte line size, 6 cycle latency
Main Memory	2048MB, 100 cycle latency
Linux Kernel	Modified v2.6.18

Table 6.2: Simulation Parameters for Transactional Memory Analysis

6.5.1 Methodology

For this work a baseline system like the one from Blake et al. [12] is assumed. This work uses a proactive conflict detection mechanism to reduce conflicts in the system. Each time a transaction is scheduled in the system, a software runtime environment evaluates the likelihood that the transaction will conflict with others running in the system. If it is likely to conflict, the runtime environment defers scheduling the transaction and tries to schedule something else instead. The boosting system leverages this by assigning credits to the transaction in the system that was labeled as a potential conflict. If any transaction exceeds a credit score, meaning it is holding up many other threads from running, it is boosted to speed the completion.

For this analysis several boosting techniques were explored. First, a system with 16 cores is examined where at most one core can boost at a time. To keep the comparison fair, the TDP is kept the same for all cores. In this analysis several boosting ratios are examined. In order to keep the designs iso-TDP, if a higher boosting ratio is used, then the nominal voltage/frequency of the other cores must be lower. Thus resulting in performance losses when there is no imbalance in the work (highly parallel). The second system analyzed is a NTC style configuration, where several cores are turned off in order to boost a core. In this system the nominal performance is not degraded when there are no boosting candidates, as the voltage doesn't need to be lowered to enable the boosting. However, since the cores are shut off during the boosted period, if there were other useful work to be done in the system some performance is lost.

6.5.2 Simulator and Benchmarks

The M5 simulator [10] was again used for this study and was modified to match the system in [12]. A modified version of the Linux kernel that supports transactional memory in M5 is used. The benchmarks used are from the STAMP [17] transactional memory benchmark suite. The simulation parameters are presented in Table 6.2.

6.5.3 Results

Two different analysis are presented. First a study of how boosting helps overcome transactional memory bottlenecks and high contention. Second a sensitivity analysis on the impact of the boost latency on the most sensitive workload.

6.5.3.1 Boosting Results

Figure 6.10 presents the speedup for different levels of boosting normalized to a uniprocessor system given the same TDP. The boosting values signify the ratio of low frequency to high frequency. For example *Boost 4x* represents a system where the boosted frequency is four times larger than the non-boosted frequency. The first observation is that for transactional workloads, for the same TDP all but *intruder* provide more performance than a single core running much faster. In the case of intruder the conflict rate is high, indicating the existence of many Amdahl bottlenecks. These bottlenecks cause the *No Boost* system to only reach 55% of the performance of a single core machine running with the same TDP. Boosting in this case, *Boost 4x*, helps to overcome some of these bottlenecks, bringing the performance to within 10% of the uniprocessor system, a nearly 50% improvement for the boosted system over the non-boosted system. In this case, too many Amdahl bottlenecks exist, and no amount of boosting can overcome them all. *Delaunay* is a similar benchmark in that it has many conflicts, and a *Boost 4x* provides the best performance. For the rest of the benchmarks, the contention rate is low and a modest boosting system, *Boost 1.6x*, provides the optimal performance. Overall boosting provides a 14% improvement on average over the non-boosted case for a 1.6X level of boosting.

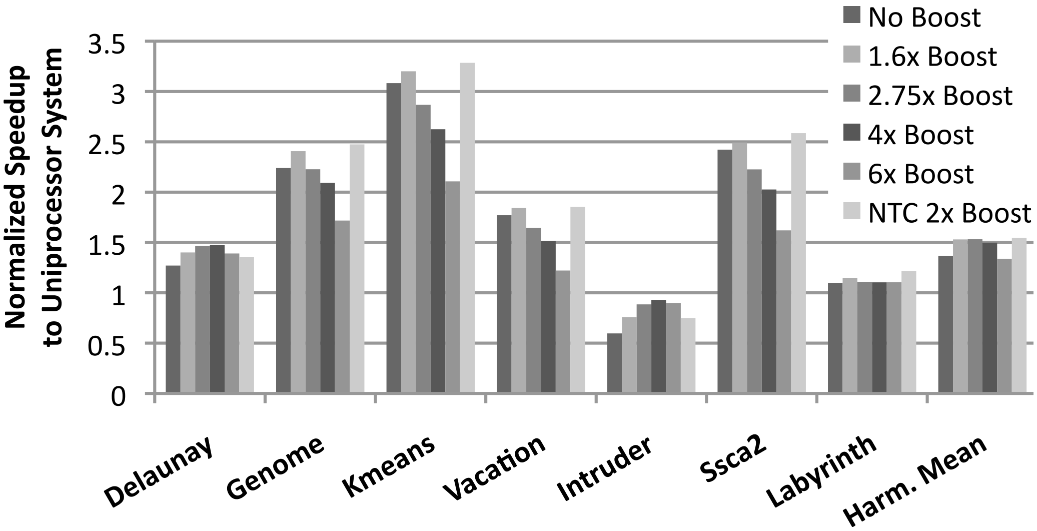


Figure 6.10: Speedup of system compared to a full voltage uniprocessor design. In cases where parallel low voltage systems do not perform as well as a uniprocessor system, boosting techniques help to regain some of the lost performance. All configurations are under same TDP.

The NTC boosting system provides a boost ratio of 2x, in Figure 6.10 the performance of the NTC system is equivalent or better than that of the 1.6X boost system. Recall that for the NTC configuration gains are made when there are no cores to boost, as in this case it performs at the same rate as *No Boost*. When boosting occurs 3 cores must be shut off, thus degrading parallel performance at the expensive of boosting the core. It is expected that for systems with low percentages of boost time NTC will outperform all other configurations, which is the case for *Genome, Kmeans, Vacation,* and *Ssca2*. In all these cases the percentage of time a core is boosted is less than 20% of cycles.

A third observation is that depending on the amount of Amdahl bottlenecks in the system, a different boosting ratio is optimal. For low contention benchmarks, like *Ssca2* 1.6X boosting is optimal, however for more imbalanced workloads with higher contention, *Delaunay*, a boosting value of 4X is better. This motivates dynamically adapted boost ratio systems, but is left as future work.

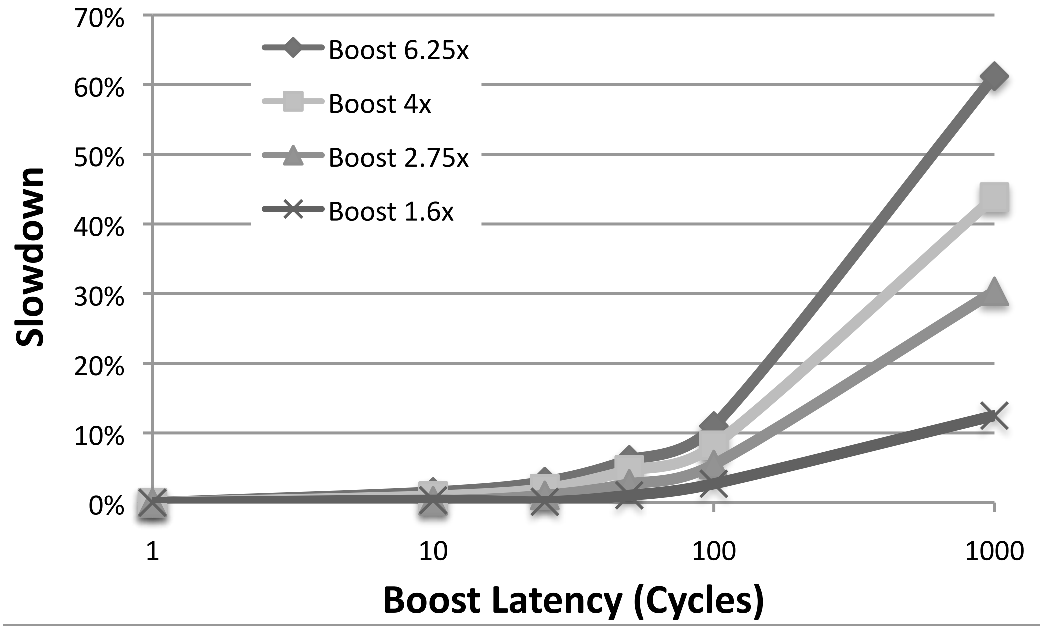


Figure 6.11: Impact of boost latency on system for the Intruder benchmark. For latencies under 100 cycles, the overhead is less than 10%.

6.5.3.2 Boosting Latency Sensitivity

The second study is an analysis of boosting latency. For this analysis the *Intruder* benchmark is chosen, as it had the highest number of boosting occurrences and presents the most sensitivity to the boosting latency.

Figure 6.11 shows the impact of boost latency on different levels of voltage boosting. The main takeaway is that for boost latencies less than 50 cycles the impact on the system is less than 5%. However, if the latency grows to beyond 100 cycles, then the impact can begin to significantly slow down the system. So it is important to be able to boost the system in 10's of cycles, which is what the dual-V$_{dd}$ design provides. DVFS on the other hand takes 10,000's of cycles to transition and could not react quickly enough to respond to the transient bottlenecks.

6.6 Web Server Latency

6.6.1 Improving Server Response Time

Recently, the number and complexity of internet-scale services has seen explosive growth. As users become accustomed to responsive and dynamic web content, data center operators have become increasingly concerned with service response time [77]. In particular, maintaining *service-level agreements* (SLAs) is critical to running robust services.

These performance constraints are often expressed by requiring the 99^{th}-percentile latency of all requests to be below a given time.

Unfortunately, server workloads are known to have service time distributions with high variance [35]. The variability in request size, coupled with often bursty traffic, leads to response times with long-tails. Currently, there are few ways to achieve the desired 99^{th}-percentile response times other than over-provisioning the system.

Boosting provides an alternative to over-provisioning. It can produce tighter latency distributions—reducing the 99^{th}-percentile latency—by selectively boosting long-running requests. Most services are designed such that they are *request-level parallel*: each server request (e.g., requesting a web page), runs on a single thread on a single cpu. Accordingly, cpus running long requests—those in the tail of the distribution—can be boosted to reduce the 99^{th}-percentile latency.

6.6.2 Methodology

To evaluate the effectiveness for server workloads, *Stochastic Queueing Simulation* (SQS) [62] is used. SQS is a event-based, stochastic queuing theoretic simulator used to model performance tradeoffs for data center workloads. Workloads are represented by empirical inter-arrival and service time distributions measured on production servers. We extend SQS to allow cpu boosting by modulating cpu service rates.

We use a simple policy to control which requests are boosted. If a request has been in service longer than the threshold time, it is marked for boosting. Provided that no other cpus are boosted, this request is immediately boosted. If another cpu is currently boosted, the current cpu is put in a queue. After a boosted cpu finishes service, the queue is inspected and cpus are boosted in *first come first serve* order if their request has not otherwise finished. Our policy is simple, yet effective and demonstrates the utility of the boosting mechanism; we leave evaluating more complex policies for future work.

6.6.3 Results

To evaluate the effectiveness of boosting, we first analyze the response time distribution of a sixteen cpu server running the Web workload using SQS. Next, we compare the response time distribution of the server with cpus running 60% slower (NTC). We investigate if we can meet the same 99th-percentile target with these slower cpus by using boosting. Our boost-enabled server can boost a single cpu to 6x the nominal frequency.

The CDF of response time for each scenario is shown in Figure 6.12 and the PDF in Figure 6.13. Viewing the response time as a CDF allows us to observe how the quantiles

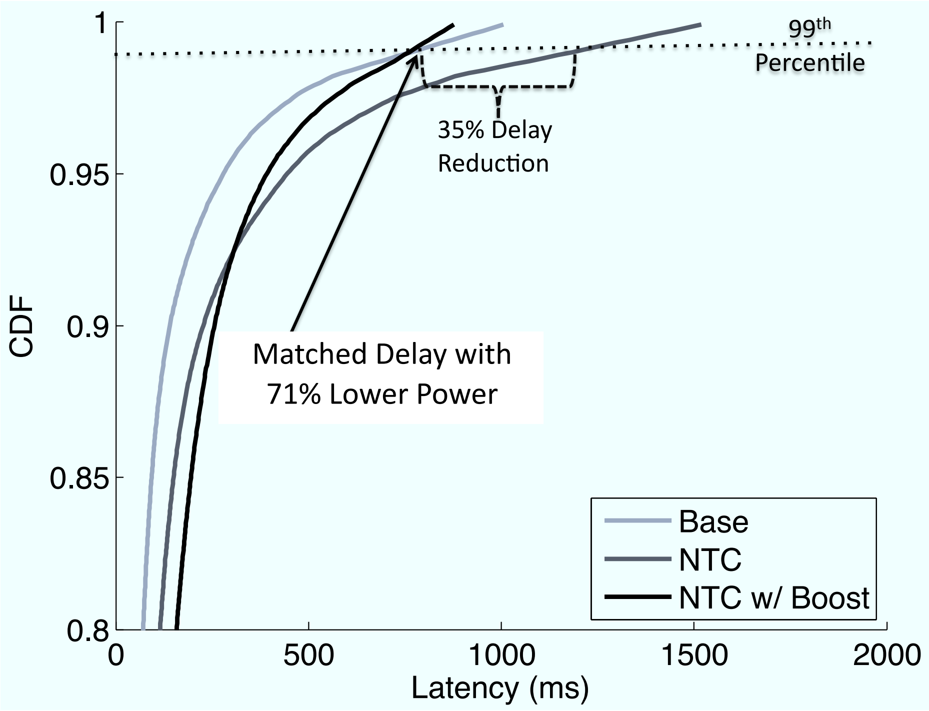


Figure 6.12: Response Time CDF.

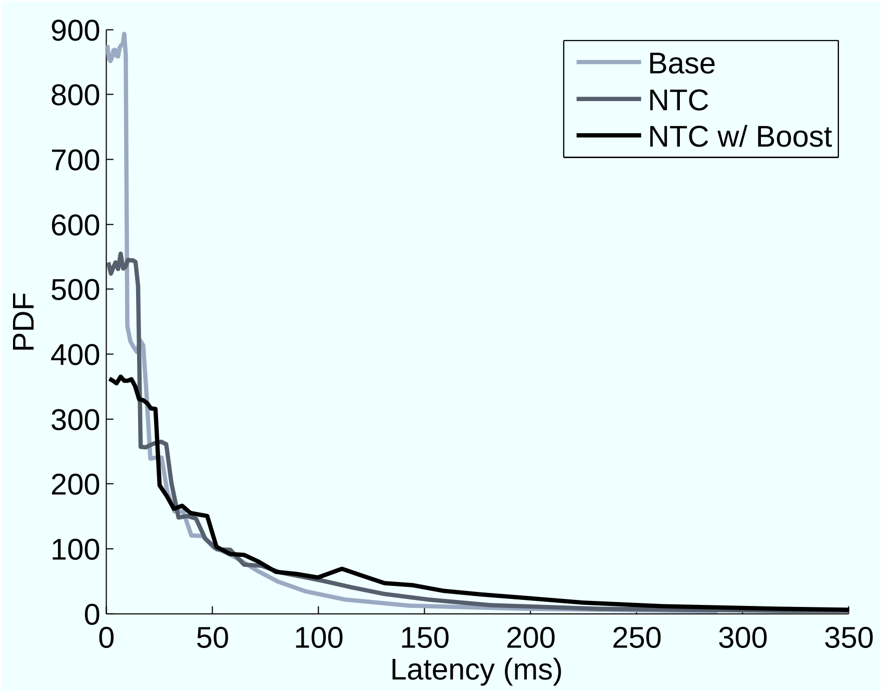
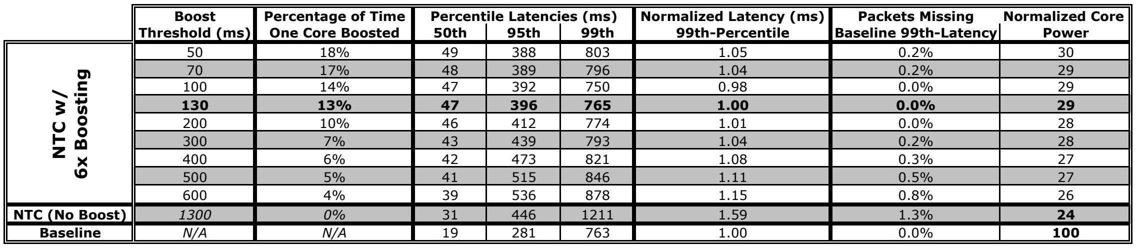


Figure 6.13: Response Time PDF.

(e.g., 99^{th}-percentile) change for each case. The response distribution of the unaltered system, *Base*, has a 99^{th}-percentile latency of 763ms. Naturally the slower cpus, *NTC*, have a much larger 99^{th}-percentile latency of 1.2s. However, with boosting, *NTC w/Boost*, we are able to achieve a 99th-percentile latency of 765ms, nearly the same as the base system.

To provide a different view of boosting's effect, Figure 6.13 provides the PDF of response time. There are two important effects in this figure. The first is that the PDF has a small value for lower latency. For example, in the 10 to 20ms region the PDF mass is much lower for *NTC w/Boost* than for *Base*. This demonstrates that jobs that would typically finish inordinately fast now take longer. The second phenomenon is that the PDF values for mid-range latency (50-200ms) are higher for *NTC w/Boost* than for the other systems. Hence, we find that voltage boosting tends to pull low-latency and high-latency requests towards the middle of the distribution.

It is important to highlight that we are reducing the latency of 99^{th}-percentile at the expense of longer delays for smaller requests. In other words, while the tail of the *NTC w/Boost* system is now shorter, the lower quantile latency increases. The median latency of the *NTC* system without boosting is 31ms, whereas it is 47ms with boosting (this region is not shown in Figure 6.12, which just provides detail at higher quantiles, but is displayed in the tabular results in Figure 6.14).



	Boost Threshold (ms)	Percentage of Time One Core Boosted	Percentile Latencies (ms) 50th	95th	99th	Normalized Latency (ms) 99th-Percentile	Packets Missing Baseline 99th-Latency	Normalized Core Power
NTC w/ 6x Boosting	50	18%	49	388	803	1.05	0.2%	30
	70	17%	48	389	796	1.04	0.2%	29
	100	14%	47	392	750	0.98	0.0%	29
	130	**13%**	**47**	**396**	**765**	**1.00**	**0.0%**	**29**
	200	10%	46	412	774	1.01	0.0%	28
	300	7%	43	439	793	1.04	0.2%	28
	400	6%	42	473	821	1.08	0.3%	27
	500	5%	41	515	846	1.11	0.5%	27
	600	4%	39	536	878	1.15	0.8%	26
NTC (No Boost)	*1300*	*0%*	31	446	1211	1.59	1.3%	**24**
Baseline	*N/A*	*N/A*	19	281	763	1.00	0.0%	**100**

Figure 6.14: Breakdown of Baseline, NTC without boosting, and several boosting thresholds. Showing the impact of boosting threshold on latency, power, and boosting percentages. 16-core system. Power normalized to 100W Baseline.

Figure 6.14 presents the impact of the boosting threshold on the performance of boosting. The entry in bold, 130ms, is the one plotted in Figure 6.12. These results are presented in graphical form in Figure 6.15. From right to left in the graph (decreasing boosting threshold) the result is that more requests are selected to be boosted, and the boosting percentage increases. This correlates to an increase in power. The percentage of lost packets also decreases as the boosting threshold is lowered, as more requests meet the deadline. For the range of 70 to 200 ms the boosting system does not drop any packets that were handled by the full voltage baseline system. If the threshold is lowered below this point non-critical requests start to boost. As a result, because only one cpu can boost at a time, critical requests become stalled behind these and start missing the deadline.

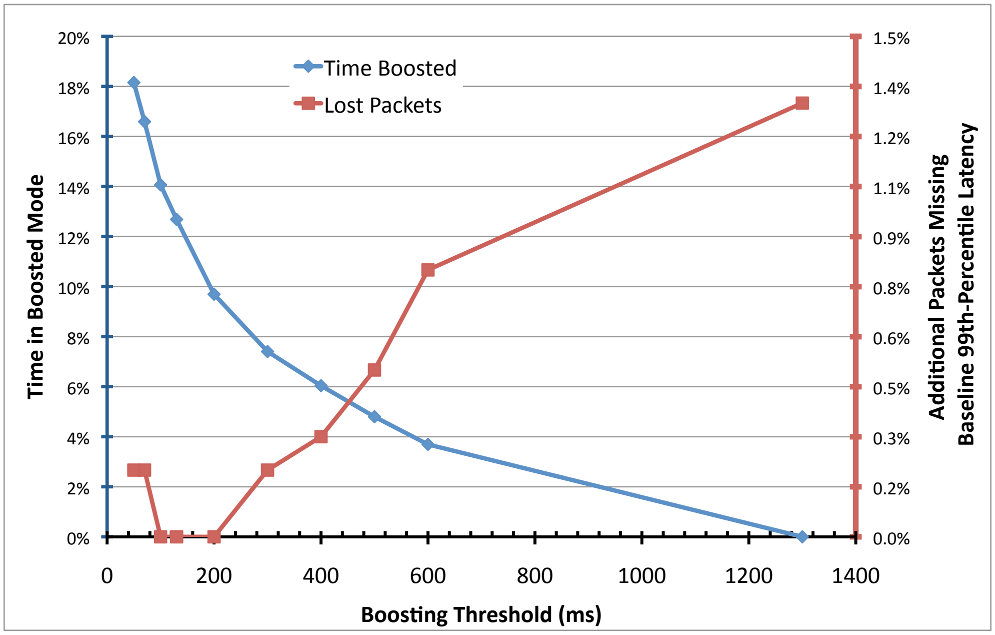


Figure 6.15: Impact of boosting threshold on the 99^{th}-Percentile latencies.

Overall, the 99^{th}-percentile latency is reduced by 35% when 6x boosting is employed compared to the non-boosted NTC system. In order for a traditional non-boosted design to achieve the same latency, it would need to run 60% faster and consume 3.4X more power.

6.7 Related Work

6.7.1 Dynamic Voltage Scaling

There has been considerable research in the area of DVFS, however, this Chapter differs in two main ways from the proposed voltage boosting work. First, this Chapter advocates designing the system for energy efficient operation and raising the voltage, where DVFS has traditionally been applied to existing high voltage designs where the voltage is simply lowered. Second, the design presented in this Chapter uses fast voltage boosting—transitioning in 10's of cycles—in order to overcome short lived bottlenecks. the goal in DVFS work is to adapt to phase changes in program behavior. The traditional time frames for DVFS techniques have been 10,000's of cycles or longer. This transition time allows for the system to adapt to longer, more phasic behavior but prevents it from being able to remove short-lived bottlenecks. DVFS is actually a complimentary technique to fast voltage boosting. A system could use fast voltage boosting to overcome transient bottlenecks or serialization points, while DVFS could be used to adapt to longer phasic changes in behavior or to adjust the level to which boosting occurs.

There is research on fast DVFS; for example, recent work by Kim et al. [51] on per core DVFS. It uses on-chip regulators to provide separate operating voltages for each cpu on chip. They seek to provide faster DVFS response time by bringing the voltage regulators on chip. While the goal of keeping the transition latency low is the same, their work focuses on providing dynamic control to all cpus. This Chapter differs in that, among other things, only one cpu is allowed to transition at a time to reduce voltage fluctuations on the power rails. This allows for a simpler on-chip power delivery system without the need of complex analog and DVFS control circuitry. The proposed dual-V_{dd} rail design enables faster voltage switching, while avoiding the power overheads associated with on-chip regulators.

6.7.2 Heterogenous Architectures for Bottlenecks

There is a long history of work in heterogeneous, or asymmetric multicore systems. Early work in such single-ISA systems was done to reduce power consumption [5, 53, 54, 55]. These early research ideas focused on creating a system with cores of the same-ISA but different performance characteristics. This Chapter differs in that it proposes using identical cores. However, it is similar in nature because asymmetry in the system is created by boosting some cores frequencies.

More recent work in the area of asymmetric systems has focused on overcoming bottlenecks. One recent approach proposed by Suleman et al. [83] looks at overcoming bottlenecks by shipping critical sections of execution to a Big Core. This work is similar in goal to the voltage boosting in this Chapter, but differs in approach. While their work migrates critical sections to a larger core, the technique in this Chapter boosts the critical section in place (since any core can be boosted). When they migrate the critical section the execution context—processor state and cache data—must migrate across the system, increasing the latency to boost a critical section. In this Chapter the approach allows boosting to happen at a much quicker granularity, because the critical section is boosted in place where cache data is already present.

6.8 Conclusions

Power considerations are now at the forefront of research in not only mobile devices, but also desktop and server markets. Traditional approaches simply try to reduce the power of conventional systems, but the power reductions are limited by variation and leakage concerns. Starting from scratch, designers can create systems optimized for NTC operation that are aware of variation and leakage concerns. This work showed that designs targeted for NTC operation can achieve, on average, 40% better energy efficiency than a traditional design. However, these systems rely on parallelization of workloads to achieve efficiency. Parallelization can present major difficulties, such as coding and verification, serial bottlenecks, and increased packet latency.

This chapter also proposes a system in which dual-V_{dd} rails are used to create NTC systems with the ability to raise the performance of a subset of cores beyond the original design point—voltage boosting. Simulation shows that voltage boosting can overcome parallelization bottlenecks introduced in transactional memory workloads, and that for a boosting level of 1.6x, the average performance of workloads can be improved by 12%. In web server workloads, reducing the frequency and increasing the parallelism of the system saves power and maintains throughput, however individual request latency is increased. Results show that boosting can reduce the 99^{th}-percentile of request latency by as much as 35% when boosting at levels of 6x, and reduces the power consumption by more than 70% compared to a traditional design with the same 99^{th}-percentile latency target.

CHAPTER 7

Many-Core Architectures

Future chip designs will contain 100's of cores—many-core systems—and this Chapter takes the NTC architecture to the extreme. 3D chip integration—where multiple layers of silicon are stacked in the z-dimension—is explored as a means to complete such a system. This Chapter presents Centip3De, a near-threshold 7-layer 3D system that contains 128 ARM Cortex-M3 cores and 256MB of stacked DRAM, Figure 7.1. Centip3De uses the unique aspects of near-threshold computing to create highly energy efficient compute clusters. By doing so, 9.28 GOPs/Watt is achieved in a relatively old, 130nm technology. Centip3De uses Tezzarons through-silicon via based 3D stacking and stacked DRAM [31]. Work in this chapter was done with the collaboration of my colleagues Dave Fick and Bharan Giridhar.

7.1 Introduction

Recent high performance IC design has been dominated by power density constraints resulting in increasingly wide multi-core designs [90]. Supply voltage scaling has not kept pace with feature scaling, resulting in die where only a portion of the transistors can be activated. The CMOS roadmap includes 3D integration [87] and FinFET devices [37], which will increase device count for a given area even further. These devices will not be usable, however, without viable strategies to reduce power consumption.

In this Chapter, we propose using near-threshold computing (NTC) to address this issue where cores are operated near the threshold voltage where power and performance are balanced and energy efficiency is increased significantly compared to operating at the highest wear-out limited supply voltage. For instance, in the Centip3De system presented in this Chapter, we operate cores at 677mV (vs. 1.5V nominal supply) where their energy efficiency is increased by 4.8X and power consumption is reduced by 9.6x. Since power

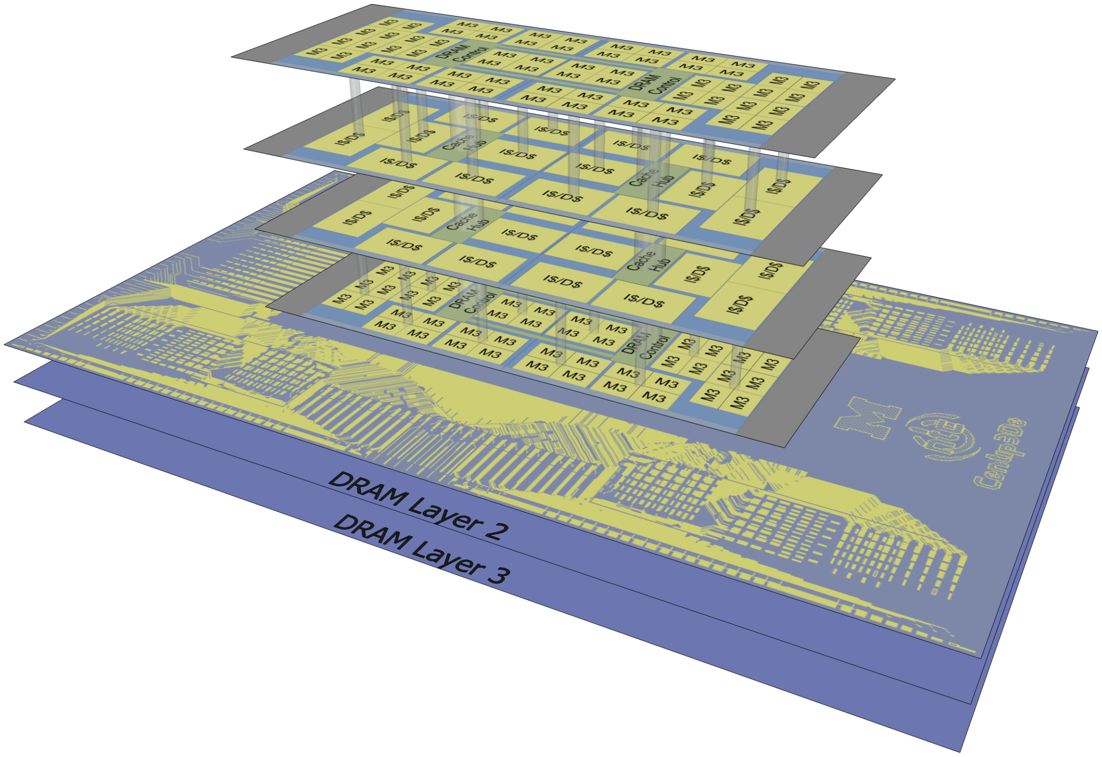


Figure 7.1: Cross section of the 3D stacked Centip3De chip.

consumption is dramatically lowered, NTC is an attractive match for 3D design which has limited power dissipation capabilities allowing designers to take advantage of the efficiency and performance benefits of 3D design. In this Chapter we present Centip3De, a 3D NTC 128-core system, which operates at 9.28GOPS/W which marks a 4.8X energy efficiency improvement compared to running at full voltage. In addition, Centip3De allows selected cores to boost momentarily by 8X in frequency by ramping them to full voltage addressing the need for single thread latency.

Recall from Chapter 4 that because of the higher leakage current in SRAMs compared to logic, memories reach their optimal energy/delay trade-off earlier than cores. Hence, we operate SRAMs at a higher supply voltage than cores in the NTC regime (870mV for SRAM and 670mV for logic in this technology). As a result, SRAMs operate faster than cores which is counter to traditional full voltage design. Centip3De takes advantage of this by connecting four cores to each cache, which each operate at 1/4th the frequency and communicate with the cache in a round-robin fashion. Attaching multiple cores to one cache has the added advantage that coherence within the 4-core cluster is automatically resolved, reducing the coherency traffic and overhead in the system. Further, as was discussed in Chapter 6, one or more cores can be disabled, allowing the remaining cores to be boosted in performance. In Figure 7.3, a single core can operate at 8X the frequency. By

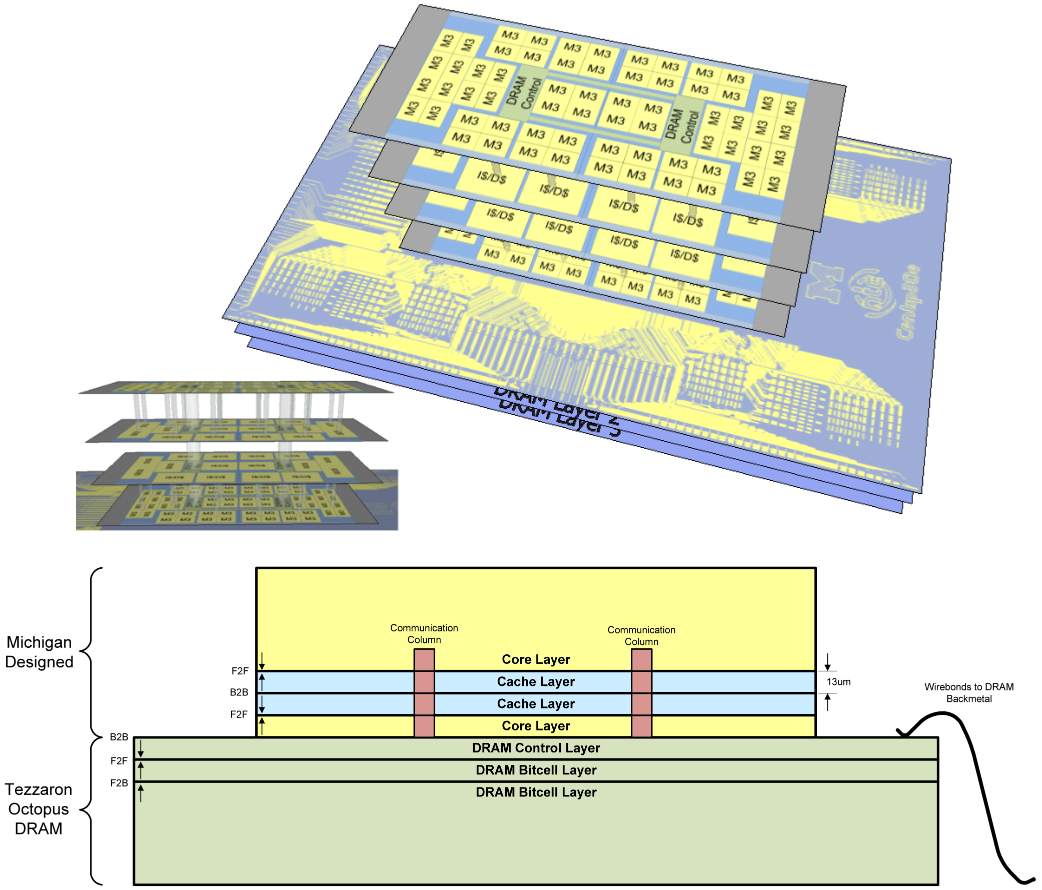
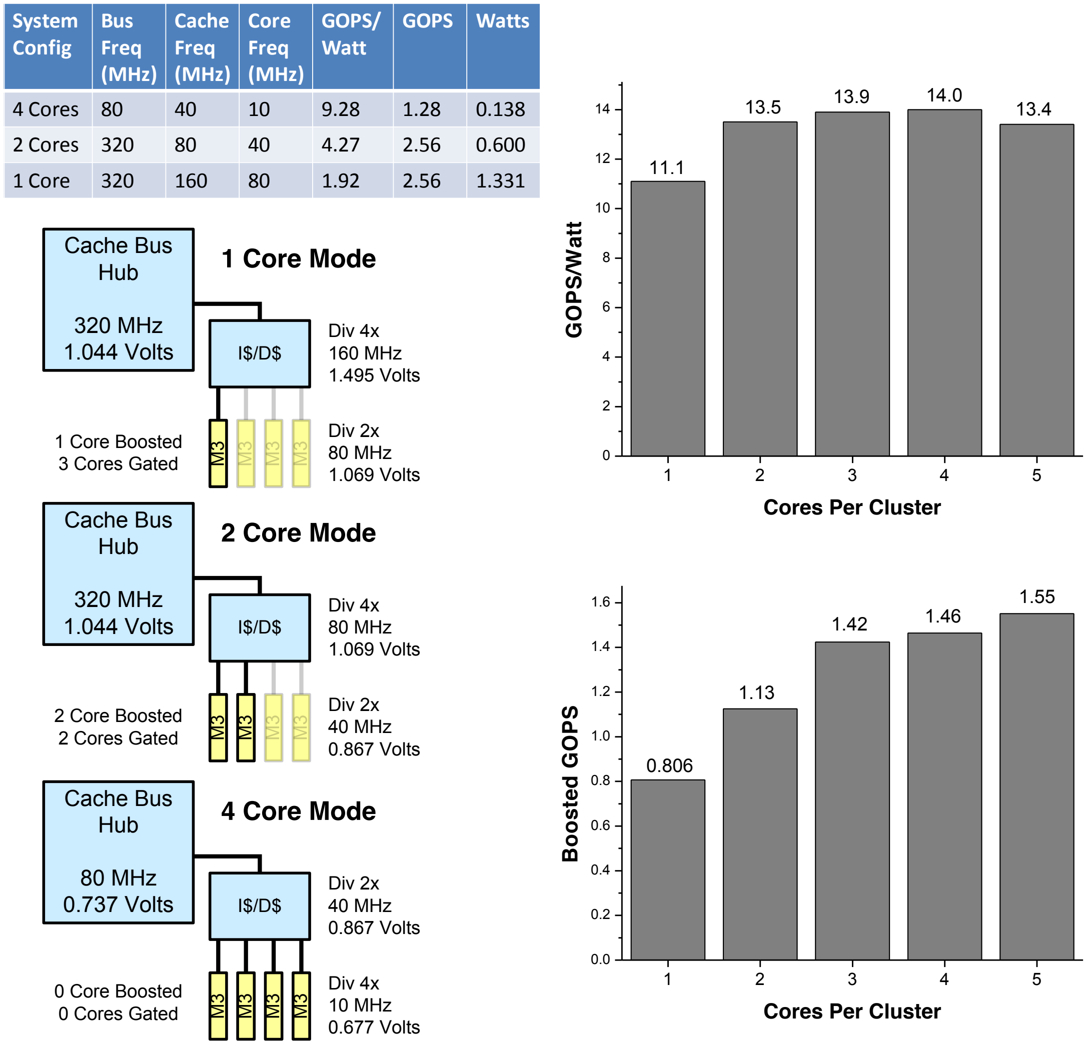


Figure 7.2: Side view of the designed Centip3De chip. The chip consists of 7 layers stacked and connected with through-silicon-vias. There are 4 logic layers consisting of cores and caches, 2 layers of DRAM cells, and 1 layer of DRAM controller and bond routing. There are 128 ARM M3 cores clustered in groups of 4 sharing a cache.

disabling the other 3 cores in the cluster, the power increase of the cluster is moderated and the boosted core gains access to the entire cache space instead of sharing it with three other cores. This further benefits the core performance by shielding it more effectively from the DRAM memory latency.



System Config	Bus Freq (MHz)	Cache Freq (MHz)	Core Freq (MHz)	GOPS/ Watt	GOPS	Watts
4 Cores	80	40	10	9.28	1.28	0.138
2 Cores	320	80	40	4.27	2.56	0.600
1 Core	320	160	80	1.92	2.56	1.331

Figure 7.3: Diagram showing how 1, 2, and 4 core mode works in the Centip3De system as well as analysis that shows the impact of clustering cores on Performance per Watt.

7.2 Top Level Architecture

At top level, Centip3De has 128 ARM Cortex-M3 processing cores which are organized into 32 four-core clusters (Figure 7.6). Each cluster has a 4-way 1kB instruction cache and an 4-way 8kB data cache, which connect to 256MB of shared DRAM through a bus-based 3D interconnect. The DRAM, bus interconnect, caches and cores can operate at several scaled frequencies of each other and have independent voltage domains. The default NTC configuration operates cores at 10MHz with 4 cores active per cluster and caches operating at 40MHz. Based on the FO4 delay trends across technology, the 10MHz is projected to translate to 93MHz in a 22nm technology. The frequency of the cache to DRAM bus operates at 80MHz. By providing 2x8 128b memory buses, 20GB/s memory bandwidth is supplied to the individual cores in NTC mode. Figure 7.11 shows the 4 core per cache configuration as well as the higher performance but lower energy efficient boosted single thread performance modes. Depending on workload, Centip3De can be configured to maximize single threaded performance, throughput, or a mixture of both.

3D design allows Centip3De to have greater bandwidth both within its bus system and to DRAM. Global interconnect resources for the bus system were reduced by roughly half by splitting the clusters across four layers, while the available maximum DRAM bandwidth, 1.6 Tbps, would be nearly impossible to achieve in a conventional design. The Tezzaron Octopus DRAM is partitioned into eight distinct vertically connected interfaces. Each interface has both a 128 bit input and 128 bit output bus, and is connected to its own DRAM controller. Centip3De's bus system has a separate 128 bit bus for each of the eight DRAM interfaces. These eight buses are duplicated in two communication columns (see Figure 7.2), where four of the eight DRAM controllers exist at the bottom of each column, and columns are bridged on the bottom-most layer.

The floorplan of the chip separates caches and cores into separate layers such that there are two cache layers and two core layers. The four cores of a computer cluster communicate with their cache through a face-to-face (F2F) interface, which reduces routing resource requirements by providing interface pins in the middle of the design for each. The caches each connect to the closest communication column on their layer. By traveling vertically between the four core and cache layers, the communication columns reduce required routing resources by approximately half when compared to a single-layer floorplan - similar gains are obtained in energy and performance for global communication.

Mask cost and design effort were reduced by designing a single core layer and a single cache layer and reusing the layers twice in the stack, post-fabrication. A core layer wafer and a cache layer wafer are bonded face to face to create a core/cache pair. Two of these pairs are created and then thinned on the cache side to 13 microns. The two pairs are then bonded back-to-back (B2B) on the cache side to create a four layer stack. Once the four layer stack is created, it is thinned and diced, and the individual dies are bonded to a thinned 3-layer DRAM wafer (Fig 1). The 4-layer logic layers must be diced first since the core and cache layers are much smaller than the DRAM layers.

Since the core and cache layers are identical, a pull-up circuit is included to identify the position of each layer. The interfacing DRAM surface provides copper tracing and wirebonding pads for the logic layers (Figure 7.2). Through-silicon vias (TSVs) in the bottom core layer connect to this copper tracing, while TSVs in the top core layer will remain isolated within the non-thinned substrate. By attaching a pull-up pad to one of these TSVs and grounding the wirebonding pad, the bottom core layer can identify itself with a zero signal, and the top core layer can identify itself with a one signal. This information is provided to the cache-cache B2B interface to control tristate buffering on the vertical buses, and to the replicated DRAM controllers, bus bridges, and PLLs to disable redundant modules.

7.2.1 Variable Cache Access Architecture

As shown in Figure 7.4, the caches are able to operate in two modes: two core mode and four core mode. In four core mode, three or four of the cores are enabled and each operate with a clock that is 1/4th the frequency of the cache and 90 degrees out of phase from each other. The cache accesses are pipelined internally to the cache while the core sees a single cycle interface. The data tags are read in the first stage of the pipeline, while the data is read in the third stage. By determining the correct way in the second stage of the pipeline, the number of accesses to the data arrays are reduced, thereby increasing energy efficiency. In two-core mode, one or two of the cores are enabled and each operate with a clock that is half the frequency of the cache clock and are 180 degrees out of phase from each other. The cache is still internally pipelined, however, the data arrays are read simultaneously with the tag arrays due to the reduced number of pipeline stages, which increases energy consumption per access. In both modes accesses are monitored for conflicts, which stall later transactions until the first finishes. A custom 8T SRAM design is used to provide adequate voltage scalability for the cache. A 128 bit cache line is used to match the bus system, but each 32 bit word may be read and written independently.

7.2.2 Clock Network

The cache and core clocks are derived from the globally distributed system bus clock. Due to the low parasitics of the TSVs, additional 3D considerations for the clock tree design were not necessary. Scaling the core and cache voltages unequally, however, causes the leaves of the distinct clock trees to become misaligned. The delay of each clock tree is adjustable via a digitally controlled, tunable delay buffer at the root of the tree, and clock tree alignment sensors are included (in situ) to assist in adjusting the trees (Figure 7.6).

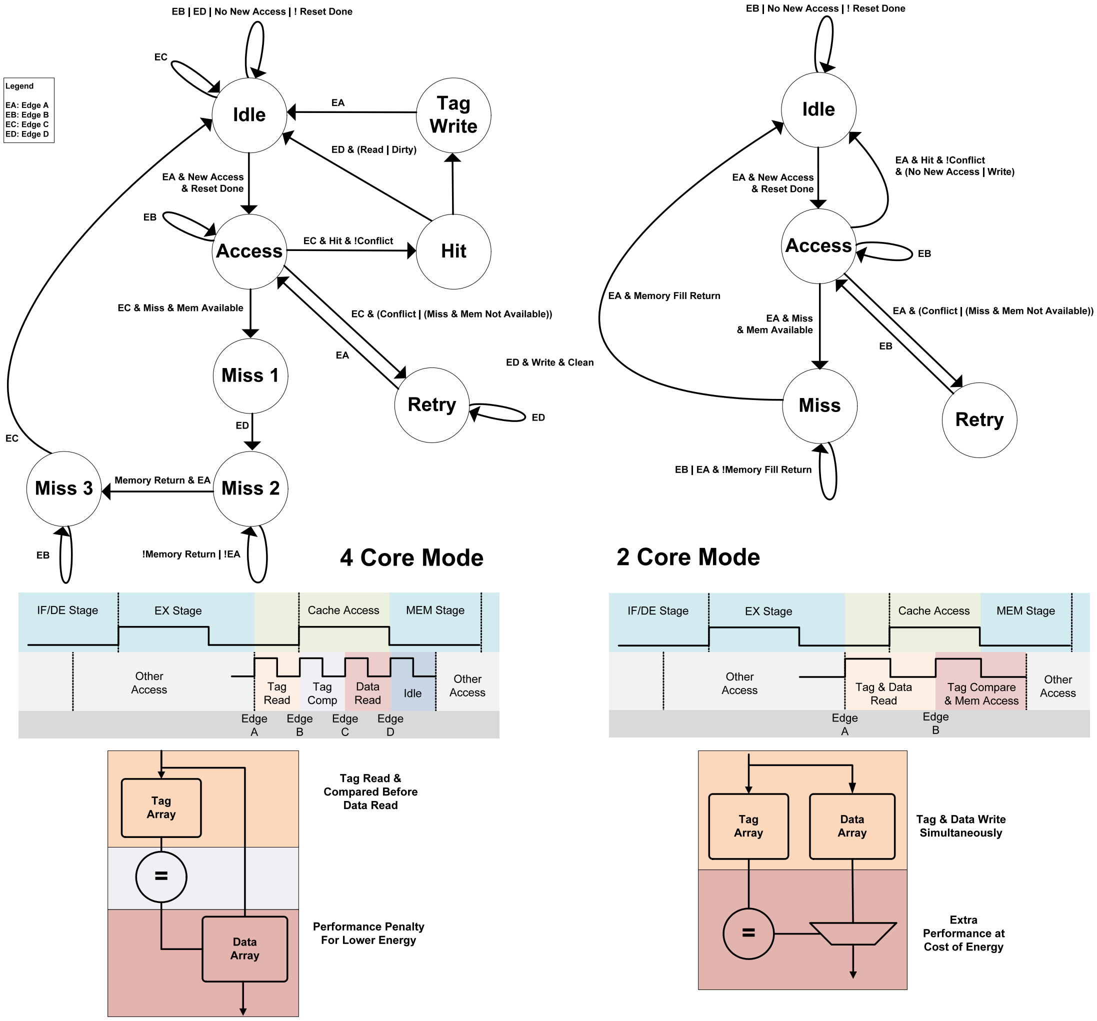


Figure 7.4: State diagram for cache accesses in 4 and 2 core mode. Note the access pattern is the same for 1 core mode.

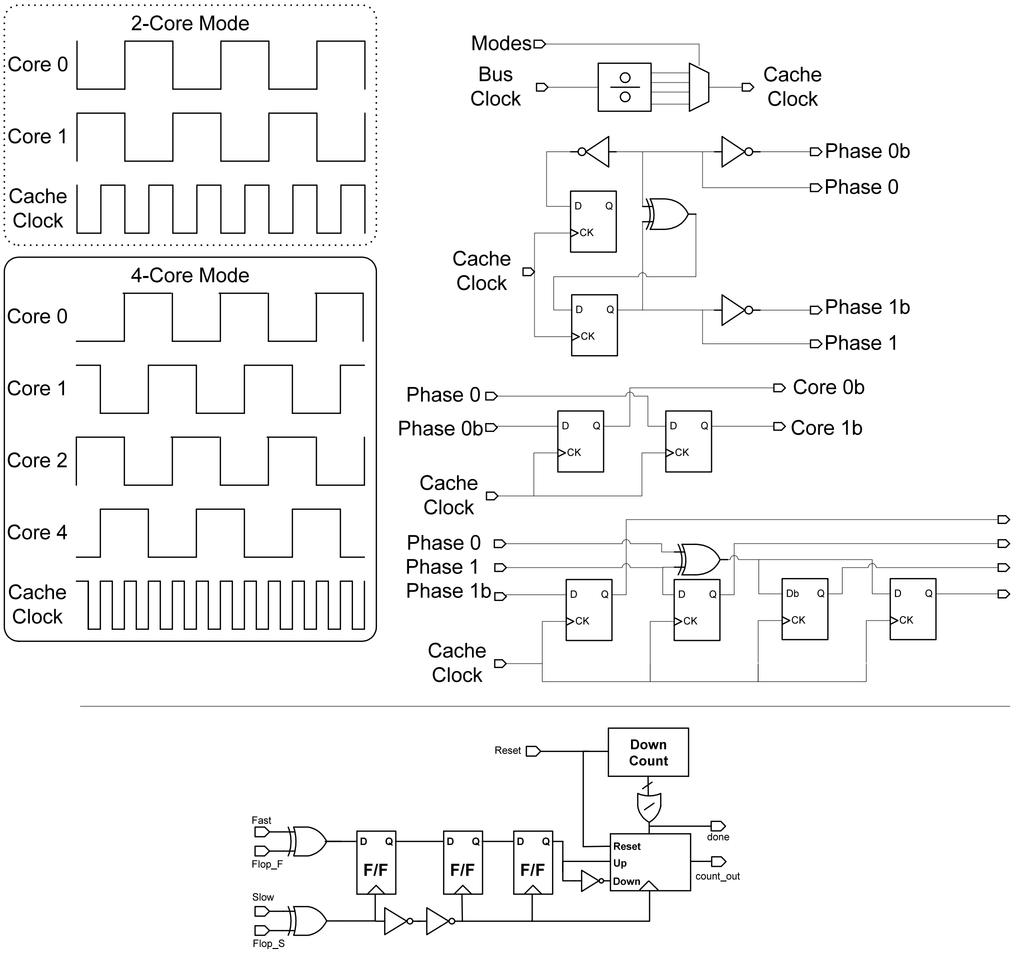


Figure 7.5: Circuit diagrams for level convertors and clock dividers in the Centip3De chip.

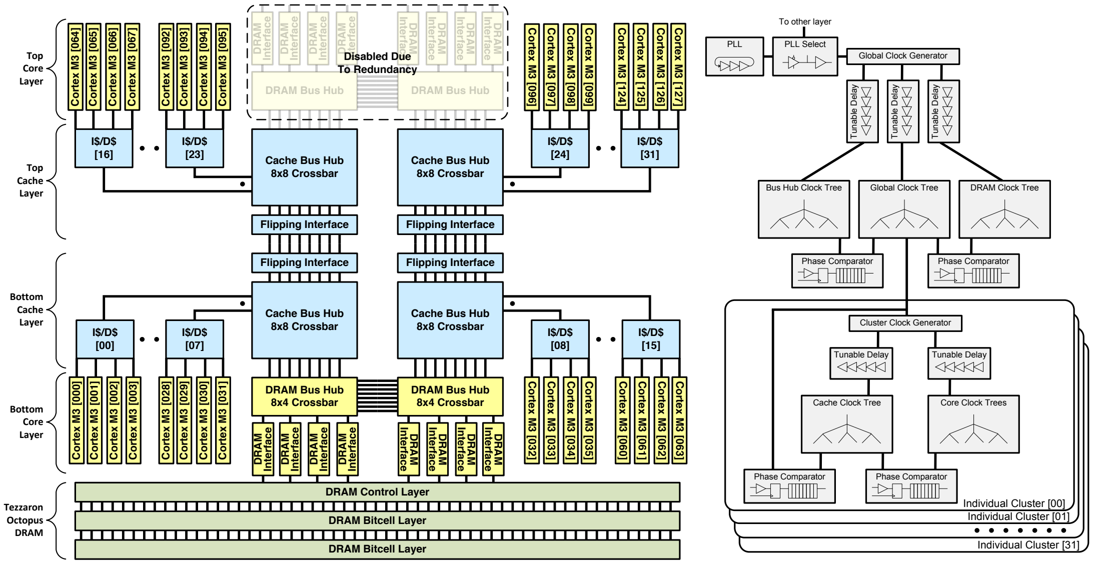


Figure 7.6: Diagram of the clock tree design for the 128 cores, 32 caches, and memory controllers..

These units and level converters are included between the core and cache interfaces, and the cache to bus interfaces and are shown in Figure 7.5. Cache modes, alignment sensor readings, delay generator settings, core enables, etc., are controlled via memory mapped IO in Core 0 of the cluster, accessible via JTAG. The JTAG port for cores 1-3 are enabled in series with Core 0 via control bits in Core 0, so that Cores 1-3 can be controlled via JTAG as well.

7.3 Final Layout

Finally the Centip3De design was prepared for fabrication in a 130nm technology using the Tezzaron 3D TSV technology [31]. Figures 7.7-7.10 show the final layout that was sent for fabrication. Note that the core tiles, consisting of 4 ARM Cortex-M3 cores is the same size as the cache tile, allowing them to be stacked directly above each other in a dense manner.

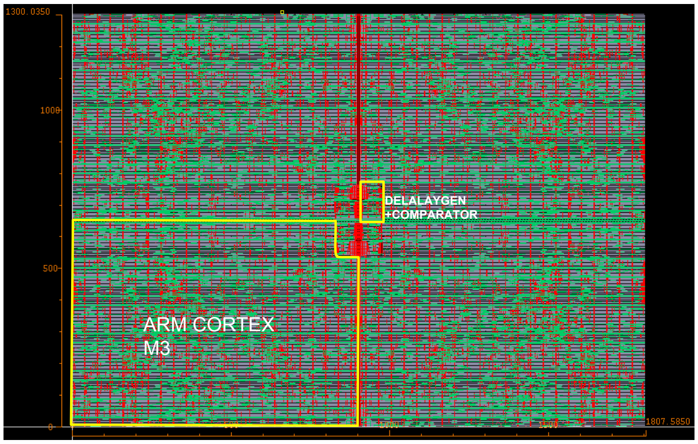


Figure 7.7: Final layout of the 4 core tile used in Centip3De.

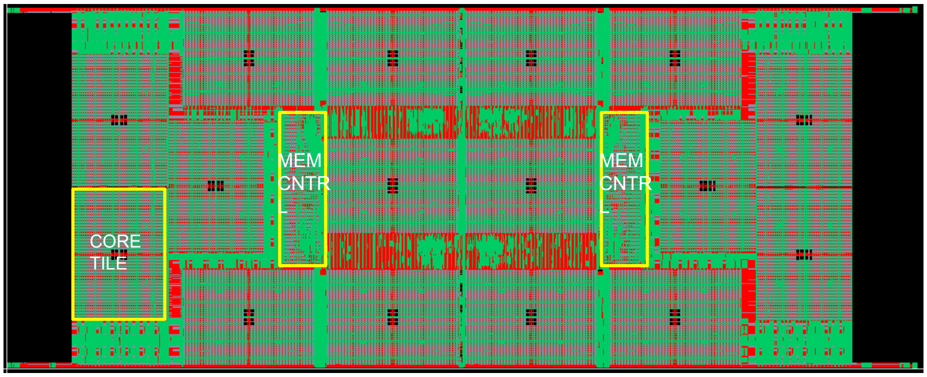


Figure 7.8: Final layout of the entire core layer in Centip3De.

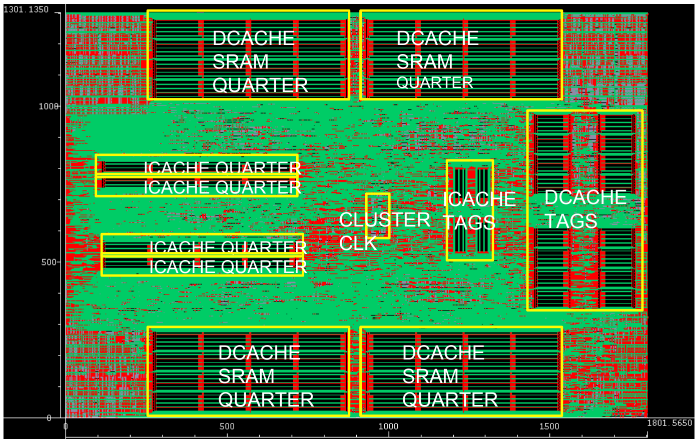


Figure 7.9: Final layout of the cache for each cluster in Centip3De.

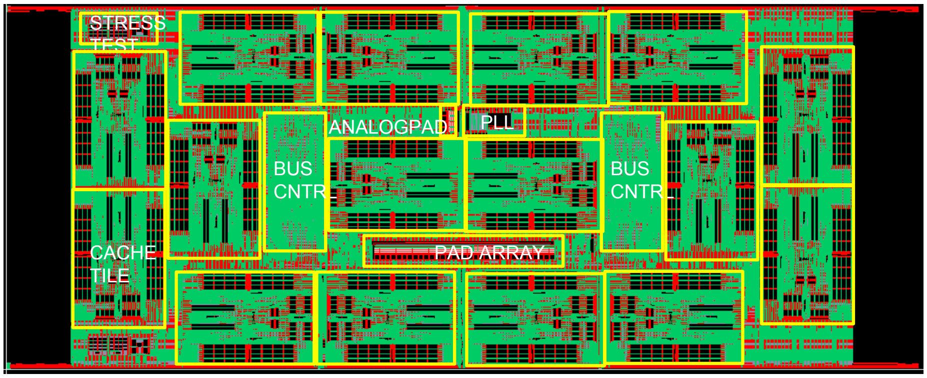


Figure 7.10: Final layout of the entire cache layer in Centip3De.

7.4 Evaluation

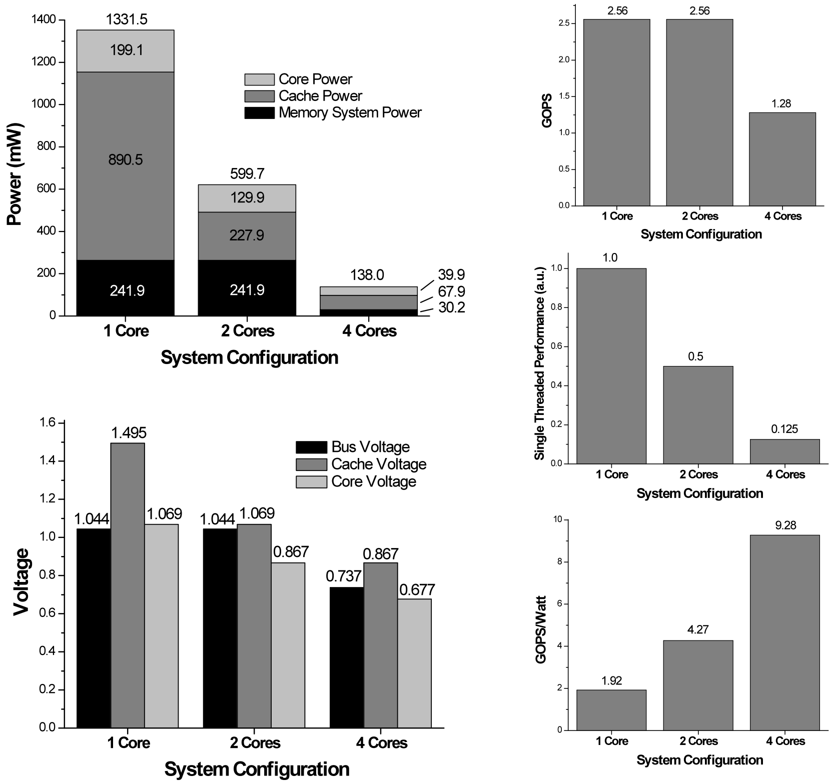


Figure 7.11: Power breakdown, operating voltages, and system analysis for the Centip3De chip in 1, 2, and 4 core mode.

In this work we evaluate three configurations which represent a range of power/performance operating points (Figure 7.3, 7.11) of the design sent for fabrication. At the time of this dissertation the chip is still at fab so post-silicon measurements are not presented. At peak efficiency, Centip3De operates the cores at near-threshold voltages and achieves 9.28 GOPS/Watt. High performance threads can operate in boosted modes which enable one or two threads with the 2-core cache mode. One core and two core boosted modes provide the same throughput, 2.56 GOPS (which is expected to scale to 24GOPS in a 22nm technology), while enabling a trade-off between latency and power (Figure 7.11). Compared to a 45nm implementation of an Intel Atom processor this 130nm Centip3De design achieves 6.4X better energy efficiency (GOPS/W).

CHAPTER 8

Related Work

Each chapter contains related work for that section. This Chapter focuses on the 2 overarching related works most relevant to the entire dissertation.

8.1 Subthreshold Circuits and Architectures

There has been numerous works looking at subthreshold design [101, 93, 23, 67, 14]. This work differs because they focus on much lower performance constraints. For applications that require only kHz of performance, the energy optimal point is lower than the threshold voltage. The energy at slightly higher voltages increases by around 2X, while the performance is an order of magnitude better. This region, near threshold, is the focus of our work.

8.2 Energy-Performance Tradeoff Analysis

The research done to determine optimal architectures and voltages for a given performance target by Azizi et al. [4] is closest in nature to this dissertation. Both seek to find the best architecture and voltage to create an energy-efficient system. However, this dissertation looks at new architectures, while the Azizi work focuses only on traditional architectures. They simply explore simple design knobs, such as issue width and in-order versus out-of-order cores. This work looks to completely redesign the architecture based on the new performance characteristics with the use of techniques such as clustering, which becomes an interesting design option at lower voltages.

CHAPTER 9

Conclusions

As Moore's law continues to provide designers with more transistors on a chip, power budgets are beginning to limit the applicability of these additional transistors in conventional CMOS design. This dissertation looked back at the feasibility of voltage scaling to reduce energy consumption. Although subthreshold operation is well known to provide substantial energy savings it has been relegated to a handful of applications due to the corresponding system performance degradation. The dissertation then turned to the concept of near-threshold computing (NTC), where the supply voltage is at or near the switching voltage of the transistors. This regime enables energy savings on the order of 10X, with only a 10X degradation in performance, providing a much better energy/performance trade-off than subthreshold operation. This dissertation then identified three major barriers to widespread adoption of NTC and current research to overcome them. The three barriers addressed were: 1) performance loss; 2) increased variation; and 3) increased functional failure.

The focus of the remaining chapters of this dissertation was on NTC architectures and techniques to overcome the performance degradation. The architectures included single-core, multi-core, digital signal processing, and many-core architectures along with the addition of voltage boosting techniques to overcome short-lived bottlenecks. The dissertation culminated with the design and fabrication of a prototype system, Centip3De. Centip3De is a 7-layer 3D stacked chip with 128 ARM Cortex-M3 cores and 256MB of DRAM. Analysis of the system in simulation showed that a 130nm implementation of Centip3De has 6.4X better energy efficiency than a Intel Atom chip implemented in 45nm.

In conclusion, with traditional device scaling no longer providing energy efficiency improvements, the solution to this energy crisis is the universal application of aggressive low-voltage operation, namely NTC, across all computation platforms.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] S. Agarwala, P. Koeppen, T. Anderson, A. Hill, M. Ales, R. Damodaran, L. Nardini, P. Wiley, S. Mullinnix, J. Leach, A. Lell, M. Gill, J. Golston, D. Hoyle, A. Rajagopal, A. Chachad, M. Agarwala, R. Castille, N. Common, J. Apostol, H. Mahmood, M. Krishnan, D. Bui, Q.-D. An, P. Groves, L. Nguyen, N. Nagaraj, and R. Simar. A 600mhz vliw dsp. In *Solid-State Circuits Conference, 2002. Digest of Technical Papers. ISSCC. 2002 IEEE International*, volume 2, pages 38 –390, 2002.

[2] D. Albonesi. Selective cache ways: on-demand cache resource allocation. In *Microarchitecture, 1999. MICRO-32. Proceedings. 32nd Annual International Symposium on*, pages 248 –259, 1999.

[3] ARM. Arm cores. `http://www.arm.com/products/CPUs`.

[4] O. Azizi, A. Mahesri, B. C. Lee, S. J. Patel, and M. Horowitz. Energy-performance tradeoffs in processor architecture and circuit design: a marginal cost analysis. In *Proceedings of the 37th annual international symposium on Computer architecture*, ISCA '10, pages 26–36, New York, NY, USA, 2010. ACM.

[5] S. Balakrishnan, R. Rajwar, M. Upton, and K. Lai. The impact of performance asymmetry in emerging multicore architectures. In *Proceedings of the 32nd annual international symposium on Computer Architecture*, ISCA '05, pages 506–517, Washington, DC, USA, 2005. IEEE Computer Society.

[6] R. Balasubramonian, V. Srinivasan, H. Dwarkadas, and A. Buyuktosunoglu. Hot-and-cold: Using criticality in the design of energy-efficient caches. In *In Workshop on Power-Aware Computer Systems, in conjunction with MICRO-36*, 2003.

[7] B. Batson and T. Vijaykumar. Reactive-associative caches. In *Parallel Architectures and Compilation Techniques, 2001. Proceedings. 2001 International Conference on*, pages 49 –60, 2001.

[8] B. Bayer. Color imaging array. *U.S. Patent 3 971 065*, July 1976.

[9] N. Binkert, R. Dreslinski, L. Hsu, K. Lim, A. Saidi, and S. Reinhardt. The M5 simulator: Modeling networked systems. *IEEE Micro*, 26(4):52–60, Jul/Aug 2006.

[10] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt. The m5 simulator: Modeling networked systems. *IEEE Micro*, 26(4):52–60, 2006.

[11] W. Bircher and L. John. Power phase variation in a commercial server workload. In *Low Power Electronics and Design, 2006. ISLPED'06. Proceedings of the 2006 International Symposium on*, pages 350 –353, oct. 2006.

[12] G. Blake, R. G. Dreslinski, and T. Mudge. Proactive transaction scheduling for contention management. In *MICRO 42: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 156–167, New York, NY, USA, 2009. ACM.

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

[14] B. Calhoun and A. Chandrakasan. Characterization and modeling minimum energy operation for subthreshold circuits. In *Proc. 2004 Int'l Symp. on Low-Power Electronics and Design*, 2004.

[15] B. Calhoun and A. Chandrakasan. Characterizing and modeling minimum energy operation for subthreshold circuits. In *IEEE/ACM ISLPED*, 2004.

[16] B. Calhoun and A. Chandrakasan. A 256kb sub-threshold sram in 65nm cmos. In *IEEE ISSCC*, 2006.

[17] C. Cao Minh, J. Chung, C. Kozyrakis, and K. Olukotun. STAMP: Stanford transactional applications for multi-processing. In *IISWC '08: Proceedings of The IEEE International Symposium on Workload Characterization*, September 2008.

[18] L. Chang, Y. Nakamura, R. Montoye, J. Sawada, A. Martin, K. Kinoshita, F. Gebara, K. Agarwal, D. Acharyya, W. Haensch, K. Hosokawa, and D. Jamsek. A 5.3ghz 8t-sram with operation down to 0.41v in 65nm cmos. In *VLSI Circuits, 2007 IEEE Symposium on*, pages 252 –253, june 2007.

[19] G. Chen, D. Blaauw, N. S. Kim, T. Mudge, and D. Sylvester. Yield-driven near-threshold sram design. In *Proc. 2007Int'l Conf.on Computer Aided Design*, 2007.

[20] G. Chen, D. Sylvester, D. Blaauw, and T. Mudge. Yield-driven near-threshold sram design. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, PP(99):1 –1, 2009.

[21] J. Chen and S.-Y. Chien. Crisp: Coarse-grained reconfigurable image stream processor for digital still cameras and camcorders. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(9):1223 –1236, sept. 2008.

[22] C. Chute. Worldwide digital still camera 20092013 forecast. *IDC*, Apr. 2009.

[23] G. De Vita and G. Iannaccone. Ultra-low-power series voltage regulator for passive rfid transponders with subthreshold logic. *Electronics Letters*, 42(23):1350 –1351, november 2006.

[24] R. Dennard, F. Gaensslen, V. Rideout, E. Bassous, and A. LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *Solid-State Circuits, IEEE Journal of*, 9(5):256 – 268, oct 1974.

[25] R. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge. Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits. *Proceedings of the IEEE*, 98(2):253 –266, feb. 2010.

[26] R. Dreslinski, B. Zhai, T. Mudge, D. Blaauw, and D. Sylvester. An energy efficient parallel architecture using near threshold operation. In *Proc. 16th Ann. Int'l Conf. on Parallel Architectures and Compilation Techniques*, 2007.

[27] R. G. Dreslinski, G. K. Chen, T. Mudge, D. Blaauw, D. Sylvester, and K. Flautner. Reconfigurable energy efficient near threshold cache architectures. *Microarchitecture, IEEE/ACM International Symposium on*, 0:459–470, 2008.

[28] M. Ekman and P. Stenstrom. Performance and power impact of issue-width in chip-multiprocessor cores. In *Parallel Processing, 2003. Proceedings. 2003 International Conference on*, pages 359 –368, oct. 2003.

[29] S. Fujii and T. Sato. Non-uniform set-associative caches for power-aware embedded processors. In *LNCS: Embedded and Ubiquitous Computing*, 2004.

[30] G. Gammie, A. Wang, M. Chau, S. Gururajarao, R. Pitts, F. Jumel, S. Engel, P. Royannez, R. Lagerquist, H. Mair, J. Vaccani, G. Baldwin, K. Heragu, R. Mandal, M. Clinton, D. Arden, and U. Ko. A 45nm 3.5g baseband-and-multimedia application processor using adaptive body-bias and ultra-low-power techniques. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 258 –611, feb. 2008.

[31] S. Gupta, M. Hilbert, S. Hong, and R. Patti. Techniques for producing 3D ICs with high-density interconnect. `www.tezzaron.com/about/papers/ieeevmic2004finalsecure.pdf`.

[32] M. Guthaus, J. Ringenberg, T. Austin, T. Mudge, and R. Brown. Mibench: A free, commercially representative embedded benchmark suite. In *4th Annual Workshop on Workload Characterization*, Dec. 2001.

[33] S. Hanson, B. Zhai, K. Bernstein, D. Blaauw, A. Bryant, L. Chang, K. Das, W. Haensch, E. Nowak, and D. Sylvester. Ultra low-voltage, minimum energy cmos. In *IBM Journal of Research Development*, page 469, 2006.

[34] S. Hanson, B. Zhai, M. Seok, B. Cline, K. Zhou, M. Singhal, M. Minuth, J. Olson, L. Nazhandali, T. Austin, D. Sylvester, and D. Blaauw. Performance and variability

optimization strategies in a sub-200mv, 3.5pj/inst, 11nw subthreshold processor. In *VLSI Circuits, 2007 IEEE Symposium on*, pages 152 –153, june 2007.

[35] M. Harchol-Balter and A. B. Downey. Exploiting Process Lifetime Distributions for Dynamic Load Balancing. *SIGMETRICS*, 1995.

[36] J. Henning. SPEC CPU2000: Measuring CPU performance in the new millennium. *IEEE Computer*, 33(7):28–35, July 2000.

[37] X. Huang, W.-C. Lee, C. Kuo, D. Hisamoto, L. Chang, J. Kedzierski, E. Anderson, H. Takeuchi, Y.-K. Choi, K. Asano, V. Subramanian, T.-J. King, J. Bokor, and C. Hu. Sub-50 nm p-channel finfet. *Electron Devices, IEEE Transactions on*, 48(5):880 –886, may 2001.

[38] J. Huh, D. Burger, and S. Keckler. Exploring the design space of future cmps. In *Parallel Architectures and Compilation Techniques, 2001. Proceedings. 2001 International Conference on*, pages 199 –210, 2001.

[39] IBM. Ibm powerpc. `http://www.chips.ibm.com/products/powerpc/`.

[40] M. IDC Farmingham. Idc's worldwide installed base forecast, 2007 -2010. *IDC*, Mar. 2007.

[41] K. Illgner, H.-G. Gruber, P. Gelabert, J. Liang, Y. Yoo, W. Rabadi, and R. Talluri. Programmable dsp platform for digital still cameras. In *Acoustics, Speech, and Signal Processing, 1999. ICASSP '99. Proceedings., 1999 IEEE International Conference on*, volume 4, pages 2235 –2238 vol.4, mar 1999.

[42] K. Inoue, T. Ishihara, and K. Murakari. Way-predicting set-associative cache for high perfromance and low energy consumption. In *Proc. 1999 Int'l Symp. on Low-Power Electronics and Design*, 1999.

[43] N. Integration and M. N. Group. Predictive technology model (ptm), [online]. *Available: http://www.eas.asu.edu/˜ptm/. [Accessed Nov. 16, 2009]*, 2009.

[44] Intel. Intel xscale. `http://www.intel.com/design/intelxscale/`.

[45] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *ISLPED '98: Proceedings of the 1998 international symposium on Low power electronics and design*, pages 197–202, New York, NY, USA, 1998. ACM.

[46] V. Joshi, D. Blaauw, and D. Sylvester. Soft-edge flip-flops for improved timing yield: design and optimization. In *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, pages 667 –673, nov. 2007.

[47] R. Katz. Research directions in internet-scale computing. *Proc. 3rd Int. Week Management of Networks and Services*, 2007.

[48] C.-I. Kim, H. Soeleman, and K. Roy. Ultra-low-power dlms adaptive filter for hearing aid applications. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 11(6):1058 – 1067, dec. 2003.

[49] N. S. Kim, K. Flautner, D. Blaauw, and T. Mudge. Single-vdd and single-vt super-drowsy techniques for low-leakage high-performance instruction caches. In *IEEE/ACM ISLPED*, 2004.

[50] T.-H. Kim, J. Liu, J. Keane, and C. H. Kim. A high-density subthreshold sram with data-independent bitline leakage and virtual-ground replica scheme. In *IEEE ISSCC*, 2007.

[51] W. Kim, M. Gupta, G.-Y. Wei, and D. Brooks. System level analysis of fast, per-core dvfs using on-chip switching regulators. In *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pages 123 –134, feb. 2008.

[52] J. Kin, M. Gupta, and B. Mangione-Smith. The filter cache: An energy efficient memory structure. In *27th Ann. Int'l Symp. on Microarchitecture*, 1997.

[53] R. Kumar, K. Farkas, N. Jouppi, P. Ranganathan, and D. Tullsen. Single-isa heterogeneous multi-core architectures: the potential for processor power reduction. In *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pages 81 – 92, December 2003.

[54] R. Kumar, K. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Processor power reduction via single-isa heterogeneous multi-core architectures. *IEEE Comput. Archit. Lett.*, 2:2–, January 2003.

[55] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas. Single-isa heterogeneous multi-core architectures for multithreaded workload performance. In *Proceedings of the 31st annual international symposium on Computer architecture*, ISCA '04, pages 64–, Washington, DC, USA, 2004. IEEE Computer Society.

[56] H. Li, C.-Y. Cher, T. Vijaykumar, and K. Roy. Vsv: L2-miss-driven variable supply-voltage scaling for low power. In *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pages 19 – 28, 3-5 2003.

[57] J. Li and J. Martinez. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*, pages 77 – 87, feb. 2006.

[58] Y. Li, K. Skadron, D. Brooks, and Z. Hu. Performance, energy, and thermal considerations for smt and cmp architectures. In *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*, pages 71 – 82, feb. 2005.

[59] Y. Lin, H. Lee, M. Who, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner. Soda: A low-power architecture for software radio. In *Computer Architecture, 2006. ISCA '06. 33rd International Symposium on*, pages 89 –101, 0-0 2006.

[60] R. Lyon and C. Mead. An analog electronic cochlea. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 36(7):1119 –1134, jul 1988.

[61] C. Mead. Analog vlsi and neural systems, 1989.

[62] D. Meisner and T. F. Wenisch. Stochastic Queuing Simulation for Data Center Workloads. *EXERT '10: Exascale Evaluation and Research Techniques*, 2010.

[63] G. Moore. No exponential is forever: but "forever" can be delayed! [semiconductor industry]. In *Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC. 2003 IEEE International*, pages 20 – 23 vol.1, 2003.

[64] A. Moshovos. Regionscout: exploiting coarse grain sharing in snoop-based coherence. In *Computer Architecture, 2005. ISCA '05. Proceedings. 32nd International Symposium on*, pages 234 – 245, june 2005.

[65] A. Moshovos, G. Memik, B. Falsafi, and A. Choudhary. Jetty: filtering snoops for reduced energy consumption in smp servers. In *High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on*, pages 85 –96, 2001.

[66] N. Nakano, R. Nishimura, H. Sai, A. Nishizawa, and H. Komatsu. Digital still camera system for megapixel ccd. *Consumer Electronics, IEEE Transactions on*, 44(3):581 –586, aug 1998.

[67] L. Nazhandali, M. Minuth, B. Zhai, J. Olson, T. Austin, and D. Blaauw. A second-generation sensor network processor with application-driven memory optimizations and out-of-order execution. In *ACM/IEEE Int. Conf. Compilers, Archit., Synthesis Embedded Syst.*, 2005.

[68] Online. Report to congress on server and data center energy efficiency. *Online*, 2006. `http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf`.

[69] B. Paul, H. Soeleman, and K. Roy. An 8x8 sub-threshold digital cmos carry save array multiplier. In *Solid-State Circuits Conference, 2001. ESSCIRC 2001. Proceedings of the 27th European*, pages 377 –380, sept. 2001.

[70] M. Pelgrom, A. Duinmaijer, and A. Welbers. Matching properties of mos transistors. *Solid-State Circuits, IEEE Journal of*, 24(5):1433 – 1439, oct 1989.

[71] T. Pering, T. Burd, and R. Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms. In *Low Power Electronics and Design, 1998. Proceedings. 1998 International Symposium on*, pages 76 –81, aug. 1998.

[72] D. Phillips. Image processing in c. *R&D Publications Inc.*, 1994.

[73] M. Powell, A. Agrawal, T. N. Vijaykumar, B. Falsafi, and K. Roy. Reducing set-associative cache energy via way-predicition and selective direct mapping. In *34th Ann. Int'l Symp. on Microarchitecture*, 2002.

[74] S. Romanovsky, A. Katoch, A. Achyuthan, C. O'Connell, S. Natarajan, C. Huang, C.-Y. Wu, M.-J. Wang, C. Wang, P. Chen, and R. Hsieh. A 500mhz random-access embedded 1mb dram macro in bulk cmos. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 270 –612, feb. 2008.

[75] C. J. Rossbach, O. S. Hofmann, and E. Witchel. Is transactional programming actually easier? In *Proceedings of the 15th ACM SIGPLAN symposium on Principles and practice of parallel programming*, PPoPP '10, pages 47–56, New York, NY, USA, 2010. ACM.

[76] T. Sakurai and A. Newton. Alpha-power law mosfet model and its applications to cmos inverter delay and other formulas. *Solid-State Circuits, IEEE Journal of*, 25(2):584 –594, apr 1990.

[77] E. Schurman and J. Brutlag. The user and business impact of server delays, additional bytes, and http chunking in web search. *Velocity*, 2009.

[78] J. Seng, D. Tullsen, and G. Cai. Power-sensitive multithreaded architecture. In *Computer Design, 2000. Proceedings. 2000 International Conference on*, pages 199 –206, 2000.

[79] S. Seo, R. G. Dreslinski, M. Woh, C. Chakrabarti, S. Mahlke, and T. Mudge. Diet soda: a power-efficient processor for digital cameras. In *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*, ISLPED '10, pages 79–84, New York, NY, USA, 2010. ACM.

[80] M. Seok, S. Hanson, Y.-S. Lin, Z. Foo, D. Kim, Y. Lee, N. Liu, D. Sylvester, and D. Blaauw. The phoenix processor: A 30pw platform for sensor applications. In *VLSI Circuits, 2008 IEEE Symposium on*, pages 188 –189, june 2008.

[81] H. Soeleman and K. Roy. Ultra-low power digital subthreshold logic circuits. In *Low Power Electronics and Design, 1999. Proceedings. 1999 International Symposium on*, pages 94 – 96, 1999.

[82] SPEC. Spec web 2005, 2005. `http://www.spec.org/web2005`.

[83] M. A. Suleman, O. Mutlu, M. K. Qureshi, and Y. N. Patt. Accelerating critical section execution with asymmetric multi-core architectures. In *ASPLOS '09: Proceeding of the 14th international conference on Architectural support for programming languages and operating systems*, pages 253–264, New York, NY, USA, 2009. ACM.

[84] R. Swanson and J. Meindl. Ion-implanted complementary mos transistors in low-voltage circuits. *Solid-State Circuits, IEEE Journal of*, 7(2):146 – 153, apr 1972.

[85] D. Talla, C.-Y. Hung, R. Talluri, F. Brill, D. Smith, D. Brier, B. Xiong, and D. Huynh. Anatomy of a portable digital mediaprocessor. *Micro, IEEE*, 24(2):32 – 39, mar-apr 2004.

[86] W. Tang, R. Gupta, and A. Nicolau. Design of a predictive filter cache for energy savings in high performance processor architectures. In *Proceedings of the 2001 International Conference on Computer Design (ICCD)*, 2001.

[87] A. W. Topol, J. D. C. L. Tulipe, L. Shi, D. J. Frank, K. Bernstein, S. E. Steen, A. Kumar, G. U. Singco, A. M. Young, K. W. Guarini, and M. Ieong. Three-dimensional integrated circuits. In *IBM Journal of Research Devekopment*, volume 50, page 491, 2006.

[88] Transmeta. Transmeta crusoe. `http://www.transmeta.com/`.

[89] K. Usami, M. Igarashi, T. Ishikawa, M. Kanazawa, M. Takahashi, M. Hamada, H. Arakida, T. Terazawa, and T. Kuroda. Design methodology of ultra low-power mpeg4 codec core exploiting voltage scaling techniques. In *DAC '98: Proceedings of the 35th annual Design Automation Conference*, pages 483–488, New York, NY, USA, 1998. ACM.

[90] 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 *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pages 98 –589, feb. 2007.

[91] N. Verma and A. Chandrakasan. A 65nm 8t sub-vt sram employing sense-amplifier redundancy. In *IEEE ISSCC*, 2007.

[92] E. Vittoz and J. Fellrath. Cmos analog integrated circuits based on weak inversion operations. *Solid-State Circuits, IEEE Journal of*, 12(3):224 – 231, jun 1977.

[93] A. Wang and A. Chandrakasan. A 180mv fft processor using subthreshold circuit techniques. In *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*, pages 292 – 529 Vol.1, 15-19 2004.

[94] A. Wang and A. Chandrakasan. A 180mv fft processor using subthreshold circuits techniques. In *IEEE ISSCC*, 2004.

[95] M. Wieckowski, Y. M. Park, C. Tokunaga, D. W. Kim, Z. Foo, D. Sylvester, and D. Blaauw. Timing yield enhancement through soft edge flip-flop based design. In *Custom Integrated Circuits Conference, 2008. CICC 2008. IEEE*, pages 543 –546, sept. 2008.

[96] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The splash-2 programs: characterization and methodological considerations. In *ISCA '95: Proceedings of the 22nd annual international symposium on Computer architecture*, pages 24–36, New York, NY, USA, 1995. ACM.

[97] H. Zen, T. Koizumi, H. Yamamoto, and I. Kimura. A new digital signal processor for progressive scan ccd. *Consumer Electronics, IEEE Transactions on*, 44(2):289 –296, may 1998.

[98] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner. Theoretical and practical limits of dynamic voltage scaling. In *DAC*, 2004.

[99] B. Zhai, D. Blaauw, D. Sylvester, and S. Hanson. A sub-200mv 6t sram in 0.13um cmos. In *IEEE ISSCC*, 2007.

[100] B. Zhai, R. Dreslinski, D. Blaauw, T. Mudge, and D. Sylvester. Energy efficient near threshold chip multiprocessing. In *Proc. 2007 Int'l Symp. on Low-Power Electronics and Design*, 2007.

[101] B. Zhai, S. Hanson, D. Blaauw, and D. Sylvester. Analysis and mitigation of variability in subthreshold design. In *Proc. 2005 Int'l Symp. on Low-Power Electronics and Design*, 2005.

[102] B. Zhai, L. Nazhandali, J. Olson, A. Reeves, M. Minuth, R. Helfand, S. Pant, D. Blaauw, and T. Austin. A 2.60pj/inst subthreshold sensor processor for optimal energy efficiency. *IEEE VLSI Technology and Circuits*, 2006.

[103] C. Zhang, X. Zhang, and Y. Yan. Two fast and high-associativity cache schemes. In *IEEE Micro*, pages 40–49, Sep/Oct 1997.

[104] C. Zheng, F. Vahid, and W. Najjar. A highly configurable cache architecture for embedded systems. In *Proc. 30th Ann. Int'l Symp. on Computer Architecture*, 2003.

[105] Z. Zhu and X. Zhang. Access-mode predictions for low-power cache design. In *IEEE Micro*, volume 22, pages 58–71, Mar/Apr 2002.