Optimal Digital System Design in Deep Submicron Technology

by

Seongmoo Heo

Bachelor of Science, Electrical Engineering,
Korea Advanced Institute of Science and Technology (1998)

Master of Science, Electrical Engineering and Computer Science,
Massachusetts Institute of Technology (2000)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 2006
[February 2006]

Author ....
Department of Electrical Engineering and Computer Science
January 13, 2006

Certified by ....
Krste Asanović
Associate Professor
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Optimal Digital System Design in Deep Submicron Technology

by

Seongmoo Heo

Submitted to the Department of Electrical Engineering and Computer Science
on January 13, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

The optimization of a digital system in deep submicron technology should be done with two basic principles: energy waste reduction and energy-delay tradeoff. Increased energy resources obtained through energy waste reduction are utilized through energy-delay tradeoffs.

The previous practice of obliviously pursuing performance has led to the rapid increase in energy consumption. While energy waste due to unnecessary switching could be reduced with small increases in logic complexity, leakage energy waste still remains as a major design challenge. We find that fine-grain dynamic leakage reduction (FG-DLR), turning off small subblocks for short idle intervals, is the key for successful leakage energy saving. We introduce an FG-DLR circuit technique, *Leakage Biasing*, which uses leakage currents themselves to bias the circuit into the minimum leakage state, and apply it to primary SRAM arrays for bitline leakage reduction (*Leakage-Biased Bitlines*) and to domino logic (*Leakage-Biased Domino*). We also introduce another FG-DLR circuit technique, *Dynamic Resizing*, which dynamically downsizes transistors on idle paths while maintaining the performance along active critical paths, and apply it to static CMOS circuits.

We show that significant energy reduction can be achieved at the same computation throughput and communication bandwidth by pipelining logic gates and wires. We find that energy saved by pipelining datapaths is eventually limited by latch energy overhead, leading to a power-optimal pipelining. Structuring global wires into on-chip networks provides a better environment for pipelining and leakage energy saving. We show that the energy-efficiency increase through replacement with dynamically packet-routed networks is bounded by router energy overhead.

Finally, we provide a way of relaxing the peak power constraint. We evaluate the use of *Activity Migration* (AM) for hot spot removal. AM spreads heat by transporting computation to a different location on the die. We show that AM can be used either to increase the power that can be dissipated by a given package, or to lower the operating temperature and hence the operating energy.

Thesis Supervisor: Krste Asanović
Title: Associate Professor

Acknowledgments

First of all, I'd like to thank my advisor, Krste Asanović, deeply for his inspiring advice and guidance and also for spending a great deal of time and energy for this thesis. Also I'd like to thank my thesis readers, Prof. Chandrakasan and Prof. Agarwal for the advices.

I thank my group mates, Ken Barr, Elizabeth Basha, Chris Batten, Steve Gerding, Mark Hampton, Ronny Krashinsky, Rose Liu, Albert Ma, Heidi Pan, Jessica Tseng, and Mike Zhang for helpful discussions. Special and extra thanks to my officemates, Mike, Jessica, Ronny and Albert, for sharing all the ups and downs of MIT grad student life together.

I thank Mary McDavitt so much for all the cares and encouragements throughout the last years of my graduate work. I thank Ken and his Sydney Grill bar for great beers and chats. I thank Prof. and Mrs. Lim for the advices on research and life.

Thanks to my MIT Korean friends: Daihyun, Jaewook, Yongwook, Junghoon, Gookwon, Hyunil, Sangwon, Sungho, Joonsung, Minkyu, Jungwon, Sejun, Myungjin, Taeksang, Min, Kyungbum, Jaeyeon, Chungyeon, Jihwan, Junmo, Daekeun, Jungin, Hyeyeon, Taesik, Seonghwan, Yongseok, Hyunjung, Sungtae, Taehong, Sungjun, Soonmin, Taeyi, Jhongwoo, Choonghyun, Hohyun, Hyemin, Jerin, Youjin, Jinyoung, Jooeun, Julie, Jungkeun, Jongchul, Miso, Kwanhong, Meekyoung, Sangil, Seungeun, Tairin, Wonyong, Yoonsung, Yoonsun, and many others.

Thanks to all friends from First Korean Church in Cambridge: Pastor. and Mrs. Kim, Mr. and Mrs. Han, Mr. and Mrs. Sung, David and Kelly, Jin, Jieha, Sunah, Jouwon, Taein, Yejeon, Eunhye, Hyeeun, Hyeju, Inyoung, Jaehoon, Jaewoo, Jihyung, Jinhee, Jiyoung, Junyoung, Kyunghee, Rayoung family, Seungwon, Seoyoung, Jaeseop, Chul, and many others. Very special thanks to Jin, Jieha, Taein, and Yejeon for all the loves and cares.

Funding for my graduate work came from a number of sources including KFAS (Korea Foundation for Advanced Studies), DARPA Grant N66001-99-2-8917, DARPA PAC/C award F30602-00-2-0562, NSF CAREER award CCR-0093354, NSF ITR award CCR-0219545, and donations from Infineon Technologies, Intel Corporation, and MOSIS Educational Program (MEP).

Finally, I want to thank my wonderful parents for supporting me, loving me, and believing in me.

어머니 아버지 감사합니다!

Chapter 1

Introduction

The advent of deep submicron technology is an exciting and challenging time for digital system designers. Emerging applications require huge amounts of computation and memory bandwidth and while deep submicron technology allows us to place billions of transistors on a single chip, thermal and power constraints are ever tightening and thus the efficient use of energy resources is more important than ever. Also, the surge in leakage current has made leakage power an important design challenge for digital system design.

The traditional optimization of a digital system often focused on only a single parameter: either improving performance or saving energy. Initially, performance was the sole goal. Energy consumption was insignificant and most innovations were unaware of energy, resulting in the rapid growth of both useful energy consumption and energy waste in digital systems. Recently, however, with the peak power reaching the thermal limit of transistors and increasing demands for mobility, power (or energy) has become a primary metric for digital system design. Though technology scaling has automatically provided around quadratic reduction of switching power for the same circuit blocks, supply voltage has scaled down sub-linearly. Die size has grown to include many sophisticated architectural innovations and soon will reach a few billion transistors on a single chip. As a result, typical power consumption of many high-performance digital systems has reached many tens of watts, which requires expensive cooling to prevent the threat of thermal emergency.

This power crisis has led to many diverse low-energy techniques. Low-energy techniques first focused on energy waste from spurious switching of parasitic capacitances. Since previous design practices were energy-unaware, even simple techniques could achieve large energy saving by cutting energy waste. However, now that most switching energy waste is well controlled, energy reduction must be done through energy-delay tradeoffs. The same principle applies to performance improvement. As energy resources are precious, innovations intended for high performance should fully understand the energy-delay curve's behavior. In this deep submicron technology era, obliviously pursuing only high performance or low-energy results in a sub-optimal design.

Though switching energy waste is well managed in modern digital systems, there is still significant leakage energy waste. Though leakage energy waste was trivial in previous technology generations, the continuation of aggressive technology scaling and resulting supply and threshold voltage scaling have led to the exponential increase of leakage current, making leakage waste energy comparable to switching energy. Thus, the development of techniques for leakage energy waste reduction is crucial for optimal digital system design.

With high performance requirements, precious energy resources, significant leakage, and hot die temperatures, optimization of digital systems has become more challenging and critical than ever. We claim that there are two basic principles for digital system optimization in deep submicron technology: energy waste reduction and energy-delay tradeoff. Increased energy resources obtained through energy waste reduction are utilized through energy-delay tradeoff.

1.1 Energy Waste Reduction

Energy waste in digital systems results from the simplicity of design. Thus, we can reduce energy waste by turning a simple and energy-inefficient design into a complex but energy-efficient design. Figure 1-1 illustrates the impact of an ideal energy waste reduction innovation on energy and execution time.

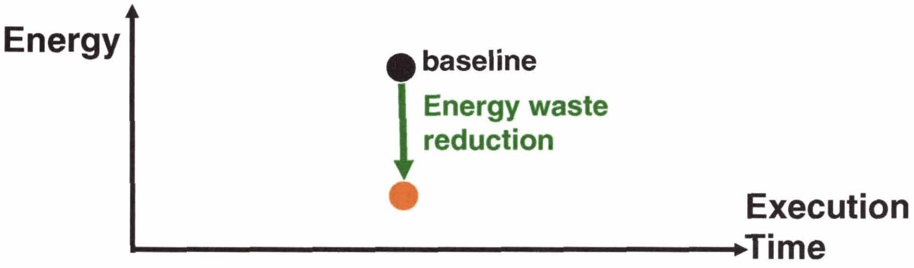


Figure 1-1: Energy waste reduction.

Energy waste is caused by spurious switching of parasitic capacitances or transistor leakage. The tight control of switching and leakage has been one of the critical themes for recent architectural innovations in digital systems. While modern energy-conscious digital systems have successfully minimized switching energy waste with the help of fine-grain clock gating for clock trees and bitline/wordline gating for SRAM arrays, the rapidly-growing leakage energy waste has not been as effectively reduced.

1.2 Energy-Delay Tradeoff

Though ideally we would like to have innovations which decrease both energy and delay at the same time, most high-performance or low-energy innovations are, in essence, energy-delay tradeoffs, even though intended for either increasing performance or decreasing energy. The essential difference between high-performance and low-energy innovations is the direction of trade. High-performance innovation increases energy consumption for a performance gain and a low-energy innovation saves energy while sacrificing performance (Figure 1-2). In fact, many innovations can be used in either direction. For example, supply voltage can be up-scaled or down-scaled to find the optimal level where the delay requirement is tightly met and the energy consumption is minimized. Similarly, the issue width of a superscalar processor can be increased for improving performance or decreased for reducing hardware complexity and saving energy.

Figure 1-2: Energy-delay tradeoffs: optimizing for either greater performance or lower energy.

The future digital systems require an urgent change in design philosophy: from purely high-performance or purely low-energy design to an energy-efficient design using energy-delay tradeoffs.

1.3 Contributions of Thesis

The major contributions of this thesis are:

- Fine-Grain Dynamic Leakage Reduction (Chapter 5): We find that leakage energy waste is the most critical target for energy waste reduction and claim that fine-grain dynamic leakage reduction, turning off small pieces for short idle intervals, is the key for continuing leakage reduction. We introduce a FG-DLR technique, *Leakage Biasing* (LB) which uses leakage currents themselves to bias the circuit into the minimum leakage state and hence has low transition overheads. We successfully apply LB to bitline leakage reduction in primary SRAM arrays (*Leakage-Biased Bitlines*) with three microarchitectural deactivation policies and to domino logic, through *Leakage-Biased Domino*. We also introduce *Dynamically Resizable CMOS* (DRCMOS), which dynamically downsizes transistors on idle paths in static CMOS logic.

- Pipelining Logic Gates and Wires (Chapter 6 and Chapter 7): We show that energy reduction for high-performance ASIC systems can be achieved with the same computation throughput and communication bandwidth by pipelining logic gates and global wires and then employing other energy-delay tradeoffs. We find that power reduction from pipelining datapaths is eventually limited by the latch energy overhead. We show that structuring global wires into a network provides a better environment for global wire optimization (e.g. pipelining). We find that the energy-efficiency increase through dynamically packet-routed network is bounded by router energy overhead.

- Hot Spot Removal with *Activity Migration* (Chapter 8): Dynamic thermal management relaxes the peak power constraint. We present a power density reduction technique, *activity migration* (AM), for hot spot removal. With AM, we spread heat by transporting computation to different locations on the die. We show how AM can be used either to increase the power that can be dissipated by a given package, or to lower the operating temperature and hence the operating energy.

1.4 Thesis Overview

This thesis is an exploration of optimal digital system design in deep submicron CMOS technology focusing on circuit- and architectural-level innovations. While SOI technology for improving CMOS performance has been under research for nearly three decades [CBF00] and is now in production, bulk CMOS technology is by far the most dominant process. We consider only bulk technology throughout the thesis. Among various levels of digital system optimization, we focus on circuit- and architectural-levels. Innovations in technology, system software, and applications are significant, but are beyond the scope of the thesis. Our main target platforms are high-performance and energy-conscious general-purpose microprocessors, and application-specific integrated chips (ASICs) for digital signal processing (DSP) applications. However, the concepts and techniques of many innovations we describe and develop in this thesis can be easily applied to other digital system platforms.

Chapter 2 provides background for this work. We first describe how to compare and combine energy-delay tradeoffs. Then we move onto power and thermal issues. In spite of continuing technology scaling, power consumption of high-performance digital systems has grown rapidly and now power is a primary metric for any digital system design. We describe two power constraints: peak power and average power. The drastic power increase has led to many low-power techniques. We find that low-power techniques trade increased delay for lower energy, or reduce energy waste, and that switching energy waste has been well managed by clock gating for clock trees and bitline/wordline gating for SRAM arrays. We present an overview of subthreshold leakage. The rapid and continuing reductions in feature size and threshold voltage have brought exponential increases in leakage. The increase of power densities has brought significant increases in die temperature. We examine the thermal behavior of digital systems and then discuss how we can overcome the thermal limit through dynamic thermal management to effectively increase the available power budget.

Chapter 3 and Chapter 4 categorize circuit- and architectural innovations for digital systems respectively, from the perspective of energy waste reduction and energy-delay tradeoff. We observe that transistors' dimensions and voltages are the two main tuning variables for circuit designs, and categorize circuit techniques according to how they tune the variables while studying the energy-delay tradeoffs with simple test circuits. We find that most architectural innovations are based on three critical techniques: exploiting parallelism, utilizing predictability, and reducing energy waste, and categorize the architectural innovations according to the techniques while analyzing their impacts on energy and delay.

Chapter 5 describes techniques for fine-grain dynamic leakage reduction (FG-DLR). We first categorize existing leakage reduction techniques into static and dynamic approaches and find that leakage on critical paths becomes dominant after static leakage reduction techniques are applied. We discuss candidates for FG-DLR and parameters for comparing different FG-DLR techniques. We introduce a FG-DLR circuit technique, *leakage biasing* (LB). LB uses leakage currents themselves to bias the circuit into the minimal leakage state. We first apply LB to primary SRAM arrays to reduce the bitline leakage, which we call *leakage-biased bitlines* (LBB). LBB dynamically biases the bitlines of unused memory subbanks into a low-leakage state. LBB has no performance impact and has very low transition energy overheads, and can convert even short idle times into leakage energy savings. We combine LBB with three microarchitectural policies: subbank deactivation for instruction caches, and idle register and read port deactivation for multiported register files. Subbank deactivation saves over 22% of leakage energy and nearly 24% of the total instruction cache energy in a 70 nm technology. Dynamically deactivating idle registers reduces the total register file energy by 57%. Dynamically deactivating read ports saves 4-22% of the total register file energy. Next, we apply LB to domino logic, presenting a new FG-DLR circuit family, *Leakage-Biased Domino* (LB-Domino) for critical functional units. LB-Domino uses sleep transistors only on non-critical paths and uses the leakage current itself to bias internal critical paths into a minimal leakage state. This technique has little impact on switching energy or delay when applied to conventional domino circuitry. A 32-bit Han-Carlson domino adder circuit is used to compare LB-Domino with conventional and dual V_T domino circuits. We find that LB-Domino provides two decades reduction in steady-state leakage current compared with low-V_T or dual-V_T domino at equal delay and noise margin, and using LB-Domino to place circuits into a sleep state can yield net energy savings even for sleep times of under 10ns in a 70nm technology.

Even within highly active critical blocks, many individual circuit paths are idle on any given cycle, though whether a path is active or inactive changes dynamically during operation. We introduce *Dynamically Resizable CMOS* (DRCMOS) logic that exploits the phenomenon to reduce leakage. DRCMOS dynamically downsizes transistors on idle paths while maintaining speed along active critical paths. DRCMOS logic is successfully applied to two key blocks of a modern superscalar processor: a static 64-entry register free list slice and a static 64-entry pick-two arbiter. We find that DRCMOS can reduce power consumption by up to 50% at equal delay for the critical components of a modern superscalar processor implemented in a 70nm technology.

Chapter 6 describes power-optimal pipelining in logic datapaths. Pipelining can be an effective power-reduction tool when used to support voltage scaling in digital systems implementing highly parallel computations. Simulation results show that power-optimal logic depth is 6 to 8 FO4 and optimal power reduction varies from 55 to 80% compared with a 24 FO4 design depending on threshold voltage, activity factor, and the presence of clock-gating in a 70nm technology. We gain some important insights from the simulation results. First, higher activity factors decrease the power-optimal logic depth and increase the optimal power reduction because pipelining is most effective at reducing the additional switching power. Second, pipelining is more effective with lower threshold voltages, resulting in lower logic depths and lower power, except for low activity factors when leakage power is dominant. Third, clock-gating enables deeper pipelining and more power reduction because it reduces timing element overhead when the activity factor is low.

Chapter 7 explores the power implications of replacing global wires with an on-chip network. We optimize network links by combining energy-delay tradeoffs such as pipelining, repeater tuning, and voltage scaling, to significantly reduce the energy to send a bit across a chip. We develop an analytic model of large chip designs with an on-chip two-dimensional mesh network, and estimate the possible power savings for two different design points: a statically wire-routed architecture, and a dynamically packet-routed tiled architecture. For statically wire-routed networks, smaller tiles give more power savings because more inter-tile wires can be power-optimized. Dynamically packet-routed designs have many advantages over wire-routed circuits, such as the reduced number of link wires through multiplexing and signal encoding, but we find that large power reductions are unlikely due to router power overhead. A tile size of around 2mm is optimal in a 70nm technology, balancing the global wire power reduction with the router overhead.

Chapter 8 examines the use of *activity migration* which reduces peak junction temperature by moving computations among multiple replicated units. Using a thermal model that includes the temperature dependence of leakage power, we show that sustainable power dissipation can be increased by nearly a factor of two for a given junction temperature limit. Alternatively, peak die temperature can be reduced at the same clock frequency.

Chapter 9 concludes the thesis, summarizing its contributions and suggesting future work.

Chapter 2

Background

Applications and technology are the major drives in digital system development, but they also have provided ample challenges for circuit designers and architects.

Useful and popular applications, also known as "killer" applications, encourage users to buy systems on which they can run the apps, and encourage system designers to optimize systems so that apps can be run faster with less energy. For personal computing, spreadsheets, word-processing, graphic design tools, and video games were such killer applications. Now, multimedia, streaming, and wireless applications are emerging killer apps throughout all digital platforms, and they are driving circuit and architecture innovations. Multimedia or vector units in high-performance microprocessors are an example of such applications' influence.

Technology has been the most crucial driving force for the rapid improvement in VLSI digital systems. The success of continuous feature size scaling has brought and will bring greater speed and reliability, lower energy, and smaller size at lower cost. But more importantly, "technology determines the kinds of structures that can be considered and thus comes to shape our view of what a digital system is" [BN71]. Deep submicron technology not only provides great opportunities, but also presents great challenges to circuit designers and architects. A few prominent examples of challenges include ever-increasing leakage from various sources, such as subthreshold, gate, and BTBT leakages, ever-growing susceptibility to process variation, and ever-slowing global interconnects.

This chapter discusses important challenges in optimal VLSI digital system design for deep submicron technology and how circuit- and architectural innovations can be employed to overcome the pending challenges, while next two chapters (Chapter 3 and Chapter 4) describe examples of circuit- and architectural innovations respectively.

Section 2.1 describes how to compare and combine different energy-delay tradeoffs. Section 2.2 describes the ever-tightening peak and average power constraints. Section 2.3 discusses two ways of reducing switching power: trading delay for lower switching power and reducing switching power waste. Section 2.4 gives an overview of subthreshold leakage current. Section 2.5 describes the thermal behaviors of digital systems and then shows how the thermal limit can be overcome and the peak power constraint relaxed using dynamical thermal management.

2.1 Energy-Delay Tradeoffs

Optimizing digital systems through energy-delay tradeoffs requires two steps. First, when an innovation is under consideration for a digital system, its energy-efficiency for the target unit and whole system should be analyzed by constructing the energy-delay trading curve. It may be a painstaking task, but the ever-tightening power budget and ever-increasing performance requirement will make this process indispensable. Second, optimal combination among the applicable innovation candidates should be sought.

2.1.1 Comparing Energy-Delay Tradeoffs

Slopes, overheads, ranges, and operating regimes are important metrics for comparing energy-delay tradeoffs. Figure 2-1 shows tradeoffs with different slopes. If the slope of a tradeoff is relatively steep (assuming that x axis is delay and y axis is energy), the tradeoff tends to be used for energy reduction. For example, dynamic voltage scaling is usually used to trade time slack for lower energy, not the other way around, because its slope is as steep as quadratic ($E \propto D^{-2}$). Likewise, a tradeoff with a relatively flat slope is often used for improving performance. While most circuit innovations have negative slopes (i.e. delay reduction requires energy increase or energy reduction requires delay increase), some architectural innovations such as caching and subbanking have positive slopes, which mean that the innovations can save energy while improving performance.

Every tradeoff has a variety of overheads and the overheads make the tradeoff deviate from the ideal slope and give less energy saving or smaller delay reduction than expected from the ideal slope (Figure 2-2). For example, power dissipated by the additional stage latches is the major overhead of pipelining for throughput increase. Each energy-delay tradeoff has its effective energy and delay ranges (Figure 2-2). Utilizing a tradeoff outside the effective ranges is impossible or too costly. For example, supply voltage downscaling is bounded as the circuit becomes too slow due to reduced

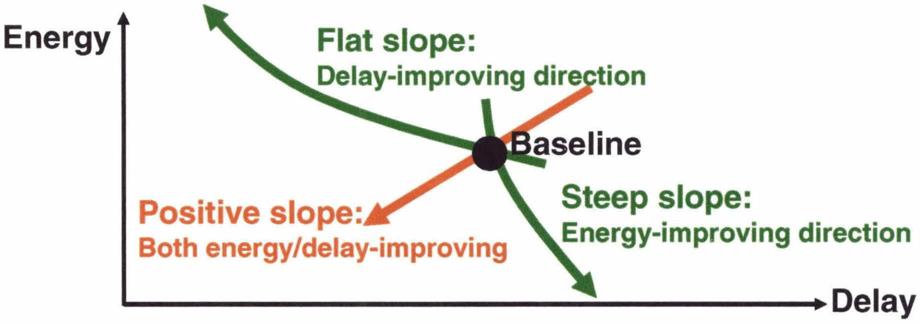

Energy
Flat slope:
Delay-improving direction
Baseline
Positive slope:
Both energy/delay-improving
Steep slope:
Energy-improving direction
Delay

Figure 2-1: Energy-delay tradeoffs with different slopes.

gate overdrive ($V_{dd} - V_T$) and too vulnerable to noise and process variation. Usually, circuit-level innovations have narrower ranges than architectural ones [Mar04]. Circuit-level techniques that scale voltages, such as supply voltage scaling, often have wider effective ranges compared to those which scale transistor dimension, such as traditional transistor sizing.

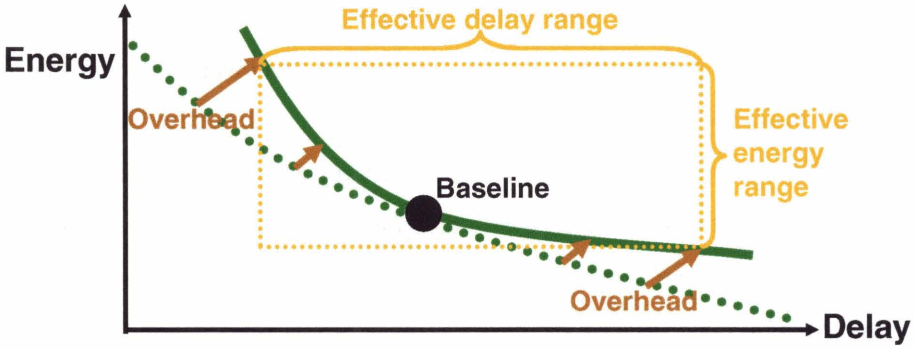


Figure 2-2: The overhead and effective ranges of an energy-delay tradeoff.

Tradeoffs have different sensitivities to different operating regimes, which consist of process and environment variables such as oxide thickness and temperature, circuit variables such as circuit styles and supply and threshold voltage levels, and architectural variables such as activity factors and microarchitectures. For example, pipelining combined with supply voltage scaling for energy reduction becomes more effective when the activity factor is high, the threshold voltage is low, and clock gating is present [HA04b].

2.1.2 Combining Multiple Energy-Delay Tradeoffs

A greater level of optimization (smaller energy at the same delay or smaller delay with the same energy) can be achieved by employing multiple energy-delay curves. For example, energy-delay curves with different delay ranges can be combined to make a new energy-delay curve with the extended effective delay range (Figure 2-3).

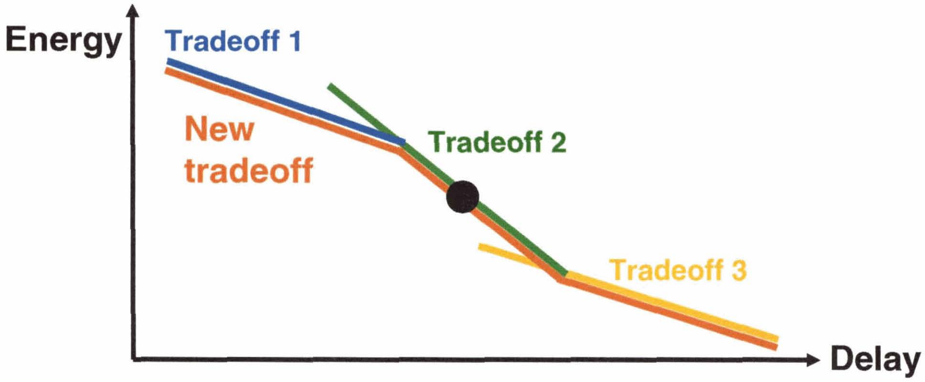


Figure 2-3: Combining energy-delay tradeoffs for larger effective energy and delay ranges.

An energy-delay tradeoff with a flatter slope can be combined with an energy-delay tradeoff with a steeper slope, to yield a performance improvement without spending more energy or achieve lower energy at the same performance. Figure 2-4 shows a combination of two tradeoffs with different slopes. From the baseline, performance is increased first with `tradeoff1` and then, using `tradeoff2`, energy can be reduced to the same energy as the baseline with delay reduction or can be further reduced to smaller energy with the same delay as the baseline. Figure 2-5 shows an optimal combination of two energy-delay tradeoffs for maximum energy saving. There exists an optimal combination point because the overhead increases as we decrease the delay or energy through an energy-delay tradeoff.

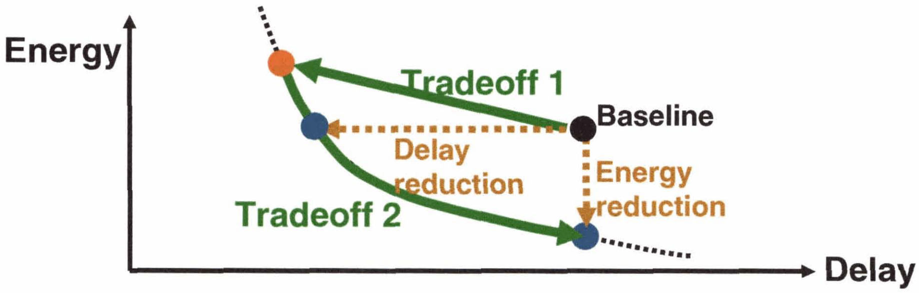


Figure 2-4: Combining energy-delay tradeoffs for delay reduction at the same energy or energy saving at the same delay.

When optimizing a digital system for minimum energy at the same delay or maximum performance at the same energy, we should carefully compare, select or reject, and combine applicable innovation candidates while observing the energy and delay impacts on the whole system as well

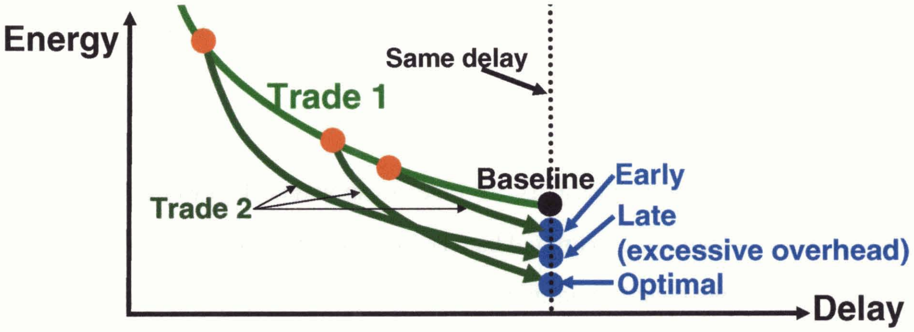


Figure 2-5: Optimal combination of energy-delay tradeoffs for maximum energy saving.

as the target unit. It also should be noted that utilizing multiple energy-delay tradeoffs might increase the design complexity. Also it is possible that some energy-delay tradeoffs are difficult to combine and even when combined, the combination can result in a worse energy-delay tradeoff due to resource conflicts.

2.2 Power – Another First-Class Metric

Power was overlooked for years. In the past, power consumption was not significant and a cheap package was enough to cool generated heat, since performance required by applications was relatively low and leakage current was ignorable. Digital system designers were not well aware of power (or energy) inefficiencies and trades between power and performance (or energy and delay). Therefore, innovations were often used for small performance gains while wasting power and other resources such as area and robustness to noise.

Intel engineers admit that transistor counts and power consumption have been increasing at rates greater than processor performance [Mar01] and over the last ten years. Even when all processors are scaled to the same process technology to isolate the circuit and architecture impacts, relative die size and power has gone up fifteen- and eighteen-fold respectively while relative performance increase is only five- or six-fold. Commonly, for modern high-performance processors, large amounts of area and power are used for work not directly related to computation, such as caching, scheduling, and predicting, to achieve as small as $2X$ improvement of instruction-per-cycle (IPC). For instance, only a quarter of MIPS R10000 CPU logic is devoted to integer and floating point datapaths, including register files and bypass circuitry [Asa98], and the rest for small performance improvement.

Now power is definitely a first-class metric as well as performance for all digital platforms. The power metric impacts the optimal digital system design through two constraints.

2.2.1 Power Constraints

There exist two important power constraints: average power and peak power (Figure 2-6). The first is the average power constraint due to limited battery life or energy cost. To maximize the operating time of a digital system, we have to minimize the energy per task or the average power. We could increase the battery capacity; however, the smaller form-factor and lighter weight requirements of modern mobile digital systems limit the size of battery, and battery technology is only slowly improving. The average power often matters due to the electricity cost ($/Watt), particularly for large computing resources such as server farms [BDH03].

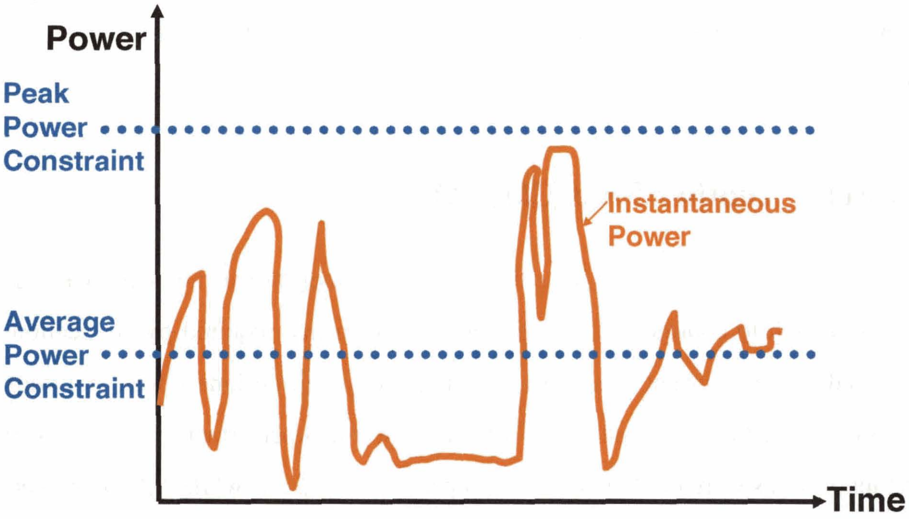


Figure 2-6: Two power constraints for digital systems.

The second is the peak power constraint due to thermal limits. Transistors dissipate power while charging and discharging parasitic capacitances, and even idle transistors consume significant leakage power. Power is dissipated as heat in digital systems. Because increase in temperature reduces system performance and reliability, heat needs to be removed through packages and fans. It is usual to set a peak power constraint to ensure thermal safety. Therefore, even when a system is plugged into main power, the power which can be traded for performance is still limited by the peak power constraint.

It is important to note that, depending on a system and its applications, there can be more demanding constraints than power. For example, for some systems, area is a critical constraint. There exists a tradeoff between power and area. A larger area budget enables more parallelism to be exploited for power reduction and allows for a smaller power consumption at a given performance requirement [Mar04]. In addition, since both area and power affect the system cost through the cost of fabrication and the costs of cooling and battery capacity respectively, the optimum is found at the point where the overall cost is minimized. Sensitivity to deterministic and non-deterministic noise and process variation is another important constraint.

2.3 Switching Power Reduction

The emergence of the power crisis in digital systems (especially in high-performance processors) has led to the advent of low-power techniques. Low-power techniques have been developed at various levels such as algorithms, architectures, circuits, and devices, and also on various platforms such as general-purpose processors, media processors, DSP ASICs, and FPGAs.

All computations and communications in digital systems are performed via switching and power consumed by switching has dominated total power. The following equation models switching power.

$$P_{switching} = AF \times C_{load} \times V_{dd}^2 \times f_{clock} \tag{2.1}$$

,where AF is the activity factor, C_{load} is the parasitic load capacitance, V_{dd} is the supply voltage, and f_{clock} is the clock frequency.

Techniques for switching power reduction can be divided into two according to how the power reduction is made. The first group reduces energy waste, energy dissipated by uncontrolled and unnecessary tasks (Section 1.1). Traditionally, energy waste was not significant and often ignored. As a result, the energy-efficiency of such energy-unaware digital systems could be significantly improved by simply finding and minimizing energy waste within the system without a complex re-design. Many architectural innovations reduce energy waste by reducing AF (Eq. 2-1) (Section 4.3). Clock gating is a representative example. Previously, it was common that the whole clock tree in a digital system ticked regardless of whether a circuit block was idle or not. Only after clock power became significantly large did power waste from spurious clock ticking get attention. Clock gating reduces the clock power waste by gating the local branches of the clock tree, if the circuit blocks which local branches feed are not doing any work. The cost for clock gating is an increase in clock control complexity. Likewise, power-aware SRAM arrays gate unnecessary bitline and wordline switching. The cost is again increased control complexity. Now that digital system designers are well aware of the importance of not wasting precious energy resources, many modern digital systems are well engineered to reduce energy lost to unnecessary capacitance switching.

The second group trades delay for lower energy, that is, through energy-delay tradeoffs (Section 1.2). A variety of circuit-level innovations attempt to tune the transistor dimensions such as the transistor width and length and voltages such as V_{dd} and threshold voltage (V_T) to balance energy and delay. The following equation models the transistor delay of a digital circuit.

$$\begin{aligned} t_{delay} &\propto \frac{C_{load} \times V_{dd}}{I_{drain}} && (2.2) \\ &\propto \frac{C_{load}}{k} \times \frac{V_{dd}}{(V_{dd} - V_T)^{\alpha}} && (2.3) \end{aligned}$$

,where I_{drain} is the average on-current, α is the carrier velocity saturation index (around 1.3 for recent short-channel transistors), and k is the gain factor [Rab96]. As can be seen from the equation, the delay model is closely related to the power model; thus, changing the transistor dimensions or voltage levels provides an intrinsic tradeoff between switching power and delay. For example, scaling down V_{dd} saves power, but increases the delay. Downsizing transistors reduces C_{load} and thus switching power, but increases delay because C_{load} does not scale as fast as k due to the other non-transistor capacitances such as interconnect caps. The design of the StrongArm 110 shows how to lower power, by sacrificing performance through direct energy-delay tradeoffs [Dob96]. The StrongARM's low-power circuit decisions include lowered supply voltage and a design change from latches to flipflops for clock load reduction. Chapter 3 presents more details on how circuit techniques for digital systems tune transistor dimensions and structures, and voltages to trade energy and delay.

2.4 Leakage Current

The recent exponential increase in leakage current results from two main factors: process scaling and die temperature increase. As the feature size shrinks, so do the supply and threshold voltages to keep the electric fields on the channels constant. Linear scale-down of threshold voltage has led to an exponential increase in leakage current. If this trend of constant threshold voltage scaling would continue, leakage current would increase by five times each generation [CBF00, DB99]. Leakage current is also exponentially proportional to die temperature. Continuous increase of total chip power has increased the average die temperature and contributed to leakage increase. Leakage power is now comparable to dynamic switching power in high-performance digital systems.

The trend toward ever more complex processors further exacerbates the situation. Large numbers of transistors are added for relatively small improvements in performance and these additional transistors may dissipate considerable subthreshold leakage power even when not actively switching. Effective leakage reduction techniques will be the key for optimizing digital systems as well as enabling further aggressive technology scaling.

2.4.1 Subthreshold Leakage

Leakage current is the drain current when the gate-to-source voltage is zero. Subthreshold leakage current due to weak inversion is now by far the most dominant component of transistor leakage among various transistor leakage components. When gate voltage is below threshold voltage, weak inversion occurs at the channel and causes the carriers to move by diffusion along the surface.

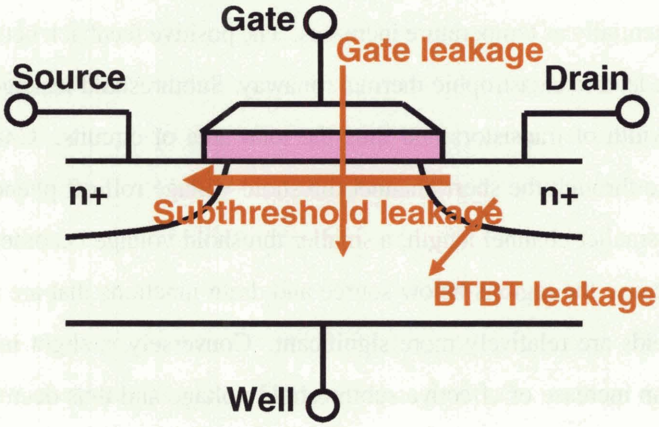


Figure 2-7: Leakage components of deep submicron transistors.

Subthreshold leakage is a complex function of many different variables such as threshold and supply voltages, temperature, and the physical and effective dimensions of channels. The following equation shows the subthreshold leakage model.

$$I_{subthreshold} \propto \frac{W}{t_{OX}L_{eff}}e^{\frac{-V_T-\gamma' V_{sb}+\eta V_{dd}}{nv_T}} \tag{2.4}$$

v_T is the thermal voltage, γ' is the linearized body effect coefficient, η is the DIBL coefficient, W is the transistor width, L_{eff} is the effective channel length, and t_{OX} is the oxide thickness. The subthreshold conduction current is exponentially proportional to threshold voltage (Eq. 2-

4) and threshold voltage is inversely proportional to transistor delay (Eq. 2-3), thus controlling threshold voltage provides a tradeoff between leakage power and delay. Supply voltage affects subthreshold leakage current through the DIBL phenomenon. When the channel length is relatively short compared to drain voltage level, drain voltage can lower the source potential barrier and as a result, effective threshold voltage level. Because of the DIBL effect, supply voltage scaling results in a superlinear reduction of leakage power rather than linear. Body voltage affects the effective threshold voltage through the body effect. Reverse body biasing (biasing the body voltage of NMOS transistors below the source voltage or biasing the body voltage of PMOS transistors above the source voltage) increases the charge stored in the depletion region and the surface potential required for strong inversion, and as a result, it raises the effective threshold voltage and reduces subthreshold leakage.

The subthreshold slope (nv_T in Eq. 2-4) flattens by a factor proportional to the absolute temperature since the thermal voltage, v_T, is proportional to the absolute temperature. As a result, leakage current grows exponentially as temperature increases. The positive feedback between leakage power and temperature can lead to catastrophic thermal runaway. Subthreshold leakage is linearly proportional to the total width of transistors and thus the total area of circuits. Channel length affects subthreshold leakage through the short-channel threshold voltage roll-off phenomenon (not shown in Eq. 2-4). With a smaller channel length, a smaller threshold voltage becomes sufficient to cause strong inversion because the regions below source and drain junctions that are already depleted by source and drain fields are relatively more significant. Conversely, a slight increase of the channel length leads to an increase of effective subthreshold voltage and thus decrease of subthreshold leakage current.

Process variation, one of the most important source of non-scalability in deep submicron technology, complicates the leakage problem further. The worsened process variation leads to significant delay spread and affects the yield severely, thus favoring faster and leakier designs. The impact on leakage spread is also significant. It has been shown that there can be 20X variation in leakage current in a 150nm technology [Aga05]. Soon, leakage reduction techniques considering the possible yield loss due to the delay and leakage spreads will be crucial.

Aggressive technology scaling has caused various short channel effects (SCE), such as drain-induced barrier lowering (DIBL), threshold voltage roll-off, and reduced on-current to off-current ratio. To mitigate SCEs, oxide thickness scaling and higher and non-uniform doping, such as halo and retrograde well, have been utilized by device engineers. The low oxide thickness gives rise to high electric field across the oxide (E_{ox}), resulting in significant gate oxide tunneling current (Figure 2-7). On the other hand, a high doping level leads to high electric field across source and drain junctions, resulting in significant junction band-to-band tunneling (BTBT) current (Figure 2-7). In the near future, gate and junction BTBT leakages are expected to be comparable to subthreshold leakage [Aga05]. Therefore, leakage reduction techniques considering all these components will be indispensable. Though the importance of other leakage components is increasing rapidly, they are beyond the scope of this research and we focus on subthreshold leakage, which requires a more urgent fix. If not specified, leakage implies subthreshold leakage in the remainder of this thesis.

2.5 Thermal Constraint

Ever-increasing power consumption and ever-decreasing form-factor of digital systems have increased heat density and die temperature rapidly. Raised junction temperature causes decreased delay and increased leakage. Increasing temperature raises the thermal agitation of the semiconductor atoms, which in turn increases lattice scattering and causes an approximately quadratic decrease of mobility [Pie96]. The reduced mobility leads to reduced circuit speed and possible timing errors.

The linear increase of temperature makes leakage power increase exponentially (Section 2.4). The leakage power increase might lead to the further temperature increase in return. This positive feedback can cause catastrophic thermal runaway, if there is no dynamic thermal management scheme to sense high temperature and reduce temperature by decreasing power density dynamically. Additionally, soft errors and transistor aging also increase exponentially with temperature.

2.5.1 Thermal Behavior of Digital Systems

Power, P, in electrical systems is mainly converted to heat energy (or heat flow), Q.

$$Q \propto P \tag{2.5}$$

Heat energy dissipation causes an increase in temperature. The following equation represents the law of heat conduction, where Q is proportional to the gradient of steady-state temperature difference, ΔT, and area, A, while inversely proportional to thickness, l.

$$Q \propto \Delta T \frac{A}{l} \tag{2.6}$$

Rearranging the above equations,

$$\Delta T \propto \frac{P}{A} \tag{2.7}$$

We see that the steady-state temperature difference is proportional to power, P, and inversely proportional to area, that is, power density (power divided by area). As power densities of high-performance digital systems rise above 100 W/cm^2, providing adequate heat removal with a low cost package is becoming particularly challenging. Cooling or packaging costs are proportional to net heat density. Heat density or flux is proportional to the amount of heat and inversely proportional to the form factor. Because cost-effective package design is beyond the scope of this thesis, we simply assume that packages with fixed dimensions are given, and heat and heat density are not differentiated in this thesis.

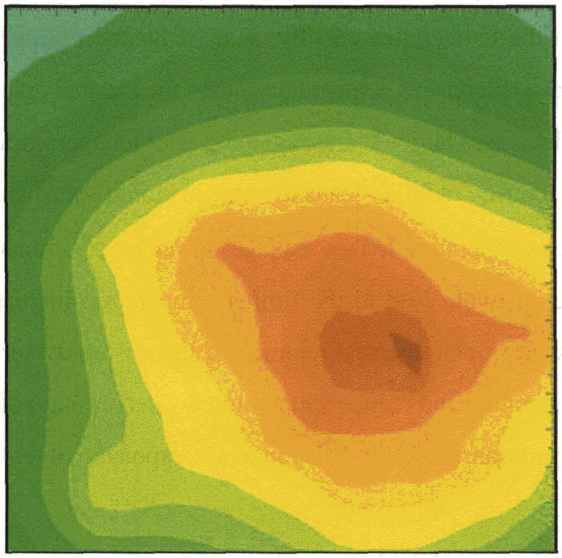


Figure 2-8: A simulated thermal plot of the Pentium 4 processor [Gun01]. Darker color indicates higher temperature.

Three basic observations can be made on the thermal behaviors of digital systems. First, temperatures on dies are far from uniform. The uneven distribution of power dissipation across a die complicates heat removal. The uneven power density leads to hot spots where regions of a die have significantly elevated junction temperatures compared to inactive regions. The reason why hot spots develop is that silicon is a relatively poor heat conductor and cannot efficiently spread heat across a die. Figure 2-8 shows a thermal plot of the Pentium 4 processor [Gun01]. The plot clearly indicates a small region with much higher temperature than the rest. Hot spots provide a serious impediment to achievable performance since they require that total power dissipation is reduced until all hot spots have junction temperatures below the reliability limit. Hot spots typically occur at the granularity of architecture-level blocks [Ska03].

Second, temperature depends not only on power density, but also on time, due to thermal capacitances. The thermal capacitances of digital systems and surrounding packages are significantly large that all thermal processes such as heating, heat spreading, and cooling take noticeable time. Lastly, there is a significant gap between the typical- and worst-case power since most applications have limited amount of parallelism, due to dependences, compared to the maximum parallelism the digital system can afford, and have low signal transition probabilities compared to the worst-case pattern.

2.5.2 Temperature-Aware Design

Thermal limit has led to strict peak power constraints (Section 2.2.1) and digital system designers optimize systems strictly under the constraints to ensure thermal safety while maintaining reasonably cheap packages. This worst-case design style is safe, but tends to be extremely conservative, resulting in limited performance.

Temperature-aware design has been proposed to attain further optimization. Temperature-aware design deals with thermal issues more directly. Instead of designing for the worst peak power, it designs for the typical high power and the rare thermal emergencies are taken care of dynamically. Through dynamic thermal management, the peak power constraint is effectively relaxed (Figure 2-9).

Temperature-aware design has two requirements: thermal sensors and dynamic thermal management units. Modern high-performance processors include on-die thermal sensors that can detect over-temperature conditions caused by inadequate heat removal. Although originally designed to detect thermal emergencies caused by poorly attached heat sinks, broken fans, or blocked air vents, thermal sensors can also be used in dynamic thermal management schemes that exploit the growing gap between typical- and worst-case power dissipations [San97]. Thermal sensors are usually distributed throughout the whole chip, making sure that the possible hot spot candidate areas are covered. Simulation with benchmark applications would help to find possible locations for thermal sensors. When the thermal sensors detect die temperature increase over a trigger point, they signal the dynamic thermal management unit, and the unit starts to reduce the power density of the problem area. When the temperature falls to a safe range, the unit stops reducing the power density.

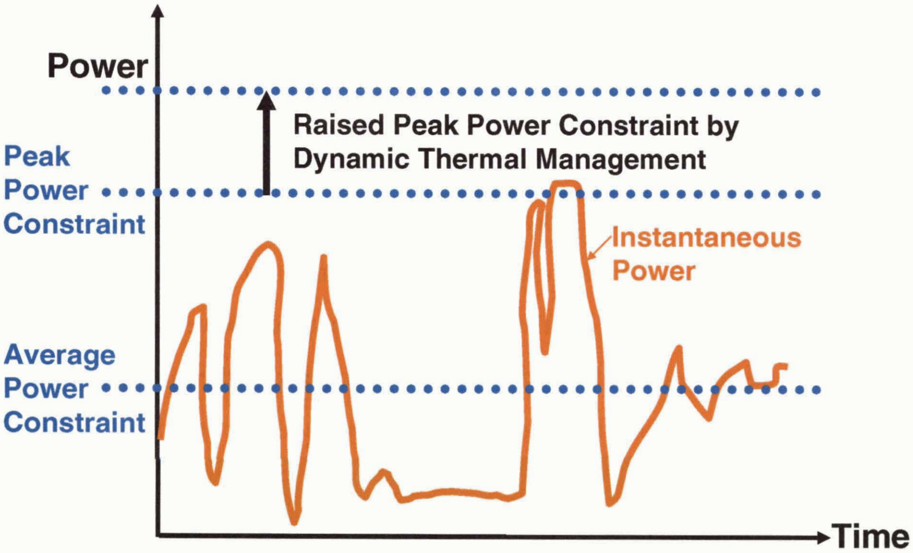


Figure 2-9: Relaxing peak power constraint through dynamic thermal management.

The performance penalty during cooling and transition (between the normal state and thermal emergency state) energy and delay costs should be minimized. Cooling only the hot spots usually gives less performance penalty than cooling the whole die, while allowing the same amount of increase in the effective power budget. Dealing with intrinsically-local heating with global cooling is not cost-effective. However, implementing localized cooling might lead to a significant increase in design complexity.

Chapter 3

Categorizing Circuit Innovations

Numerous circuit- and architectural innovations have led to the rapid improvements of VLSI digital systems. This chapter categorizes circuit innovations from the perspective of energy waste reduction and power-performance tradeoff.

Circuit innovations have two goals (Figure 3-1). First, they can trade time slack on non-critical paths for lower energy. Second, they can reduce the delay of critical paths and decrease the cycle time, and thus total execution time.

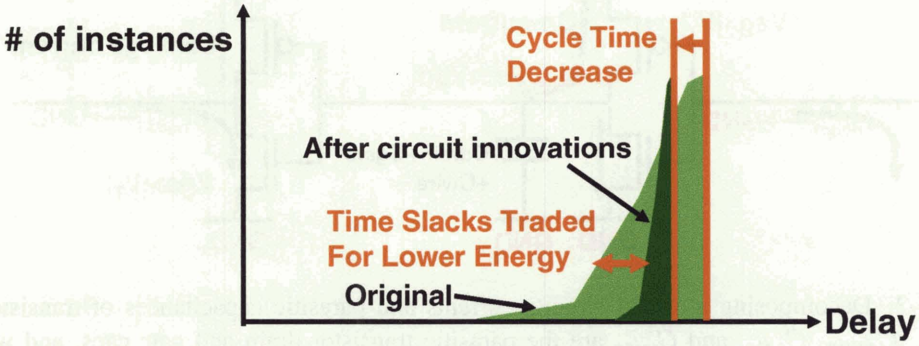


Figure 3-1: Goals of circuit innovations.

Circuit innovations often do not stand alone. Commonly, they are combined with architectural innovations. For example, dynamic voltage scaling is a circuit technique which dynamically scales supply voltage and saves energy while meeting the performance requirement. However, an accompanying architectural innovation is necessary for finding time slack. Chapter 4 categorizes the architectural innovations for digital systems.

3.1 Digital Circuits

We observe that digital circuits are constructed with two major components: first, currents and parasitic capacitances of transistors, and second, voltages (Figure 3-2). A digital system is basically a web of switches, and computes and communicates by switching modes between zero and one. Modern electrical digital systems built in CMOS technologies use transistors as the switches and switching is done through charging and discharging the parasitic capacitances using the on-currents of the transistors. Thus, tuning transistors' dimensions has substantial effect on the energy and delay of the system since it directly impacts both the parasitic caps and on-currents (and also leakage currents). Sizing transistors is the most basic form of circuit-level optimization.

A few discrete voltage levels greatly affect digital circuits. The digital values, zero and one, are established by ground (GND) and supply voltage (V_{dd}) respectively. Transistors are turned on and start to flow on-currents when the gate voltages exceed the threshold voltage (V_T). In addition, the body voltage (not shown in Figure 3-2) affects V_T and thus the on- and off-currents. Thus, it is intuitive that manipulating V_{dd} and V_T can provide direct control of the systems' performance and power.

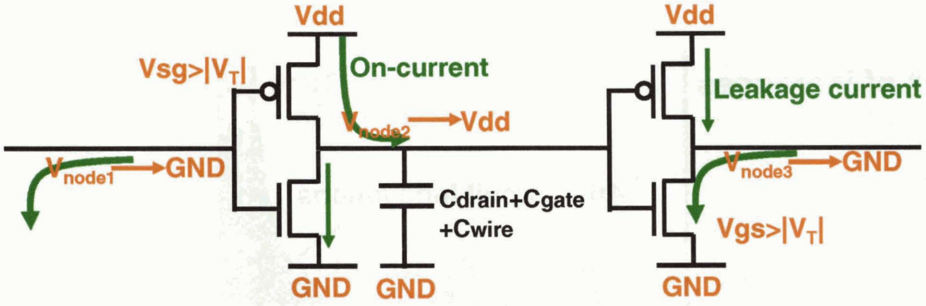


Figure 3-2: Decomposing a digital circuit: currents and parasitic capacitances of transistors and voltages. C_{drain}, C_{gate}, and C_{wire} are the parasitic transistor drain and gate caps, and wire cap respectively.

We divide circuit innovations into two, according to whether a circuit innovation focuses more on sizing transistors (Section 3.2) or scaling voltages (Section 3.3).

If not specified otherwise, all the energy-delay plots in this chapter were made through Hspice simulation of a 16 FO4 inverter chain (Figure 3-3). BPTM 70nm technology [Dev01] was used and important parameters of the process are summarized in Table 3.1.

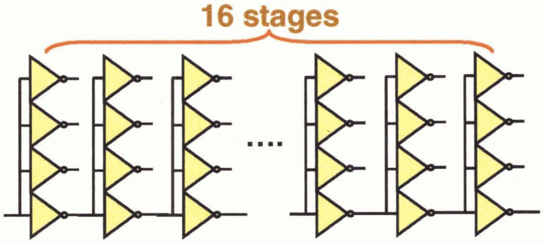


Figure 3-3: An FO4 inverter chain.

Table 3.1: Process parameters.

Nominal supply voltage	1.0V
Nominal NMOS threshold voltage	0.21V
Nominal PMOS threshold voltage	-0.24V
Temperature	100 ^oC

3.2 Tuning Transistors' Dimensions

Sizing transistors impacts both on-current and leakage currents ($I_{on}, I_{subthreshold} \propto \frac{W}{L_{eff}}$, where W and L_{eff} are the width and effective length of a transistor) and also the parasitic capacitances ($C_{gate} \propto WL_{eff}, C_{diffusion} \propto W$), giving a tradeoff among delay, switching energy, and leakage power. The basic rule of transistor sizing is to upsize transistors on critical paths while downsizing non-critical ones such as feedback transistors in sequential logic or keepers in dynamic circuits. Energy saved by downsizing non-critical transistors and reducing the amounts of parasitic capacitance and leakage current can be exploited by upsizing transistors for delay reduction.

Figure 3-4 shows the effects of varying the PN ratio ($\frac{W_{PMOS}}{W_{NMOS}}$) when the total sum of PMOS and NMOS widths is fixed. Since the total parasitic capacitance remains constant, the switching energy does not change much. Reducing NMOS transistor width (increasing the PN ratio) decreases leakage power since the leakage current of NMOS transistor is larger than that of PMOS transistor when the sizes are identical.

Most circuit blocks consist of cascaded stages, and load balancing between stages is necessary to achieve the lowest power at a given delay requirement [SSH99]. While the minimum delay design requires the delay of every stage to be the same, the power-optimal sizing downsizes the largest gates toward the output and increases the effective fan-out from the minimum delay point [Sto95] (The fan-out is defined as the amount of logic gates connected to the output of a logic gate).

Figure 3-5 shows the effects of varying the fan-outs and number of stages when the load capacitance is fixed. Increasing the fan-out further after the minimum latency point results in the excessive

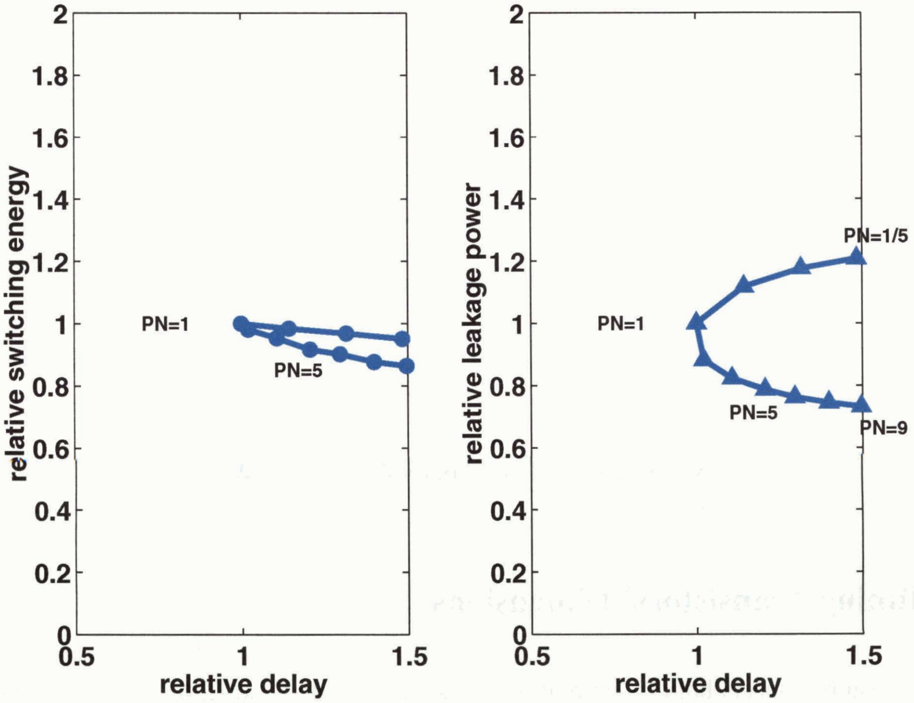


Figure 3-4: Varying the PN ratio. Switching energy, leakage power, and delay are normalized to the minimum delay point.

energy increase due to overheads such as the increases of parasitic capacitance and leakage current. Around 20% of switching energy and 70% of leakage power can be saved by increasing the delay by 50% from the minimum delay point.

Figure 3-6 shows that lengthening the transistor can be used as an effective leakage reduction tool (Section 5.1.1). A small increase in transistor length away from the minimum gives a significant reduction in leakage current with a small impact on delay, because the effective threshold voltage depends on the length when it is short. The StrongARM processor design reduces leakage current by lengthening devices in cache arrays and pad drivers by 10–30% [Dob96].

3.3 Tuning Voltages

This section discusses the effects of varying the supply, threshold, and body voltages on the energy and delay of the digital circuit.

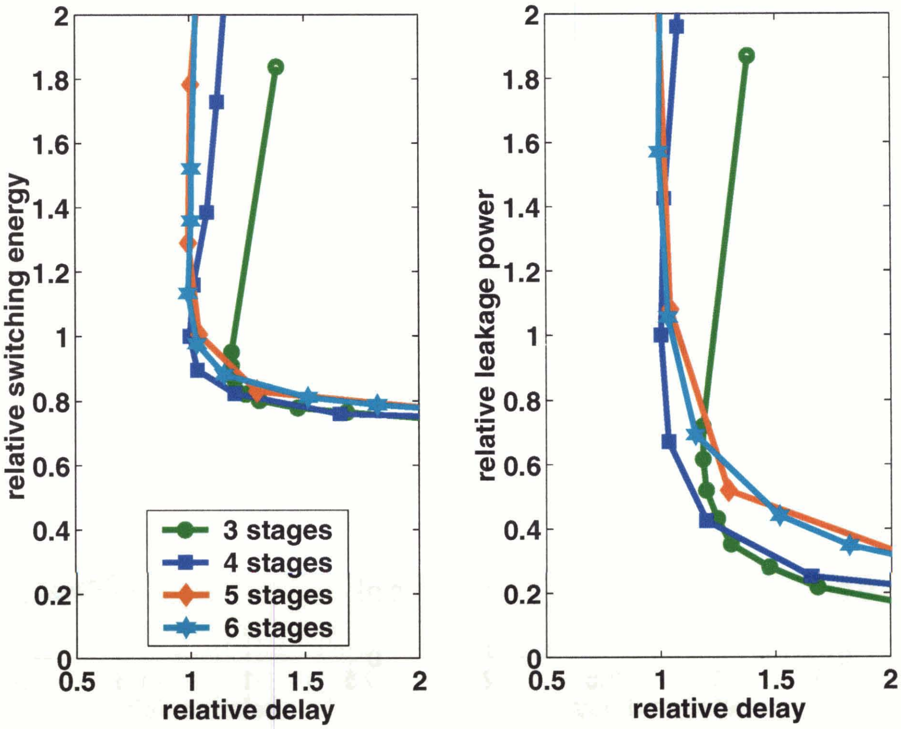


Figure 3-5: Varying the fan-out and hence number of stages when the load capacitance is fixed. Switching energy, leakage power, and delay are normalized to the minimum delay point.

3.3.1 Scaling Supply Voltages

In a deep submicron process, the nominal V_{dd} is already around the critical voltage [Cha92], and the amount of current is less dependent upon the supply voltage due to the velocity saturation effect. Hence, increasing performance through voltage scaling requires excessively large energy. On the other hand, supply voltage scaling can be used as a highly efficient energy-saving tool. The following equations show that roughly 2% reduction of switching energy can be achieved with only 1% delay increase (assuming $V_T << V_{dd}$ and the mobility degradation due to the short channel is not significant).

$$E_{switching} = C_{load}V_{dd}^2 \tag{3.1}$$

$$Delay \propto \frac{1}{V_{dd}} \tag{3.2}$$

$$E_{switching} \propto \frac{1}{Delay^2} \tag{3.3}$$

$$\Delta E_{switching} = -2\Delta Delay \tag{3.4}$$

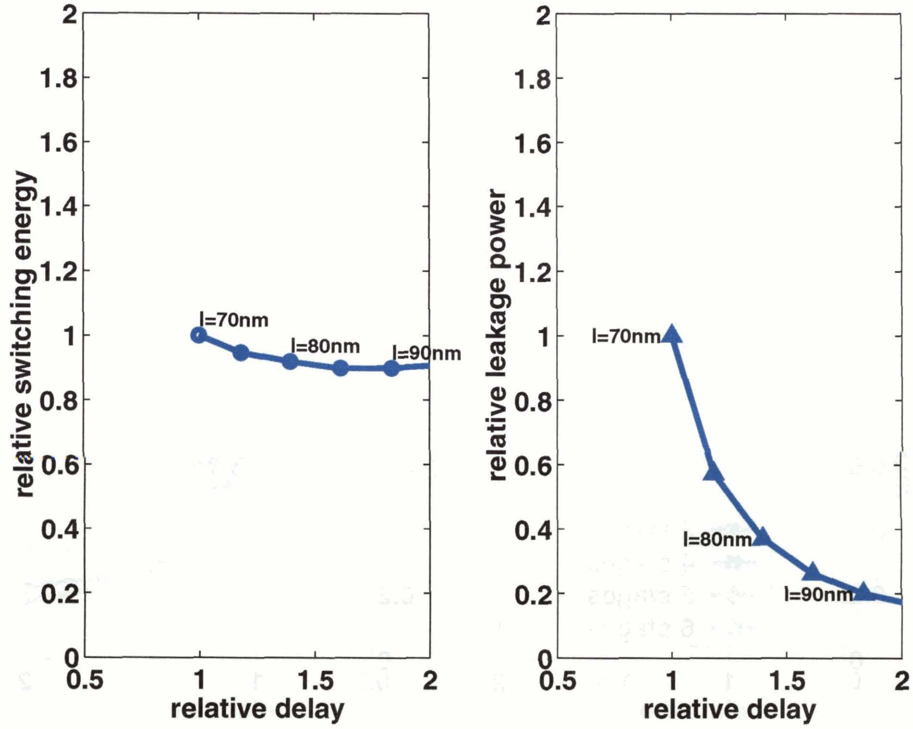


Figure 3-6: Lengthening transistors for leakage reduction. Varying the fan-out and number of stages when driving a fixed load. Switching energy, leakage power, and delay are normalized to the minimum length point.

In deep submicron technology, even the leakage current is dependent upon V_{dd} through the drain-induced-barrier-lowering (DIBL) phenomenon and as a result, the leakage power is super-linearly dependent upon V_{dd}. Figure 3-7 shows the effects of V_{dd} scaling. It is clearly seen that the V_{dd} scaling is effective at both switching energy and leakage power saving.

To keep the gate overdrive, $V_{dd} - V_T$, reasonably large and avoid excessive slowdown ($Delay \propto \frac{V_{dd}}{(V_{dd}-V_T)^{\alpha}}$, where α is the mobility degradation factor and around 1.3), V_{dd} downscaling has been forcing V_T to scale down too. The V_T downscaling has led to the rapid increase of leakage power (Section 2.4), which is the major obstacle to V_{dd} scaling.

Some emerging applications such as medical applications and distributed sensor networks have low energy as the primary concern rather than performance. Subthreshold operation has been used for these ultra low-energy applications because minimum energy operation for low performance situations occurs in the subthreshold region. An ultra-low voltage FFT processor whose V_{dd} can be lowered to 180mV was implemented in a 180nm process and was measured to be 8x more energy-efficient than the standard superthreshold implementation [Wan02a].

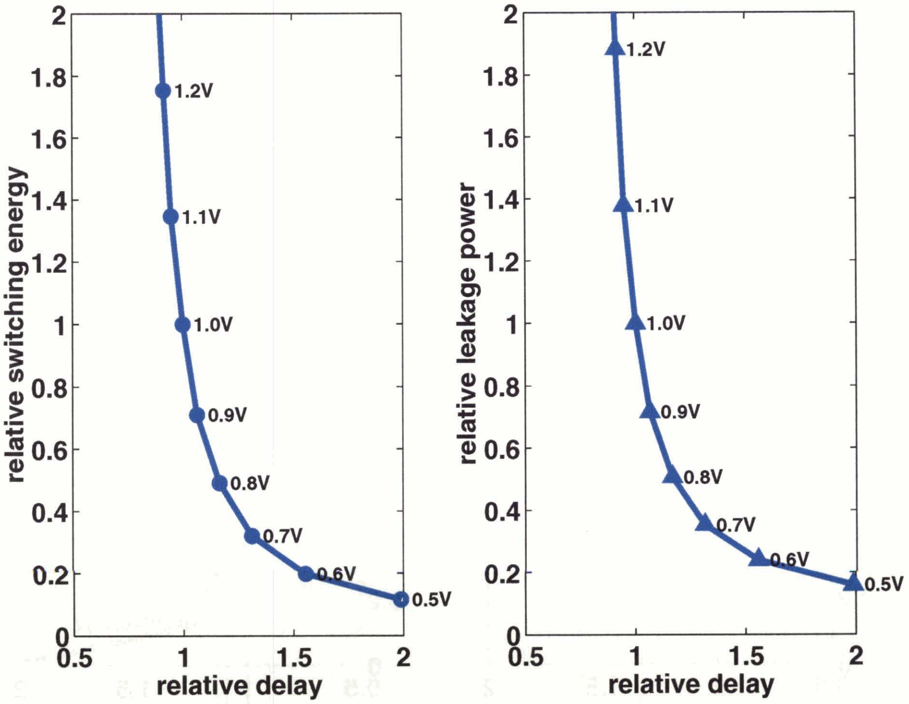


Figure 3-7: Scaling supply voltage. Switching energy, leakage power, and delay are normalized to the nominal voltage, 1V.

3.3.2 Scaling Threshold Voltages

Previously, V_T was less important than V_{dd} in terms of both energy and delay. It was too small to affect the gate overdrive, $V_{dd} - V_T$ and too large to cause a significant subthreshold leakage power consumption. However, continuous supply voltage down-scaling has led threshold voltage to become large enough to impact circuit delay through reduced gate overdrive and small enough to cause leakage power to be comparable to switching power. Figure 3-8 shows the effects of V_T scaling on delay, switching energy, and leakage power. Switching energy decreases as V_T increases because the short-circuit current (the current flowing when PMOS and NMOS transistors are simultaneously turned on and the path from V_{dd} to GND is briefly made) is reduced. It is clearly seen that V_T scaling provides an effective leakage power reduction tool. At our 70nm process, around 25% delay increase can save 90% of leakage power.

Deep submicron technology often requires the simultaneous scaling of supply and threshold voltages to meet ever-high performance goals while keeping ever-tight power constraints. Figure 3-9 shows the optimal leakage power ratio when both V_T and V_{dd} are scaled. We measured the total power and the ratio between leakage and total power while varying V_T and V_{dd} for a given delay

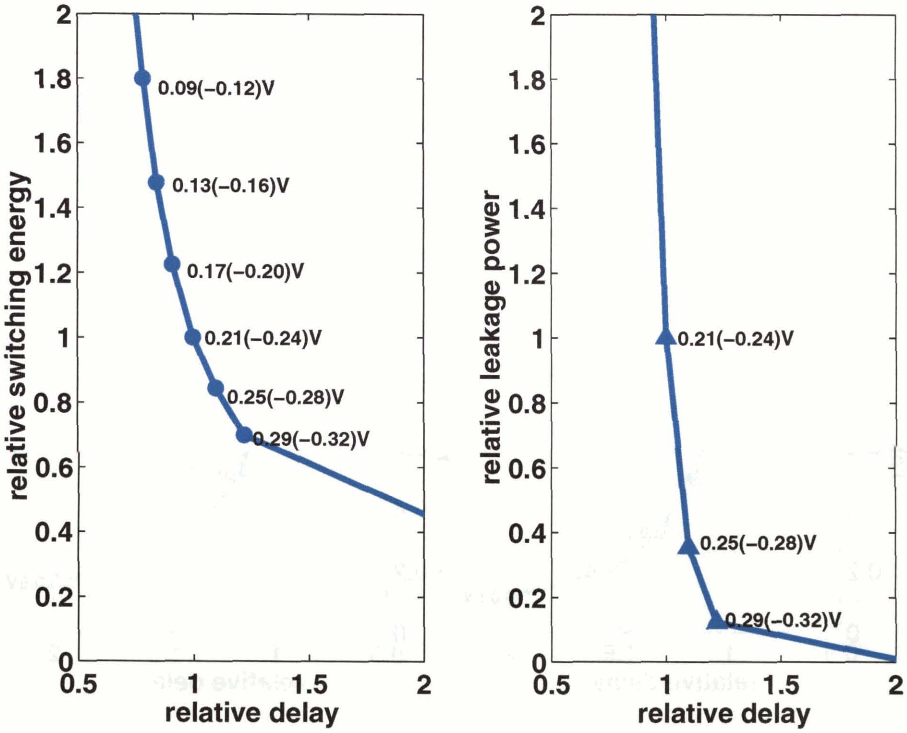


Figure 3-8: Scaling threshold voltage scaling. Switching energy, leakage power, and delay are normalized to the nominal threshold voltages, 0.21/-0.24V.

requirement and activity factor (AF). It can be seen that the optimal ratio is rather wide and the total power curves are quite flat around the 50% ratio regardless of the delay requirement. As AF increases, the optimal ratio tends to increase slightly. An analysis by closed-form equations [NS00] and a circuit simulation [Mar04] also show that the total power has a very shallow minimum when represented as a function of leakage and total power ratio, and the optimal ratio is around 30-40%, and that the value of ratio is not changed even over a wide range of design parameters such as activity factor, logic depth, and clock frequency. Figure 3-10 shows the optimal V_{dd} and V_T that gives the lowest total power. The simulation results show that the optimal threshold voltage is strongly dependent on the activity factor. When AF is low, the optimal V_T become high to reduce leakage power. On the other hand, when AF is high, the optimal V_T becomes low to reduce V_{dd} and thus switching power. The optimal supply voltage is a function of both the delay requirement and AF.

With additional manufacturing cost, multiple V_T transistors can be used for a further optimization. One straightforward way of exploiting multiple V_T transistors when V_{dd} is fixed is to make non-critical transistors high V_T for leakage power saving while employing low V_T transistors on

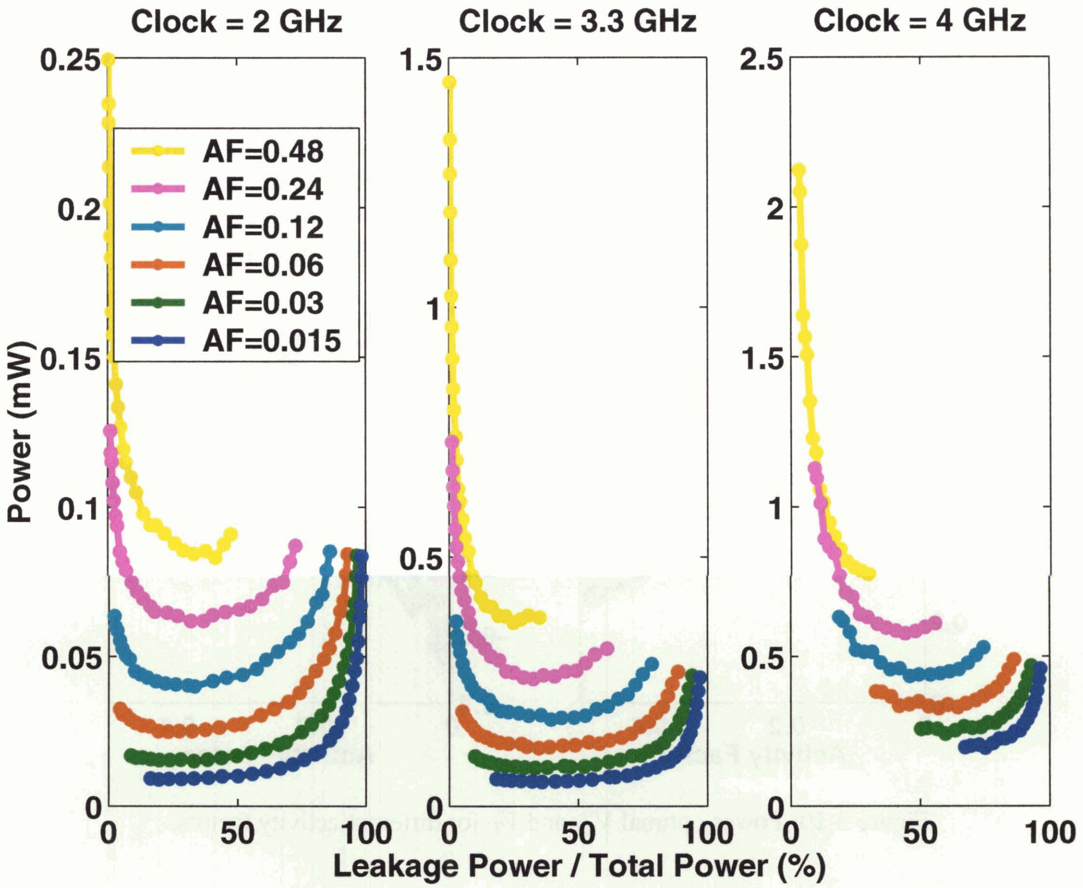


Figure 3-9: Power-optimal ratio of leakage power and total power for different delay requirements and activity factors.

critical paths. V_{dd} scaling can also be improved by utilizing multiple V_T transistors. Figure 3-11 shows that the best energy-delay curve differs according to the AF. LVT requires the lowest total energy at the same delay when AF is high, because the V_{dd} of LVT is the smallest and so is the switching energy. Conversely, HVT is the best when AF is low, since it has the lowest leakage while its V_{dd} is larger than others.

The optimal voltage levels (V_{dd} and V_T) in individual circuit blocks rarely coincide because of differences in structure when circuit blocks are combined. While allowing multiple V_{dd} and V_T leads to a better optimization point, it requires additional manufacturing cost and increases area and design complexity. For example, employing multiple V_{dd}s, where a low V_{dd} is used for non-critical paths and a high V_{dd} on critical paths, requires multiple power planes and level converters between different V_{dd} regions.

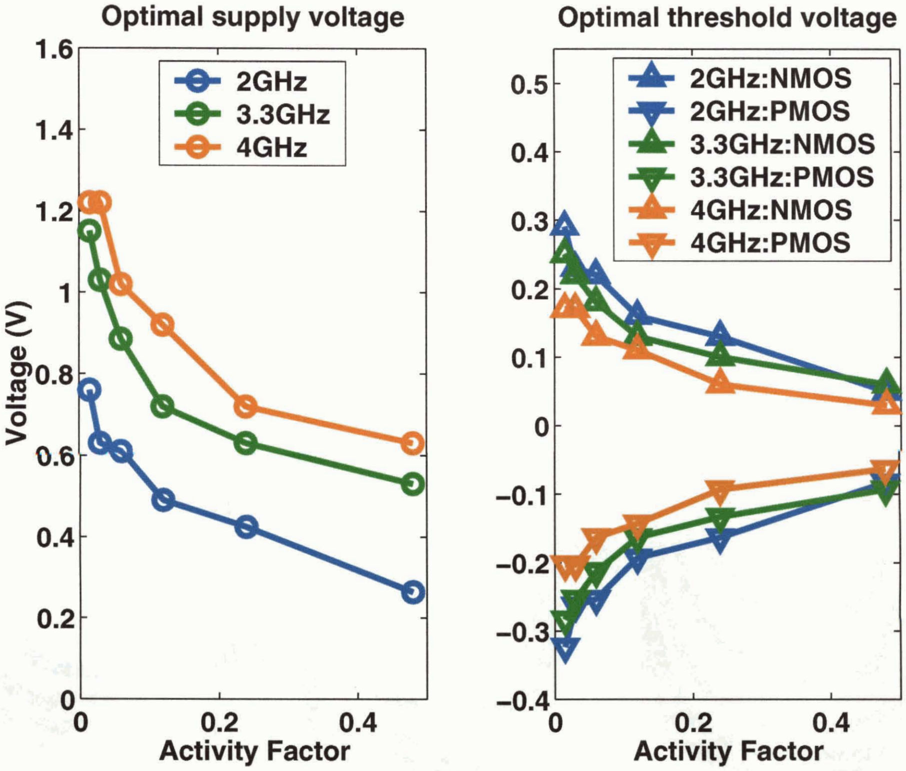


Figure 3-10: Power-optimal V_{dd} and V_T for different activity factors.

3.3.3 Scaling Body Voltages

Traditionally, bodies were tied to supply voltages and grounds in bulk CMOS processes. However, energy-delay tradeoff using body biasing has received attention as leakage power has become significant. Body biasing takes advantage of the body effect to adjust the effective threshold voltage and reduce the leakage power waste. Eq. 3-5 models the body effect. In the equation, Φ_F is the Fermi potential, γ is the body effect coefficient, V_{T0} is the zero bias threshold voltage, and V_{sb} is the source-body voltage.

$$V_T \quad = \quad V_{T0} + \gamma(\sqrt{|-2\Phi_F + V_{sb}|} - \sqrt{|-2\Phi_F|}) \qquad (3.5)$$

Figure 3-12 shows the impact of body biasing on the delay, switching energy, and leakage power. Reversely body biasing (0∼-0.4V) results in reduced switching energy since the short-circuit current decreases when V_T increases. Around 20% delay increase can bring as much as 80% leakage power saving.

Body biasing can be done in two ways: reverse body biasing (RBB) or forward body bias-

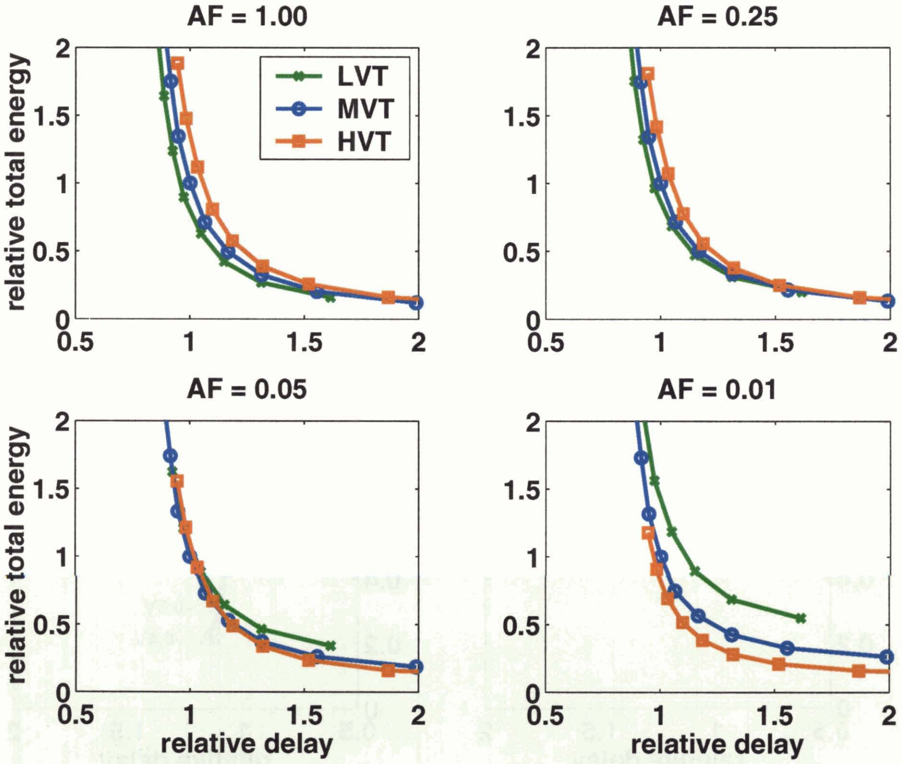


Figure 3-11: Scaling supply voltage for different threshold voltages and activity factors. AF is the activity factor and LVT, MVT, and HVT are low-threshold, medium-threshold, and high-threshold transistors respectively.Switching energy, leakage power, and delay are normalized to the MVT and nominal V_{dd} point.

ing (FBB) (Figure 3-13). RBB dynamically increases the effective threshold voltage of originally low-threshold transistors when they are idle. On the other hand, FBB dynamically decreases the effective threshold voltage of originally high-threshold transistors when they are in operation. In deep submicron technology, FBB is preferable to RBB because it uses high-threshold transistors, which have less short-channel effect and threshold voltage variation [Nar03].

3.4 Summary

Circuit innovations have improved the energy and delay of digital systems through sizing transistors and scaling voltages. V_{dd} downscaling effectively reduces both switching energy and leakage power. Lengthening the transistor, increasing V_T, or reversely biasing body voltage provide effective leakage power reduction for delay increase. When both V_{dd} and V_T are scaled, the optimal leakage power ratio is around 50% regardless of the activity factor or delay requirement and the

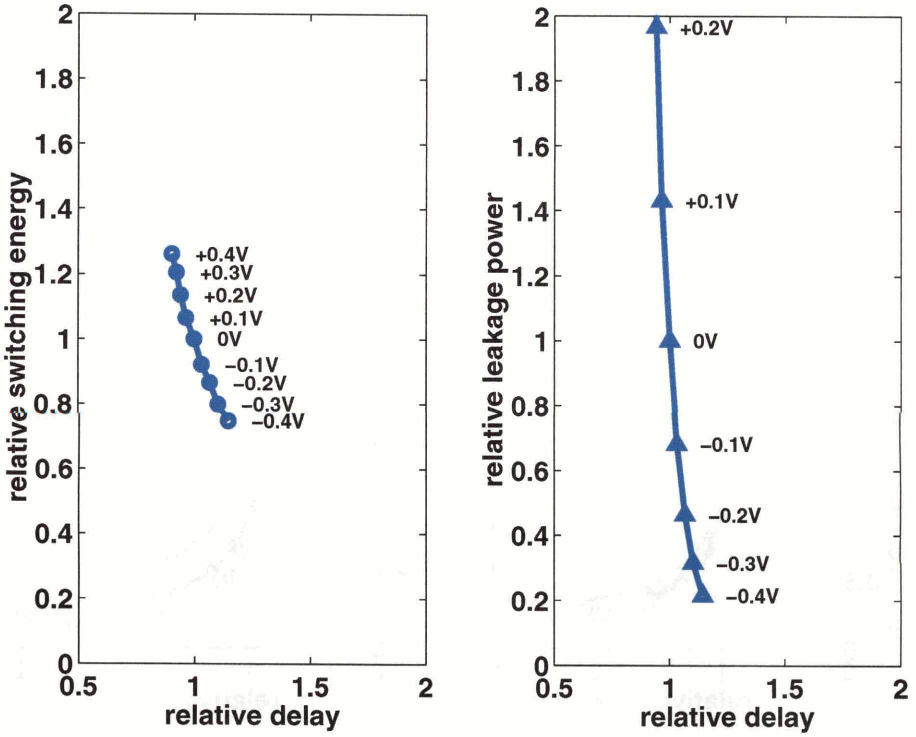


Figure 3-12: Biasing body voltages. Switching energy, leakage power, and delay are normalized to the zero body bias point.

optimal V_T depends on AF. In a multiple V_T process, Low-V_T gives the best energy-delay curve for V_{dd} scaling at high AF while high-V_T gives the best energy-delay curve for V_{dd} scaling at low AF.

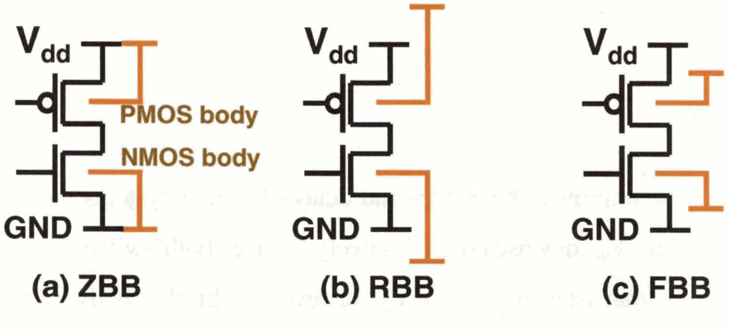


Figure 3-13: Zero body biasing (ZBB), reverse body biasing (RBB), and forward body biasing (FBB) for digital circuits.

Chapter 4

Categorizing Architectural Innovations

The goal of architectural innovations for processors is to provide a structure that can run instruction streams with maximum concurrency, minimum stalls, and minimum waste.

We find that most architectural innovations are based on three critical techniques: employing parallelism, exploiting predictability, and reducing energy waste. Architectural innovations seek and make use of the intrinsic parallelism and predictability in programs and computation models to maximize the utilization of system resources, while minimizing the inefficient use of energy resources. This chapter categorizes architectural innovations for processors from the perspective of energy waste reduction and energy-delay tradeoff principles.

It should be noted that the categorization in this chapter is not exclusive. Many architectural innovations can fall into two or three categories at the same time. We focus on architectural innovations for general-purpose processors since the processors are the most architecturally-complex digital systems which require the most sophisticated architectural innovations.

4.1 Employing Parallelism

Employing various levels of parallelism has been a key theme in architectural innovation. Parallel execution is a straightforward way of increasing throughput. Especially, in this deep submicron technology era, transistors are numerous and cheap, and providing additional parallel circuitry for parallel execution is one of the best uses of abundant transistors [Cha92]. Figure 4-1 shows the benefit of employing parallelism in terms of energy and execution time. Ideally, employing parallelism shortens execution time while dissipating the same amount of energy for the same task.

In this category, we include architectural innovations overcoming the factors that limit ex-

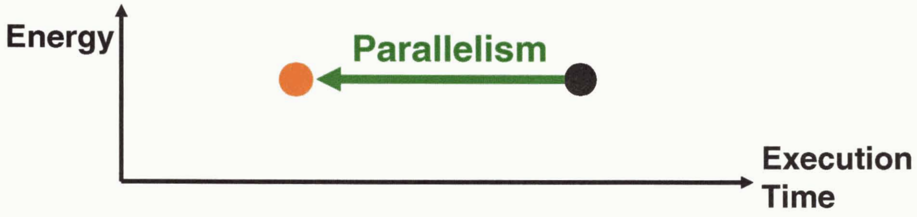


Figure 4-1: Employing parallelism.

ploitable parallelism as well as those employing the available parallelism directly. We divide this category into three sub-categories: pipelining and overcoming data dependences, running multiple execution units, and running multiple instruction streams.

4.1.1 Pipelining and overcoming data dependences

Pipelining is the most fundamental technique taking advantage of parallelism. In deep submicron technology, most high-performance processors employ deep pipelining.

Pipelining

Instructions are overlapped in execution by pipelining to increase throughput [HP96]. In even the most complex digital systems such as general-purpose processors, different instructions go through similar stages, which enables simple pipelining.

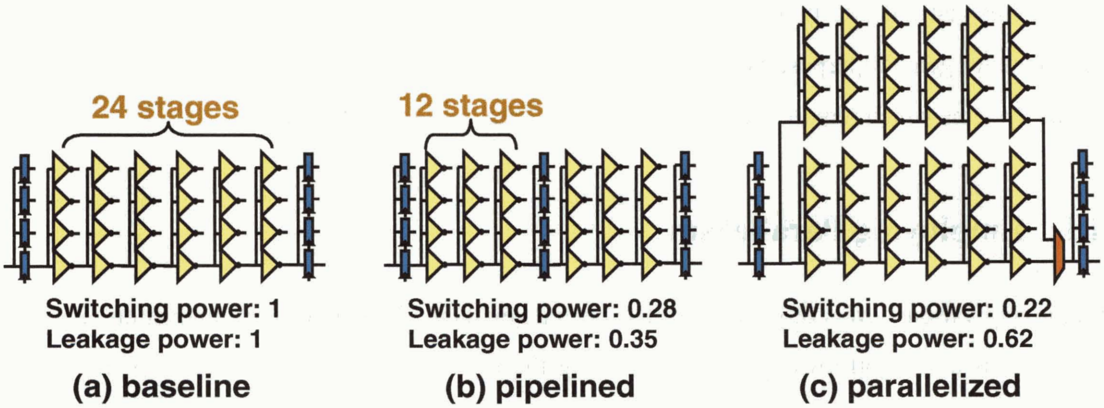


Figure 4-2: Comparing pipelined and parallel architectures. Throughput is fixed at 2 Giga-operations-per-sec (GOPS).

Compared to parallel architecture, pipelined architecture has better area- and leakage-efficiencies. Though they show similar energy-delay tradeoffs when leakage is insignificant, the pipelined architecture becomes superior when leakage power becomes comparable to switching power. Figure 4-2 shows the result of circuit simulations of pipelined and parallel architectures. A 24 FO4 inverter chain was built in a 70nm BPTM process [Dev01] and used as the baseline. The pipeline architecture dissipates only around half the leakage power of the parallel architecture.

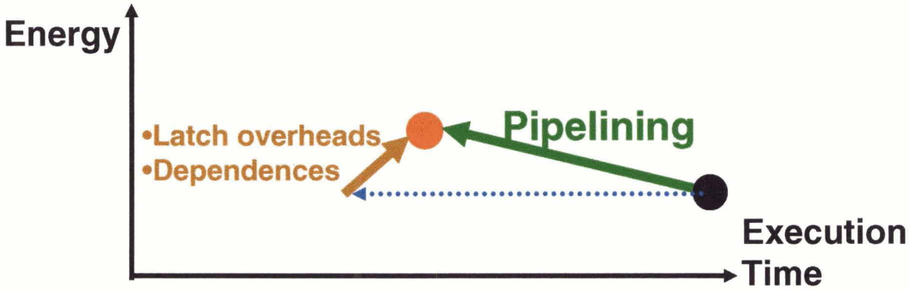


Figure 4-3: Pipelining.

Figure 4-3 shows the energy and delay effects of pipelining. Pipelining allows faster clock and greater throughput, thus initially providing a linear relation between power and performance (or constant energy with reduced execution time). The energy and delay of stage latches are the major circuit overheads [HA04b]. More details on power-optimal pipelining for logic datapaths can be found in Chapter 6.

Pipelining for general-purpose processors exploits instruction level parallelism (ILP) in programs. Exploiting ILP is limited by data and control dependences in programs (Figure 4-3). Architectural innovations such as bypassing and scheduling have been employed to reduce or hide the data dependences. On the other hand, speculative execution through branch prediction has been used to overcome control dependences by taking advantage of the intrinsic predictability within programs (Section 4.2).

Bypassing

Data hazards with short data-dependences can be overcome simply by providing shortcuts between pipeline stages. Bypassing forwards a functional unit's output to the same or other functional units' inputs and reduces the execution time. As the current trend toward a wider issue and deeper pipe continues, the size and complexity of bypass logic including wires, buffers, muxes, and control circuitry increase rapidly (eq. 4-1), as does energy and delay. (Figure 4-4).

$$complexity(bypassing) \quad \propto \quad width(pipe)^2 \times depth(pipe) \qquad (4.1)$$

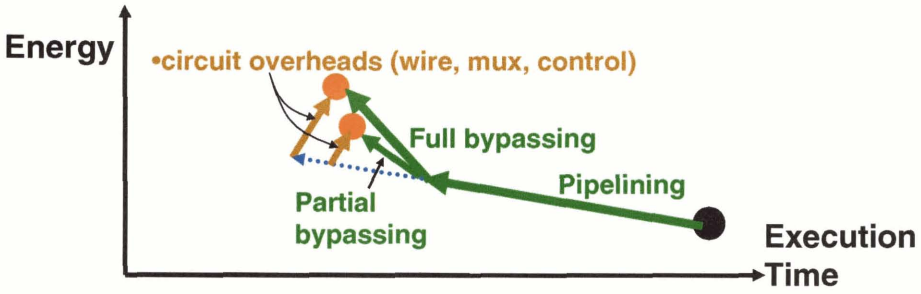


Figure 4-4: Bypassing to overcome data dependences.

Since full-bypassing in processors with wide and deep pipes is both energy- and delay-limiting, a proper subset of full-bypassing is often implemented instead while enduring some clocks-per-instruction (CPI) increase.

Scheduling

Instead of executing in program order, instructions can be reordered to reduce pipeline bubbles and effectively hide data dependences. While static scheduling depends on a compiler's dependence analysis to group independent instructions for simultaneous issue, dynamic scheduling (or scoreboarding) keeps un-issued instructions in large instruction windows, checks the dependences, and issues independent instructions with the help of wide issue logic. Dynamic scheduling often performs better because the data dependences which are unknown at compile time can be dynamically resolved with run-time only information. Tomasulo's algorithm extends scoreboarding with register renaming to avoid name dependences or false data dependences for a greater level of out-of-order execution. However, dynamic scheduling with register renaming comes with significant hardware costs (Figure 4-5). It requires multi-ported SRAM arrays with associative search and complex arbitration logic, which consumes a significant amount of switching and leakage energy as well as requiring large area. Aggressive dynamic scheduling techniques have led to the rapid increase of schedulers' area and energy but with diminishing returns in instructions-per-clock (IPC).

4.1.2 Running Multiple Execution Units

Processors can be extended to have multiple execution units to exploit a larger amount of parallelism. This sub-category includes the architectural innovations that help the efficient utilization of multiple execution units. The maximum achievable throughput is proportional to the number of execution units and most high-performance processors include multiple parallel execution units.

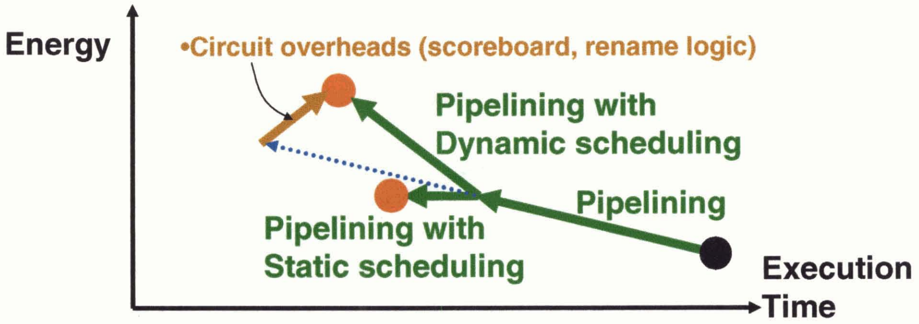


Figure 4-5: Dynamic and static schedulings to overcome data dependences.

Providing multiple execution units is relatively cheap and easy because the execution units are relatively small and can be duplicated, but keeping them busy is expensive and hard. Though processors tend to have a similar number and sort of execution units, they show a substantial difference in how they feed the multiple execution units. For example, a vector architecture keeps a single instruction flow, but a single vector instruction controls the simultaneous execution of multiple operations on multiple execution units. Superscalar or VLIW processors also keep a single instruction flow, but issue multiple independent instructions to the multiple execution units. On the other hand, many modern larger-scale processors have multiple physical or virtual cores and run multiple instruction streams on the multiple cores (Section 4.1.3). Each stream can also execute multiple operations on multiple executions units.

Issuing multiple instructions

To utilize multiple execution units, superscalar or VLSI processors dispatch multiple instructions per cycle, often in an out-of-order fashion. The essence of multi-issuing is to update and arbitrate ready instructions (the instructions with all operands ready and with no control dependence) as fast and energy-efficiently as possible.

Increasing the issue width further after a certain point brings only a marginal performance gain with an excessive increase of energy. While the intrinsic limit is the bounded ILP in programs, the non-scalability of some key centralized structures such as instruction windows, multi-ported register files, and bypass logic is another critical limit (Figure 4-6). Increasing the width and depth of pipelining requires the increase of number of ports and elements, which brings the rapid size increase of centralized structures, impacting area, cycle time, and energy significantly. Decentralization provides a solution for the scalability problem.

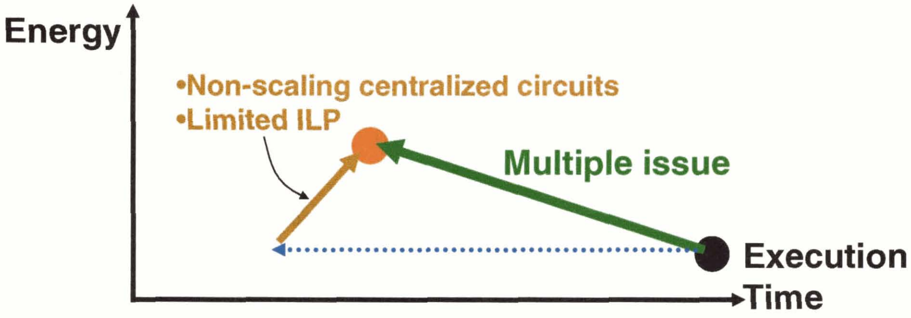


Figure 4-6: Issuing multiple instructions.

Decentralizing

Some centralized units in processors with multiple pipes do not scale well as the pipeline width increases. Decentralization divides the centralized units into smaller units to force area, delay, and energy to scale, while trying to minimize the pipeline bubbles due to the communication overhead among the decentralized units. Subbanking storage is an example of fine-grain decentralization. It can reduce the number of ports with a small increase in the total number of entries while keeping the same throughput requirement. It allows for a smaller, faster, and lower-energy storage design, but requires inter-bank communication, which can cause pipeline bubbles. Also, the increased control complexity can impact the cycle time. Figure 4-7 shows the effects of subbanking on energy and delay.

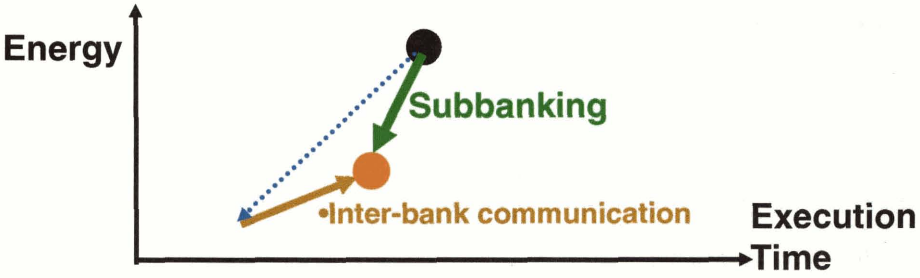


Figure 4-7: Subbanking storage.

Clustering is an example of coarse-grain decentralization. Splitting a microarchitectural block into distributed clusters makes it more amenable to scaling to larger pipeline widths. Clustered superscalar architectures are claimed to be more complexity-effective [Pal97] and inherently lower-power [Zyu00]. But, they require complex control logic to map instructions to clusters and handle inter-cluster dependences [TA03]. The accuracy of mapping heuristics is one of the most important factors for any successful clustered architecture.

4.1.3 Running Multiple Instruction Streams

As pipeline width increases, processors have reached the point where parallelism in a single instruction stream is not enough. Increasing the number of execution units gives only an incremental increase in performance for a significant amount of complexity, energy, and area increase. Coarser-grain parallelism has been sought for further performance increase with small energy and area penalties. In particular, exploiting thread-level parallelism (TLP) and running multiple instruction streams has been favored because of its energy-efficiency.

Multi-threading

A thread is a sequence of instructions and TLP indicates independent threads which can be executed in parallel. Multi-threading exploits TLP and improves performance with a small increase in transistor count [Mar01], by overlapping threads on a core. Overlapping can be time-multiplexed or it can be time-space-multiplexed, as in simultaneous multi-threading (SMT). Multi-threading lets a single physical core be viewed as multiple virtual cores by the operating systems and applications, and allows each instruction stream to run on each virtual core. Multi-threading is cost-effective in terms of area and leakage energy compared to having multiple physical cores, since most resources are shared among threads. But it still requires increases in some resources such as temporary storage to reduce the conflicts among threads. Because different and rather independent threads are multiplexed, the activity in functional units or storages is increased and so is the switching energy. There is also a possibility of performance degradation because the data locality is decreased due to the blending of different threads' data and the prediction based on the locality becomes harder. Figure 4-8 shows the effects of multi-threading on the energy and delay.

Multiple cores

Using multiple physical cores in a chip multi-processor (CMP) is a straightforward way of running multiple instruction streams. CMPs can provide higher throughput and consume less energy per operation than a wide-issue uniprocessor when applications have significant thread-level parallelism [ZA05]. Localizing computations and minimizing communication overheads among the cores, and controlling leakage on idle cores efficiently is critical for performance and energy-

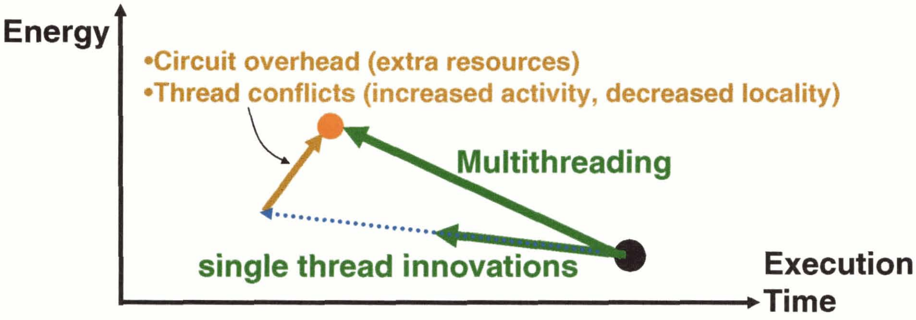

Energy
•Circuit overhead (extra resources)
•Thread conflicts (increased activity, decreased locality)
Multithreading
single thread innovations
Execution Time

Figure 4-8: Multithreading.

efficiency. Instead of full cores, vector [Asa98, Esp02] or multimedia units have been successfully added to main cores as slaves.

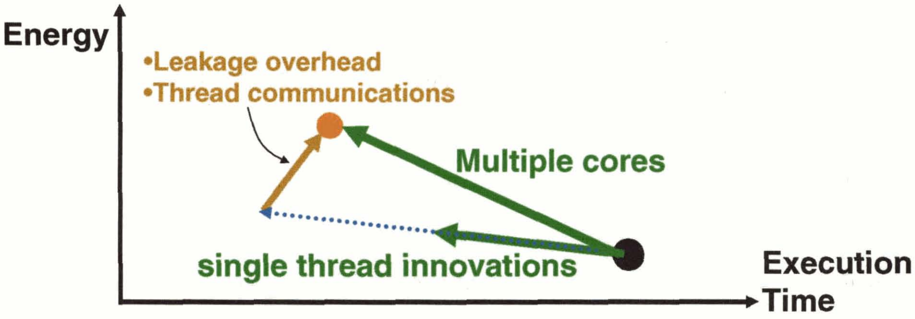


Figure 4-9: Utilizing multiple cores.

4.2 Exploiting Predictability

Control flow, data, and addresses in processors are not random but often correlated and predictable. Utilizing the intrinsic predictability has been a vital idea for many architectural innovations. The locality of reference [HP96] is the most prevalent predictability in processors. There are two types of locality of reference: temporal and spatial localities. Temporal locality is the concept that a resource that is referenced will be referenced again in the near future. On the other hand, spatial locality implies that a memory location has a higher chance of being referenced if a memory location nearby was just referenced. Constant-strided memory access is a form of spatial locality of reference. There is also the locality of value, which means that the same values keep being referenced. It exists because the majority of static instructions exhibit very little variation in the values that they write during the course of a program's execution [LS96]. Another important category of predictability is the independence of reference. We can sometimes optimistically assume no dependence among instructions and data, and execute them out-of-order.

Predictions are often combined with speculative execution (or speculation). Speculation through disambiguation, data prediction, and branch prediction are such examples (Figure 4-13). Misspeculation requires a verification and a full recovery to the previous state on misprediction, which leads to considerable performance and energy penalties. Consequently, the prediction accuracy is particularly important for speculation.

The intrinsic predictability has been exploited in two main ways: to reduce access latency and to overcome control dependences.

4.2.1 Reducing memory latencies

The latency gap between cores and off-chip memory has been one of the most critical obstacles to the performance improvement for processors and is getting worse as the pipeline depth increases.

Caching

Caching is a fundamental tool used to reduce the effective latency of reference. A cache is a small memory placed near the execution units, collecting copies of values that were originally stored elsewhere or computed earlier. Because of the small size and proximity, a cache can provide faster access latency. A cache stores not only recently-accessed references to exploit temporal locality, but also larger data chunks including the referenced values to exploit spatial locality. Caching can be done hierarchically for further latency reduction. Caching also saves energy because the cache accesses switch a smaller amount of parasitic capacitances compared to off-chip memory accesses.

Keeping the hit ratio high is crucial to effective caching, that is, lower access latency and lower energy. The hit ratio is affected by multiple factors such as capacity, line size, replacement policy, and associativity [SG83].

While increasing the capacity can directly increase the hit ratio and decrease the number of expensive trips to off-chip memory, the increased area might increase delay and leakage power. Subbanking can improve the delay while dynamic deactivation can reduce the idle subbanks' switching energy and leakage power.

Instruction and data caches in microprocessors are the most typical implementations of caching. However, caching can be used for other purposes than memory access latency reduction. For example, modern virtual memory systems take advantage of a small caching structure called the trans-

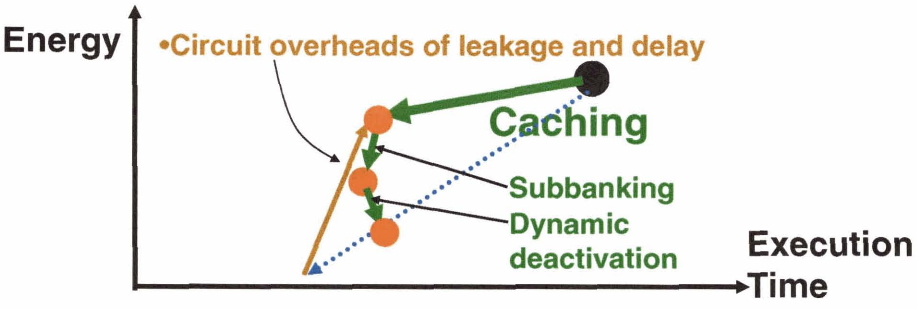


Figure 4-10: Caching.

lation look-aside buffer (TLB) for a faster and lower-energy virtual-to-physical address translation. The texture cache in a graphics processor, which captures texture mappings, is another important application of caching.

Prefetching

Prefetching is another approach to hide the long memory access latency. Prefetching does not wait until a miss occurs as caching does, but speculates what data would be referenced in the near future and fetches them in advance. Prefetching does not require verification or recovery mechanisms like other speculative techniques. Prefetching techniques rely on some heuristics for generating addresses of future memory references, such as exploiting common unit-stride or constant-stride accesses [LS96]. Figure 4-11 shows the impact of prefetching on energy and delay.

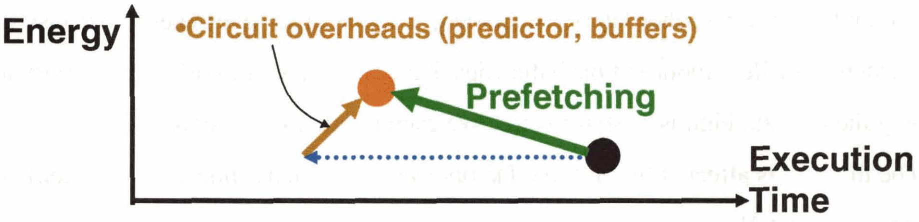


Figure 4-11: Prefetching.

Pre-execution

Pre-execution fetches data beforehand by executing loads earlier than the program order. Some delinquent loads are hard to prefetch because the accesses are irregular, and prone to cause secondary-level (L2) cache misses. For these problematic loads, we can issue a helper thread including the loads that runs ahead of the main thread. This makes the L2 cache misses occur earlier and the data values are ready in caches before the main thread needs them.

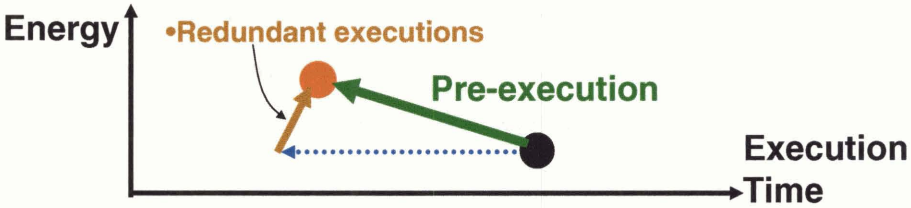


Figure 4-12: Pre-execution.

Figure 4-12 shows the effects of pre-execution on the energy and delay. Depending on the latency of L2 cache and the criticality of the delinquent loads that are selected for pre-execution, performance can be significantly improved by pre-execution. Even though the redundant executions always result in increased switching energy, the reduced execution time and increased idle time can lead to reduced overall leakage energy if leakage during the idle time can be dynamically deactivated.

Disambiguating memory accesses

Speculative disambiguation selects possibly-independent memory references following address-unknown memory references, and speculatively issues them in parallel for effective memory latency reduction. For example, memory operations inside loops often access array elements with a regular pattern, and tend to be independent. Figure 4-13 shows the impact of speculation through memory disambiguation on the energy and delay.

Predicting data values

Value locality is the likelihood of a previously-seen value recurring repeatedly within a storage location. Value prediction directly speculates on values by exploiting value locality [LS96], while previous techniques, such as caching, prefetching, pre-execution, and speculative disambiguation, speculate on the addresses. Value prediction can be extended to non-memory instructions as well as memory loads. Figure 4-13 shows the impact of speculation through data prediction on the energy and delay.

4.2.2 Overcoming Control Flow Dependence

The second sub-category uses predictability to overcome control dependences. Conditional branches are prevalent and frequent, especially in general-purpose processors with large issue widths. Branches limit the amount of exploitable ILP since we do not know in advance which path is correct to follow for the instruction fetch. Branch prediction foretells two things, direction and target address. While static branch prediction uses simple heuristics based on program analysis, dynamic branch prediction is based on the temporal locality of reference. Dynamic branch prediction is more accurate than static since the past behaviors of a branch strongly influence the current outcome and branch behaviors can change depending on input data. Usually, both predicted branch direction and target addresses are stored and updated in a small table indexed by the instruction address. The predicted branches are verified when branches are later resolved. In case of misprediction, machine state should be reverted to the state before the speculative execution. Figure 4-13 shows the impact of speculation through branch prediction on energy and delay.

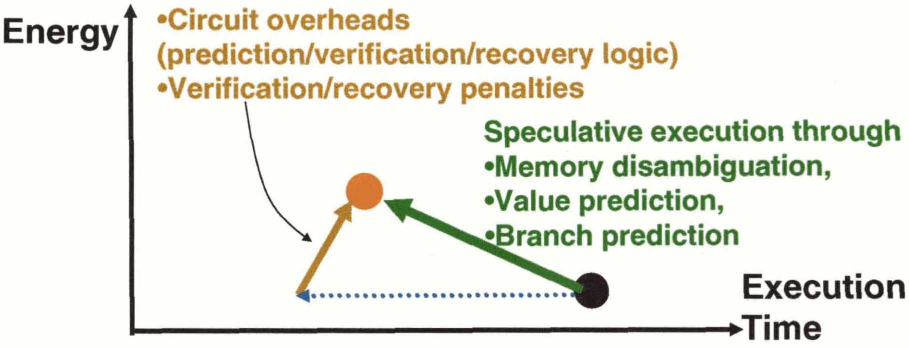

Energy
•Circuit overheads
(prediction/verification/recovery logic)
•Verification/recovery penalties
Speculative execution through
•Memory disambiguation,
•Value prediction,
•Branch prediction
Execution
Time

Figure 4-13: Speculative execution through memory disambiguation, data prediction, and branch prediction.

4.3 Reducing Energy Waste

Optimal digital system design requires architects to minimize energy inefficiencies (Chapter 1). Energy waste can be reduced at design time, or dynamic waste reduction techniques can be applied during run time. Figure 4-14 shows ideal energy waste reduction.

These architectural innovations sacrifice ease of design for increased energy-efficiency. To minimize unused and redundant tasks, they rely on temporally and spatially finer-grain controls. For example, breaking a memory into smaller subbanks and activating only the needed banks saves the

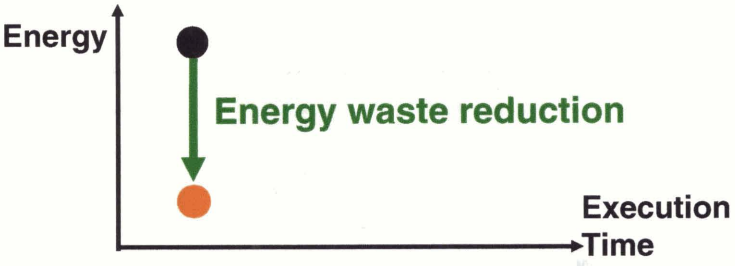


Figure 4-14: Ideal energy waste reduction.

energy dissipated by unused subbanks, but it requires individual control for each subbank and the increased complexity in control logic could impact cycle time.

4.3.1 Dynamically deactivating idle units

The basic idea of dynamic deactivation is simple: turning off idle units until needed later to save energy waste. Thus, dynamic deactivation causes frequent transitions in and out of deactivated and active states (more frequent for finer-grain dynamic deactivation) and requires a controller which controls the transitions. To find or make more deactivation opportunities, more complex microarchitectural policies are employed. The transition energy and delay costs and complex control circuitry are the main overheads for dynamic deactivation (Figure 4-15).

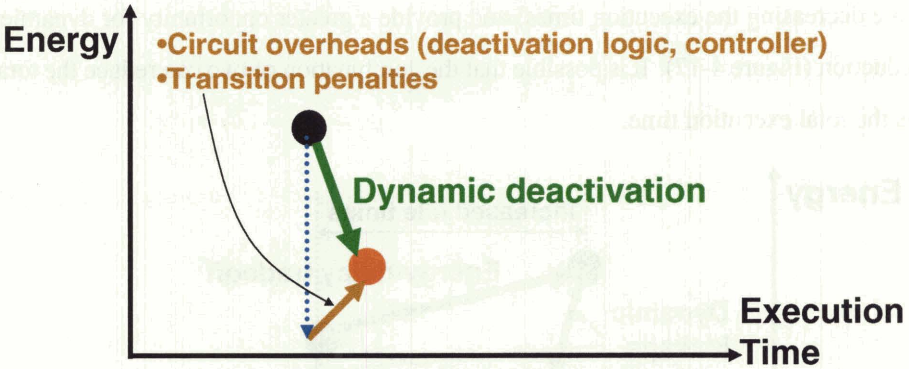


Figure 4-15: Dynamic deactivation.

The granularity of deactivation is diverse. The spatial granularity can vary from a few gates or subbanks to the whole core. Likewise, the temporal granularity can vary from a couple of cycles to OS context switching times. As long as the overheads are minimized, temporally and spatially fine-grain dynamic deactivation is more effective at energy waste saving. Figure 4-16 illustrates fine-grain dynamic deactivation.

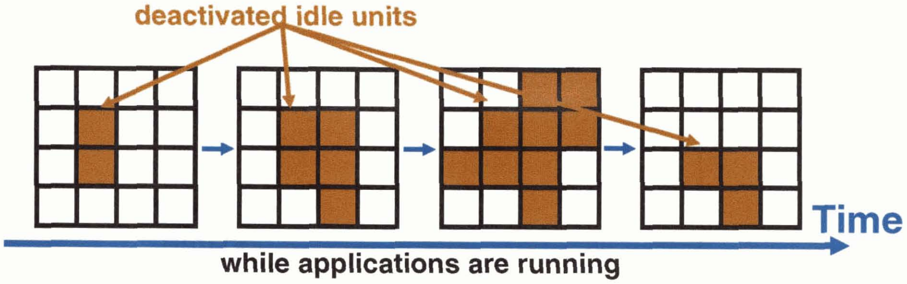


Figure 4-16: Temporally and spatially fine-grain dynamic deactivation.

Clock gating provides an efficient way of saving switching energy by gating the clock signal to the idle units. As well as spurious clock ticking, spurious data switchings within and after the idle units are decreased. For finer-grain control, hierarchical clock gating is usually employed. The main overhead is the increased complexity in clock control logics, which might impact clock cycle time.

Dynamic leakage deactivation forces the idle units to low-leakage states after some transition time. The co-design of low-overhead circuits and efficient architectural policies is necessary for effective dynamic leakage deactivation. Spatially and temporally fine-grain dynamic leakage reduction is preferred as long as the transition switching energy and delay penalties are kept small (Chapter 5). Performance-improving energy-delay tradeoffs increase the idle times between executions (while decreasing the execution times) and provide a greater opportunity for dynamic leakage energy reduction (Figure 4-17). It is possible that the combination of two can reduce the total energy as well as the total execution time.

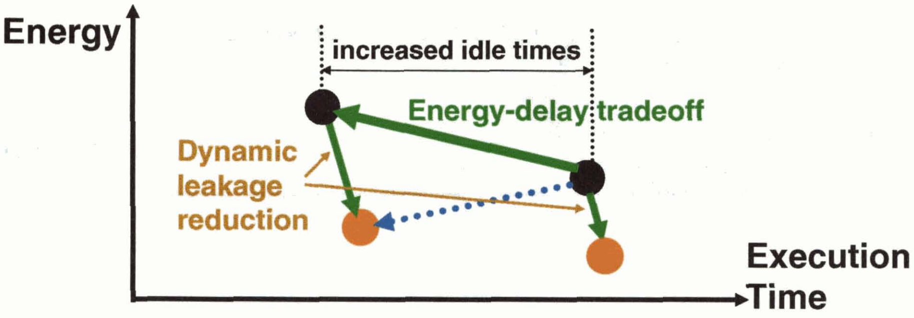


Figure 4-17: Combining an energy-delay tradeoff with a dynamic leakage reduction technique.

Dynamically adapting the effective sizes of resources is an example of dynamic deactivation. Every application shows a different resource utilization and even the utilization dynamically varies as an application is running. For example, the number of issue queue entries can be dynamically adapted according to the level of ILP in programs to save switching and leakage energy [ACG03].

4.3.2 Factoring out common operations

Common operations among instructions or operations can be factored out to save both energy and total execution time (Figure 4-18). For example, the vector architecture factors out instruction fetch and decode for multiple elements [Asa98] (Figure 4-19). The direct addressed cache is an example of microarchitectural-level factoring [Wit01], which factors out hardware tag checks when the compiler can guarantee that an access will be applied to the same line as the earlier access.

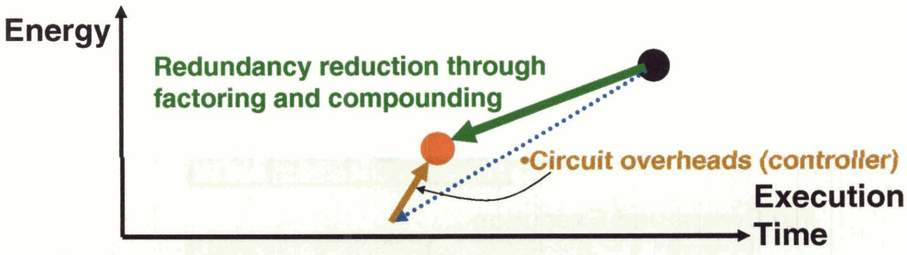

Energy
Redundancy reduction through factoring and compounding
•Circuit overheads (controller)
Execution Time

Figure 4-18: Factoring and compounding for waste reduction.

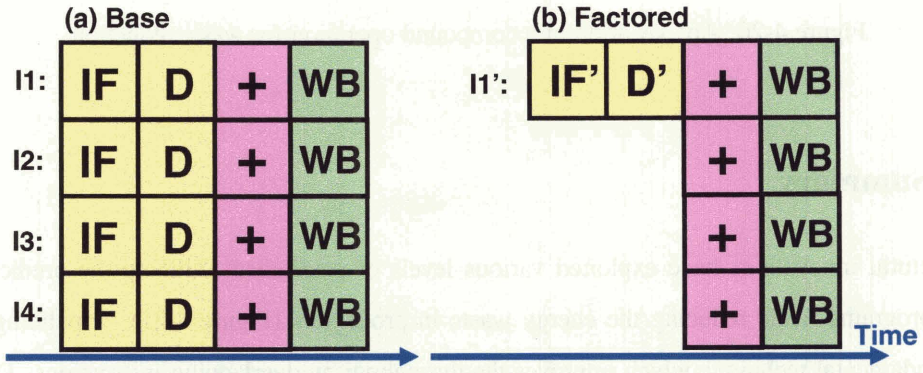


Figure 4-19: An illustration of factoring for waste reduction. IF, D, and WB represent instruction fetch, decode, and write back respectively. IF' and D' are factored instruction fetch and factored decode respectively.

4.3.3 Compound computations

Figure 4-20 illustrates an example of compound computations, where multiple operations are chained together without needing intermediate results to turn on storage such as register files or caches. It reduces the round trips to storages such as register files or caches if the intermediate computation results are used only once, which happen frequently in many applications. In addition to the total execution time reduction, compounding computation saves storage access energy and the energy for communications among functional units and memories (Figure 4-18). Many processors [Asa98, Kha01, San03] designed for computation-intensive multimedia tasks support compound execution. The increased control complexity is the major overhead.

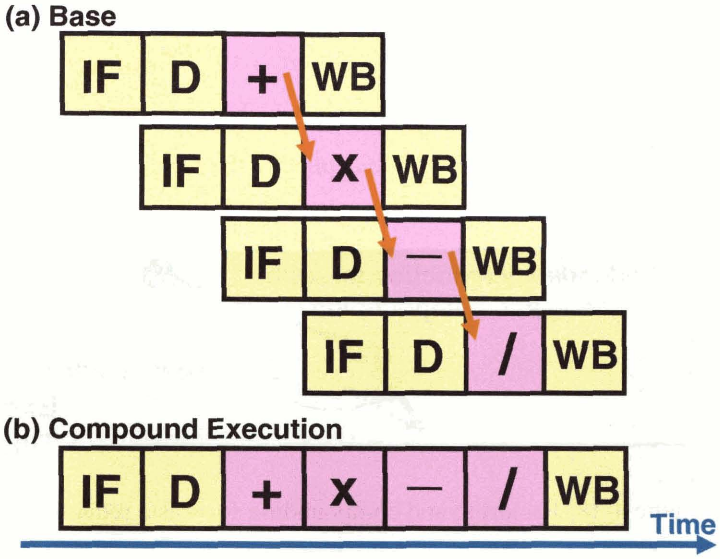


Figure 4-20: An execution of a compound operation for waste reduction.

4.4 Summary

Architectural innovations have exploited various levels of parallelism utilizing the predictability within programs while reducing the energy waste in processors (Figure 4-21). Pipelining is the most fundamental technique which improves the throughput by overlapping instructions. Data dependences limit the effectiveness of naive pipelining. Bypassing and scheduling can overcome some of the dependences with increased hardware cost. For a further performance improvement, multiple execution units have been employed. The major overhead of multiple issue is the non-scalable, centralized units such as issue logic and register files, but these can be reduced by decentralization. The efficiency of decentralization is limited by the inter-block communication costs. Multithreading and utilizing multiple cores exploit a coarser grain of parallelism, and run multiple instruction streams simultaneously. Communications and conflicts among threads are the main costs of exploiting TLP. The intrinsic predictability in programs are utilized in two ways: reducing memory latencies (caching, prefetching, pre-execution, memory disambiguation, and value prediction) and overcoming control dependences (branch prediction). Speculative execution requires accurate predictors and sometimes verification and recovery mechanisms, which all can be costly in terms of both energy and delay. Energy waste reduction techniques, such as dynamic deactivation, factoring and compounding, sacrifice the ease of design. The increased control complexity is one of the major performance-harming overheads for all energy waste reduction techniques, while dynamic waste reduction techniques can incur significant transition (between idle and active states) overheads.

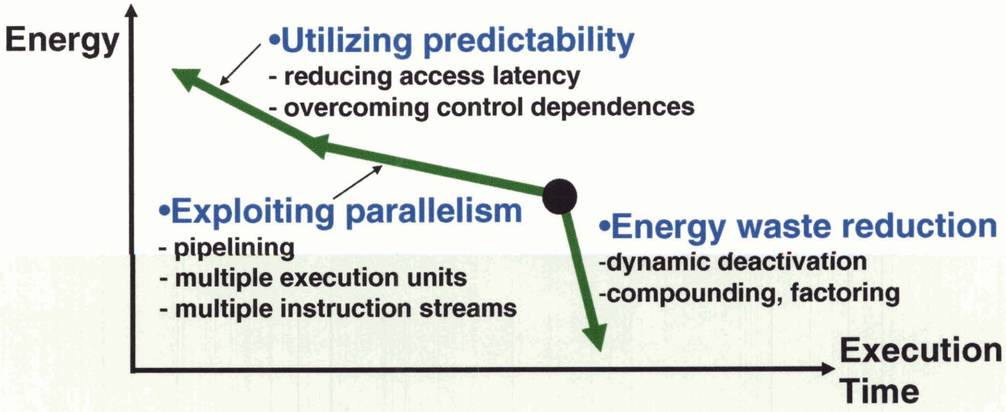

Energy
•Utilizing predictability
- reducing access latency
- overcoming control dependences
•Exploiting parallelism
- pipelining
- multiple execution units
- multiple instruction streams
•Energy waste reduction
-dynamic deactivation
-compounding, factoring
Execution Time

Figure 4-21: Architectural innovations.

Chapter 5

Fine-Grain Dynamic Leakage Reduction

The surge in leakage power has changed the optimal design philosophy for high-performance VLSI digital systems. It has made being idle expensive and being busy energy-efficient. Exploiting cheap transistors by adding complex structures for small performance increases is no longer effective, due to the leakage power. Stalls have become expensive in terms of energy as well as performance.

While switching power is unavoidable since digital information is transferred and stored through capacitance switching, leakage power is pure waste due to the intrinsic imperfection of transistors as switches and cannot be traded for performance. Leakage power is by far the most critical target for energy waste reduction. The energy saving can be reserved for longer battery life or converted to useful switching energy for further performance improvement.

In this chapter, we show that successful leakage power waste reduction leads to further optimized systems and that fine-grain dynamic leakage reduction (FG-DLR) is the key for further efficient leakage reduction [1]. Section 5.1 categorizes previous leakage reduction techniques into two groups: static and dynamic. After static leakage reduction techniques are applied, the leakage on critical paths becomes dominant, which requires dynamic leakage reduction. Section 5.2 describes the requirements for effective FG-DLR techniques, shows candidate microarchitectures for FG-DLR with a microprocessor example, and presents the metrics for comparing different FG-DLR techniques.

The later sections introduce new FG-DLR techniques that we have developed. We introduce a FG-DLR circuit technique, *leakage biasing* (LB), which uses leakage currents themselves to bias the circuits into the minimum leakage state and has very low transition delay and energy overheads. In

[1] The work in this chapter was a joint work with Kenneth Barr, Mark Hampton, and Krste Asanović and was previously published in [HBHA02, HA02, HA04a].

Section 5.3, LB is first applied to bitlines within primary memory arrays such as instruction caches and multiported register files for bitline leakage reduction. *Leakage-biased bitlines* (LBB) uses leakage currents themselves to bias the bitlines of unused memory subbanks into a low-leakage state. In Section 5.4, we apply LB to domino logic, presenting a new FG-DLR circuit family, *Leakage-Biased Domino* (LB-Domino) for critical functional units. LB-Domino uses sleep transistors only on non-critical paths and uses the leakage current itself to bias internal critical paths into a minimal leakage state.

Even within highly active critical blocks, many individual circuit paths are idle on any given cycle, though whether a path is active or inactive changes dynamically during operation. In Section 5.5, we introduce *Dynamically Resizable CMOS* (DRCMOS) logic that exploits the phenomenon to reduce leakage. DRCMOS dynamically downsizes transistors on idle paths while maintaining speed along active critical paths.

5.1 Leakage Reduction Techniques

We divide previous approaches to reducing leakage power into two categories. Techniques that trade increased circuit delay for reduced leakage current include: conventional transistor sizing, lower Vdd [Usa98, Tak98], stacked gates [Nar01, YBD98, HN97], longer channels [Mon96], higher threshold voltages [Lee97, Wei98, McP00, KC00, AAE00], and thicker T_{ox}; we collectively refer to these as statically-selected slow transistors or in short, static leakage reduction. Techniques for dynamic run-time deactivation of fast transistors include body biasing [Mut95, Kur96, Kur98, Mak98, Kes01], sleep transistors [Mut95, Shi97, KC00, Inu00, Kos01], and sleep vectors [YBD98, HN97]; we collectively refer to these as dynamically-deactivated fast transistors or in short, dynamic leakage reduction. Static and dynamic leakage reduction techniques are complementary approaches: Static leakage reduction techniques reduce leakage on non-critical paths and dynamic leakage reduction techniques reduce leakage on critical paths. Both can be simultaneously applied to yield larger overall savings [JSCR02].

Although many leakage-reduction techniques are implemented at the circuit or device level, digital system architects have considerable scope to influence digital systems' leakage power [BS00]. One approach is to increase the use of static leakage reduction by finding additional parallelism, so that a given throughput can be achieved with a larger parallel array of units built with slower, less-leaky transistors, rather than with a smaller number of lower-latency units built with faster but leaky transistors. Unfortunately, available parallelism is limited in single-threaded general-purpose applications, and much of the complexity of modern high-performance processors is due to the difficulties of finding such parallelism.

Alternatively, system architects can focus on finding opportunities to exploit dynamic leakage reduction, whereby fast, leaky circuits are deactivated when not required. This approach can potentially maintain the lowest latency for applications with little parallelism, while reducing leakage power to acceptable levels. The difficulty with this approach is that most existing circuit techniques for dynamic leakage reduction are only effective at reducing leakage energy if a circuit block will be inactive for a long time. This limits the scope for applying dynamic leakage reduction within an active digital system, where some blocks may only be inactive for a small number of cycles.

5.1.1 Static Leakage Reduction Techniques

Static leakage reduction techniques replace fast transistors with slow transistors on non-critical paths. This tradeoff has been done through two main ways: tuning the dimensions and structures of transistors and scaling voltages.

Tuning Transistors' Dimensions and Structures

In fact, static leakage reduction through transistor tuning has been a common design practice for many decades. Traditional transistor sizing reduces transistor gate width on non-critical paths to reduce parasitic load on critical nodes and save switching power. Leakage is proportional to gate width, and so these narrower transistors also have lower leakage. Non-critical paths also use slower, more complex gate topologies to reduce area. These more complex gates have deeper transistor stacks, which also reduce leakage. Recently, researchers have forced the transistor stack effect for leakage reduction [Nar01, YBD98, HN97]. It was reported that the forced stack is faster than lengthened transistors in a 180nm process for the same leakage saving [Nar01].

Leakage decreases superlinearly with gate length and a small increase in transistor length away from minimum can give a significant reduction in leakage current with a small impact on delay. This is because the effective threshold voltage is dependent on channel length when the channel lengths are short, a phenomenon called threshold-voltage roll-off. Accordingly, the designers of the StrongARM-1 slightly lengthened cache and pad transistors to reduce leakage in standby mode, yielding a five-fold reduction in leakage with only a small performance penalty [Mon96]. The Alpha 21164 used this approach to control the effects of leakage on dynamic gates [Gro96]. Lengthening gates is only effective for a small increment in channel length, and has the disadvantage of increasing switching power because of increased gate capacitance.

Scaling Voltages

Lowering supply voltage causes effective leakage reduction as well as switching power saving because it decreases the leakage current due to the drain induced barrier lowering (DIBL) effect. Multiple supply voltage schemes, where non-critical transistors are connected to a lower supply voltage, have been effectively utilized for low-power DSP applications [Usa98, Tak98].

Increasing threshold voltage reduces subthreshold leakage exponentially. The tradeoff is reduced gate drive, the difference between supply voltage and threshold voltage ($V_{dd} - V_T$) and thus reduced circuit speed. At the expense of additional mask processing steps, it is possible to manufacture transistors with several different threshold voltages on the same die. By using slower, high-threshold transistors on non-critical paths it is possible to reduce leakage current without impacting performance [Lee97, Wei98, McP00, KC00, AAE00]. PowerPC G6 microprocessors used low threshold-voltage transistors only at the most critical paths and minimized the percentage of the low threshold-voltage transistors to 3% [McP00]. Low threshold-voltage transistors were not used in arrays or dynamic circuits because of noise issues. Currently, high-threshold transistors are realized through higher source and drain doping and the higher doping leads to the increased electric field across the source and drain areas. In the near future, the increased junction BTBT leakage due to the increased electric field will make leakage reduction by multiple threshold transistor designs less effective [Aga05].

When we are allowed to manipulate both supply voltage and threshold voltage simultaneously, it is interesting to note that regardless of the delay requirement and activity factor, there exists an optimal ratio of leakage power. An analysis taking into account the short-channel effects and variation of threshold voltage and temperature finds that a simple guideline to optimize the power consumption is to set the ratio of maximum leakage power to total power around 30%. The value of about 30% is not changed over a wide range of design parameters such as activity factor, logic depth, and frequency [NS00]. If leakage power ratio is smaller than the optimal ratio, it indicates that some dynamic power can be converted to leakage power (for example, decreasing threshold voltage and decreasing supply voltage) while increasing performance. The increased performance can be reverted to total power saving through a power-performance tradeoff. Likewise, if leakage power ratio is larger than the optimal ratio, leakage power can be converted to dynamic power and performance can be increased.

When supply and threshold voltages are tuned simultaneously, the optimal threshold voltage level is not a strong function of either the clock frequency or the logic depth but of the activity factor. For example, memory blocks with low activity are optimized with high V_T transistors while clock circuits are built with very low V_T transistors. On the other hand, the optimal supply voltage level is a strong function of required performance.

5.1.2 Dynamic Leakage Reduction

Even though most transistors are non-critical, the achievable leakage reduction is limited, because the non-critical transistors have already been reduced in width and stacked into complex gates and hence have low leakage. Especially, after application of static leakage reduction techniques to non-critical paths, leakage is even more highly concentrated in the critical path transistors. One example is a recent embedded PowerPC 750, that employs three threshold voltages: high, standard, and low. The low threshold transistors account for only 5% of the total transistor width, but around 50% of the total leakage [Gei02].

Static replacement of slow transistors on critical paths would lead to increased cycle time. As a result, dynamic techniques that attempt to switch critical paths transistors between active, but leaky operation and inactive low-leakage states are necessary for significant leakage power saving.

Dynamic leakage reduction techniques exploit the voltage dependence or the input dependence of leakage for leakage reduction. Techniques in the first category increase the effective threshold voltage or decrease supply voltage or gate supply voltage dynamically, while those in the second category minimize leakage by forcing the lowest-leakage state on the given circuit, that is, turning low leakage transistors off and high leakage ones on.

Exploiting Voltage Dependence

One dynamic leakage reduction technique, popular in low-power digital systems for portable devices, is a dynamically varying body bias to modulate transistor threshold voltages [Mut95, Shi97, KC00, Inu00, Kos01]. Reverse body biasing (RBB), by setting the p-well voltage higher than Vdd and the n-well voltage lower than GND, increases V_T because of the body effect, thereby reducing leakage current. This technique requires twin or triple well processes and therefore increases manufacturing costs. RBB effectiveness diminishes as technology scales primarily because of worsening short-channel effects such as V_T roll-off and DIBL, particularly when V_T is low [Kes01].

A variation on the body biasing approach is to fabricate high-V_T transistors, then actively forward body-bias (FBB) the wells during normal operation to lower V_T [Miy00]. In the idle state, the forward bias is removed returning the transistors to their natural high-V_T state. An advantage of this technique is that it has less threshold variation than using low-V_T devices directly because of improved V_T roll-off and DIBL and hence can allow higher speed operation for a given leakage current budget [Miy00]. PMOS FBB has been successfully applied to an on-chip router [Nar02] and microprocessor ALUs [Tsc03] with small area and performance penalty.

Because of the large capacitance and distributed resistance of the wells, charging or discharging the well has a relatively high time constant and dissipates considerable energy. To allow the latency and energy costs of transitioning into the low leakage state to be amortized, these schemes are used when the system enters a sleep state where it will be idle for at least 0.1–100 μs [Set95, Kur96, Tak98].

An alternative dynamic leakage reduction approach is supply power gating [Mut95, Shi97, KC00, Inu00, Kos01]. The power supply to circuits can be cut off by inserting a high V_T *sleep transistor* between Vdd and virtual Vdd (or GND and virtual GND). When turned off, the sleep transistor adds an extra high-V_T transistor in series with the logic transistors, dramatically reducing leakage current. Some of the disadvantages of sleep transistors are that they add additional impedance in the power supply network that reduces circuit speed, they require additional area and routing resources for the virtual power supply nets, and they may consume considerable deactivation energy to switch between active and inactive states. By sizing the sleep transistor [KC00], boosting the gate voltage for the sleep transistor [Inu00], or forward-biasing the sleep transistor [Kos01], the delay penalty can be reduced in exchange for greater sleep leakage currents and increased deactivation energy. Fine-grained sleep regions based on local sleep devices were proposed and implemented [Cal04].

Instead of being gated, supply voltage can be lowered when the circuit is idle. Due to the DIBL effect, the reduced drain voltage results in reduced leakage current. Drowsy caches [Fla02] implement two supply voltage levels for SRAM arrays and change between them according to the activity of cache lines while preserving cache state.

Exploiting Input Dependence

Another interesting dynamic leakage reduction technique exploits the fact that the leakage current of a block depends on the input pattern and internal state [YBD98, HN97, KC00]. For example, embedded dual-V_T domino requires clock and inputs to be high to force the domino circuit to the lowest leakage state [KC00] when the circuit is idle. A *sleep vector* is a combination of input patterns and internal state which minimizes the leakage current, and is applied by forcing internal latches into the correct state and forcing inputs to the correct polarity. However, the application of the sleep vector can require additional circuitry, which reduces performance, and can cause spurious circuit switching, which results in significant deactivation energy.

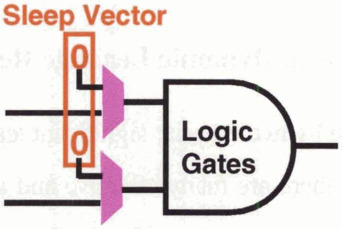


Figure 5-1: An example of sleep vector. We assume that the input vector, `00`, makes the logic gates spend the lowest leakage power and that there is no internal latch. The sleep vector is fed through input muxes.

5.2 Fine-Grain Dynamic Leakage Reduction

All dynamic leakage reduction circuits require a policy to decide when to switch to a low-leakage mode. Current energy-aware VLSI digital systems use a simple policy, usually implemented by the operating system, whereby the entire system is deactivated when it enters a sleep mode. This coarse-grain policy cannot reduce active mode leakage power. It is intuitively clear that turning off small pieces for short intervals is more effective at saving leakage. The tradeoff is the increased complexity.

We believe that fine-grain dynamic leakage reduction (FG-DLR) is the key for successful leakage power saving for digital systems in deep submicron technology. FG-DLR techniques focus on critical microarchitectural units and exploit short inactive times.

There are two components to effective FG-DLR techniques. First, we need circuit techniques with low active delay penalty, low energy overhead when moving in and out of sleep, and fast wakeup time. Many existing dynamic leakage reduction techniques are only effective given a long sleep time since the circuit techniques have long wakeup time or high energy costs when switching between inactive and active modes. Deep understanding of leakage characteristics within circuits will give insight to novel FG-DLR circuit designs.

Second, we need microarchitectural scheduling to keep the sleep time as long and often as possible. Most of previous work focused on only circuit-level techniques and ignored the importance of microarchitectural policies. However, even the best FG-DLR circuit technique will be useless if no sleep time is available. By studying microarchitecural behaviors of units, we can find when FG-DLR circuit techniques can be applied to the units and how sleep opportunities for the units can be increased.

5.2.1 Candidates for Fine-Grain Dynamic Leakage Reduction

Candidates for FG-DLR are chosen by two criteria: significant leakage power and critical impact on cycle time. For non-critical units, there are many effective and applicable static leakage reduction techniques that trade delay for lower leakage by replacing fast and leaky transistors with slow and low-leakage ones. For instance, an L2 cache can have large leakage due to a large number of transistors, but is not chosen as a candidate unit since static leakage reduction techniques, such as using a longer channel, lowering the supply voltage, or employing high threshold transistors, can be effectively applied. A large portion of units in a microprocessor's core satisfy these two criteria. Figure 5-2 shows examples of target units within a superscalar microprocessor.

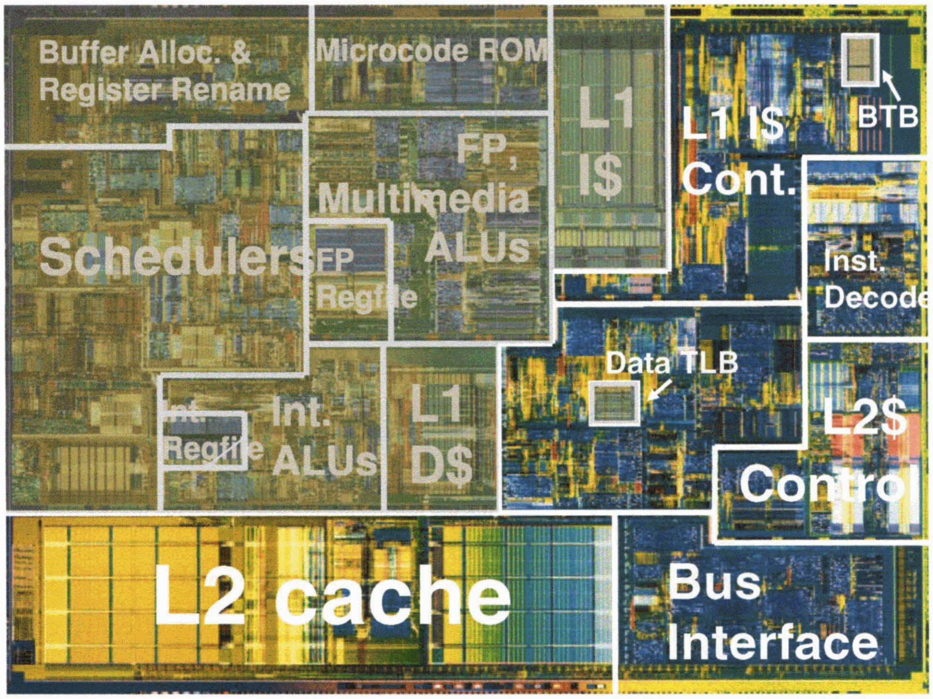


Figure 5-2: Target microarchitectural units in a typical modern superscalar microprocessor, Pentium 4 [Hin01a]. The target units are shaded.

We find that the possible candidates for FG-DLR can be categorized into three groups according to structure and circuit style.

The first is a group of primary SRAM arrays that provide fast temporary storage. They are usually multi-ported and multi-banked. L1 instruction and data caches and register files are such examples. The delays are so critical that the sizes are limited to make sure that the access latencies take only one or few cycles. The massive number of transistors makes them significant leakage contributors. Most leakage currents flow through bitlines as access transistors must be fast.

The second is a group of critical functional units. These are the essential blocks responsible for the arithmetic and logic calculations. Many modern processors have multiple fast integer and floating-point ALUs to exploit available parallelism. Commonly, they are part of critical datapath loops, and they are built with large and fast transistors and thus tend to be leaky. A dynamic circuit style is dominantly used for them, but the reduced noise margin due to the leakage increase has made more stable static circuit styles more popular again.

We call the last group complex logic arrays. They are prevalent, especially in modern out-of-order superscalar microprocessors. They consist of SRAM arrays and intermixed complex logic. Tag arrays, branch target buffers (BTB), translation look-aside buffers (TLB), rename tables, schedulers, and instruction windows are typical examples. As the issue width and level of speculation increases, the size and number are rapidly growing and so is the leakage power.

5.2.2 Comparing Fine-Grain Dynamic Leakage Reduction Techniques

The ultimate goal of leakage reduction is total energy reduction with minimum performance loss. Thus, when attempting to deactivate a block for a short period of time, the performance and energy impacts of entering and leaving the low-leakage state must be considered.

Figure 5-3 introduces the different parameters we use to compare FG-DLR techniques. The left-hand side of Figure 5-3 shows the evolution of leakage current over time on entering the deactivated state. Once deactivated, a block requires some time to reach the lowest leakage state. For example, a substrate biasing scheme will require time to bias the wells, and a virtual-GND scheme requires time for leakage currents to charge up the virtual-GND node. During the transition, leakage current can be substantially higher than in the steady state.

For clarity, this graph only shows the leakage current. When switching into and out of the low-leakage state there can be substantial switching current spikes. Switching between active and deactivated modes requires additional transition energy, for example, to switch the gates of power-

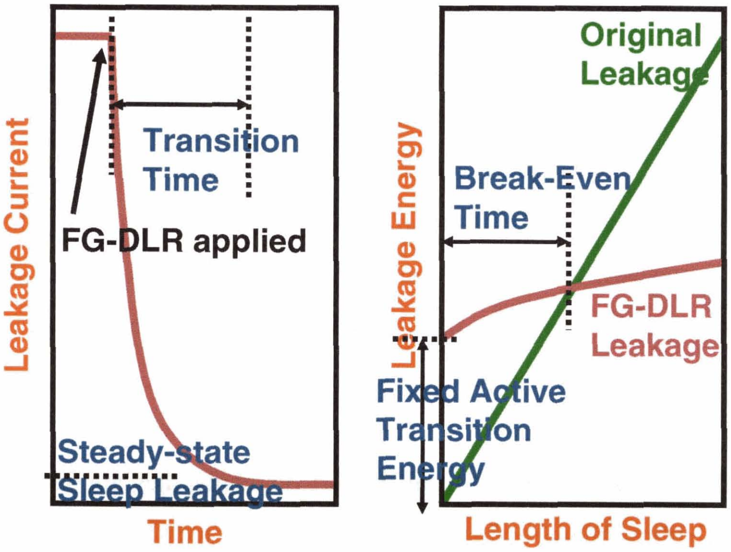


Figure 5-3: Transition time, steady-state leakage current, and break-even time of FG-DLR techniques.

gating transistors or to charge and discharge well capacitances. The right-hand side of Figure 5-3 illustrates how we compare the overall energy consumed over time when idling in a normal, high-leakage state versus transitioning into a low-leakage state. The original idle leakage energy is shown by the straight line which rises at a constant rate dependent on the leakage current. On the same graph, we show an example curve for a dynamic leakage reduction technique. The fixed energy costs of first moving to the low-leakage state, then moving back to the active state, are summed to give the fixed transition energy cost. In addition to the fixed transition energy, there may be additional variable transition energy costs proportional to the time that the block is deactivated. For example, in a virtual GND scheme, the virtual GND node is slowly charged over the transition time. The amount of energy dissipated when the block wakes up and discharges the virtual GND depends on the idle time. These variable transition energy costs are factored into the energy curve. The curve rises more steeply initially during the transition time, where variable transition energy costs are being incurred and as leakage current drops to its steady state value. After the transition time, the energy curve rises more slowly, as only the steady-state leakage current is being dissipated.

We define the break-even time as the time at which the two curves cross, i.e., when the leakage energy of remaining in an active idle state matches the energy consumed when switching to the low-leakage state. The circuit must be idle for considerably longer than the break-even time to save significant energy.

Another important factor in comparing FG-DLR techniques is the wakeup latency (not shown). The wakeup latency is the time for a block to become usable after being in an inactive state. Faster wakeup time is usually preferable to faster transition time because it reduces any performance penalty. Wakeup latency can sometimes be traded for transition energy, for example, using a wider transistor to accelerate discharge of a biased well increases the transition energy to switch the transistor.

Although FG-DLR techniques do not use slow transistors to reduce leakage power, some dynamic leakage reduction techniques affect the delay and power of the active state. For example, the NMOS sleep transistor technique causes virtual GND to be a slightly higher potential than GND and so the circuit is somewhat slower.

5.3 Leakage-Biased Bitlines for SRAM Arrays

In this section, we introduce *leakage-biased bitlines* (LBB) that uses leakage currents themselves to bias the bitlines of unused memory subbanks into a low-leakage state. We apply LBB to the instruction cache and register file of an out-of-order superscalar microprocessor. For the instruction cache, we deactivate idle subbanks with LBB. There can be a small performance penalty from the latency increase due to possibly-delayed precharge. For register files, we exploit idleness in two spatial dimensions. We deactivate subbanks when their registers are unused, and disable unused read ports when there is not enough parallelism to keep them busy. Leakage energy can be saved with no loss in performance and minimal area overhead.

Section 5.3.1 describes how we estimated the process parameters for future process technologies. Section 5.3.2 introduces leakage-biased bitlines and describes how we apply it to an instruction cache. Section 5.3.3 describes how we apply LBB to a multiported register file. Section 5.3.4 discusses the results from our evaluation. Section 5.3.5 describes related work.

5.3.1 Process Technologies

To evaluate our dynamic leakage reduction techniques, we used models of four dual-V_T processes, including 180 nm, 130 nm, 100 nm, and 70 nm process generations. The 180 nm high-V_T and low-V_T transistors were modeled after 0.18 μm TSMC low-leakage and medium-V_T processes respectively. The parameters of the 180 nm process were scaled to future technologies using the SIA road-map [Int00]. For example, the SIA road-map predicts that I_{on} remains the same, but I_{off} jumps twice for each technology generation. Because of the difficulty in predicting future leakage numbers, we bracket our results using our own pessimistic and optimistic estimates of how leakage currents will scale. The pessimistic estimates assume 4× leakage increase per generation while the optimistic estimates assume 2× leakage increase per generation. Important parameters of the processes are summarized in Table 5.1. We believe future leakage currents might be considerably higher that even our pessimistic numbers indicate, as these are based on a low-leakage, moderate-performance base case.

Based on the table, we estimated the scaling of switching and leakage power for circuits. The results are shown in Figure 5-4, where numbers are normalized to the 180 nm process. It is important to note that leakage power per transistor increases significantly although Vdd and the total area of the circuit decreases. The switching power is decreasing quadratically as expected from constant

Table 5.1: Process parameters.

Parameter (nm)	180	130	100	70
Vdd (V)	1.8	1.5	1.2	0.9
Temp (Celsius)	100	100	100	100
FO4 delay (ps)	61.1	47.4	36.7	24.0
16 FO4 freq. (GHz)	1.0	1.3	1.7	2.6
LVT I_{on} (μA/μm)	732	732	732	732
LVT I_{off} (nA/μm) (optimistic)	21.8	43.6	87.2	174
LVT I_{off}(nA/μm) (pessimistic)	21.8	87.2	349	1395
HVT I_{on} (μA/μm)	554	554	554	554
HVT I_{off} (nA/μm) (optimistic)	0.35	0.71	1.42	2.83
HVT I_{off} (nA/μm) (pessimistic)	0.35	1.42	5.68	22.6

field scaling. If the leakage power was 10% of the total power at the 180 nm process, it will increase to 47-87% for the 70 nm process if the circuit is scaled unchanged. In practice, devices, circuits, and microarchitectures will be redesigned to limit leakage to a manageable fraction of total power. The techniques in this work can be used to help keep leakage current within this budget without sacrificing performance.

5.3.2 Leakage-Biased Bitlines for Caches

The primary caches (L1 caches) can cause significant leakage current, as they contain a large number of transistors which must be high-speed to avoid impacting system performance. Figure 5-5 shows the structure of an L1 cache SRAM cell together with the two primary leakage current paths when the word line is off. One leakage path, L_1, is from the precharged bit-line, through the access transistor, and across the turned-on n-type pull-down. The other leakage path, L_2, is from the enabled p-type pullup to the turned-off n-type in the cross-coupled inverters. The pullup transistors have been made high-V_T so that there is negligible leakage current across the turned-off p-type ($L_3 \simeq 0$), and the access and pulldown transistors have been made low-V_T to maintain circuit speed. The current through the leakage path, L_4, is insignificant since the path has two turned-off transistors and the V_{DS} of the access transistor is zero.

With technology scaling, the leakage currents from non-accessed bits will reduce the effective signal from the accessed bit, requiring that SRAMs have fewer cells connected to each bitline segment to obtain sufficient noise margin. We assume that only 32 bit cells are attached to each local bitline within a subbank, and that these local bitlines are connected through pass-transistor switches to a global bitline which in turn connects to the senseamp.

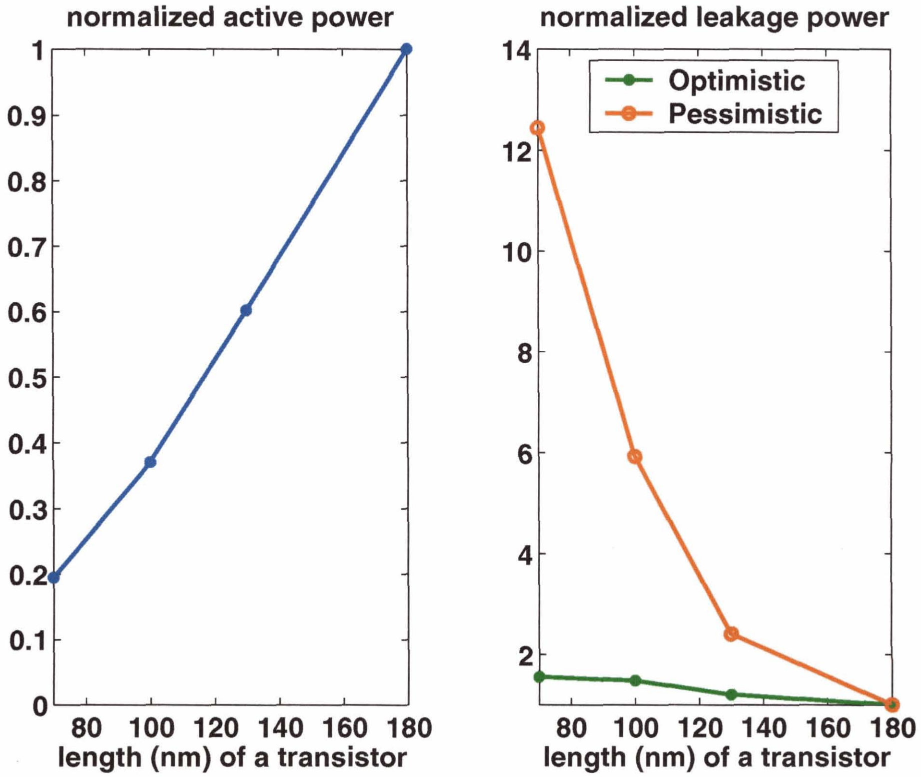


Figure 5-4: Normalized switching and leakage power for different processes.

A key observation is that the leakage current, L_1, from each bitline into the cell depends on the stored value on that side of the cell; there is effectively no leakage if the bitline is at the same value as that stored in the cell (L_4). We might consider using a sleep vector on the bitlines to force the SRAM subbank into a low leakage state. For example, it is known that there are usually more zeros than ones stored in a cache [VZA00], so if we force the true bitline to a zero value while keeping the complement bitline precharged, we could statistically reduce the bitline leakage of an inactive cache subbank. There are two disadvantages to this approach. First, if the percentage of zero bits is under 50%, the sleep vector technique *increases* leakage energy. Second, this technique requires additional transition energy to force the bitlines into and out of the sleep vector state.

We have developed a simple circuit technique, *leakage-biased bitlines* (LBB) (Figure 5-6), that reduces bitline leakage current due to the access transistors of these structures with minimal transition energy and wakeup time. Rather than forcing zero sleep values onto the read bit lines of inactive subbanks, this technique just lets the bitlines float by turning off the high-V_T NMOS precharging transistors. The leakage currents from the bit cells *automatically* bias the bitline to a mid-rail volt-

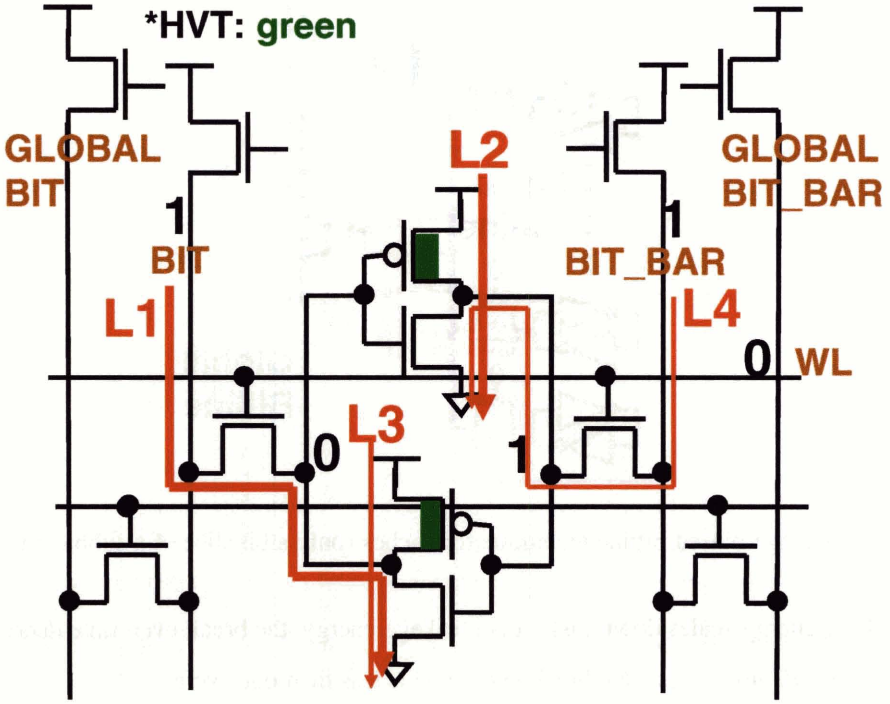


Figure 5-5: A dual-V_T SRAM cell.

age that minimizes the bitline leakage current. If all the cells store a zero, the leakage currents will fully discharge the non-inverted bitline ("BIT" in Figure 5-5) while the inverted bitline ("BIT-BAR") will be held high. If all the cells store a one, the non-inverted bitline will be held high and the inverted bitline will discharge. For a mix of ones and zeros, the leakage currents bias the bitline at an appropriate mid-rail voltage to minimize leakage. Although the bitline floats to mid-rail, it is disconnected from the senseamp by the local-global bitline switch, so there is no static current draw. This technique has little additional transition energy because the precharge transistor switches exactly the same number of times as in a conventional SRAM—we only delay the precharge until the subbank needs to be accessed. The wakeup latency is just that of the precharge phase.

Figure 5-7 compares the steady-state leakage power of the leakage-biased bitline and the forced-zero/forced-one sleep vector techniques with the original leakage power for a 32-row × 16B SRAM subarray with varying numbers of stored ones and zeros. It is clear that the leakage-biased bitline technique has the lowest leakage power independent of stored bit values.

Figure 5-8 compares the cumulative idle energy and the LBB energy consumption for different processes. The LBB technique must replace the lost charge on the bitline before the attached memory cells can be used. The break-even time, is around 200 cycles in a 180 nm process. However,

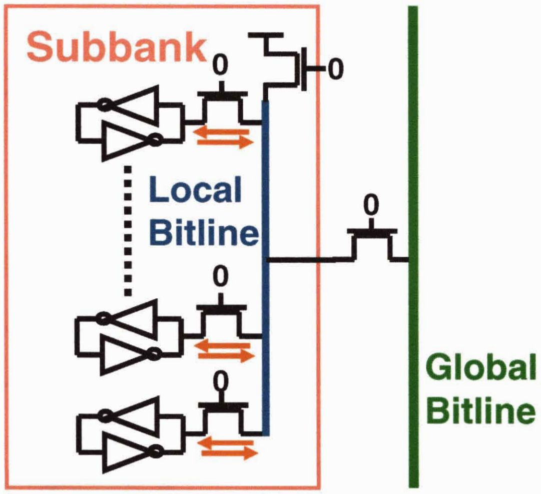


Figure 5-6: Leakage-biased bitline technique for caches (only a bit slice of a subbank is shown).

since switching energy scales down faster than leakage energy, the break-even time decreases with feature size. In a 70 nm process, the break-even time is less than one cycle.

Each subbank must be precharged before use, which will add latency to the cache access if the subbank is not known in time. We focus in this work on the application of LBB fine-grain dynamic leakage reduction technique to the instruction cache, because of its predictable access pattern. For an N-way set-associative cache structure, each way consists of some number of subbanks and we access N subbanks in parallel, where each subbank returns a fetch group of instructions. In the most optimistic case, we can assume that the simple subbank decode happens sufficiently before the more complex word-line decode to allow precharge to complete before word-line drive; in this case, there would be no performance penalty. In the most pessimistic case, we can assume that the additional precharge latency adds an additional cycle to the fetch pipeline, and hence increases the branch misprediction penalty by one cycle.

5.3.3 LBB for Multiported Register Files

Multiported register files can also consume considerable leakage power. For example, in the proposed Alpha 21464 design, the multiported register file was several times larger than the 64 KB primary caches [Pre02]. Figure 5-9 shows an 8-read port, 4-write port, register file cell. Because there are many leakage paths in a multiported register file cell, we chose a baseline design that was already optimized for leakage power. The cell has a high-V_T storage cell connected to multiple low-V_T single-ended read ports. The write ports are not as latency critical and so these access

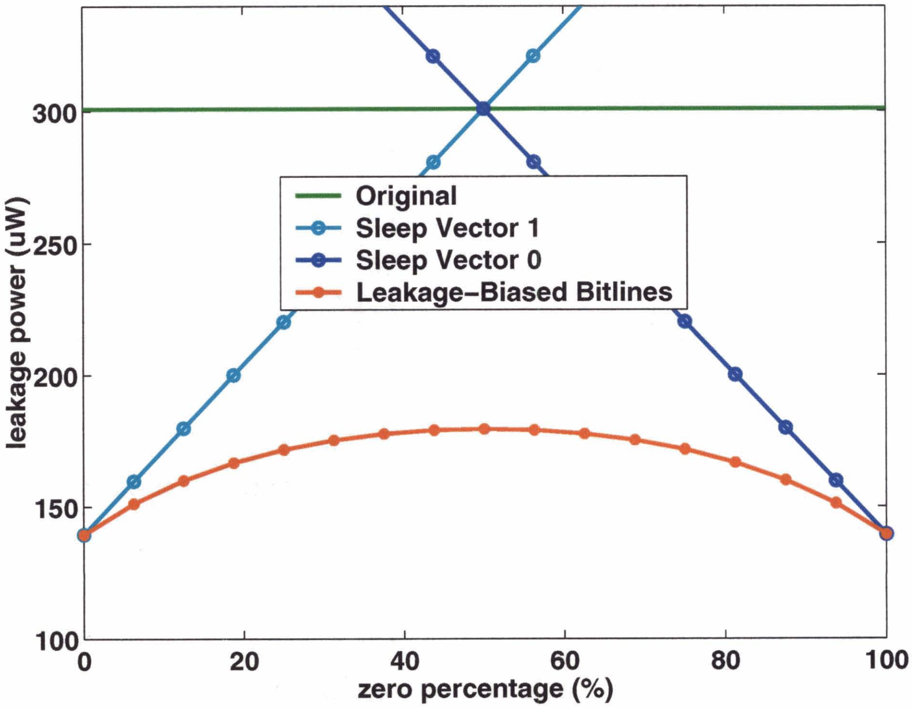


Figure 5-7: The leakage power of 32-row×16B SRAM subbank for forced-zero and forced-one sleep vectors and leakage-biased bitlines versus percentage of stored zero bits.

transistors are high-V_T. To reduce switching and leakage energy further, we make the cell asymmetrical, with all read ports arranged so that if the cell stores a zero, the single-ended bitline is not discharged [TA00]. Our experiments showed that around 75% of the bits read from the register file are zero.

As with the cache SRAM, the register file array is divided into subbanks with local bitlines connected to global bitlines to save switching energy and to increase speed and noise margin. The LBB technique can be applied to the single-ended read port bitlines. By turning off the precharger on an idle subbank read port, leakage currents will discharge the bitlines towards ground if any bits are holding a one, reducing bit line leakage current significantly. If the dead time is long enough, the energy overhead to precharge the bitline before an access becomes relatively small compared to the leakage energy saved. Note that this technique does not corrupt the state stored in the register file. Figure 5-10 shows the hierarchical bitlines and a modified column cell for the LBB multiported register file.

We deactivate the register file for lower leakage using two orthogonal techniques (Figure 5-11). The first deactivates dead registers whose contents are not needed. We exploit the fact that in a

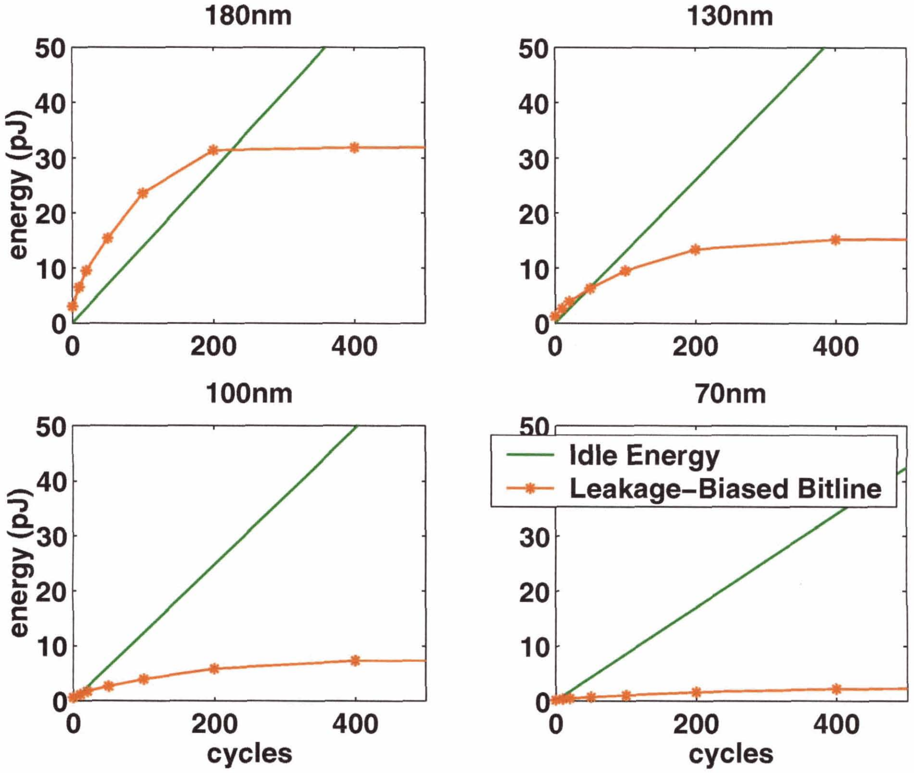


Figure 5-8: Idle energy and LBB energy of 32-row×16B SRAM subbank for different processes (optimistic leakage current was used).

superscalar machine with register renaming, the contents of a physical register are not needed from the time the register enters the free list until the time it is next written. If all the registers in a subbank are dead, then all subbank read ports can be turned off. Because the register is allocated in the decode stage of the pipeline and written to several cycles later (and before any read access), there is ample time to precharge the floating bitline with no performance impact. As a comparison, we considered an alternative dynamic leakage reduction approach to turn off dead registers using a virtual-GND sleep transistor (Figure 5-11). This approach has the advantage that registers can be turned off individually, rather than a subbank at a time, but has the disadvantage that the read access time increases due to the sleep transistor in the pull-down path. The delay penalty can be reduced by increasing the size of the sleep transistor, but this also increases the steady-state leakage current and the transition energy. We sized the sleep transistor to give an overall 5% slowdown.

The second technique deactivates idle read ports (Figure 5-11). In a superscalar machine, when fewer than the maximum number of instructions issue, some register file read ports will be idle. There is no performance impact when the port is reactivated because it is known whether a read port

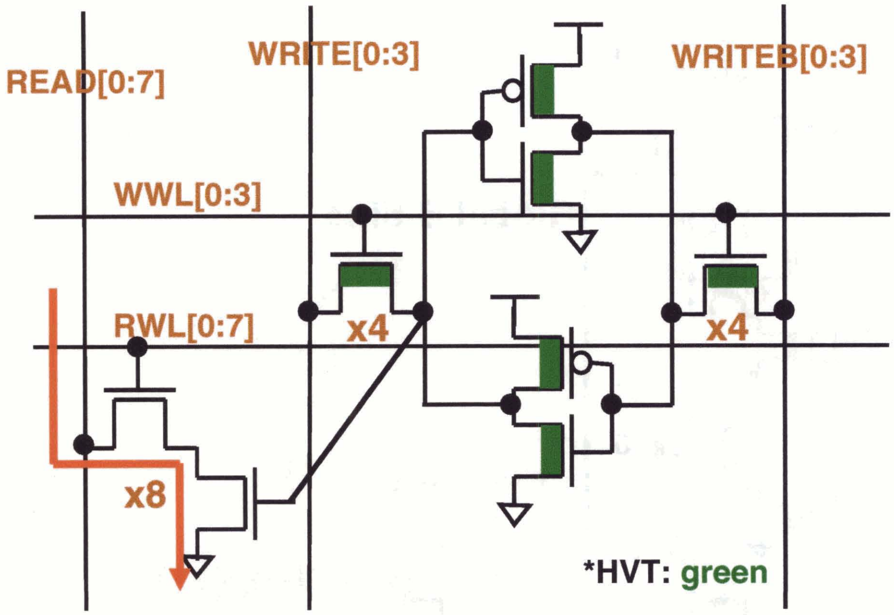


Figure 5-9: An embedded dual V_T unbalanced 8-read, 4-write register file cell.

Table 5.2: The switching read and write energy consumption of 32×32b multiported register file subbank for different processes.

transistor length(nm)	180	130	100	70
zero read E(pJ)	6.0	2.9	1.4	0.5
one read E(pJ)	17.3	8.2	4.0	1.4
average read E(pJ) (E_r)	8.8	4.2	2.0	0.7
0-to-0 write E(pJ)	0.7	0.4	0.2	0.1
0-to-1 write E(pJ)	16.5	7.9	3.8	1.3
1-to-0 write E(pJ)	2.2	1.0	0.5	0.2
1-to-1 write E(pJ)	13.0	6.2	3.0	1.0
average write E(pJ) (E_w)	4.7	2.3	1.1	0.4

is needed before it is known which register will be accessed in the pipeline. The port precharge time can be overlapped with register file address decode.

Table 5.2 shows the energy consumption when reading/writing 32-bit zeros or ones from the 32×32-bit register file with the unbalanced embedded dual V_T cells. All read/write energy numbers are per single read/write port. The energy consumption for 180 nm was measured using Hspice simulation and those for other processes were scaled using Figure 5-4. The average read and write energy numbers were calculated assuming 75% of values stored in the register files and write data are zero and that the values are statistically independent. The total switching energy consumption is simply the sum of total read energy and total write energy.

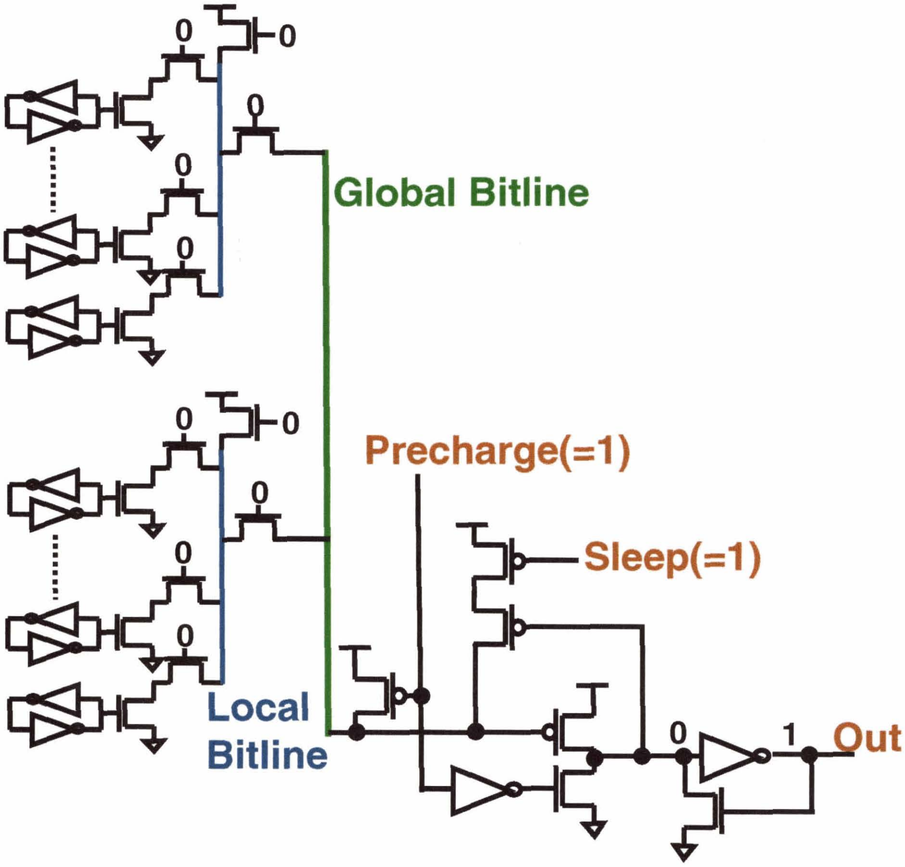


Figure 5-10: Leakage-biased bitline scheme for multiported register file. Each local bitline can be left unprecharged, biased by local leakage currents.

Table 5.3 shows the steady-state leakage power when different leakage techniques are applied to the register file and the idle leakage power of the original circuit for different processes. We again assumed 75% of the bits in the register file are zeros when measuring the leakage power. We also include numbers for the sleep vector fixed-zeros technique. All three techniques, sleep vector (SV), leakage-biased bitline (LBB), and NMOS sleep transistor (NST) reduce the leakage power to less than 1.5% of the original idle power when in the steady state.

Figure 5-12 shows the sleep-time dependent energy consumption of the register file dynamic leakage reduction techniques across the set of process technologies. We can see that all of the dynamic leakage reduction techniques become applicable at shorter time scales as transistors scale down. This is partly because leakage current grows as a fraction of switching power, but also partly because most of the transition energy cost scales with switching power and so the relative overhead of switching is reduced. Figure 5-13 is an expanded view of the graph for the 70 nm

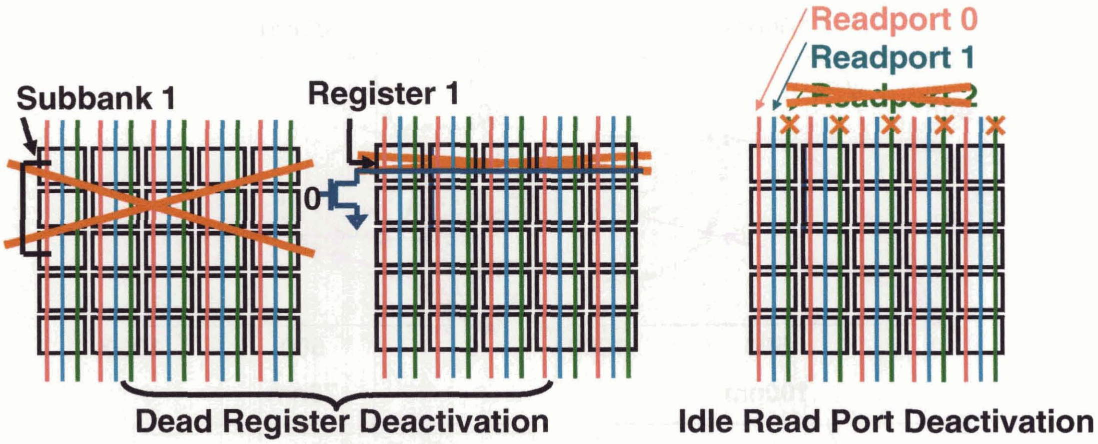


Figure 5-11: Dead register deactivation and idle port deactivation for register files.

Table 5.3: The leakage power of 32×32-b multiported register file subbank (optimistic leakage current was used).

Process Technology (nm)	180	130	100	70
Original (uW)	177.9	214.1	263.6	276.7
SV steady-state (uW)	2.0	2.4	3.0	3.1
LBB steady-state (uW)	2.0	2.4	3.0	3.1
NST steady-state (uW)	1.8	2.2	2.7	2.9

process technology.

We see that for the sleep vector technique, the break-even time is around 200 cycles at the 180 nm process, but shrinks to only 24 cycles in the 70 nm process. The sleep vector technique has high fixed transition energy costs, and so below the break-even time, the energy consumption is much higher than the original leakage energy.

For the leakage-biased bitline, the break-even time in the 180 nm process is only around 10 cycles. Moreover, the cumulative energy rises slowly from the initial deactivation time, and is not much larger than the original leakage before the break-even time. With technology scaling, the break-even time becomes less than a cycle and this technique can therefore give useful leakage energy savings even for a few cycles of dead time.

The NMOS sleep-transistor performs better than the leakage-biased bitlines in the coarser feature sizes, but suffers from a long transition time in the finer-pitch process technologies. The time taken to charge the virtual GND node leaves this scheme with higher cumulative leakage energy for small numbers of cycles in the 70 nm technology, though at large numbers of cycles the cumulative energy drops below that of the leakage-biased bitline scheme.

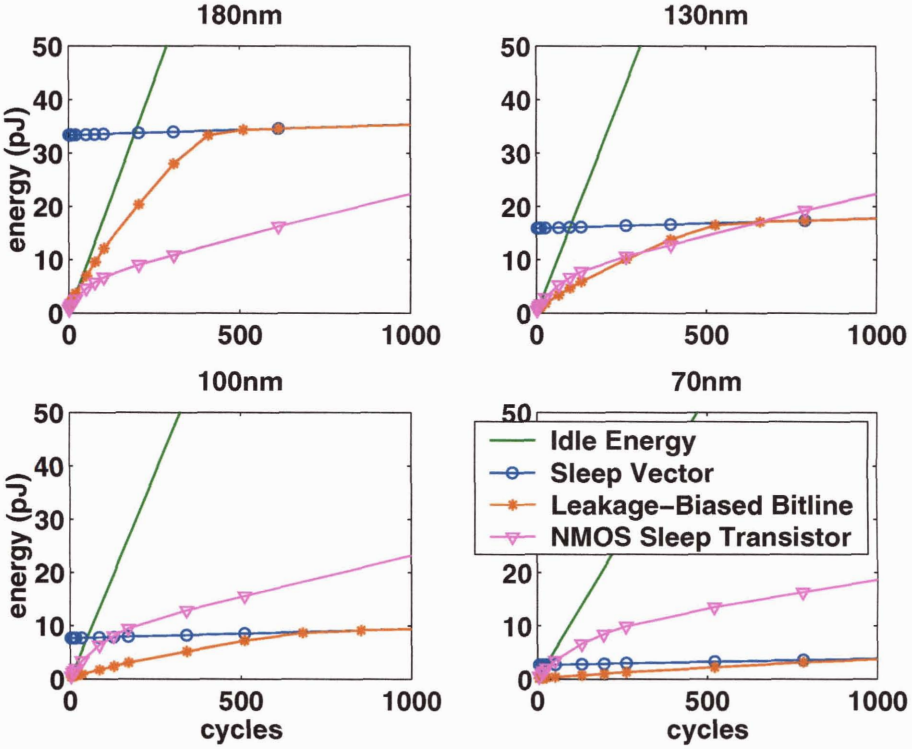


Figure 5-12: Sleep-time-dependent cumulative leakage energy of different register file dynamic leakage reduction techniques for different processes (optimistic leakage current was used).

5.3.4 Evaluation

In this section, we use detailed simulation of an out-of-order microprocessor to estimate the energy savings that can be achieved by using dynamic leakage reduction techniques on instruction cache subbanks and a multiported register file.

Simulation Methodology

We instrumented SimpleScalar 3.0b [BA97], an out-of-order, superscalar processor simulator, to track the activity of the instruction cache and the physical register file. We obtained results for a four-wide issue machine with the configuration shown in Table 5.4. We also simulated a four-wide issue machine with 128 register-update-units (RUUs) and performed cache simulation on an eight-wide issue machine with 256 RUUs, but the results were similar and thus we omit them for brevity. We used the SPECint95 benchmark suite and the benchmarks were run on their reference data sets until 100 million instructions had committed. In the figures that follow, black bars denote the optimistic assumptions of future leakage as described in subsection 5.3.1. White bars denote the

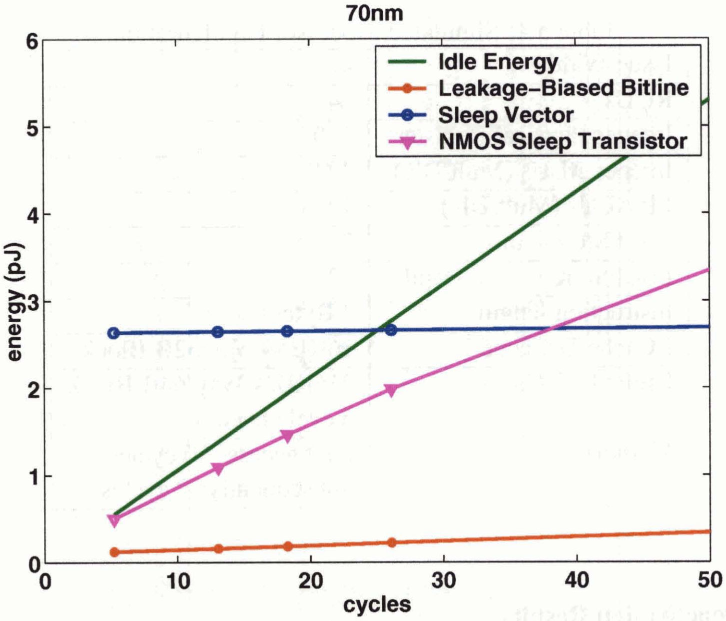


Figure 5-13: Expanded view of cumulative leakage energy in a 70 nm process technology (optimistic leakage current was used).

pessimistic view of the future (greater leakage).

Cache Subbank Deactivation Results

Figure 5-14 shows the energy savings achieved for the instruction cache subbank deactivation scheme. For the 180 nm generation, there is a net energy increase, but for all other process technologies there is a net energy savings. In the 70 nm generation, over 20% of total instruction cache energy is saved.

As discussed in subsection 5.3.2 there can be a performance penalty from the additional precharge latency if the subbank precharge cannot be overlapped with the rest of the bank address decode. We modeled the effect of lengthening the fetch pipeline by one cycle to allow for subbank precharge, which increases branch misprediction latency from 3 to 4 cycles. Our results show that this decreases IPC by around 2.5% on average across all benchmarks. We note that this estimate of performance impact is highly pessimistic, as the precharge latency is much less than one cycle and the extended pipeline could be used to support a much larger instruction cache.

Table 5.4: Simulated Processor Configuration

Issue Width	4
RUUs	64
Integer Physical Registers	100
Integer ALUs (Mult/Div)	4 (1)
FP ALUs (Mult/Div)	1 (1)
Load/Store Units	2
Load/Store Queue Depth	32
Instruction length	4 Bytes
I-Cache/D-Cache	16KB/4-Way/32B Block
Unified L2-Cache	256KB/4-Way/64B Block 6 cycle latency
Memory Latency	First access: 50 cycles. Subsequently: 2 cycles.

Dead Register Deactivation Results

To quantify energy saved by deactivating dead registers, we modified SimpleScalar to model a machine with a separate unified physical register file pool holding both committed architectural registers and renamed registers. We maintain a set of physical register tags which move between a free list and the register update units. We restricted our study to the integer register file. The number of physical registers is determined by the number of writable architected registers (fixed by the ISA at 33) plus the number of values that can be produced by in-flight instructions.

Figure 5-15 presents results for the dead register deactivation techniques. For reference, we present two variants of the NMOS sleep transistor (NST) technique. The two variants of NST are FIFO and LIFO free list policies. A FIFO policy (or circular queue) is the conventional free list policy, but a LIFO policy (stack) has the advantage of keeping some registers dead for very long times. Experiments with the 70 nm process reveal that LIFO gives an additional 2.4 to 10.0% savings over FIFO in terms of total register file energy saved. The figure also shows the benefits of LIFO increasing as feature size decreases.

We also show results for LBBs used in a subbanked register file, where a subbank's read ports are deactivated when all registers in the subbank are dead. The allocation policy is a stack of subbanks, where registers are allocated from a new bank only when the previous bank is empty. As shown in Figure 5-15, despite the increased granularity of deactivation, LBB is competitive with NST in terms of energy savings. Because the cumulative sleep energy of LBB circuits is less than that of NST circuits for the majority of sleep times encountered in practice, LBB outperforms NST

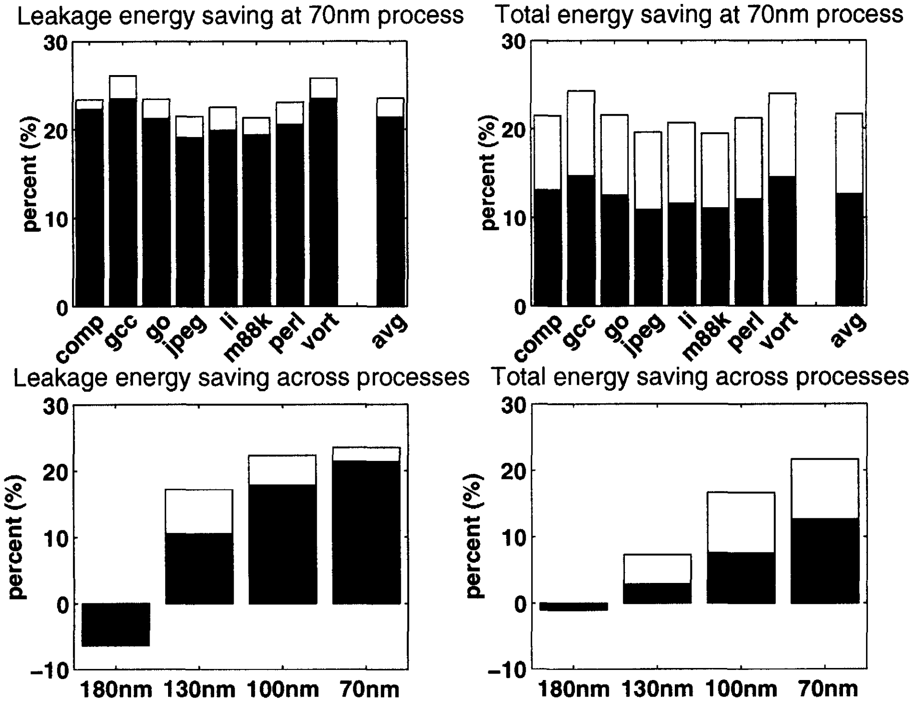


Figure 5-14: I-cache energy savings for subbank deactivation.

by 31.5% for the worst case leakage. In addition, LBB has no performance penalty, whereas NST has a 5% performance slowdown. The figure also shows that having fewer registers per bank (eight rather than sixteen) allows deactivation at a finer granularity which translates to greater savings.

Regfile Global Read Port Deactivation Results

Figure 5-16 shows the energy savings achieved by deactivating the read ports. As with cache subbank deactivation, there is a net energy increase for the 180 nm generation, but for the remaining process technologies, there is a net energy savings. In the 70 nm generation, nearly half of the leakage energy is removed, resulting in a total register file energy savings of over 20% with no performance penalty.

As the processor's issue width increases, the peak number of read ports increases. However, IPC does not scale linearly with issue width, so in general a greater percentage of read ports will be idle. Thus, we expect the energy savings to be greater for wider-issue processors.

The savings from global read port deactivation can be combined with those from dead subbank deactivation, giving greater total energy savings while still avoiding any performance penalty.

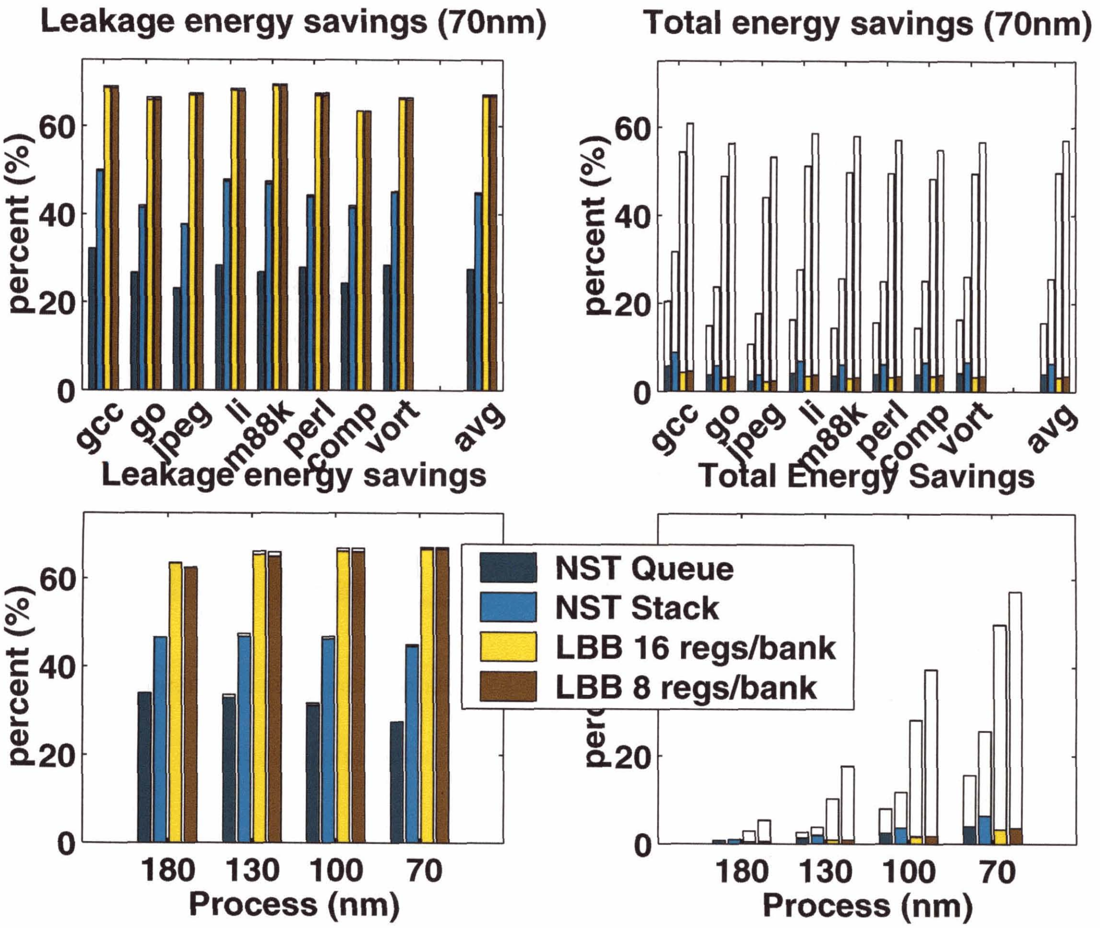


Figure 5-15: Register file energy savings by dead register deactivation.

5.3.5 Related Work

A few researchers have proposed fine-grain deactivation techniques that place portions of an active processor into low-leakage states. The dynamically-resized instruction cache [Pow00] uses a virtual-GND power gate to supply power to just enough RAM subbanks to hold the active working set of the current application. An adaptive hardware algorithm is used to determine an adequate cache capacity by monitoring miss rates as the active partition size is varied. This scheme is more complex than using leakage-biased bitlines, and is limited to a direct-mapped instruction cache, but reduces leakage further as both storage cell and access port leakage is cut off. Cache decay [KHM01] dynamically predicts which cache blocks are unlikely to be accessed in the near future, marks them invalid, then powers them down using a power gate. Both of these techniques have long deactivation times of thousands of cycles. Hamzaoglu et al. briefly describe a "precharge-as-

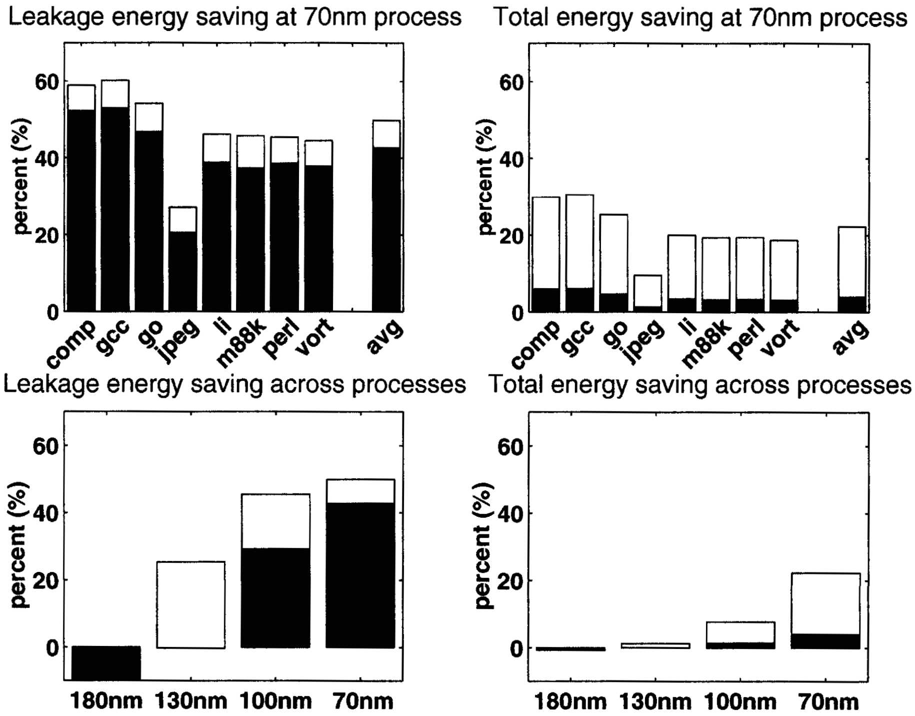


Figure 5-16: Register file energy savings by global read port deactivation.

needed" scheme [Ham00], apparently similar to our leakage-biased bitlines, but do not describe the dynamic transient effects of the leakage reduction or the use of this technique within a microprocessor.

5.4 Leakage-Biased Domino Logic for Critical Functional Units

In this section, we present *Leakage-Biased Domino* (LB-Domino) that uses sleep transistors only on non-critical paths and uses the leakage current itself to bias internal critical paths into a minimal leakage state. A 32-bit Han-Carlson domino adder circuit is used to compare the LB-Domino with the conventional single and dual V_T domino circuits.

Section 5.4.1 describes related work. Section 5.4.2 shows how LB-Domino works. Section 5.4.3 describes how we evaluate LB-Domino logic family. Section 5.4.4 shows the evaluation results with the Han-Carlson domino adder circuits.

5.4.1 Related Work

Domino logic is often used on critical paths, and several fine-grain dynamic leakage reduction techniques have been proposed to reduce leakage on idle domino blocks. Dual-V_T domino [KC00] requires additional input gating to force the internal nodes into a sleep state which reduces performance and increases switching energy. Also, as shown below, the high-V_T keepers increase switching energy once noise margin is equalized [KC00]. MHS-Domino [AAE00] modifies a clock-delayed keeper circuit to force internal dynamic nodes into a low leakage state. However, the internal node is pulled down through a PMOS leaving the possibility of an intermediate voltage on the dynamic node of the first stage of a domino chain if the data inputs are not high. This can cause short-circuit current in the static output inverter until the leakage through the input transistors finally pulls the dynamic node to ground.

5.4.2 Leakage-Biased Domino

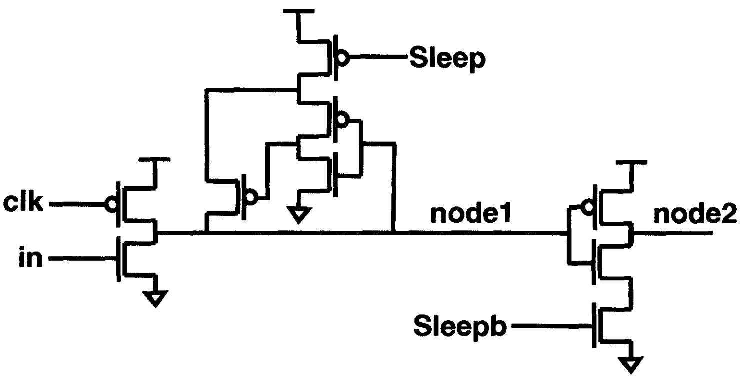


Figure 5-17: A leakage-biased domino buffer.

An LB-Domino buffer is shown in Figure 5-17. This example is a footless domino buffer without a clock transistor in the dynamic pull-down stack, but the LB technique can also be applied to footed domino stacks. Only two small sleep transistors are added to a conventional CMOS domino gate: a high-V_T PMOS in series with the keeper power supply and a high-V_T NMOS in series with the static output logic pulldown. When the sleep signal is de-asserted, the circuit operates as a conventional domino gate with minimal performance degradation because there are no additional series transistors in the critical evaluate path.

To place the circuit into sleep mode, the clock signal is left high after an evaluate cycle and the sleep signal is asserted (`sleep=1` and `sleepb=0`). If the data input was high, `node1` would have been discharged. If the data input was low, `node1` is high but the leakage through the NMOS dynamic pull-down stack will slowly discharge the node to ground (the precharge and keeper pull-up transistors are high-V_T devices with significantly lower leakage than the pull-down stack). The NMOS sleep transistor is added to prevent any short-circuit current in the static output logic while the dynamic node discharges to ground. The static output, `node2`, will rise as the static pull-up turns on. As the leakage current of one domino gate causes its output node to rise, this will cause the NMOS transistors in the pulldown stacks of the following domino gates to turn on, accelerating the discharge of their internal dynamic nodes. In this way, LB-Domino gates bias themselves into a low-leakage state where the internal dynamic nodes are discharged low and static nodes are charged high regardless of input vector state.

When the internal dynamic node is discharged, the main leakage is across the high-V_T PMOS precharge transistor which is turned off by the clock signal remaining high. The leakage path of the static output includes at least two series NMOS transistors, one of which is a high-V_T device. A conventional precharge cycle is used to move from sleep mode back to active mode.

Compared with MHS-Domino, LB-Domino has a simpler sleep mechanism that is compatible with, but does not require, a clock-delayed keeper. LB-Domino also avoids short-circuit current in the static output inverter of the first gate of a domino chain.

Table 5.5: Processes.

Process	180 nm	70 nm
High V_T (NMOS/PMOS)	0.46V/-0.45V	0.39V/-0.40V
Low V_T (NMOS/PMOS)	0.27V/-0.23V	0.15V/-0.18V
Vdd	1.8V	0.9V
Temperature	100 °C	100 °C

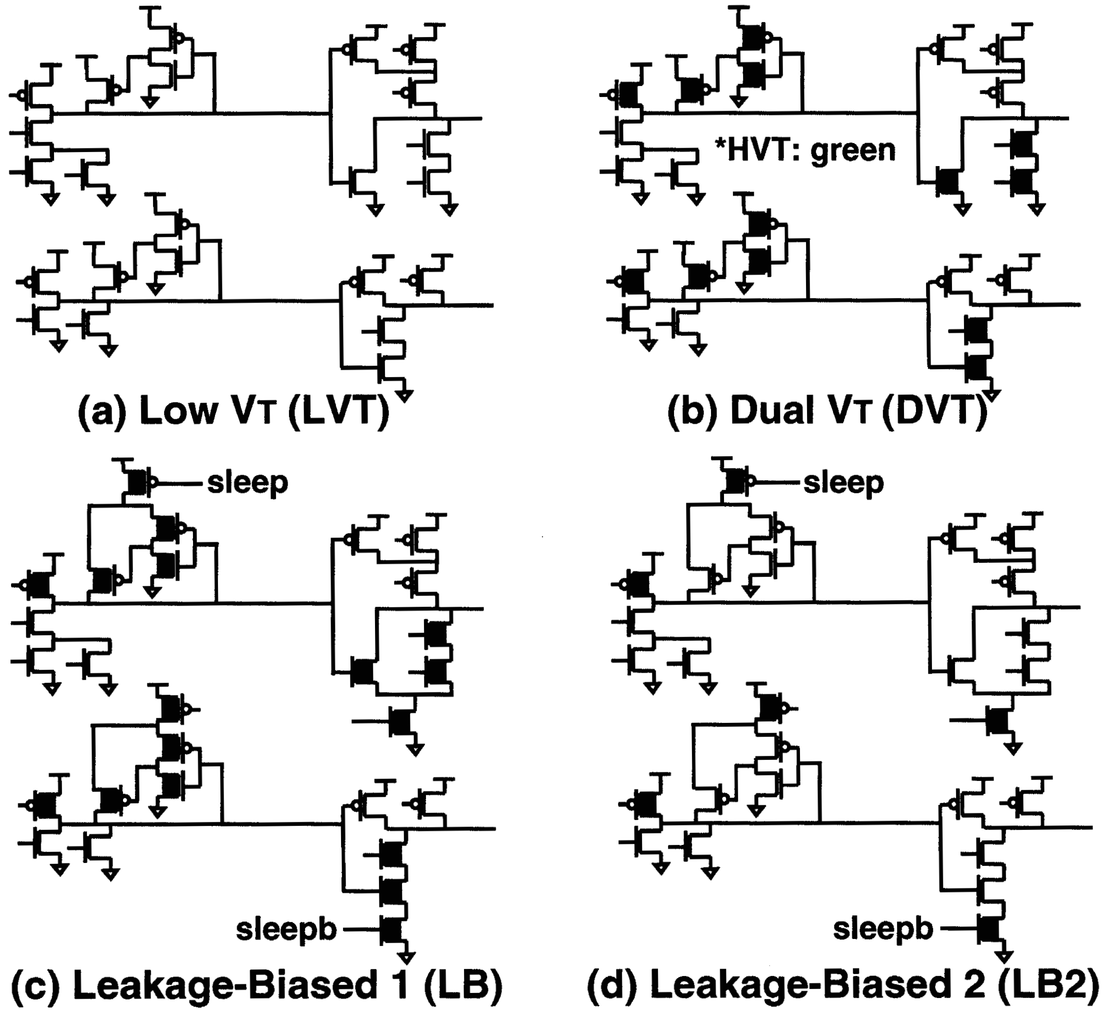


Figure 5-18: Cells for a 32-bit Han-Carlson adder.

5.4.3 Evaluation Methodology

The carry-generation circuit of a 32-bit Han-Carlson adder [Mat01] was used to evaluate LB-Domino. The carry-generation circuit is pure domino with six levels of alternating dynamic and static logic. The basic propagate-generate cells are shown in Figure 5-18. Four variants of the design were compared. The first uses only low-V_T transistors (LVT), while the second is a dual-V_T (DVT) design where only evaluation phase transistors are low-V_T. The third variant is an LB-Domino (LB) design based on the DVT design but with high-V_T sleep transistors added to the keeper feedback circuits and the static logic pull-downs. The fourth variant (LB2) is another LB-Domino design which only uses high-V_T for the precharge transistors and for the added sleep transistors.

For all four designs, the input and output noise margin of all dynamic circuits was set to 10% of

Table 5.6: Input vectors.

	a	b	ci
Vector 1	`0x00000000`	`0x00000000`	`0`
Vector 2	`0xffffffff`	`0x00000000`	`0`
Vector 3	`0xffffffff`	`0xffffffff`	`1`

the supply voltage and the precharge/evaluation delays were equalized to within 1% error through transistor sizing. The circuits were designed for an existing TSMC 180 nm process and a projected 70 nm process obtained from the BPTM project [Dev01] (Table 5.5). All simulations used HSPICE.

Since both switching energy and leakage power are dependent upon inputs, three different input vectors were considered (Table 5.6): `vec1` does not discharge any dynamic nodes, `vec3` discharges all dynamic nodes, and `vec2` discharges half and leaves half high.

5.4.4 Results

Figure 5-19 and Figure 5-20 show the delay and switching energy consumption for 180 nm and 70 nm processes respectively. The switching energy of DVT is greater than that of LVT because the high-V_T keeper transistors must be sized up to give equal noise margin and equal precharge delay. For the same reason, the switching energy of LB is greater than that of DVT. However, LB2 can meet the delay constraints with only a small increase in switching energy over the LVT design because it uses only a small number of high-V_T transistors.

Figure 5-21 and Figure 5-22 show the steady-state leakage power for 180 nm and 70 nm processes respectively. The leakage power of DVT is very sensitive to input values. For `vec3` with `clk`=1, all the high-V_T transistors in DVT are turned off and the lowest leakage power is obtained. On the other hand, for `vec1` with `clk`=1, the leakage is comparable to the LVT design. The sleep-state leakage power of LB and LB2 is independent of input vector because leakage currents bias the internal nodes into the lowest leakage state over some transition time. The LB schemes have worst-case sleep-state leakage currents that are around two decades lower than the LVT and DVT designs. For the 180 nm process, the LB scheme is preferred for circuits that spend enough time in sleep mode as it has lower leakage than LB2, but for circuits that are more active, LB2 has lower switching energy and reasonable steady-state leakage. For the 70 nm process, LB2 is always better than LB since it has lower switching energy and lower steady-state leakage than LB.

Figure 5-23 and Figure 5-24 show how energy consumption evolves over time when the circuit is put into a sleep state for 180 nm and 70 nm processes respectively. The energy curves show the

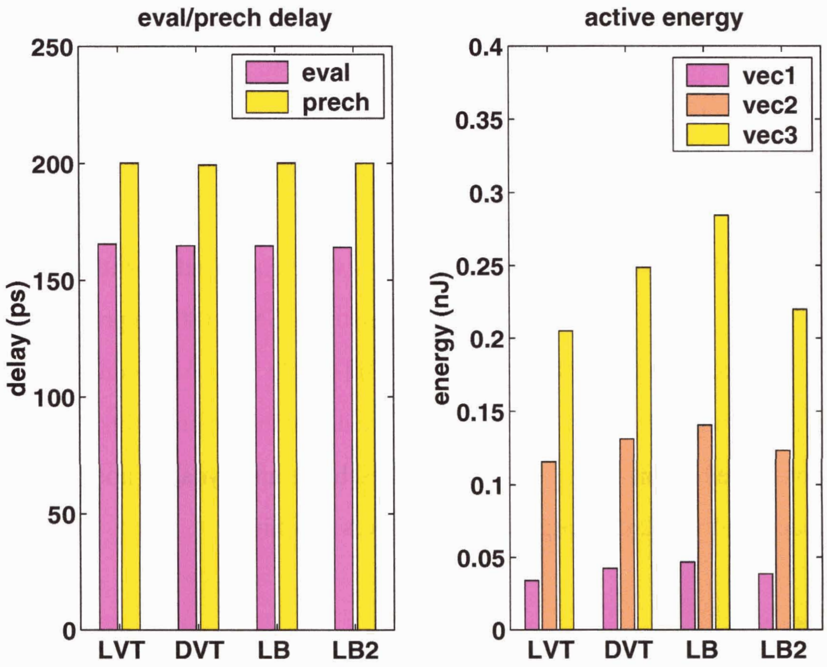


Figure 5-19: Delay and switching energy consumption : 180 nm process.

energy consumption when the circuit sleeps for the specified time, including the cost to transition the circuit into and out of the sleep state (e.g., the energy to switch the gates of the sleep transistors). For LVT and DVT schemes, the sleep energy is just linearly proportional to sleep time as leakage currents are constant. The sleep energy curve of LB shows a very different characteristic. There is a large jump in energy after a short sleep time (around 20 ns for 180 nm and around 1 ns for 70 nm). At this point, the static output of the first domino stage charges up to the threshold voltage, and causes the following stage to move rapidly to the low-leakage state. This process quickly ripples through the chain of domino gates. The energy stored in any precharged dynamic nodes is lost and must be restored during precharge when the circuit is next woken up, hence the steep rise in effective sleep energy dissipation. After this point, the energy curve has a very shallow slope due to the lowered leakage currents.

For short sleep times, the LB schemes require more total energy than simply idling an LVT or DVT circuit. But for longer sleep times the energy cost of discharging the internal dynamic nodes is amortized and the lower sleep leakage current yields lower overall energy. For LB and LB2, the cross-over point is around 2 μs in the 180 nm process for the worst case (vec1). However, the cross-over point in the 70 nm process is under 10 ns because switching energy scales down faster than leakage power.

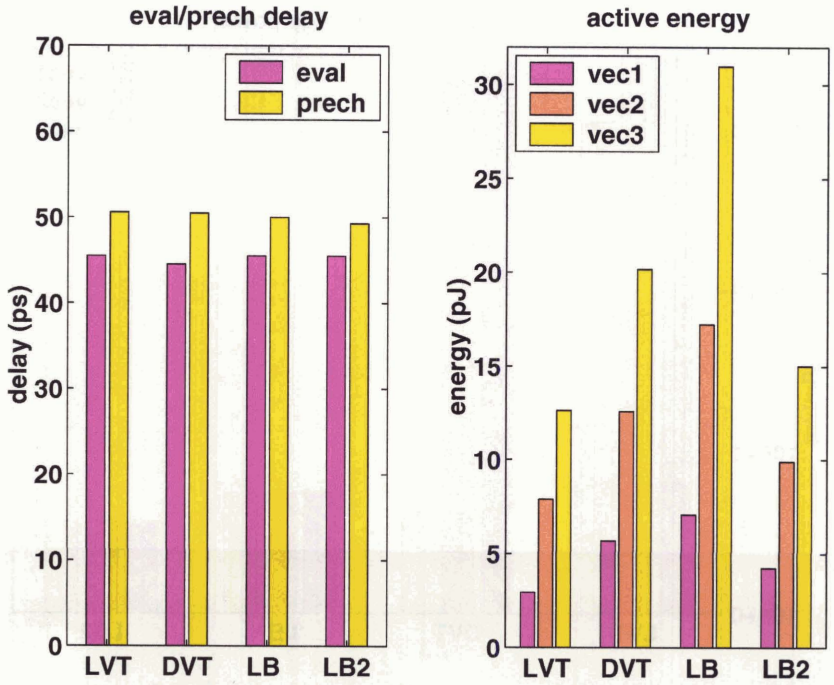


Figure 5-20: Delay and switching energy consumption : 70 nm process.

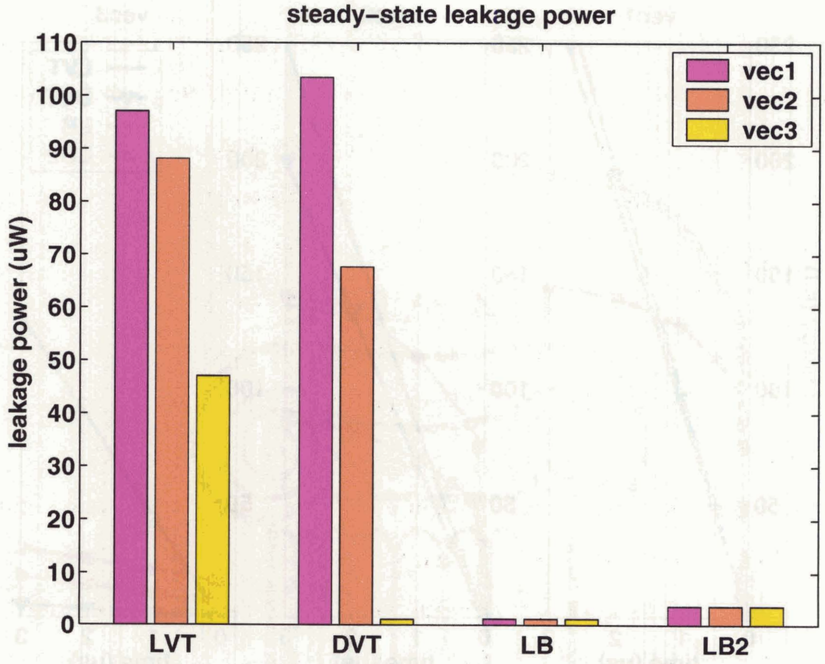


Figure 5-21: Steady-state leakage power : 180 nm process. `clk` is high for all and `sleep` is asserted for LB and LB2.

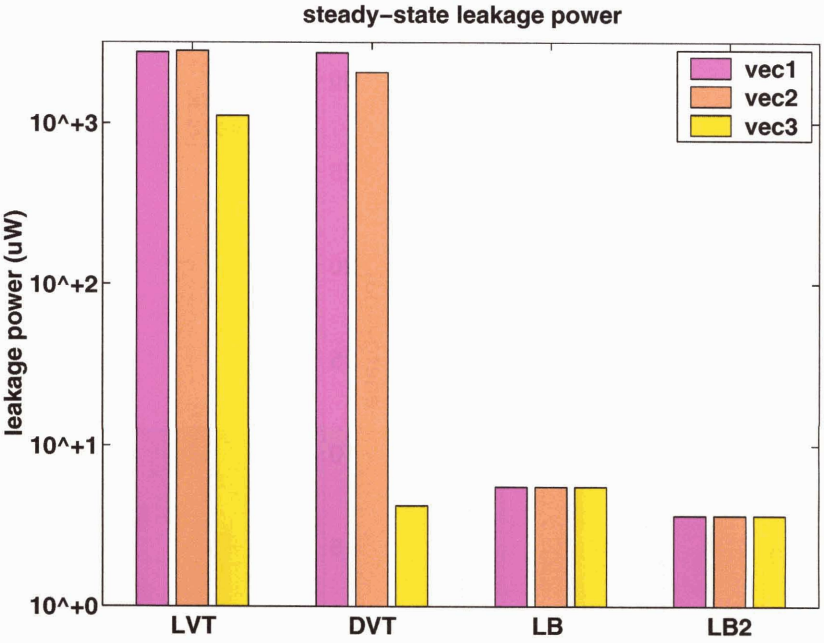


Figure 5-22: Steady-state leakage power : 70 nm process. `clk` is high for all and `sleep` is asserted for LB and LB2. Note that y-axis is log-scale.

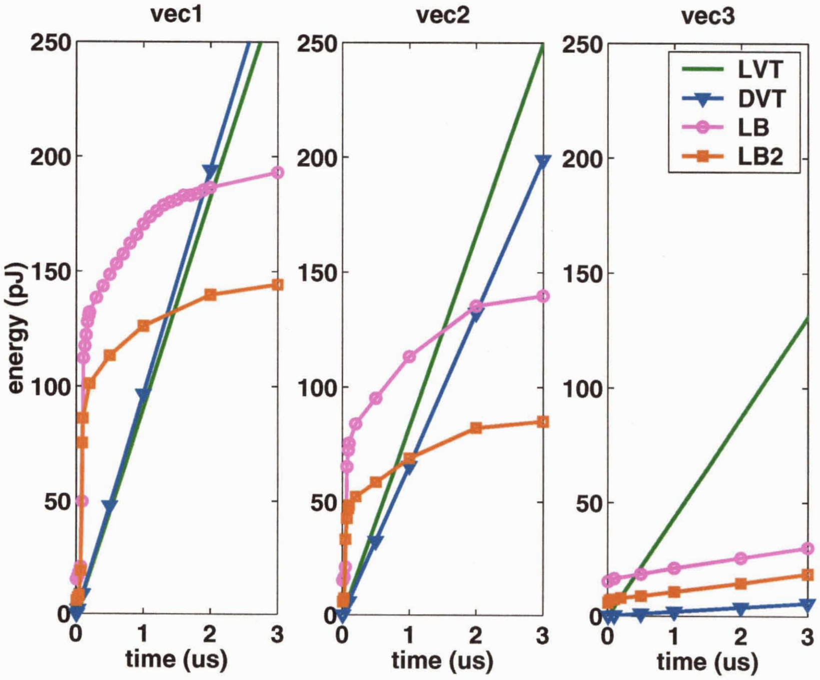


Figure 5-23: Cumulative sleep energy : 180 nm process.

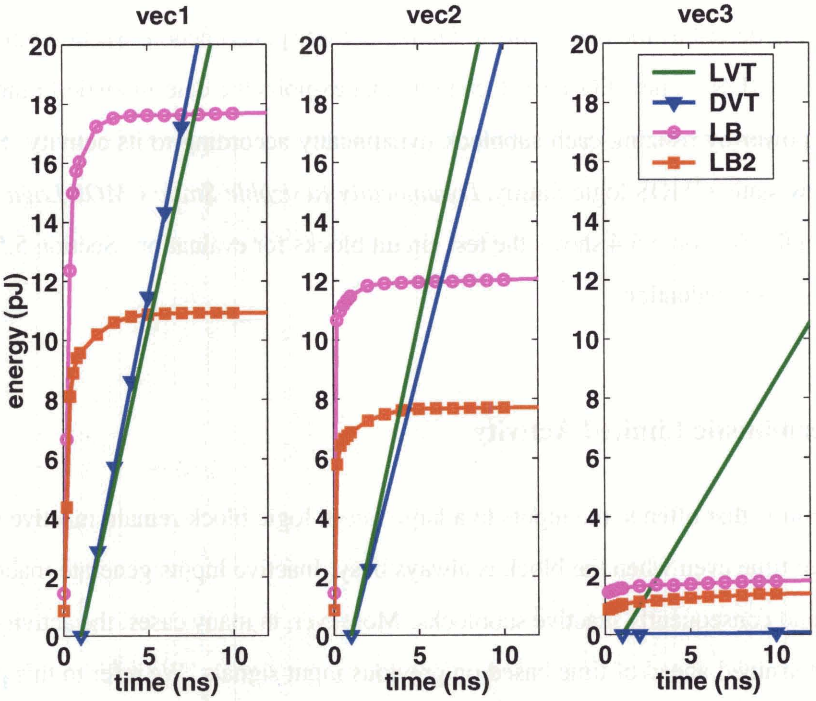


Figure 5-24: Cumulative sleep energy : 70 nm process.

5.5 Dynamically Resizable Static CMOS Logic (DRCMOS)

For some critical microarchitectures such as SRAM arrays and ALUs, fine-grain power gating or leakage biasing [HBHA02, HA02] can be used to save leakage power, provided the unit's activity pattern includes sufficiently long idle times to repay the often large energy cost of switching in and out of a low-leakage mode. Unfortunately, many critical logic blocks within a high-performance digital system are busy every cycle and so are not amenable to the block-level deactivation. For such blocks, we introduce *Dynamically Resizable CMOS* (DRCMOS) logic that dynamically downsizes transistors on idle paths while maintaining speed along active critical paths.

Section 5.5.1 describes the *deterministic limited activity* phenomenon. Section 5.5.2 introduces *Dynamic resizing* (DR), a new FG-DLR technique that exploits the deterministic limited activity to save leakage power by resizing each subblock dynamically according to its activity. Section 5.5.3 proposes a new static CMOS logic family, *Dynamically Resizable Static CMOS Logic (DRCMOS)* to implement DR. Section 5.5.4 shows the test circuit blocks for evaluation. Section 5.5.5 discusses the results from our evaluation.

5.5.1 Deterministic Limited Activity

The observation is that often some inputs to a large fan-in logic block remain inactive for a significant amount of time even when the block is always busy. Inactive inputs generate inactive intermediate signals and consequently inactive subblocks. Moreover, in many cases, the activity pattern can be exactly determined ahead of time based on previous input signals. We refer to this phenomenon as *Deterministic Limited Activity*. For example, many logic blocks attached to queues or arrays, which are ubiquitous in modern superscalar processors, exhibit deterministic limited activity. Figure 5-25 shows an example of deterministic limited activity. The top six inputs are determined to be inactive for a while. Busy subblocks constitute critical paths. Only a subset of the subblocks are on the critical path and the rest remain idle for a while.

There are two key concerns when exploiting deterministic limited activity to save leakage power. First, idle subblocks should maintain their output values to preserve the functionality of the entire block. Second, the critical path within a block changes dynamically; and therefore, soon-to-be active subblocks must be woken from a low-leakage and probably low-speed state sufficiently early to avoid a delay penalty.

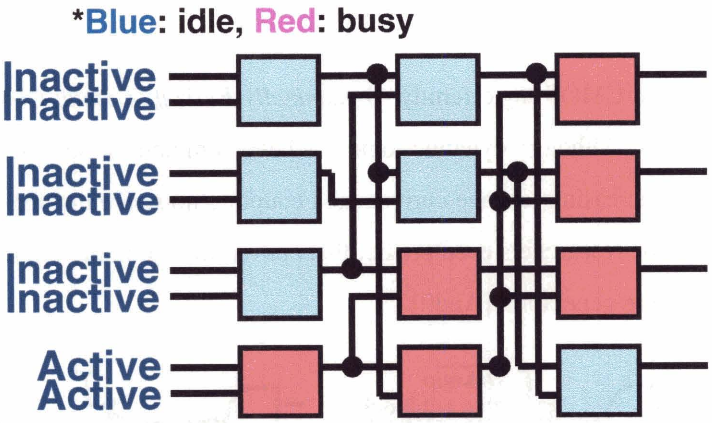


Figure 5-25: Deterministic limited activity.

5.5.2 Dynamic Resizing

Dynamic resizing (DR) is a new FG-DLR technique that exploits deterministic limited activity to save power by resizing each subblock dynamically according to its activity. DR determines the idle subblocks for the first stage of logic based on input patterns. Subsequent stages of logic will be idle if all subblocks feeding their inputs will be idle. When a subblock is determined to be idle, DR downsizes the transistors in the subblock to save leakage power. To maintain speed along critical paths, DR upsizes transistors in soon-to-be active subblocks before critical paths change. Figure 5-26 shows an example of dynamic resizing. It is determined that the top six inputs will be inactive for subsequent cycles. The idle subblocks are resized small to save leakage, while the active subblocks are resized large to maintain speed.

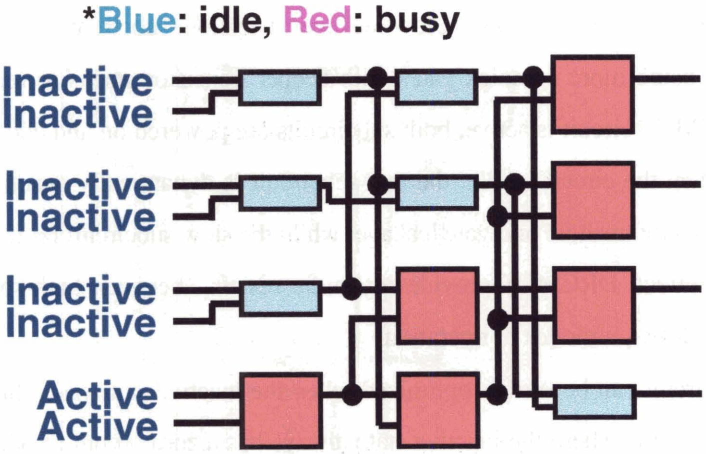


Figure 5-26: Dynamic resizing.

5.5.3 Dynamically Resizable Static CMOS

We propose a new static CMOS logic family, *Dynamically Resizable Static CMOS Logic (DRCMOS)*, to implement DR. Although dynamic logic has been common in the critical paths of high-speed digital systems, increasing leakage currents and coupling noise are making static logic more attractive [And03]. Some researchers even predict that conventional domino circuits could cease to be useful below the 70 nm generation [And01].

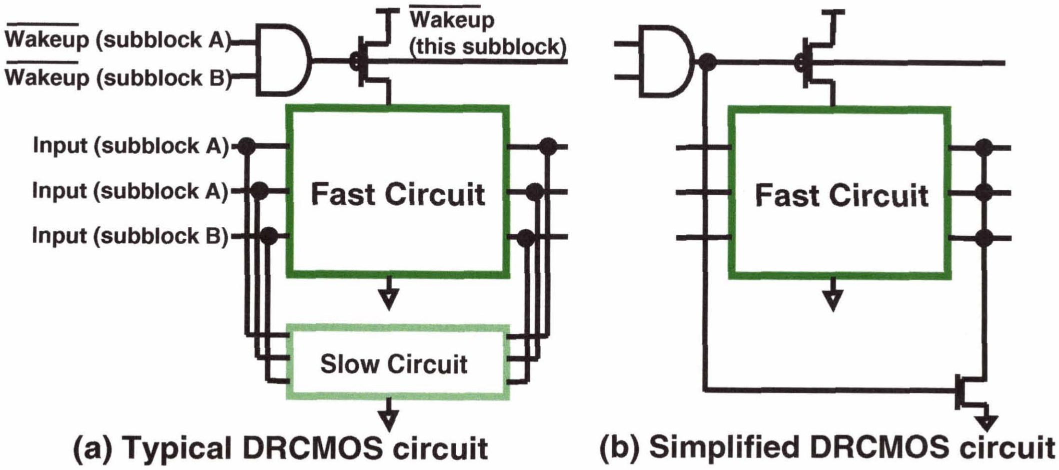

$\overline{\text{Wakeup}}$ (subblock A)
$\overline{\text{Wakeup}}$ (subblock B)
$\overline{\text{Wakeup}}$ (this subblock)
Input (subblock A)
Input (subblock A)
Input (subblock B)
Fast Circuit
Slow Circuit
(a) Typical DRCMOS circuit
Fast Circuit
(b) Simplified DRCMOS circuit

Figure 5-27: Dynamically resizable static CMOS logic (DRCMOS).

A DRCMOS circuit consists of a fast subcircuit and a slow subcircuit connected in parallel between inputs and outputs (Figure 5-27(a)). The fast subcircuit is built with large, low V_T transistors that are high speed but leaky. The slow subcircuit has the same functionality but is built using smaller, high V_T transistors to give low leakage. The slow subcircuit might also implement the logic function using more complex gates with deeper transistor stacks to reduce leakage further. When a DRCMOS circuit is active, both subcircuits are powered on and cooperate to generate output results. When the circuit is idle, the fast subcircuit is dynamically deactivated using sleep transistors to cut its subthreshold and gate leakage, while the slow subcircuit remains on to preserve output values. In effect, DRCMOS provides dynamic resizing between high-speed/high-leakage and low-speed/low-leakage modes of operation.

The slow subcircuit can be further optimized when the inactive state has a limited set of input patterns. A trivial case is where the inactive state always has a zero output in which case the slow subcircuit degenerates to a single NMOS pulldown transistor as shown in Figure 5-27(b).

DRCMOS requires additional control logic to generate the wakeup signals. The wakeup signal must be generated early enough (typically at least one clock cycle earlier) to ensure each subcircuit will be activated in time to propagate a critical transition at full speed. The wakeup signals for the first stage subblocks are generated from external logic associated with the inputs. Subsequent stage subblocks generate their wakeup signals by OR-ing the wakeup signals from their input subblocks.

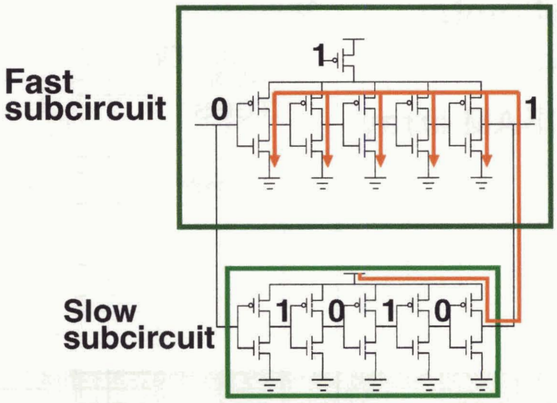


Figure 5-28: Sneak leakage path problem.

A sneak leakage path can occur when the fast subcircuit is deactivated as shown in Figure 5-28. To prevent this, the fast subcircuit must use both NMOS and PMOS sleep transistors, or its output stage must have separate power gating.

5.5.4 Evaluation Methodology

To evaluate DRCMOS logic, two key blocks of a modern superscalar processor are chosen: a static 64-entry register free list slice (Figure 5-29), and a static 64-entry pick-two arbiter (Figure 5-30).

The register free list slice is a FIFO containing a list of currently unassigned 9-bit physical register numbers (Figure 5-29). The FIFO is implemented as a small circular RAM with two pointers giving the head and tail of the list. The whole free list would use multiple slices to allow parallel access to multiple distinct registers. The free list uses a static mux tree to provide the read port. The read mux trees exhibit deterministic limited activity and so DR can be applied. Only one input to the mux tree, the input from the entry pointed to by the read pointer, is active. Thus, many muxes in the tree remain idle for multiple cycles. Also, the location of the next active input to the tree is predetermined to be circularly sequential from the current input owing to the circular FIFO structure. In the DRCMOS register free list design, only the muxes in the first stage of the mux tree where the read pointer is currently pointing or will point next cycle are upsized. In the second stage, any mux which has any upsized children is also upsized. The root mux is always on the critical path and is not resized.

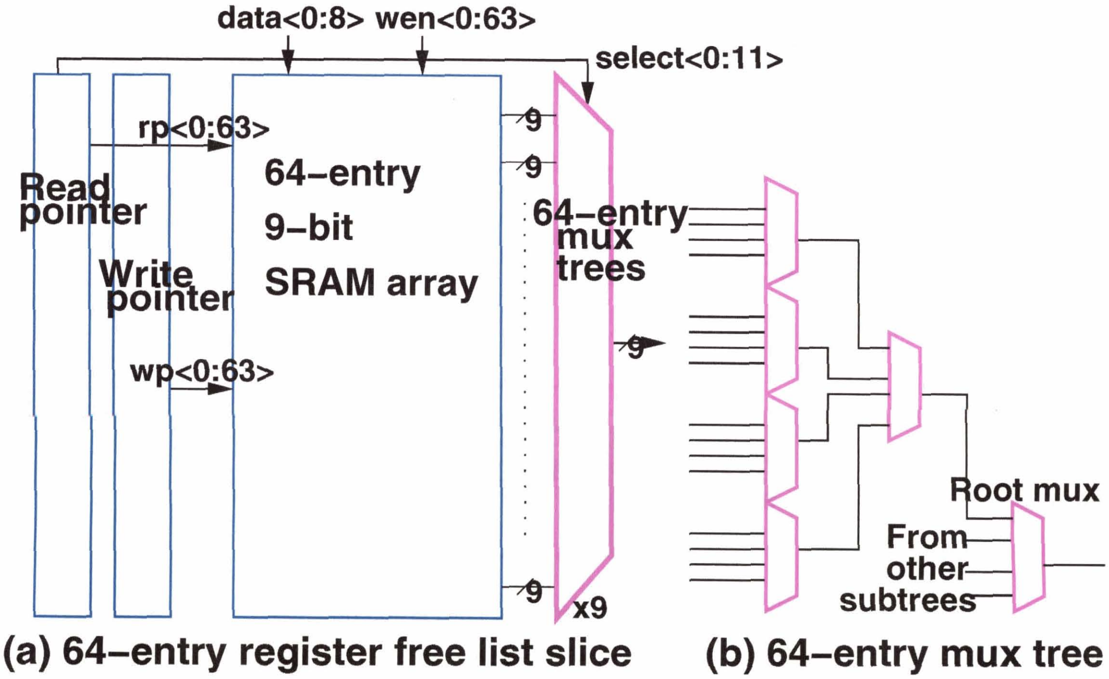
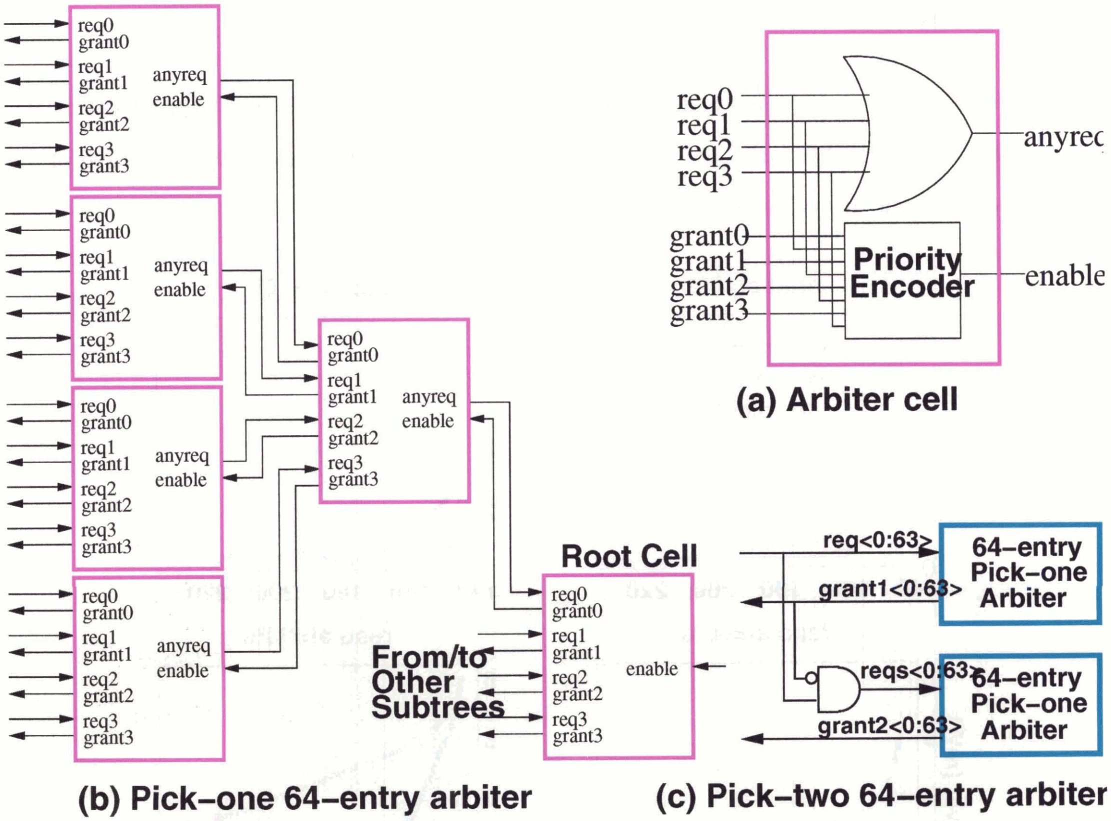


Figure 5-29: A static 64-entry register free list slice.

The arbiter selects instructions for execution from the pool of ready instructions in the issue window. The inputs to the arbiter are request signals from ready instructions and the outputs are issue grant signals. Our arbiter selects the two oldest ready instructions (Figure 5-30). The issue window contains full and empty regions delimited by the read and write pointers (Figure 5-31). To simplify control, we treat the empty region as inactive inputs to the arbiter and the entire full area as active, even though some entries in the full area will be inactive as they are not ready for issue. The arbiter shows deterministic limited activity as the borders between the full and empty areas move sequentially as instructions are fetched and retired. When idle, the arbiter cell has zero outputs and the DRCMOS slow subcircuit can be simplified as shown in Figure 5-27(b).

In addition to comparing DRCMOS to baseline designs, architectural pipelining is also evaluated as another way to lower active and leakage power for the free list and arbiter. A pipelined structure has relaxed performance demands and so can use slower and lower-power transistors. However, pipelining adds clock and switching power. Also, the increased area and number of transistors can lead to more leakage power. Although pipelining lowers local cycle time, it adds global latency and so can impact the overall CPI (clocks-per-instruction) performance of the synchronous digital system by adding more hazards. When pipelining the register free list, timing elements are inserted



(a) Arbiter cell

(b) Pick-one 64-entry arbiter

(c) Pick-two 64-entry arbiter

Figure 5-30: A static 64-entry pick-two arbiter.

between the mux select control logic (read pointer) and the mux tree, that is, `select<0:11>` in Figure 5-29(a) is pipelined. For the pipelined arbiter, timing elements are inserted between the AND gates and the second pick-one arbiter, that is, `reqs<0:63>` in Figure 5-30 is pipelined. All flip-flops have appropriate clock gating.

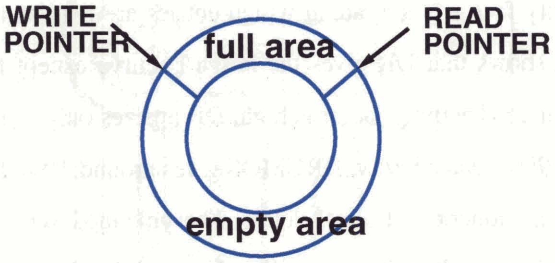


Figure 5-31: Logical structure of issue window.

5.5.5 Results

The circuits were designed for a projected 70 nm process obtained from the BPTM project [Dev01]. All simulations used HSPICE and the results for the DR scheme include the power overhead of the wakeup control logic and of switching the sleep transistors.

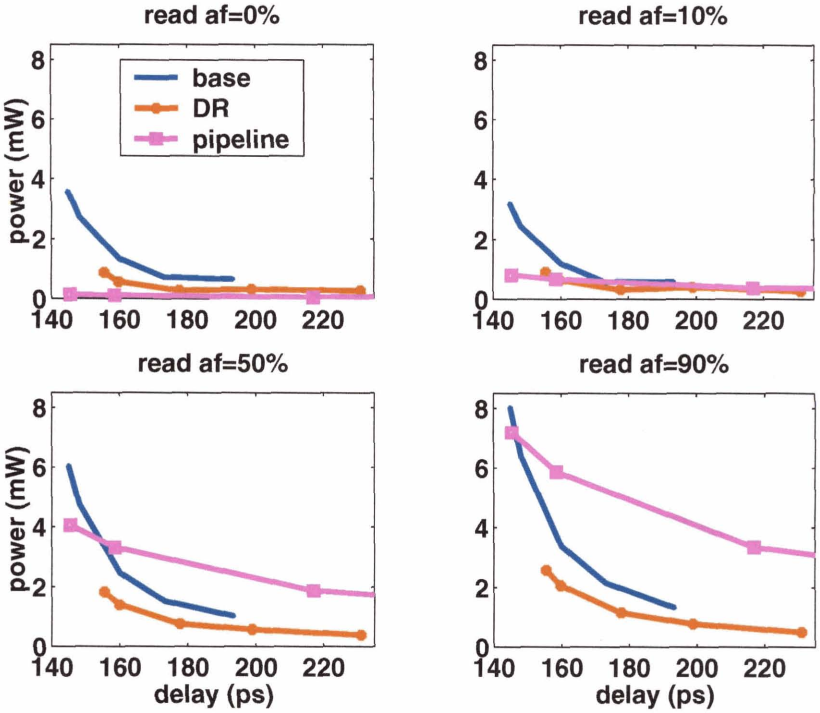


Figure 5-32: PD curves for register free list's mux tree using supply voltage scaling.

Figure 5-32 shows power-delay (PD) curves for the baseline, DRCMOS, and pipelined versions of the static 64-entry register free list under supply voltage scaling. Supply voltage was varied from 0.7 V to 1.35 V. Temperature was set at 100 °C. The SRAM array was assumed to contain 50% zero bits. The read activity factor is the rate at which entries are read and was varied from 0% to 90%. The graph clearly shows that DR gives the best PD curve except for the 0% read activity factor case. Even when the read activity factor is high, DR upsizes only a small subset of the muxes keeping others small. At 90% read activity, DRCMOS gives around 10% delay reduction for equal power or 1.5× total power reduction at equal delay. The pipelined version suffers from flip-flop switching power overhead and only performs well at low activity factors where the advantage of small low-leakage transistors becomes apparent.

Figure 5-33 shows alternate PD curves of the baseline, DRCMOS, and pipelined free lists when

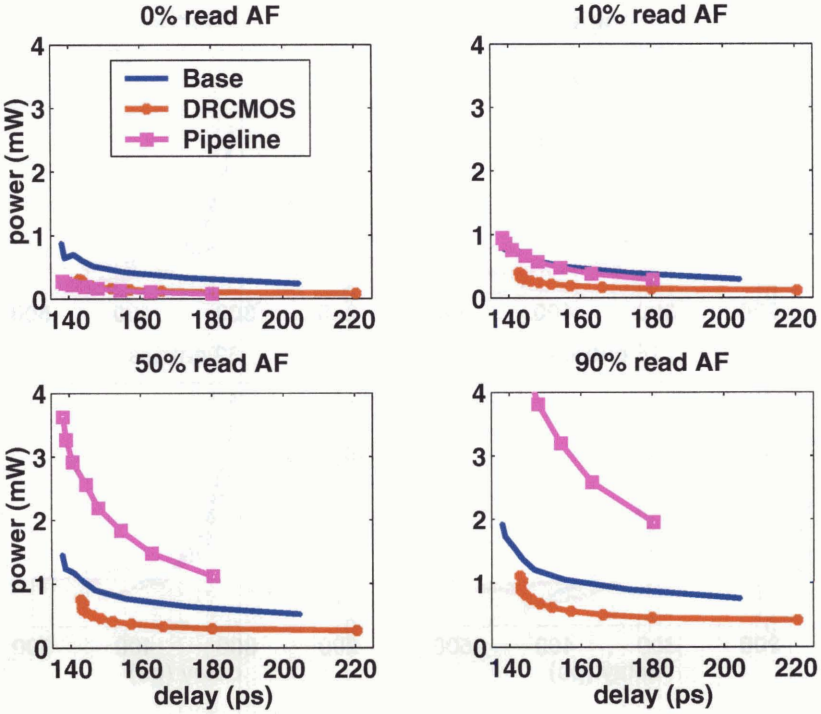


Figure 5-33: PD curves for register free list's mux tree using transistor sizing.

conventional design-time transistor sizing is used to vary delay with supply voltage fixed at 0.9 V. The graph clearly shows that DRCMOS gives the best PD curve. At 90% activity factor, DRCMOS gives at least 10% delay reduction for equal power or around 50% total power reduction at equal delay.

Figure 5-34 shows PD curves of the alternate designs of the static 64-entry pick-two arbiter using supply voltage scaling, and varying the number of the entries in full area from 0 to 32. DR performs better when the instruction window is emptier as more arbiter cells remain small reducing total leakage power. Instruction window occupancies vary greatly during program execution depending on application program characteristics. With only 6 entries in the full area, around 10% delay reduction for equal power or 2× total power reduction for equal delay can be achieved with DR. On the other hand, when half the entries in the issue window are ready, the DRCMOS curve is close to the baseline at shorter delays. The pipelined version supports lower delays at low power, but at looser delay constraints, the flip-flop power overhead overwhelms the power saving from the use of small and high V_T transistors.

Figure 5-35 shows PD curves of the arbiter with transistor sizing at a 0.9 V supply. When there are 16 ready entries, around 3% delay reduction for equal power or 50% total power reduction

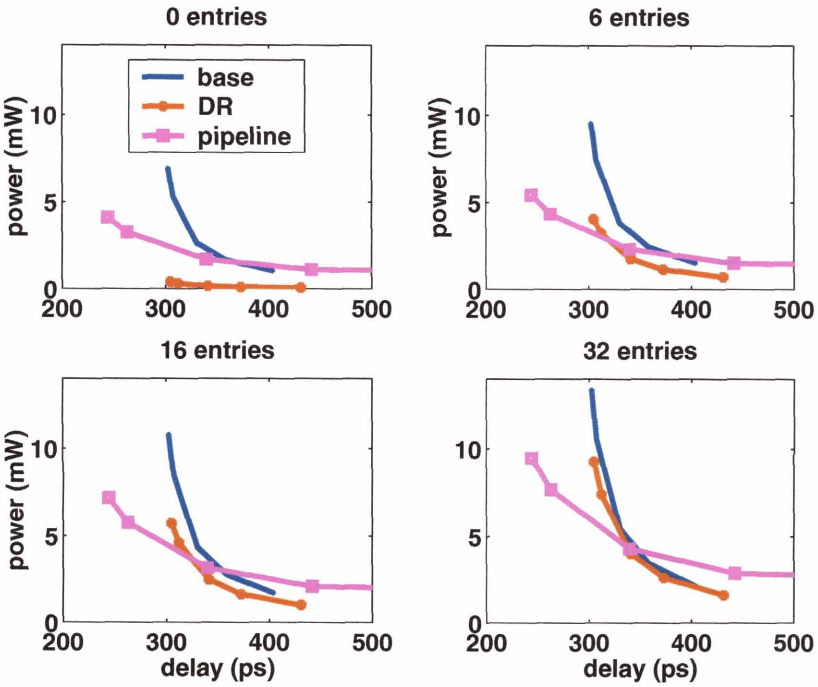


Figure 5-34: PD curves for arbiter using supply voltage scaling.

for equal delay can be achieved by using DRCMOS logic, compared to the baseline. Pipelining always gives a better PD curve than baseline as transistors can be downsized to take advantage of the pipelining.

Although pipelining works well for the arbiter, it introduces an architectural hazard that prevents dependent instructions from issuing in consecutive cycles, which causes a significant global system performance penalty [Pal97] and so would usually be avoided in a high performance design.

5.6 Summary

Most leakage current is dissipated on critical paths, especially after slower, low-leakage transistors are used on non-critical paths. To reduce leakage energy further without impacting performance, it is desirable to dynamically deactivate the fast transistors on the critical path. We have shown that fine-grain leakage reduction techniques, whereby a small piece of a digital system is placed in a low-leakage state for a short amount of time, can yield significant energy savings in future process technologies. To attain savings, the circuit-level leakage reduction technique must have low transition energy and rapid wakeup times.

We present leakage-biased bitlines, a circuit technique that has these properties. To exploit a

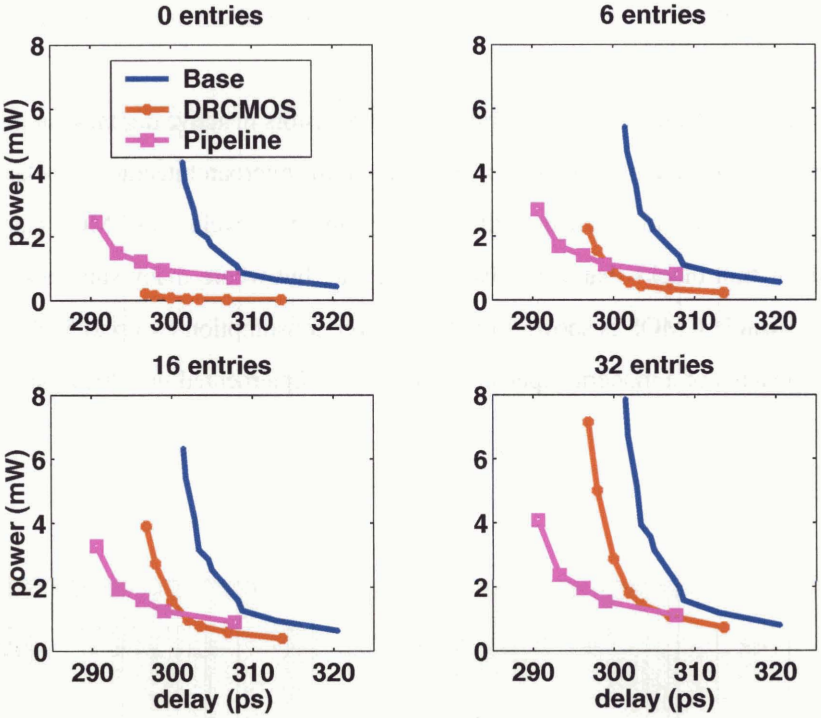


Figure 5-35: PD curves for arbiter using transistor sizing.

dynamic leakage reduction technique, the microarchitecture must be designed to force blocks to be idle for multiple cycles and preferably to give early notice when the blocks are to be reawakened. We have presented three applications of leakage-biased bitlines that apply these principles and have shown how they enable leakage current reductions in the context of a wide superscalar processor. SRAM read path deactivation saves over 22% of leakage energy and nearly 24% of total I-cache energy when using a 70 nm process. Dynamically deactivating idle registers reduces register file leakage energy by up to 67.1% and total register file energy by 57.1%. Dynamically deactivating read ports within a multiported register file saves 42.7-49.8% of leakage energy and 3.9-22.3% of total energy depending on the prediction of the future process.

We apply leakage biasing to domino logic and introduce LB-Domino logic. When used to dynamically deactivate critical path circuits in projected 70 nm process technologies, LB-Domino provides two decades reduction in steady-state leakage current compared with low-V_T or dual-V_T domino at equal delay and noise margin. LB-Domino has sub-cycle deactivation and reactivation latencies, and because leakage currents are used to bias the circuit, LB-Domino also has low transition energy overheads. Using LB-Domino to place circuits into a sleep state can yield net energy savings even for sleep times of under 10 ns. This makes dynamic fine-grain circuit deactivation practical, where small pieces of an active system can be powered-down for short periods of time to save leakage energy.

DRCMOS reduces leakage power of critical path transistors in active digital systems. DRCMOS exploits the regular predictable patterns of activity within microarchitectural blocks to downsize transistors that are known to be off the critical path in the next cycle. DRCMOS can be used at a very fine-grain within blocks that are active every cycle, but where many subblocks will be idle. Dynamically resizable CMOS is shown to reduce power consumption by up to 50% at equal delay in critical components of a modern superscalar processor implemented in a 70 nm technology.

Chapter 6

Pipelining Logic Datapaths

Pipelining is a representative energy-delay trading tool. We describe various aspects of energy-delay tradeoffs for digital system optimization using pipelining as an example throughout two chapters. Optimal pipelining is sought in two contexts: pipelining logic datapaths for lower energy (this chapter) and pipelining global wires and structuring them into a wire network (Chapter 7).

In this chapter, we show how power-optimal pipelining of datapaths varies for different operating regimes in deep submicron technology [1]. We examine the tradeoffs among pipeline depth, supply voltage, threshold voltage, and total power using circuit-level simulations and analytical models. We also explore the effects of the activity factor and of clock gating.

6.1 Power-Optimal Pipelining

Pipelining exploits parallelism among the instructions in a sequential instruction stream [HP96] (Section 4.1.1). Because of ample instruction-level parallelism in computation models and application software, it has been and is likely to continue to be one of the most effective and popular architectural innovations for many high-performance VLSI digital systems such as processors, DSPs, FPGAs, and even memory systems.

Pipelining has been extensively used in logic datapaths. It reduces the number of logic gates between stages by inserting more stage latches or flip-flops and overlapping more instructions. The reduced amount of computation for a clock cycle is exploited to increase clock frequency and improve throughput. Clock frequency increase through pipelining has been a crucial technique for the success of high-performance microprocessors in recent years. Deep pipelining (or super-pipelining)

[1] The work in this chapter was a joint work with Krste Asanović and was previously published in [HA04b].

has made modern high-performance digital systems include enormous number of timing elements such as latches and flip-flops. Their power consumption contributes to a significant portion of total power consumption now.

The time slack obtained from pipelining can also be used to reduce power consumption by employing other energy-delay tradeoffs in a different direction, such as lowering supply voltage, at a fixed clock frequency (Figure 6-1). This technique can be effective for digital systems with fixed throughput requirements and highly parallel computations.

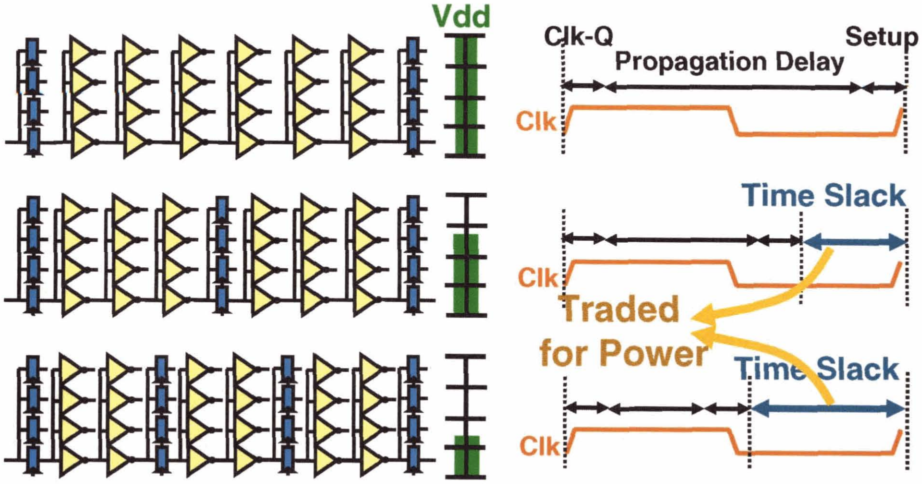


Figure 6-1: Pipelining datapath for lower energy.

A parallel architecture could also be used to trade the excess performance for lower energy, but pipelining has the advantage of less area penalty and lower leakage energy. Power reductions from pipelining are eventually limited by the power overhead of the pipeline latches or flip-flops required for additional pipe stages, with a resulting power-optimal pipelining.

6.2 Related Work

The trend towards deeper pipelines in high-performance digital systems is clearly seen in the evolution of the Intel x86 family, with a factor of 7 reduction having taken place in logic depth per stage over the last decade [Hri02]. This reduction in logic depth has combined with improvements in transistor speed resulting from technology scaling to yield an even larger increase in clock frequency. An increase in the number of pipeline stages for an operation also increases latency in clock cycles, which in turn increases the number of pipeline stalls experienced by dependent operations. The resulting reduction in instructions completed per cycle (IPC) reduces the performance advantage from greater clock frequency and has a greater impact on codes with lower instruction-level parallelism (ILP).

Digital system architects have explored this tradeoff between increased clock frequency and reduced IPC to determine performance-optimal pipelining depth. Early work by Kunkel and Smith [KS86] considered pipelining in vector supercomputers and found that 8–10 ECL gate levels was performance-optimal for scalar code, and as little as 4 gate levels was optimal for more parallel vector code. Recently, several authors have investigated the performance-optimal pipeline depth for superscalar microprocessors [HP02, Hri02, SC02], with a consensus in the range of 8–11 FO4 delays for SPEC integer codes and around 6 FO4 delays for SPEC floating-point codes, which generally have higher ILP. These performance-optimal numbers ignore power issues as well as the design and verification complexities that would inevitably accompany such high-frequency designs (roughly twice the clock rate of existing systems [SC02]).

Several authors have extended superscalar performance models with power models that include the power overhead of additional pipeline latches [Sri02, HP03]. Srinivasan et al. [Sri02] found that power-performance optimal logic depth is about 18 FO4 for SPEC benchmarks and around 24–28 FO4 for TPC-C, a commercial application. Hartstein and Puzak [HP03] found 22.5 FO4 is the power-performance optimum, according to their power-performance metric. They also found that clock gating pushes the optimum back to deeper pipelines [HP03], which agrees with our results.

Previous work focuses on processor performance, where limited instruction parallelism reduces the benefits of deep pipelines, and these studies limit power optimization to the selection of the correct number of additional pipeline stages. Other types of digital systems, including digital signal processors, network processors, and graphics engines, have much greater levels of parallelism and often have fixed throughput requirements. For these systems, pipelining can be used together with supply and threshold voltage scaling to reduce total energy consumption without variations in a clock rate. The use of pipelining for power reduction was proposed by Chandrakasan et al.[Cha92] but without an attempt to determine a power-optimal pipelining strategy.

6.3 Methodology

Our main target is a logic-dominant pipeline stage. We make several simplifying assumptions in our analysis. Since we are interested in fixed-throughput designs for highly parallel computations, we do not include any performance loss from an increased frequency of pipeline stalls as pipeline depths increase. Global wire delay does not scale as fast as gate delay since feature size is reduced, and some modern microprocessors have so-called drive stages which include only wires and repeaters [Hin01b]. We leave wire-dominant pipeline stages for next chapter (Chapter 7) but note that the wire RC delay in logic-dominant pipeline stages becomes relatively less important as supply voltage is scaled down in a fixed technology, because wire resistance remains constant while effective transistor resistance increases. We do not include local wire capacitance due to the absence of detailed circuit layouts, but note that wire cap can be an important component of total load in deep submicron technology even for a logic-dominant stage.

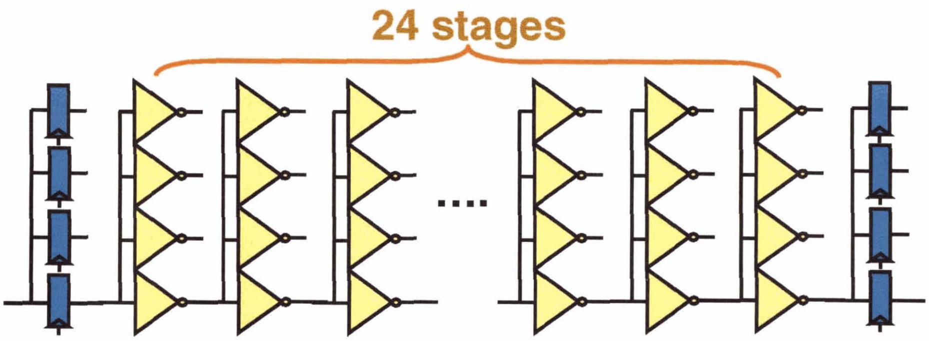


Figure 6-2: Baseline pipeline stage model. Input and clock buffers are not shown.

Figure 6-2 shows the baseline pipeline stage model assumed in our study. To model a well-designed path in a circuit, we use a simple static inverter chain with each inverter driving four copies of itself to yield a FO4 load. We use 24 FO4 delays as a baseline clock period, representing a current high-performance microprocessor circuit (the high-frequency Pentium-4 has a 20 FO4 cycle time while the core execution units have an even smaller cycle time, 10 FO4 [SC02]. Most other designs have shallower pipelines).

Even though different circuit styles and logic gates might lead to different power-optimal pipelining results, we assume that our FO4 inverter chain model is fairly representative and that insights gathered from our simulation results can be applied to other cases. Flip-flops were chosen as the timing elements rather than latches due to their simplicity of usage, and the PowerPC transmission-gate flip-flop was chosen because it is a popular choice due to its robustness and energy-efficiency [HKA01]. While the transistor sizes of inverters and flip-flops were fixed, the sizes of clock buffers were varied to ensure the appropriate clock rise and fall times when varying the depth of pipelining.

We used the BPTM 70 nm transistor models with different threshold voltages [Dev01] and

HSPICE for circuit simulation. In this analysis, clock frequency was fixed at 2 GHz and temperature was constant at 100 ^{o}C. We only considered subthreshold leakage; although gate leakage might become significant at some point in these technology generations. It is also possible that new gate dielectrics will make gate leakage insignificant again.

6.4 Pipelining and Supply Voltage

We begin by showing the effect of pipeline depth on supply voltage. With delay fixed, supply voltage scales down as pipelining deepens because the logic amount per pipeline stage decreases. Synchronous circuit delay is approximately given by

$$delay \quad \propto \quad (N+k) \times \frac{V_{dd}}{(V_{dd}-V_T)^{\alpha}} \tag{6.1}$$

where N is the logic depth per pipeline stage in term of FO4 delay (or the number of FO4 inverters per pipeline stage), k is the timing element delay normalized by FO4 delay, α is a velocity saturation effect factor, V_{dd} and V_T are supply and threshold voltages respectively. Assuming α is 2 (actual value of α in deep submicron technology is close to 1.5 due to the short-channel effect),

$$N+k \quad \propto \quad V_{dd} - 2V_T + \frac{V_T^2}{V_{dd}} \tag{6.2}$$

Now assuming $\frac{V_T^2}{V_{dd}}$ is close to zero, we get a simple linear equation between V_{dd} and N, where a_0 is a constant:

$$V_{dd} \quad = \quad a_0 N + a_1 \tag{6.3}$$

$$(\frac{a_1}{a_0} \quad = \quad k + \frac{2}{a_0}V_T) \tag{6.4}$$

Figure 6-3 shows the simulated supply voltages when varying the number of FO4 inverters per stage for different threshold voltages. In the figure, LVT, MVT, and HVT represent low, medium, and high threshold voltages respectively and their values are shown in Table 6.1. Low threshold voltage results in low supply voltage for the same delay. The least square method was used to calculate a_0 and a_1 and the obtained values are shown in Table 6.1. We can see that a_0 is proportional to V_T as well as a_1 (our simplified equations fail to explain the effect).

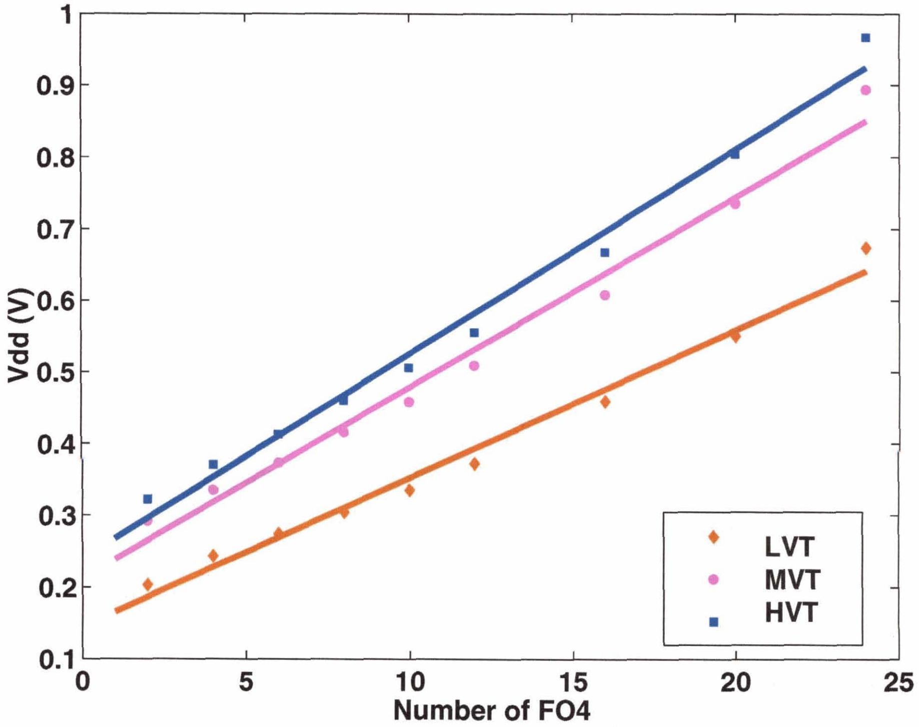


Figure 6-3: Supply voltage scaling shown as voltage required to achieve 2 GHz with given number of FO4 logic levels per pipeline stage.

V_T (V) NMOS(PMOS)	name	a_0	a_1
0.17 (-0.20)	LVT	0.0207	0.1450
0.19 (-0.22)	MVT	0.0266	0.2119
0.21 (-0.24)	HVT	0.0286	0.2389

Table 6.1: Threshold voltages and supply voltage scaling coefficients.

6.5 Pipelining Power Components

In this section, we explore the impact of pipelining on the components of total power consumption when delay is fixed. We use the supply voltage scaling results shown above in Section 6.4 and investigate switching, leakage, and idle components of power consumption assuming no clock-gating mechanism.

6.5.1 Pipelining and Switching Power

Switching power remains the dominant component of total power consumption when the activity factor is high, even in leaky deep submicron technology. Switching power is the power consumed while charging and discharging load capacitances. The load capacitances include transistor parasitic and wire capacitances. Because we assume our pipeline stage is logic-dominant, wire capacitances are not included in our simulation.

The switching power of a pipelined logic stage can be divided between power due to logic gates and power due to timing elements, and can be modeled as:

$$\begin{aligned} P_{switching} &= (b_0 + \frac{b_1}{N})V_{dd}^2 & (6.5) \\ &= b_0 a_0^2 (1 + \frac{b_1}{b_0}\frac{1}{N})(N + \frac{a_1}{a_0})^2 & (6.6) \end{aligned}$$

The overhead includes clock and switching power of timing elements and it is inversely proportional to, N, the number of logic gates per stage. We assume that the number of latches increases linearly with the number of pipeline stages. All the switching power components are proportional to V_{dd}^2. The ratio of switching power coefficients $\frac{b_1}{b_0}$ is approximately the ratio of the parasitic capacitances of one FO4 inverter and one timing element.

When N is much greater than $\frac{a_1}{a_0}$ and $\frac{b_1}{b_0}$, $P_{switching}$ becomes quadratic to N as shown in Eq. 6.7, which represents the dominance of logic gate switching power.

$$P_{switching} \approx b_0 a_0^2 N^2 \qquad (6.7)$$

On the other hand, if N gets much smaller than $\frac{a_1}{a_0}$ and $\frac{b_1}{b_0}$, $P_{switching}$ becomes inversely proportional to N, as shown in Eq. 6.8, which represents the dominance of timing element switching power:

$$P_{switching} \approx b_1 a_1^2 \frac{1}{N} \qquad (6.8)$$

Note that the $(N + \frac{a_1}{a_0})^2$ term in Eq. 6.6 makes relative $P_{switching}$ scale down slowly when $\frac{a_1}{a_0}$ is large. Since a higher V_T process has a higher $\frac{a_1}{a_0}$, as shown in Table 6.1, a higher V_T process gets less switching power reduction from pipelining.

The optimal logic depth N^* is given by:

$$N^* = \frac{1}{4}(\sqrt{\frac{b_1}{b_0}^2 + 8\frac{a_1}{a_0}\frac{b_1}{b_0}} - \frac{b_1}{b_0}) \qquad (6.9)$$

The equation indicates that the capacitance ratio of a timing element and an FO4 inverter, $\frac{b_1}{b_0}$, is positively correlated to N^*. That is, larger timing element parasitics lead to less deep pipelining. However, N^* is not as sensitive to $\frac{b_1}{b_0}$ as it is to $\frac{a_1}{a_0}$. This means that V_T and timing element delay k affect N^* and correspondingly optimal power reduction more significantly (Eq. 6.4).

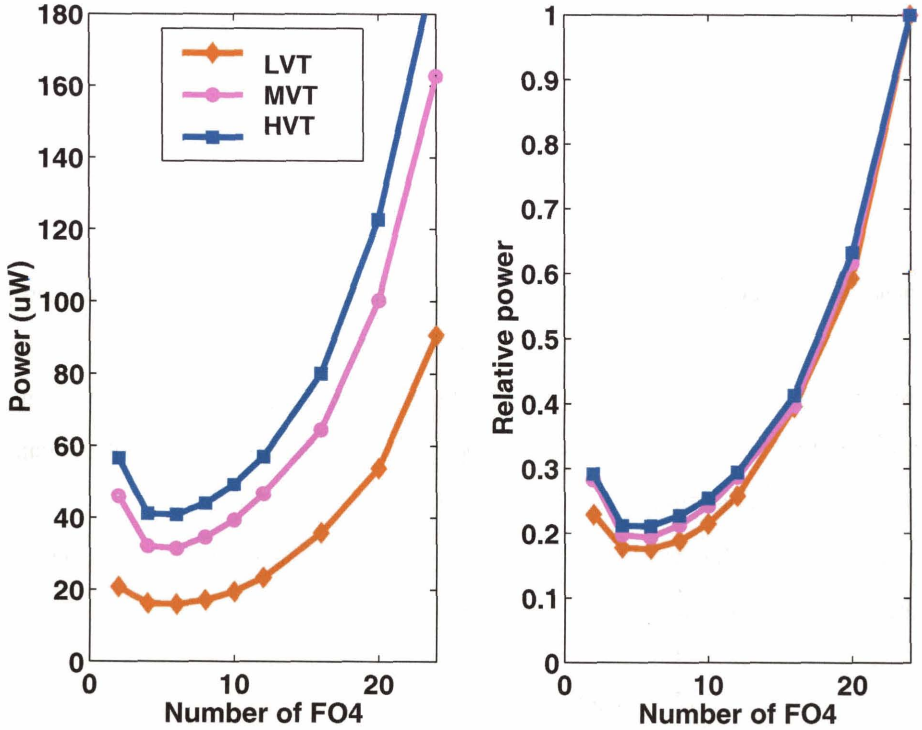


Figure 6-4: Switching power scaling.

Figure 6-4 shows the switching power obtained through HSPICE simulation of our model pipeline stage. Optimal logic depth N^* was six and optimal power reduction was from 79 to 82% compared to the baseline of $N = 24$. The graphs show that lower threshold voltages gives slightly lower optimal logic depth and also slightly greater switching power reduction. The variances are quite small since the variance of $\frac{a_1}{a_0}$ is small.

6.5.2 Pipelining and Leakage Power

The leakage power of our pipelined circuit can be given by the following equations:

$$P_{leak} = (c_0 + \frac{c_1}{N})V_{dd}\, e^{\frac{-V_T+\eta V_{dd}}{n v_T}} \tag{6.10}$$

$$= c_0 a_0\, e^{\frac{-V_T}{n v_T}} (1 + \frac{c_1}{c_0}\frac{1}{N})(N + \frac{a_1}{a_0})\, e^{\frac{\eta a_0}{n v_T}(N+\frac{a_1}{a_0})} \tag{6.11}$$

where nv_T is a constant representing leakage current slope, η is a Drain-Induced-Barrier-Lowering (DIBL) coefficient, and $\frac{c_1}{c_0}$ is the ratio of leakage power of one FO4 inverter versus one timing element. As in the switching power model, the leakage power in a stage can be divided into logic gate leakage and timing element leakage, with timing element leakage inversely proportional to N.

When N is much greater than $\frac{a_1}{a_0}$ and $\frac{c_1}{c_0}$, P_{leak} becomes proportional to the product of N and the exponential term $e^{\frac{\eta a_0}{nv_T}N}$:

$$P_{leak} \approx c_0 a_0 e^{\frac{-V_T}{nv_T}} N e^{\frac{\eta a_0}{nv_T}N} \tag{6.12}$$

The exponential term, $e^{\frac{\eta a_0}{nv_T}(N+\frac{a_1}{a_0})}$ represents the dependence of leakage current on the drain voltage (from DIBL). In modern deep submicron technology, for an appropriate supply voltage range, this term is larger than $O(1)$ but smaller than $O(N)$, therefore leakage power is reduced in a super-linear fashion as N decreases, though less than the quadratic reduction for switching power. Also, it is noted that the exponential term scales down faster as N decreases when a_0 is larger. A higher V_T process has higher a_0 as shown in Table 6.1, and so it is expected that higher V_T process will see greater leakage power reduction from pipelining, which is opposite to the switching power case, but higher V_T processes have less absolute leakage to begin with.

On the other hand, if N becomes much smaller than $\frac{a_1}{a_0}$ and $\frac{c_1}{c_0}$, P_{leak} becomes inversely proportional to N just as in the $P_{switching}$ case:

$$P_{leak} \approx c_1 a_1 e^{\frac{-V_T}{nv_T}} \frac{1}{N} e^{\frac{\eta a_1}{nv_T}} \tag{6.13}$$

Figure 6-5 shows the simulated leakage power while varying the number of logic gates per stage. Optimal logic depth N^* was around six and optimal power reduction was around 70–75%. The graphs show that lower threshold voltages gives less leakage power reduction and slightly greater optimal logic depth.

6.5.3 Idle Power without Clock-Gating

Clock-gating is a popular switching power reduction technique which inactivates the clock signal to timing elements within an inactive block when a circuit block is idle. But clock gating is not always possible due to the increase control complexity or the insufficient setup time of the clock

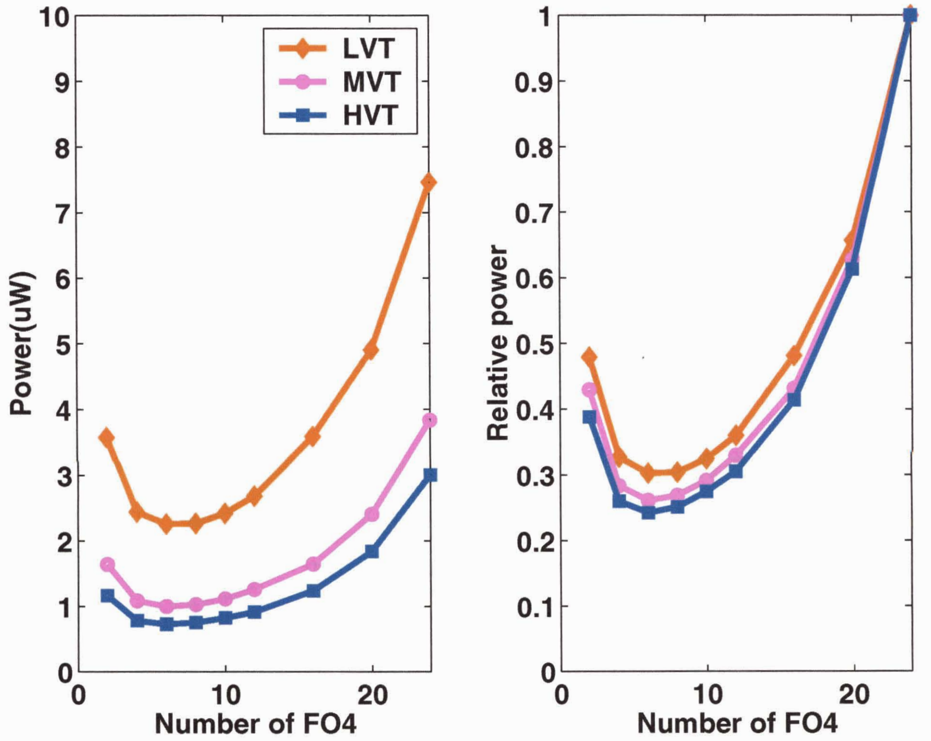


Figure 6-5: Leakage power versus logic depth per stage.

enable signal. This section focuses on the impact of pipelining on an idle pipeline stage without clock-gating. The following section discusses the effects of clock-gating.

The following equations model idle power with no clock-gating mechanism as simply the sum of the switching power of the timing elements and the total leakage power. Because of the exponential dependency of leakage current on V_T as represented in the $e^{\frac{-V_T}{nv_T}}$ term, P_{idle} approximately follows the switching power of the timing elements when V_T is high and follows the total leakage power when V_T is low.

$$\begin{aligned} P_{idle} &= \frac{b_1}{N}V_{dd}^2 + (c_0 + \frac{c_1}{N})V_{dd}e^{\frac{-V_T+\eta V_{dd}}{nv_T}} && (6.14) \\ P_{idle} &= b_1 a_0^2 \frac{1}{N}(N + \frac{a_1}{a_0})^2 + && (6.15) \\ & \quad c_0 a_0 e^{\frac{-V_T}{nv_T}}(1 + \frac{c_1}{c_0}\frac{1}{N})(N + \frac{a_1}{a_0})e^{\frac{\eta_0 a_0}{nv_T}(N+\frac{a_1}{a_0})} && (6.16) \end{aligned}$$

When N is much greater than $\frac{a_1}{a_0}$ and $\frac{c_1}{c_0}$, P_{idle} becomes proportional to the product of N and the exponential function of N or just proportional to N, depending on V_T:

$$P_{idle}(HighV_T) \approx b_1 a_0^2 N \tag{6.17}$$

$$P_{idle}(LowV_T) \approx c_0 a_0 e^{\frac{-V_T}{nv_T}} N e^{\frac{\eta a_0}{nv_T} N} \tag{6.18}$$

When V_T is high, P_{idle} shows a linear reduction as N decreases, which is slower than a quadratic reduction as in switching power or a super-linear reduction as in leakage power. Thus, we can expect that idle power reduction from pipelining is lower than those of switching and leakage power reduction when V_T is high.

On the other hand, if N is much smaller than $\frac{a_1}{a_0}$ and $\frac{c_1}{c_0}$, P_{idle} becomes inversely proportional to N:

$$P_{idle}(HighV_T) \approx b_1 a_1^2 \frac{1}{N} \tag{6.19}$$

$$P_{idle}(LowV_T) \approx c_1 a_1 e^{\frac{-V_T}{nv_T}} \frac{1}{N} e^{\frac{\eta a_1}{nv_T}} \tag{6.20}$$

Figure 6-5 shows the simulated idle power without clock-gating, varying the number of FO4 inverters per pipeline stage. Optimal logic depth N^* was eight, which is greater than the optimal logic depths for switching and leakage power. Also, optimal power reduction was smaller (50 to 70%) compared to the switching and leakage power cases. For idle stages, the overhead of timing elements is more significant compared to active stages. The graphs show that lower threshold voltages gives more idle power reduction and slightly lower optimal logic depth.

6.6 Combined Results

In this section, we combine the results for the individual power components to calculate optimal logic depths and optimal power reduction for different operating regimes including threshold voltage, activity factor, and presence of clock-gating. Power-optimal pipelining varies depending on the activity factor and V_T because these change the proportion of switching power and leakage power (or idle power with no clock-gating mechanism), and each impacts pipelining power differently as seen in Section 6.5. This section is divided into two parts: the first part details the case with a clock-gating mechanism for pipeline stages and the second part considers the case without clock-gating.

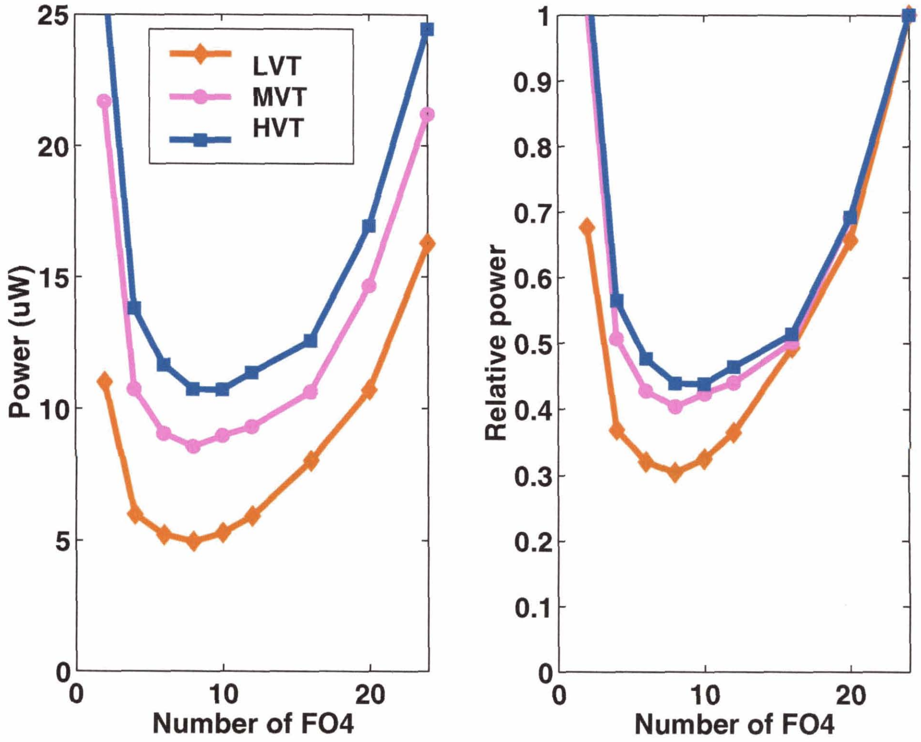


Figure 6-6: Idle power scaling.

6.6.1 Case 1: Clock-Gating Present

Figure 6-7 shows the simulated total power when a clock-gating mechanism is present for different activity factors. With a low activity factor, total power curves follow leakage power curves and high V_T leads to more power reduction by pipelining. As the activity factor increases, total power curves follow switching power curves and high V_T leads to less power reduction by pipelining.

Figure 6-8 shows the simulated optimal total power reduction when a clock-gating mechanism is present. With zero activity factor, optimal power reduction compared to a 24 FO4 design vary from 70 to 75% depending on V_T. Since switching power reduction from pipelining is less dependent upon V_T, optimal power reduction reaches around 80% regardless of V_T as activity factor increases.

Because both switching power and leakage power are minimized when N is six as seen in Section 6.5.1 and Section 6.5.2, optimal logic depth was found to be six regardless of activity factor or threshold voltage when a clock-gating mechanism is present. However, as seen in Figure 6-4 and Figure 6-5, both switching and leakage power curves are quite flat around the optimum, and the total power reduction is quite insensitive to modest deviations from the optimum. Therefore, 8 FO4 delays per stage might be a better choice since it simplifies design complexity with a small loss of

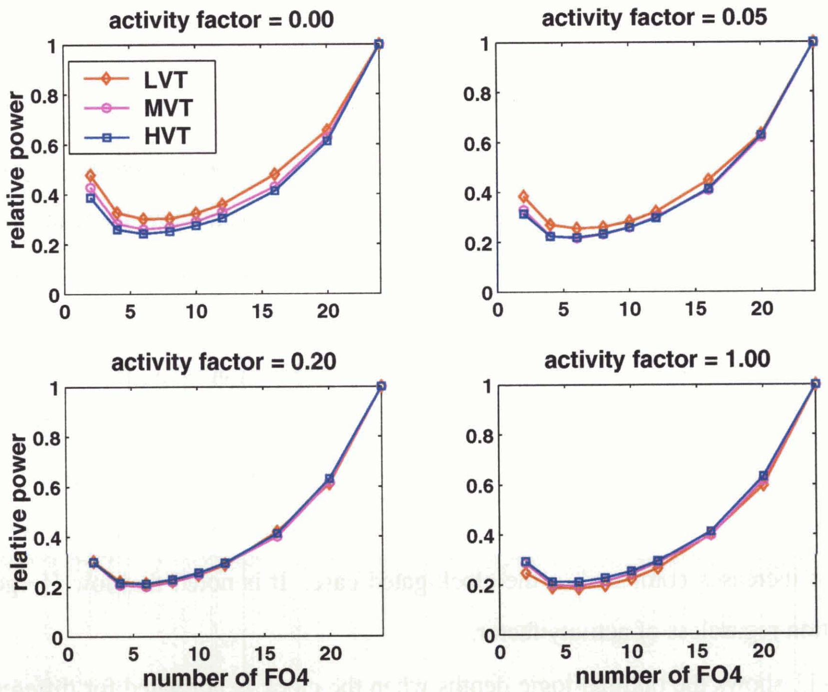


Figure 6-7: Total power scaling with a clock-gating mechanism.

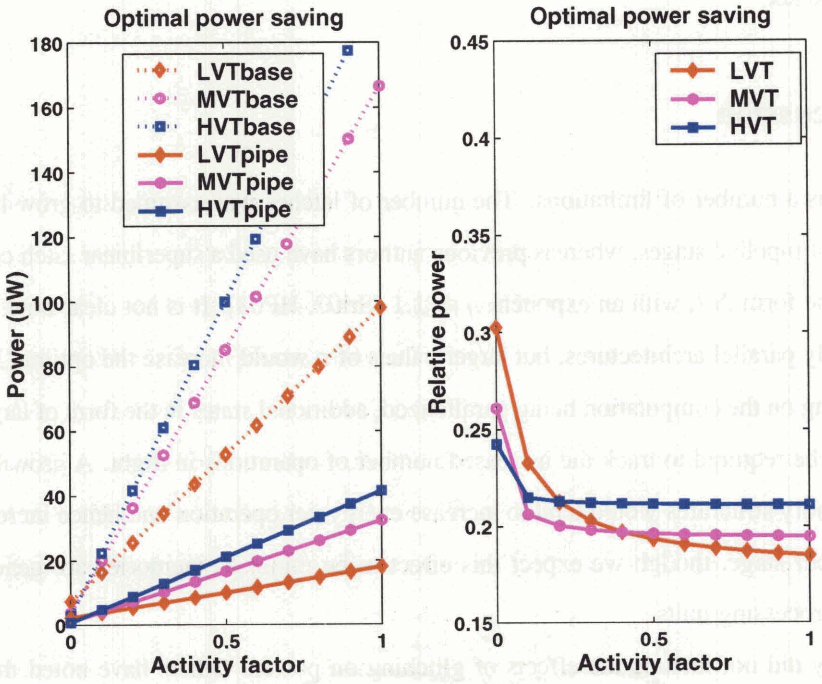


Figure 6-8: Optimal power saving with a clock-gating mechanism.

power reduction.

6.6.2 Case 2: No Clock-Gating Present

Figure 6-9 shows the simulated total power without clock-gating for different activity factors. With a low activity factor, total power curves follow idle power curves and low V_T leads to more power reduction (Section 6.5.3). As the activity factor increases, total power curves follow switching power curves.

Figure 6-10 shows the simulated optimal total power reduction when there is no clock-gating mechanism. With zero activity factor, optimal power reduction is around 5 to 15% less than the clock-gating present case because of the timing element switching power overhead which is not present when there is a clock-gating scheme. Optimal power reduction reaches 80% slowly as activity factor increases compared to the clock-gated case. It is noted that low V_T gets the most power reduction regardless of activity factor.

Figure 6-11 shows the optimal logic depths when the clock is not gated for different threshold voltages. Because the idle power is minimized when N is eight (Section 6.5.3), optimal logic depths remain at eight until the activity factor reaches around 0.2 (0.3 at high V_T) and after 0.2 (0.3 at high V_T), it falls to six.

6.7 Discussion

Our study has a number of limitations. The number of latches was assumed to grow linearly with the number of pipeline stages, whereas previous authors have used a superlinear latch count scaling formula of the form N^η, with an exponent $\eta \approx 1.1$ [Sri02, HP03]. It is not clear how latch counts scale in highly parallel architectures, but larger values of η would increase the optimal logic depth.

Depending on the computation being parallelized, additional states in the form of larger memory arrays might be required to track the increased number of operations in flight. A growth in the size of these memory structures would tend to increase energy per operation and hence increase optimal logic depth per stage, though we expect this effect to be minor as memories are generally lower power than processing units.

Our study did not include the effects of glitching on power. Others have noted that glitching activity reduces linearly with pipeline depth as it becomes less likely that inputs to a gate would have very different path lengths [Sri02]. This effect would tend to push the optimum towards shallower

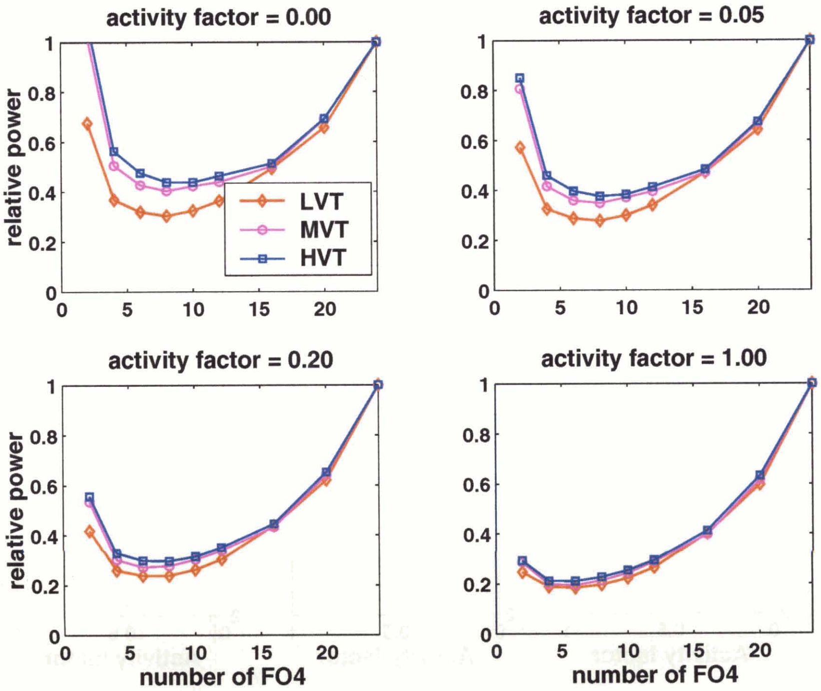


Figure 6-9: Total power scaling with no clock-gating mechanism.

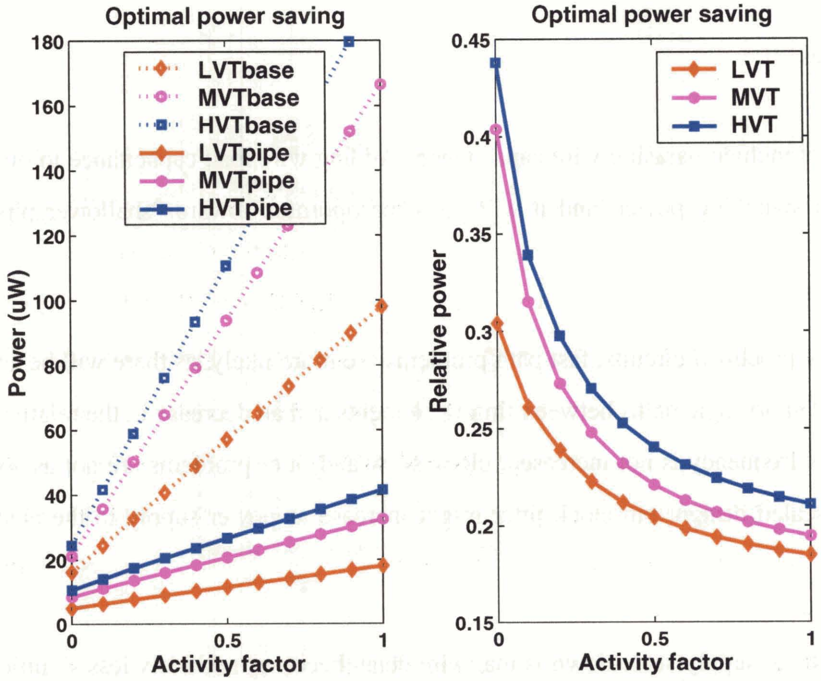


Figure 6-10: Optimal power saving with no clock-gating mechanism.

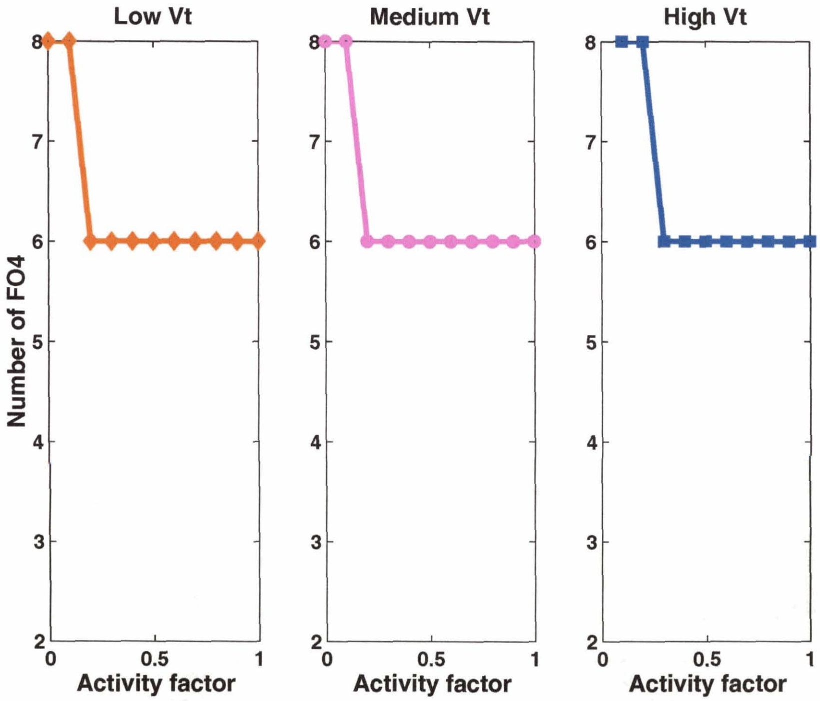


Figure 6-11: Optimal logic depth with no clock-gating mechanism.

pipeline stages.

We did not include parasitic wire capacitance. Adding wire load capacitance to our model will increase total switching power, and it will push the optimum towards shallower pipeline stages again.

For deeply pipelined circuits, fast path problems are more likely, as there will be an increase in the number of short logic paths between timing elements and an decrease in the relative wire delay. Because clock frequency is not increased, clock skew and jitter problems are not as apparent as in a frequency-scaled design, but clock jitter might increase as power supply to the clock drivers is reduced.

One benefit of supply scale-down is that wire delay becomes relatively less significant as gates slow down. This helps reduce some of the design effort of building a highly pipelined circuit compared with pipelining for increased clock frequency.

6.8 Conclusions

Pipelining can be an effective power-reduction tool when used to support voltage scaling in digital systems implementing highly parallel computations. Simulation results show that power-optimal logic depth is 6 to 8 FO4 and optimal power reduction varies from 55 to 80% compared with a 24 FO4 design depending on threshold voltage, activity factor, and the presence of clock-gating.

Even though the exact power-optimal pipelining is technology-dependent, we can gain some important insights from the simulation results. First, higher activity factors decrease the power-optimal logic depth and increase the optimal power reduction because pipelining is most effective at reducing the additional switching power. Second, pipelining is more effective with lower threshold voltages, resulting in lower logic depths and lower power, except for low activity factors when leakage power is dominant. Third, clock-gating enables deeper pipelining and more power reduction because it reduces timing element overhead when the activity factor is low.

Therefore, power-optimal pipelining with clock gating should be an efficient low-power technique for high throughput blocks in systems implementing highly parallel computations.

Chapter 7

Power-Optimal On-Chip Networks

Cross-chip global wires are becoming progressively problematic as feature sizes shrink, with their delay and power consumption rapidly increasing relative to individual logic gates [Ho 01]. These worsening trends have led to proposals that replace design-specific global wires with structured on-chip networks in large ASIC designs [Sgr01, DT01].

This chapter explores the power implications of use of an on-chip network in two steps [1]. We first develop detailed power models for power-optimized wires, including the effects of leakage currents. Deep wire pipelining can reduce communication power with a small increase in latency, and the increased latency can be tolerated by exploiting the ample parallelism present within the typical applications that the custom ASIC or FPGA chips run. We next examine the use of power-optimized wires in two contexts: 1) conventional ASIC or FPGA designs where dedicated global wires are replaced with dedicated but power-optimized wires, and 2) tiled architectures where all inter-tile global communication is via a dynamic packet-routed on-chip network using power-optimized links. We vary the tile size and use Rent's rule [CS00] to estimate interconnect density. Smaller tiles put more connections on the power-optimal wires of the on-chip network, but require more routers. Although power-optimized wires can reduce global wire power significantly in wire-routed ASIC or FPGA designs, it is difficult to achieve significant power reduction in packet-routed tiled designs due to the energy expended in routers even for highly multiplexed inter-tile traffic.

[1] The work in this chapter was a joint work with Krste Asanović and was previously published in [HA05].

Table 7.1: Delay scaling of logic gates and wires. k is the process scaling factor.

Components	Delay
Logic gates	$\frac{1}{k}$
Local wires	1
Unrepeated global wires	k^2
Repeated global wires	k

7.1 Global Wires

Previously, pipelining was solely used for logic gates. On-chip wires were ignorable in terms of delay and power, compared to the gates. However, the continuous scaling of wire dimensions, as with transistor feature sizes, has changed the situation totally, particularly for global wires. Global wires have become comparable to gates in terms of both delay and power [Ho 01]. In this deep submicron technology era, global wires are definitely precious resources for optimal VLSI digital system design and pipelining is often indispensable.

Constant scaling, where all the dimensions of wires scale linearly proportional to the scaling factor, k, makes wire resistance, R, increase proportional to k ($R \propto \frac{l}{t \times w} \propto k$, where l, t, and w are the length, thickness, and width of wire respectively), while making wire capacitance, C, decrease inversely proportional to k ($C_{vertical} \propto \frac{w \times l}{h} \propto \frac{1}{k}, C_{horizon} \propto \frac{t \times l}{s} \propto \frac{1}{k}$, where s and h is the wire spacing and height respectively and fringe caps are ignored), leading to the constant RC wire delay when the delay of transistors scales down linearly ($d_{transistor} \propto \frac{1}{k}$). As for global wires, the situation is worse. With the chip size, L, remaining constant, the global wire length, l_{global}, does not scale but remains constant ($l_{global} \propto L$). Thus, R becomes quadratically proportional to $k_{scaling}$ ($R_{global} \propto \frac{L}{t \times w} \propto k^2$) while C remains constant ($C_{vertical} \propto \frac{w \times L}{h} \propto 1, C_{horizontal} \propto \frac{t \times L}{s} \propto 1$), resulting in the quadratic increase of RC global wire delay. To make the problem even worse, the average chip size is, in fact, slowly increasing, resulting in the increase of average global wire length.

To enhance speed, numerous repeaters built with large transistors for speed are inserted. In global wires, repeaters segment the wires into multiple segmented wires with fixed length and make the wire delay proportional to the wire length. However, even with the repeaters, the global wires' delay does not scale with the transistor and linearly increases as the technology shrinks (Table 7.1).

Along with delay, global wire power has also increased rapidly. The power consumed by repeaters is already comparable to the power from switching wire caps, and some dimensions of global wires, such as height, thickness, and spacing, do not scale down linearly for speed, resulting in increased wire caps. To make the power problem worse, the number of cross-chip global wires has also increased, as larger-scale digital systems, such as system-on-chip (SoC) and chip multiprocessor (CMP), tend to include multiple cores and previous off-chip interconnects are being replaced with on-chip cross-chip global wires. All these changes have made the power consumption of global wires comparable to that of logic gates. In addition to delay and power increases, the problems of traditional automatic ASIC routing such as routing congestion and noise coupling are being exacerbated, as the relative length of global wires increases.

As a solution for all these problems of global wires, two levels of innovations can be made: optimizing wires through pipelining and structuring wires [Sgr01, DT01]. For many applications such as streaming multimedia, software can be mapped to hardware such that global communication between units can be made latency-tolerant. Pipelining and other energy-delay trading tools, such as supply voltage scaling and repeater sizing and spacing, can achieve significant energy reduction while keeping the same wire bandwidth, with only a small increase in latency (Figure 7-1).

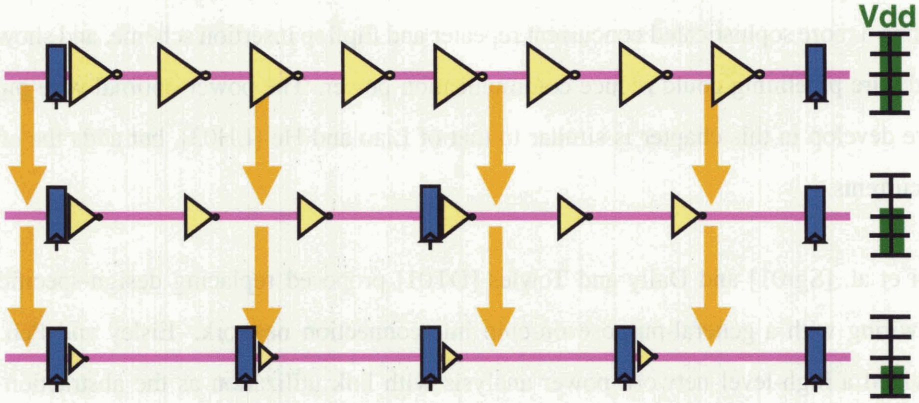


Figure 7-1: Pipelining wires and scaling supply voltages for energy reduction at a fixed bandwidth.

Structuring global wires or, more specifically, replacing global wires with an on-chip network, provides a higher level of optimization. The on-chip network links can be highly optimized by controlling their electrical environment to allow the use of optimized signaling techniques. Wiring efficiency can be improved by replacing a large number of low activity dedicated wires with fewer multiplexed communication links.

7.2 Related Work

Bakoglu [Bak90] reported the delay-optimal repeater sizing and spacing for a repeated wire. Ho et al. [Ho 01] pointed out that the power consumption of the delay-optimal repeated wire is prohibitively large, and suggested increasing repeater spacing and decreasing repeater size to save power while sacrificing some speed. Kapur et al. [Kap02] and Banerjee and Mehrotra [BM02] calculated the power-optimal repeater sizing and spacing for global interconnects using a simple first-order RC repeated wire model. Gupta et al. [Gup03] described a high-level interconnect power model for wires of a single core chip.

Chandrakasan et al. [Cha92] first suggested the use of pipelining for power reduction in digital circuits. Heo and Asanovic [HA04b] examined power-optimal pipelining for logic datapaths in deep submicron technology both analytically and through circuit simulation. Cocchni [Coc02] estimated the effect of concurrent flipflop and repeater insertion while considering routing tree topology, but focused only on minimizing wire latency. Liao and He [LH03] modeled full-chip interconnect power using a more sophisticated concurrent repeater and flipflop insertion scheme, and showed how increased wire pipelining could reduce communication power. The power-optimal wire pipelining model we develop in this chapter is similar to that of Liao and He [LH03], but adds the effects of leakage currents.

Sgroi et al. [Sgr01] and Dally and Towles [DT01] proposed replacing design-specific global on-chip wiring with a general-purpose on-chip interconnection network. Eisley and Peh [EP04] first provided a high-level network power analysis with link utilization as the abstraction of network power. Some previous work focused on the low-level power estimation of routers. Wang et al. [Wan02b] provided a low-level general framework for different types of routers, Orion, and verified the simulator with Alpha 21364 and Infiniband router examples [Wan02c]. Chen and Peh [CP03] added a leakage power model to the Orion simulator. Ye et al. [Ye 02] focused only on the power consumption of the switch fabric in a router.

We build a complete system-level power model for on-chip networks including routers and power-optimal links. We examine the trade off in tile size versus communication power, using Rent's Rule to estimate inter-tile and intra-tile interconnect.

7.3 Wire Power Model

In this section, we present an analytical latency and power model for a pipelined and repeated wire when throughput is fixed. We include the switching and leakage power consumed by repeaters and flipflops in addition to the switching power of the wire capacitance.

7.3.1 Methodology

We choose BPTM 70 nm technology as our deep submicron process technology [Dev01]. Base supply voltage is 0.9 V and V_T is 0.20(-0.22) V. V_T is set quite high to reduce leakage power of repeaters and flipflops. We assume that clock period is fixed at 24 FO4 delays (clock frequency of 2 GHz), representing a high-performance digital circuit [HA04b]. For most designs, clock frequency will be set by logic within a tile, and we assume the network links run at the same frequency with a fixed throughput requirement.

We assume three categories of metal interconnect: local, semi-global, and global. Table 7.2 shows the characteristic dimensions and RC components of these wires.

70 nm Cu tech	Local	Semi-global	Global
Width (μm)	0.10	0.14	0.45
Spacing (μm)	0.10	0.14	0.45
Thickness (μm)	0.20	0.35	1.20
Height (μm)	0.20	0.20	0.20
Resistivity (Ωcm)	2.2	2.2	2.2
$C_{wire}(fF/mm)$	152	178	228
$R_{wire}(\Omega/mm)$	1100	449	41
$R_{wire}C_{wire}$ (FO4)	8.03	3.84	0.45
Max. distance in 24 FO4 (mm)	2.10	3.04	8.88
Max. distance in 24 FO4 (mm)	2.10	3.04	8.88
Latency-optimal repeater spacing (mm)	0.29	0.42	1.22

Table 7.2: Wire characteristics of our example 70 nm technology.

7.3.2 First-Order RC Wire Model

Figure 7-2 shows a first-order RC model of a wire [Ho 01]. We assume minimum-sized flipflops. In the RC circuit, the wire segment is modeled inside the dotted box and the repeaters are outside.

Wire delay is represented as a function of the repeater sizing (ratio of the repeater gate cap and

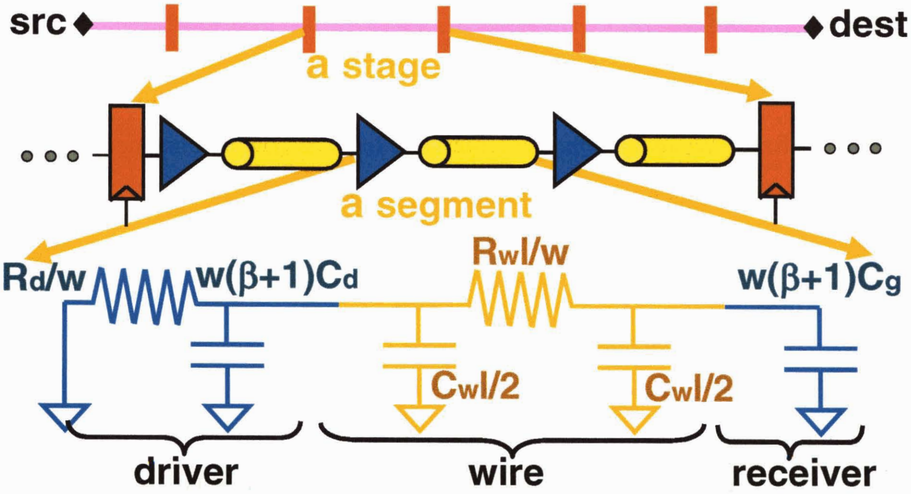


Figure 7-2: First-order RC model of wire. The length of the whole wire is L. β is the PN ratio, w is the width of the repeater NMOS transistor, C_w, C_d, and C_g are the unit-length wire cap and drain and gate caps of the minimum-sized inverters respectively.

wire cap within a wire segment), r.

$$r = \frac{w(\beta+1)C_g}{C_w l}$$

where l is the wire segment length.

When interconnect supply voltage is scaled, the scaling factor,

$$S = \frac{V_{dd}}{V_{dd0}} \times (\frac{V_{dd0} - V_T}{V_{dd} - V_T})^{\alpha}$$

is used. α is a velocity saturation effect factor.

Wire delay and link latency are calculated as:

$$\begin{aligned}
Delay &= 0.7\frac{l_{st}}{l}\frac{R_d}{w}(w(\beta+1)(C_d+C_g)+lC_w) && (7.1)\\
&\quad +0.7\frac{l_{st}}{l}(l^2\frac{R_wC_w}{2}+lR_ww(\beta+1)C_g) && (7.2)\\
&= 0.7l_{st}(\frac{1}{3l}+\frac{2}{9rl}+\frac{kl}{2}+krl)\times FO4 && (7.3)\\
Latency &= \frac{0.7L}{T-D_{ff}}(\frac{1}{3l}+\frac{2}{9rl}+\frac{kl}{2}+krl) && (7.4)
\end{aligned}$$

where T is the clock period, D_{ff} the flipflop delay, and k is the unit-length wire RC delay in FO4.

We calculate power as:

$$\begin{aligned} Power &= \frac{1}{2}AF((1+\frac{3}{2}r)C_wL + C_{ff}latency)fV_{dd}^2 & (7.5)\\ &+ (k_{rep}(\frac{3}{2}r)C_wL + k_{ff}C_{ff}latency)V_{dd}^{1+\gamma} & (7.6)\end{aligned}$$

where AF is the activity factor and C_{ff} is the flipflop cap. The repeater drain cap is assumed to be half of the repeater gate cap. k_{rep} and k_{ff} are leakage power coefficients for repeaters and flipflops. The former term of the equation is the switching power component and the latter is the leakage power component. We assume leakage current remains constant regardless of input patterns or internal states. Leakage power scales super-linearly in deep submicron technology ($1 < 1+\gamma < 2$) [HA04b].

7.3.3 Pipelining Wire

Latency is minimized when r^* is $\frac{1}{\sqrt{3}}$ and l^* is $\sqrt{\frac{2}{3k}}$, where wire delay and repeater delay are equal. However, this minimum latency point requires very large, power-consuming repeaters [Ho 01].

We can save power by using deeper pipelining to provide additional time slack in each wire stage. Although there are many ways of exploiting the time slacks obtained from pipelining, we focus on two variables: repeaters and supply voltage. We can either reduce the size and increase the spacing of repeaters, or scale down supply voltages, or both.

Figure 7-3 shows latency-power curves of wires while varying repeater sizing, spacing, and supply voltage. In the figure, the `sizing` curve shows power-latency tradeoff through repeater sizing only, while repeater spacing is fixed at the minimum latency point and supply voltage is constant at the nominal voltage. Increasing repeater size over the minimum latency point results in larger latency and power. The `+spacing` curve shows power-latency tradeoff through repeater sizing and spacing. Repeaters are power-optimally sized and placed while supply voltage is fixed.

Finally, the `+scaling` curve adds supply voltage scaling to the optimally sized and spaced repeaters. Supply voltage scaling is by far one of the most effective techniques for trading time slack for power. Supply voltage reduction leads to a cubic reduction in switching power and also a super-linear reduction in leakage power, as leakage current has a strong dependency on drain voltage in deep submicron processes [HA04b]. Adding supply voltage scaling enables much greater power saving compared to optimal repeater sizing and spacing alone, especially when latency is allowed to increase by greater than 10%, but requires a second power supply to be distributed to the

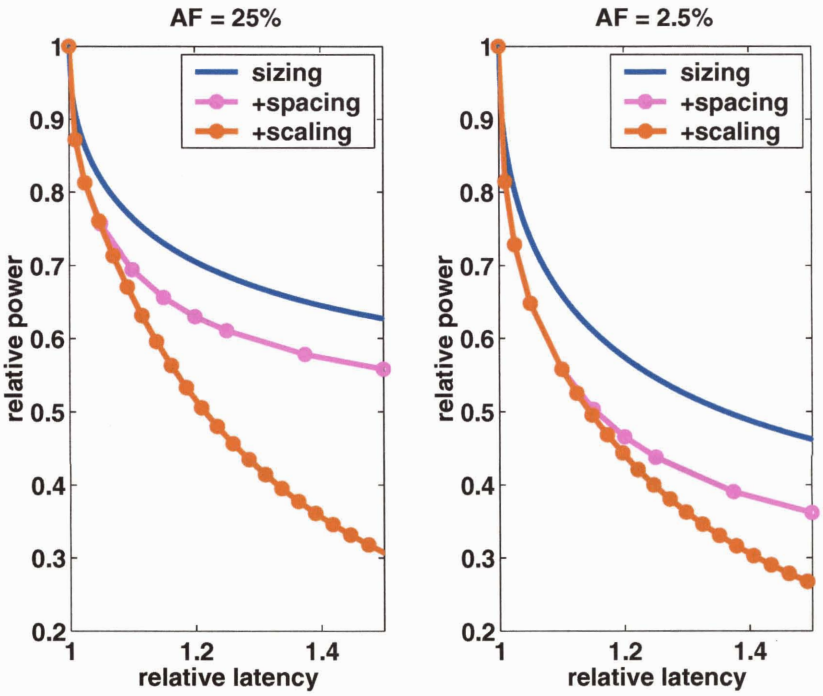


Figure 7-3: Wire latency-power curves for activity factors of 25% and 2.5%, while changing repeater sizing, spacing, and supply voltage. Both axes are normalized to the minimum latency point.

interconnect network.

Repeater sizing and spacing is more effective when the activity factor is low since it reduces the leakage power of repeaters as well as their switching power. On the other hand, at higher activities, the power due to wire capacitance switching becomes dominant and thus the total wire power is less affected by repeater sizing and spacing.

Overall, the combined techniques achieve a three- to four-fold reduction in communication power when latency is allowed to increased by 50%.

7.4 On-Chip Interconnect Network Power Model

We develop a system-level interconnect power model for large digital designs. We assume the design is for a highly parallel system, such as a DSP engine or network processor, and assume the design can be divided easily and flexibly into any number of smaller tiles. Each tile represents a computation module including local memory. We examine three design points: 1) a single tile containing the whole design, 2) a tiled design where tiles are connected with a statically wire-routed network, and 3) a tiled design with a dynamically packet-routed inter-tile network.

We focus only on power consumption for this analysis and assume that the performance impact from additional inter-tile latency is not significant. When we divide the chip, we keep the logic and local memory ratio the same regardless of the tile size. We assume the total power consumed by logic and memory transistors remains roughly the same regardless of network configuration, and focus on power consumed by communication wires (intra-tile or inter-tile) and supporting transistors (repeaters or registers). Communication power is already comparable to logic and memory power and increases as digital systems become more communication-centric than computation-centric.

Communication power can be divided into three parts: inter-tile wire power, router power, and intra-tile wire power. The intra-tile wire is the power consumed by design-specific wires connecting logic and memory transistors. As tile sizes shrink, more intra-tile wires turn into the inter-tile wires depending on tile and network architecture.

Table 7.3 summarizes the dimensions of our example chips. Estimates are based on ITRS 2004 [Int04]. We assume that gates are uniformly distributed on a chip. Although Rent's rule provides the estimates of number of wires, it does not give the bandwidth requirements. For simplicity, we assume a uniform activity factor regardless of wire length.

Chip length (mm)	20
Tile length (mm)	0.5 – 20
Gate density (gates/mm^2)	3.2×10^5
Gate pitch (μm)	1.8

Table 7.3: Dimensions of our example chips.

7.4.1 Single Tile Baseline

We first use Donath's method to estimate the wire distribution for the whole chip treated as a single tile. Estimation of wire distribution is an important application of Rent's rule, which is an empirical rule stating that the number of wires leaving a circuit block is exponentially proportional to the number of gates in the block. We assumed a nominal value, $\frac{2}{3}$ for the Rent exponent, p which agrees with the wire distribution of Intel microprocessors ($N(l) \propto l^{-1.65}$) [Yan98].

We chose the following wire distribution equations, which divide wires into two regions [CS00]:

$$\begin{aligned} N(l) &\propto (\frac{l^3}{3} - 2Ml^2 + 2M^2l)l^{(2p-4)}(l \leq M) && (7.7) \\ &\propto (2M - l)^3 l^{(2p-4)}(M < l \leq 2M) && (7.8) \end{aligned}$$

where l is the length of a wire, M is the chip length, and $2M$ is the length of the longest wire within the chip assuming Manhattan wiring.

We assumed that a local wire is used if the wire length is less than 100 gate pitches, a semi-global wire if less than 1400, and a global wire if 1400 or more. Figure 7-4 shows the wire distribution of our base chip. We can see that the number of wires decreases drastically as we reach the very longest wires in region 2. Figure 7-6 shows the wire power of the base chip, at the point where the tile size is 20 mm (that is, when the whole chip is one tile).

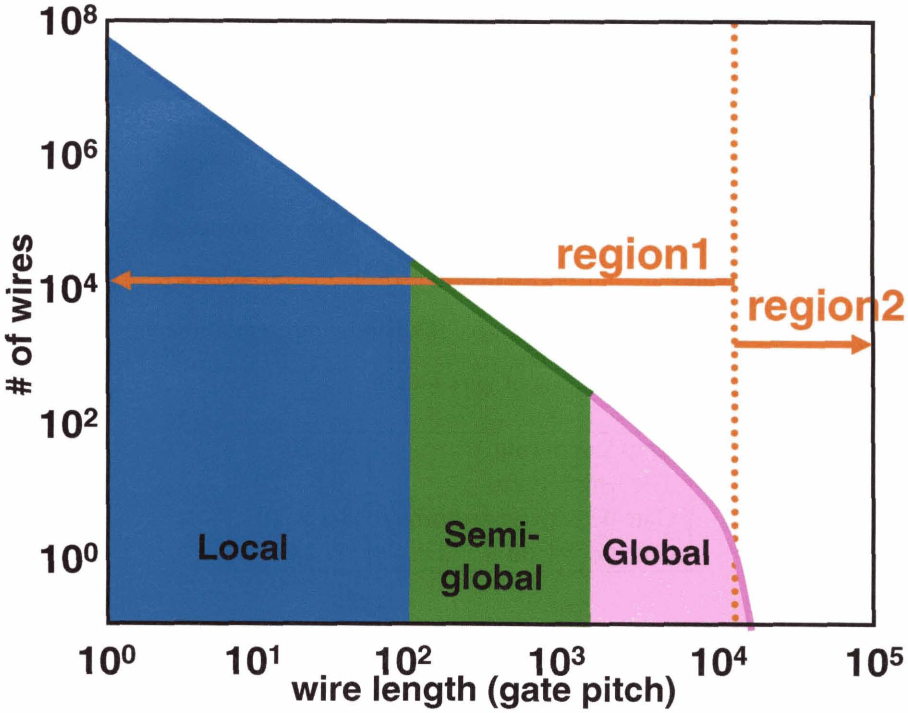


Figure 7-4: Wire distribution of our base chip. Two dotted vertical lines divide wires into local, semi-global, and global wires. The vertical solid line shows the boundary between region 1 and 2 wires.

7.4.2 Wire-Routed Tiles

We now divide the chip into multiple tiles, and replace inter-tile wires with power-optimal pipelined wires. First, we build a statically wire-routed on-chip network by making wire channels between tiles and moving global wires onto the wire grid (Figure 7-5). The number and total length of wires do not change. We assume that all the wires longer than twice the tile size ($2L$) are replaced with inter-tile wires, while the rest remain intra-tile wires.

We assume that all the intra-tile wires regardless of dimensions are latency-optimized. In par-

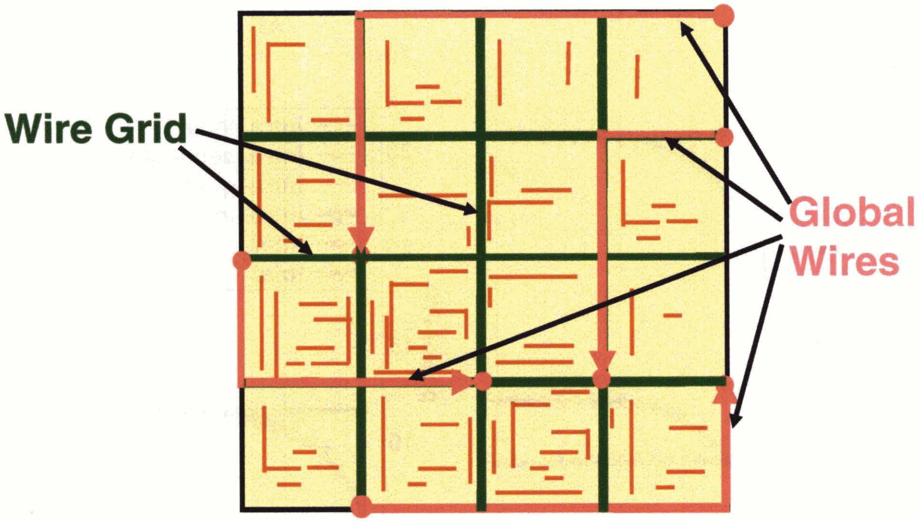


Figure 7-5: Wire-routed ASIC design.

ticular, compared to inter-tile wires, they are more performance-critical. The following equations show that the total power of the intra-tile wires.

$$\begin{aligned} N(l) &\propto l^{2p-3} && (7.9) \\ P_{intra} &\propto \sum_{l=1}^{2L} N(l)l && (7.10) \\ &\propto L^{2p-1} && (7.11) \end{aligned}$$

where L is the tile size.

Figure 7-6 shows the wire power consumption of a tiled wire-routed chip. As the tile size decreases (smaller than half of the chip), intra-global wire power decreases exponentially while intra-semi-global or intra-local wire power remain roughly unchanged. However, the increase in power on inter-tile wires matches the power loss of the intra-global wires and so the total wire power stays roughly the same.

Figure 7-7 shows the power saving of a tiled wire-routed chip when inter-tile latency is increased by 25% and the network wires are pipelined. Smaller tiles result in greater power saving as more signals are pipelined. In particular, when leakage power is significant (AF = 0.1%), pipelining through repeater optimization and voltage scaling is more effective at reducing the inter-tile wire power, since pipelining is more effective at saving leakage power (Section 7.3.3). When AF is 0.1%, almost half of the total power can be saved.

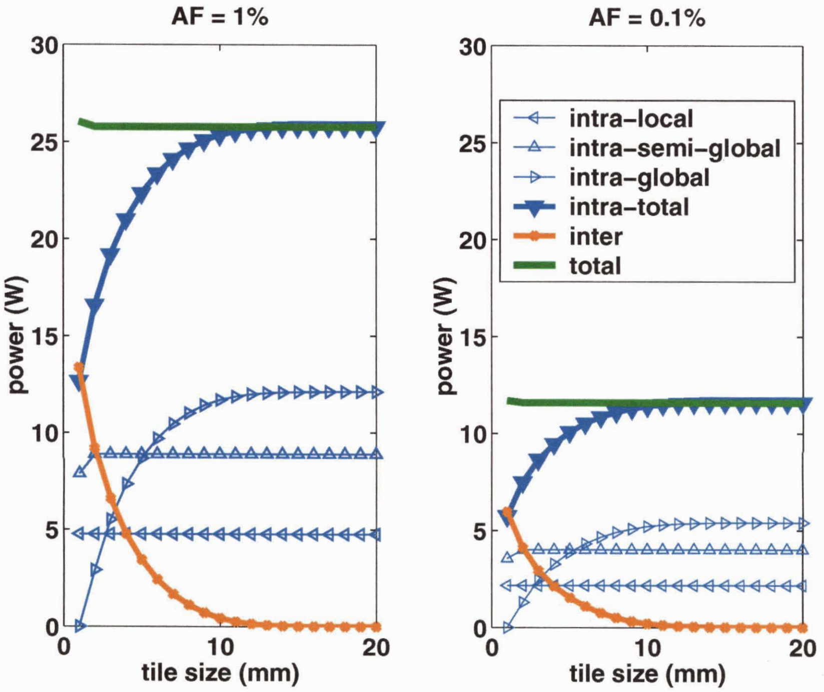


Figure 7-6: Wire power consumption of tiled wire-routed design for varying tile sizes, assuming uniform activity factors of 1% and 0.1%.

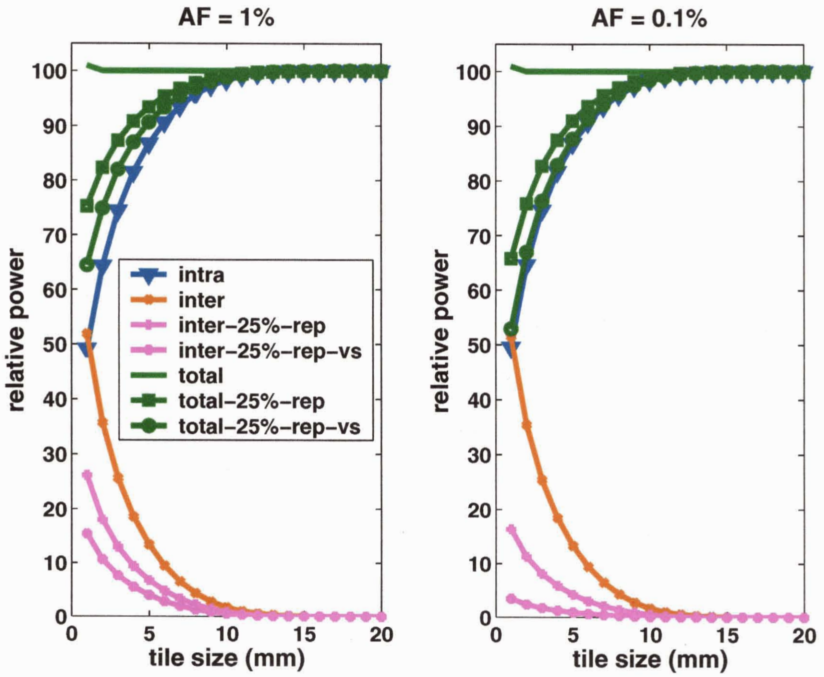


Figure 7-7: Tiled wire-routed design: power saving by pipelining. `inter-25%` means 25% increased latency requirements for the inter-tile wires. `rep` represents repeater sizing and spacing and `vs` also includes voltage scaling.

7.4.3 Packet-Routed Tiles

We finally consider a packet-routed tiled architecture. We build a dynamically packet-routed on-chip network by adding routers and IOs and connecting routers with links. Figure 7-8 shows an example of a tiled ASIC architecture (4 by 4) and a mesh interconnect network. A mesh interconnect network was chosen since it is simple to design, power-efficient, and scalable. Tiles communicate with others only through routers and links between routers.

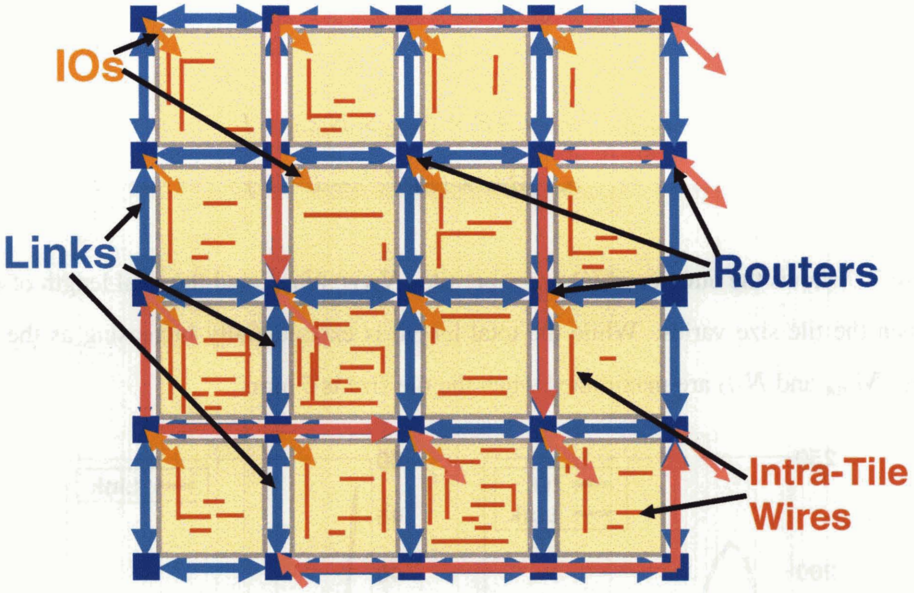


Figure 7-8: Tiled packet-routed ASIC design.

On-chip network links are much cheaper in terms of area and power, than traditional off-chip network links. Thus it is natural that tiles exploit wider links than a single-tile chip does. However, an excessive number of on-chip IO wires result in a huge power and area overhead for routers. Usually, some degrees of multiplexing and packet encoding are employed to reduce the number of IO wires while increasing the activity on link wires. We define the multiplexing factor, MF, as the ratio between activity on packet-routed link wires and that on the inter-tile wire-routed links they replace. The multiplexing factor will vary according to application and tile architecture.

We assume the total chip bandwidth (BW) is conserved and the total sum of global wire length times activity factor remains the same regardless of the tile size. The left side of the following equality shows the BW of a wire-routed tiled chip, and the right side shows that of the base single tile chip.

$$LAF \times (N_{link} \times N_{tile} \times L) \quad = \quad AF \times \sum_{l=2L}^{2M} N(l)l \tag{7.12}$$

where LAF is the activity factor on link wires and N_{link} is the number of wires between two adjacent routers.

The following equations show the total inter-tile wire power. The total inter-tile wire power increases as the tile size decreases at the same rate as the decrease of the intra-tile wire power.

$$P_{inter} \quad \propto \quad LAF \times N_{tile} \times N_{link} \times L \tag{7.13}$$

$$\propto \quad MF \times (M^{2p-1} - L^{2p-1}) \tag{7.14}$$

Figure 7-9 shows the number of IO wires per tile (N_{IO}), N_{link}, and the total length of inter-tile wires when the tile size varies. While the total length is exponentially increasing as the tile size decreases, N_{link} and N_{IO} are maximized when the tile size is 5 mm.

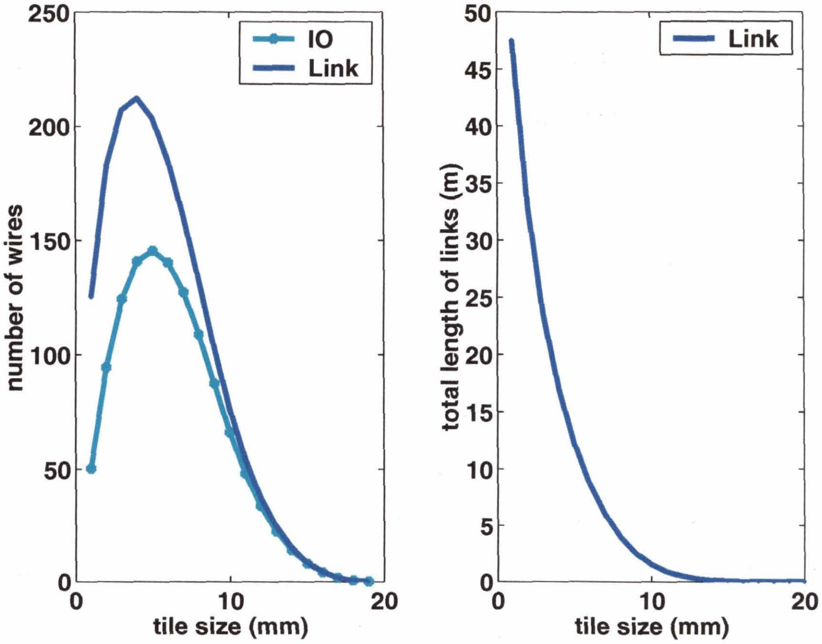


Figure 7-9: Number and total length of inter-tile wires.

We assume a low-latency virtual channel router [Mul04]. Virtual-channel flow control maintains high throughput even when the packet traffic is high. The inter-tile wire latency becomes relatively significant since the router has a low-latency. A router design can be divided into three main components: input and output packet storage, switch fabric, and arbiters. The power consumption of arbiters is insignificant and thus is ignored here [Wan02c]. We choose a matrix crossbar for a switch fabric implementation because it is easy to design and low-power. Since the number of ports for the switch is rather large, the wires dominate power consumption of the crossbar [Ye 02], and we ignore power consumed by the internal switches.

Table 7.4 describes the router parameters we assumed. Buffers are implemented with SRAM arrays.

Phit (bits)	32
Number of input ports	5
Number of output ports	5
Number of virtual channels per physical channel	2
Number of input buffers per virtual channel	4

Table 7.4: Virtual channel router parameters. Phit is the physical transfer size of the link.

We fix phit size and the size of routers, and instead allow multiple routers per tile rather than one large router. In case the number of IOs between routers or between a tile and the router connected exceeds the phit size, multiple phits are sent simultaneously through multiple fixed-sized routers.

The following equations show that the total power of routers, P_{router}, grows even faster than the total power of inter-tile wires, P_{inter}, as the tile size decreases.

$$\begin{aligned} P_{router} &\propto N_{tile} \times N_{link} && (7.15) \\ &\propto \frac{(M^{2p-1} - L^{2p-1})}{L} && (7.16) \end{aligned}$$

Figure 7-10 shows the power consumption of a packet-routed tiled chip. We assume zero-power routers and vary the multiplexing factor (MF) and activity factor (AF). We can see that smaller and hence more numerous tiles results in large achievable power reduction, as more global wires are replaced with fewer and thus lower-leakage network wires. When the leakage is more dominant (AF=0.1%) and more multiplexing is employed (MF=25), more inter-tile wire power is saved. Figure 7-11 shows the power saving by pipelining. When MF is 25 and AF is 1%, more than 35% of power can be saved through pipelining the network wires.

Figure 7-12 and Figure 7-13 show the power consumption when considering the power overhead of routers. In all cases, the power saving is limited by the energy cost of the routers. The router overhead limits peak wire power savings to around 0–20% with optimal tile sizes of around 2 mm.

7.5 Conclusions

We have developed a system-level interconnect power model that predicts the power savings possible by moving global traffic onto a power-optimized on-chip network. The switch to packet-routed on-chip networks has many advantages over wire-routed circuits, but we show that large power reductions are unlikely due to router power overheads. A tile size of around 2 mm is optimal in a 70 nm technology, balancing global wire power reduction with router overhead. Additional work is needed to develop low-power on-chip router units.

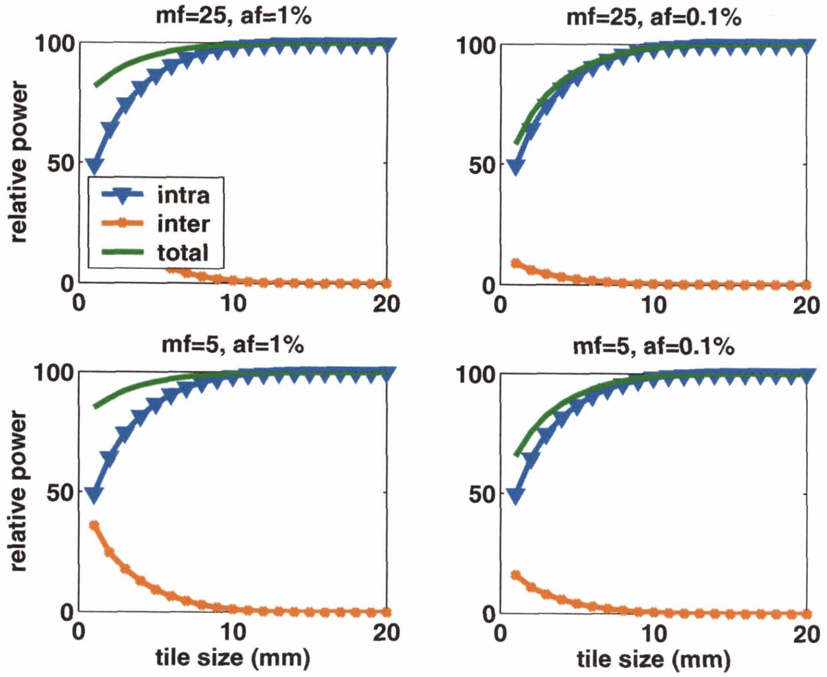


Figure 7-10: Packet-routed ASIC with ideal zero-power routers.

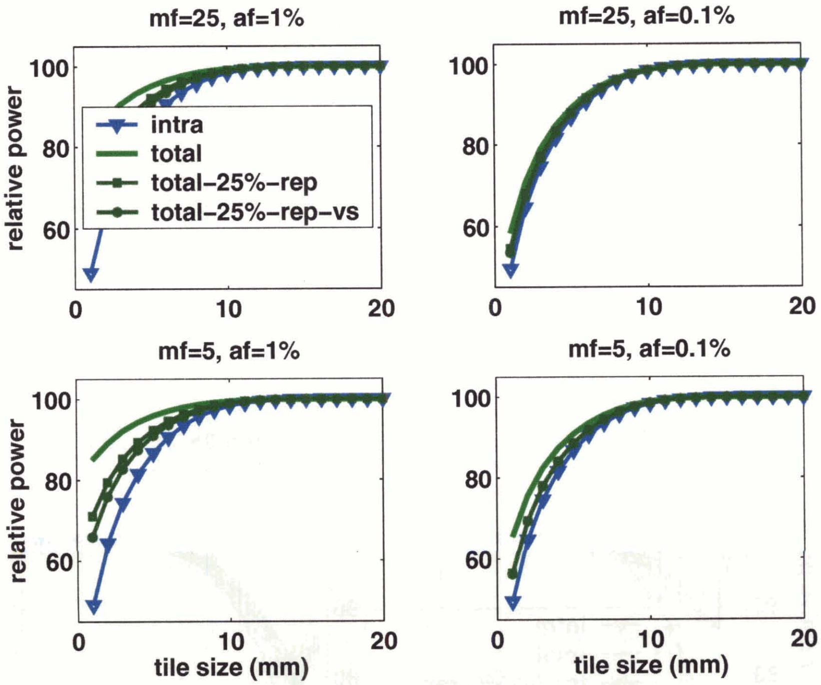


Figure 7-11: Packet-routed ASIC with ideal routers: power saving by pipelining.

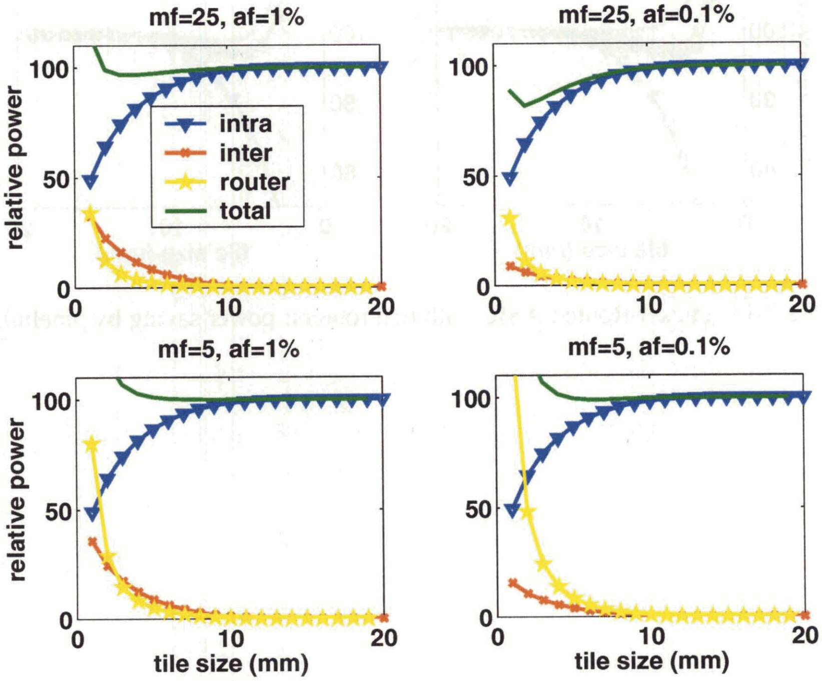


Figure 7-12: Packet-routed ASIC with real routers: power consumption.

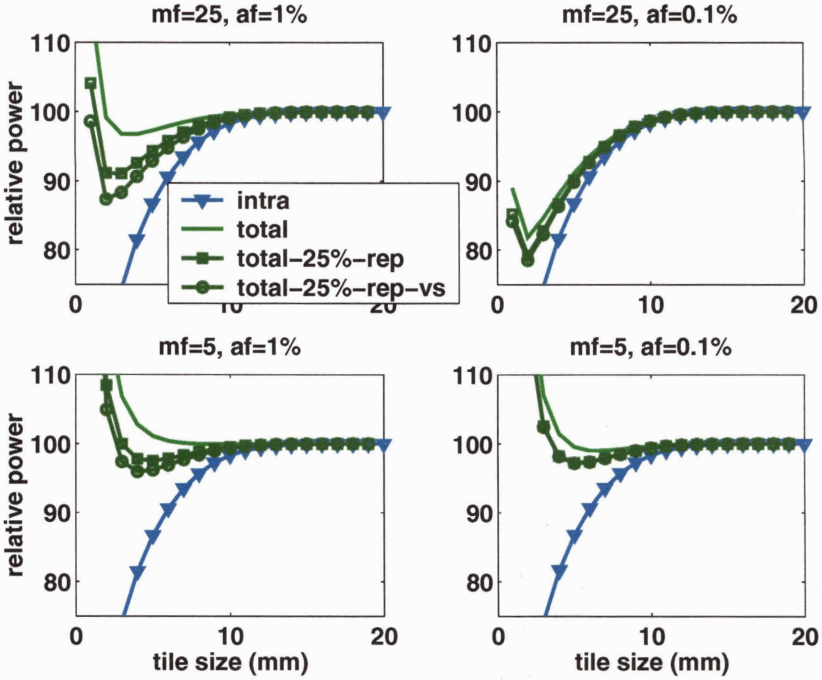


Figure 7-13: Packet-Routed ASIC with real routers: power saving by pipelining.

Chapter 8

Power Density Reduction through Activity Migration

The rapid increase of power densities has caused a significant die temperature increase, which leads to the reduced circuit speed and reliability. Previously, designers set a peak power constraint to ensure the thermal safety and optimized the system conservatively under the peak power constraint. However, we can overcome the thermal limit directly for a further optimization with dynamic thermal management. This chapter introduces and evaluates the use of *activity migration* (AM) to mitigate the power density problem [1].

Elevated die temperatures reduce device reliability and transistor speed, and increase leakage currents exponentially. Providing adequate heat removal with low cost packaging is particularly challenging as high-performance digital systems' power densities rise above 100 W/cm^2. Moreover, power dissipation is unevenly distributed across a die. Highly active portions of the design such as the issue window or execution units may have more than twenty times the power density of less active blocks such as a secondary cache [Dee02]. Even with dynamic thermal management, these hot spots limit performance because total power dissipation must be reduced until all hot spots have acceptable junction temperatures. Hot spots develop because silicon is a relatively poor heat conductor and cannot spread heat efficiently across a die.

With activity migration (AM), we instead spread heat by transporting computation to different locations on the die. Computation proceeds in one unit until it heats past a temperature threshold. The computation is then transferred to a second unit allowing the first to cool down. We show

[1] The work in this chapter was a joint work with Kenneth Barr and Krste Asanović and was previously published in [HBA03].

how AM can be used either to double the power that can be dissipated by a given package, or to lower the operating temperature and hence the operating power. Our thermal model predicts that the intervals between migrations needed to obtain maximum benefit can be much smaller than typical operating system context swap times, and will scale to smaller intervals in future technologies. We evaluate the performance impact of such small migration intervals, and examine various alternative architectural configurations that trade area for power/performance.

8.1 Related Work

We divide related work into two parts: temperature-aware simulators and dynamic thermal management (DTM) techniques.

8.1.1 Temperature-Aware Simulators

There has been a great deal of research effort on temperature-aware simulators. Dhodapkar et al. developed a thermal enabled power/performance simulator [Dho00]. They also used thermal resistance and capacitance to model the heating and cooling of a chip or a functional block. Brooks and Martonosi modeled temperature as a moving average of chip-wide power dissipation, which is a poor model since it cannot explain hot spots spatially or temporally [BM01]. Huang et al. built a framework with two goals in mind: temperature control and energy efficiency control while minimizing any slowdown. The framework makes use of various energy-management techniques in a spatially and temporally fine-grain manner according to a given policy [Hua00]. Skadron et al. proposed the formal feedback control theory for adaptive fetch toggling to manage temperature dynamically [Ska02a]. The work uses the lumped RC based thermal model with the granularity of individual functional blocks. Recently, the authors developed an architecture-level floorplan-based dynamic thermal simulator with the functional block granularity, "HotSpot" [Ska02b, Ska03]. It models an equivalent circuit of thermal resistances and capacitances that correspond to microarchitecture blocks and essential aspects of the thermal package. The simulator is based on "Wattch" [Bro00] power model.

8.1.2 Dynamic Thermal Management

We divide DTM schemes into two groups. The first category reduces power consumption through energy-delay tradeoff techniques. This category is sometimes called "thermal throttling". Performance loss is a major disadvantage. On the other hand, the second category increases the effective area to lower power density while keeping power consumption the same.

Decreasing Power through Energy-Delay Trades

Some DTM schemes actively gate the clock toggling of hot spots to lower power and temperature while sacrificing performance, though clock gating is usually utilized to cut down the power waste of unused blocks. The clock gating for DTM can be done on the chip-level [Hua00, Ska03, Gun01] or in a finer-grain, controlling the toggling rates of individual functional blocks (e.g., fetch block) [Ska03, BM01]. Dynamic voltage and/or frequency scaling is another popular energy-delay trading tool utilized by many DTM schemes [Hua00, Ska03, Gun01]. Throttling instruction flows [San97, BM01], turning on low-power features [Hua00], or turning off performance-oriented features such as speculation [BM01] are other examples of DTM techniques that trade performance for lower power.

Techniques such as clock gating and throttling reduce the activity rate and save only the switching energy. Thus, the techniques become less effective when leakage power is dominant. Even though thermal emergencies are rare events, the performance loss by DTM techniques can be significantly large enough that the overall performance degradation can be severe.

Increasing Effective Area

Increasing effective area can reduce power density while keeping power and performance the same. A straightforward way of trading area for reduced power density is to spread out hotspot blocks such as register files, but it is not an attractive option as latency and switching energy and delay are increased due to increased wire length. Using empty space on the die can help spread heat [Dee02] for thicker wafers, but this becomes much less attractive for thinner wafers as the lateral conductance decreases relatively.

A few researchers have tried to increase the effective area by migrating computations on duplicated blocks. Unlike spreading, the longer wires are placed out of the critical paths and infrequently used. Another advantage of migration is that we can run the multiple duplicated blocks in parallel to increase throughput when the application has that type of parallelism. Intel described a scheme called ”core hopping” [Int02]. However, swapping threads at OS timescales is too slow to get the maximum benefit and has a large area overhead. Time constants will also get smaller in absolute time in future technologies. Intel also described this only as a means of avoiding thermal crisis rather than as a way of improving performance or reducing energy. Skadron et al. emphasized the importance of fine-grain dynamic thermal management and proposed a scheme, called "migration computation" [Ska03]. The dual-pipeline scheme [Lim02] can be considered a coarse-grain form of activity migration. When the primary out-of-order high-power pipeline gets too hot, the secondary in-order low-power pipeline is turned on and utilized until the primary pipeline becomes cool.

8.2 Thermal Model

In this section, we develop a thermal model we later use to evaluate AM. To determine the maximum achievable benefit, we can simplify the model to consider a single circuit block with a given power density.

We are primarily interested in the case where performance is constrained by thermal packaging and hence we are only worried about the worst case hot spot. Considering a digital system in thermal crisis, one block (for example, the scheduler or regfile or ALUs) will hit the peak temperature. It does not actually matter which block it is, which is why we use a single power density number to model the power density of the hot block. The rest of the blocks on the die will have varied power densities of course, and the average power density across the whole die will be much lower. At thermal crisis, the best thing to do is to alternate between the two sites to cool the hot spot.

One concern is that the detailed dynamic behavior of a program will determine whether or not a digital system hits thermal crisis, and so will determine how often our scheme gives a benefit. Our limited study assumes that one of the various forms of thermal management allows us to deliberately run as hot as possible without violating the junction temperature, i.e., the optimal performance point is when we run fast enough to always be at the verge of a thermal crisis. As absolute power grows, and the gap between maximum and typical power dissipation grows, we expect this to be the usual case, i.e., it will be rare that an active digital system is not on the verge of overheating – otherwise we have wasted money on packaging (Energy-constrained systems will have other criteria, of course).

If the system is not operating at this extreme point, then periods of lower dissipation mean that we can run for longer without triggering AM, hence lowering the performance overhead.

From this model, we can find the necessary thermal time constants and the expected benefit in terms of sustainable power increase or temperature reduction.

8.2.1 Thermal and Process Properties

Table 8.1 shows the thermal properties and process technology data we used in this model. We present two technology cases: a present mature technology (Current case) and a near-future technology (Future case). The transistor models for 180 nm and 70 nm processes were based on TSMC 180 nm and BPTM 70 nm [Dev01] processes respectively.

	Symbol	Current case	Future case
Die thickness (μm)	t	250	100
Die conductivity (W/K/m)	k	100	100
Die specific heat (J/K/m^3)	c	1e6	1e6
Die area (mm^2)	A_{die}	100	100
Block area (mm^2)	A_{block}	2	2
Block switching power density (W/mm^2)	PD_{act}	5	7.5
Block leakage power density (110^{o}C)	PD_{leak}	0.015	0.15
Isothermal point (^{o}C)	$T_{isothermal}$	70	70
Leakage-Temperature coefficient (1/K)	β	0.036	0.017
Channel length (nm)	L	180	70
Supply voltage (V)	V_{DD}	1.5	1.0
NMOS threshold (V)	NV_{th0}	0.269	0.120
PMOS threshold (V)	PV_{th0}	-0.228	-0.153

Table 8.1: Process technology, thermal properties and transistor data.

Because most heat is removed through the back of the die, an important parameter in the thermal properties of digital systems is die thickness. During manufacture, wafers are approximately 750 μm thick, but are then thinned to improve thermal properties. For the mature technology, we assume a 250 μm thickness, and for the future case we assume the wafer is thinned to 100 μm. We note that current smart card chips are already thinned to 100 μm thicknesses, and commercial vendors offer thinning to 50 μm.

The switching power density numbers were chosen to be aggressive for current technology, at 5 W/mm^2. Under ideal scaling theory, power density remains constant. However, we expect power density to scale higher in the future because voltages will not scale down as rapidly as feature size, and clock frequency is increasing faster than transistor speed due to deepening of pipelines. We chose 7.5 W/mm^2 as the scaled future power density.

8.2.2 Equivalent *RC* Thermal Model

Junction temperature was modeled by a simple first-order equivalent *RC* circuit (Figure 8-1) using the well-known analogy between electrical circuits and thermal models [Ska02a]. Temperature and power are represented by voltage and current respectively.

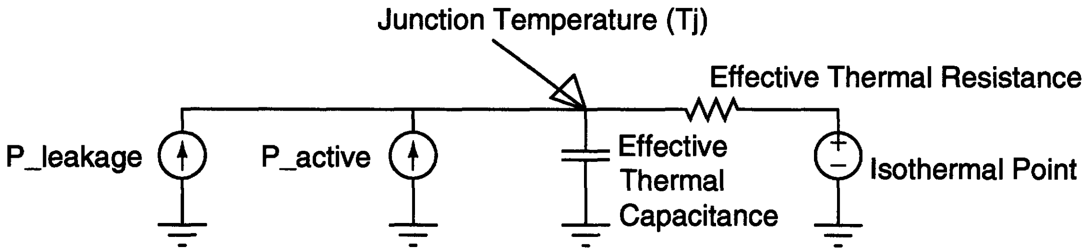

Junction Temperature (Tj)
Effective Thermal Resistance
P_leakage
P_active
Effective Thermal Capacitance
Isothermal Point

Figure 8-1: Thermal model: an equivalent *RC* circuit.

Vertical thermal resistance comes from silicon and packaging and we can assume that those two resistances are connected in series.

$$R_{silicon,vertical} = \frac{t}{k \times A_{block}} \tag{8.1}$$

$$R_{package,vertical} = \gamma \times \frac{t}{k} \times \frac{A_{tot}}{A_{block}} \tag{8.2}$$

$$R_{total,vertical} = (1 + \gamma \times A_{die}) \times \frac{t}{k \times A_{block}} \tag{8.3}$$

The formula for $R_{package}$, thermal resistance from the die to the isothermal point in the package for the specific block, is from [Bar02]. [Bar02] assumed that $R_{package}$ is proportional to the thickness of silicon die since the isothermal point in the package approximately is proportional to the die thickness. [Bar02] shows that this simple formula matches the time-consuming 3D simulation results well. (γ was measured by 3-D thermal simulation to be 120).

Most heat is dissipated vertically in a chip, particularly when wafers are thinned, since horizontal cross-section is usually much smaller than the area of a typical hotspot. Therefore, we assumed infinite lateral thermal resistance, which will lead to the worst-case temperature gradients between junctions in a chip.

Deeney [Dee02] notes that lateral conductance can be used to spread the heat of hot spot and suggests the use of blank silicon around hot spots, but these experiments were done on an unthinned 750 μm HP PA-8700 die. As die thickness drops, vertical conductivity improves and lateral conductivity increases.

The lateral resistance of the heat spreader can change our results somewhat, but because the heat spreader has a much lower resistance than silicon, its main effect is to make the heatsink act as a constant thermal source (this is where the isothermal point in the package is derived from). We have taken the worst case by assuming no lateral conductance but we don't believe this will qualitatively change our results.

Vertical thermal capacitance of die can be derived as

$$C_{silicon,vertical} = c \times t \times A_{block} \tag{8.4}$$

Thermal capacitance of the packaging was ignored, because the package capacitance is so massive that it effectively acts as a temperature source. Lateral thermal capacitance of silicon can be ignored for the same reason as that of the lateral thermal resistance case.

8.2.3 Temperature Dependency of Leakage Power

Leakage power is a significant part of total power in deep submicron technology. It has exponential dependency on temperature since the effective threshold voltage decreases linearly with the increase of temperature. Therefore, for the accurate temperature calculation, the dependency of leakage power on temperature should be included in the thermal model. The dependency of switching power on temperature is comparably weak and ignored in this work.

This exponential temperature dependency was measured by simulating an inverter chain with Hspice for the 180nm and 70nm processes and the temperature-dependency was extracted from the temperature-voltage curves (Figure 8-2). (In this work, β is defined as the thermal dependency of leakage on temperature: $P_{leakage} \propto e^{\beta \times Temp}$.) It is a well-known physical fact that β decreases as channel length shrinks, as seen in our measurements.

We modeled leakage power in our thermal model as a voltage-dependent current source using β ($P_{leak110}$ is leakage power measured at 110^oC.):

$$P_{leak} = P_{leak110} \times e^{\beta(T_j - 110)} \tag{8.5}$$

Figure 8-3 compares the simulation results for a thermal power model with constant leakage power at the worst-case temperature 110^oC and our temperature-dependent thermal model at different conditions.

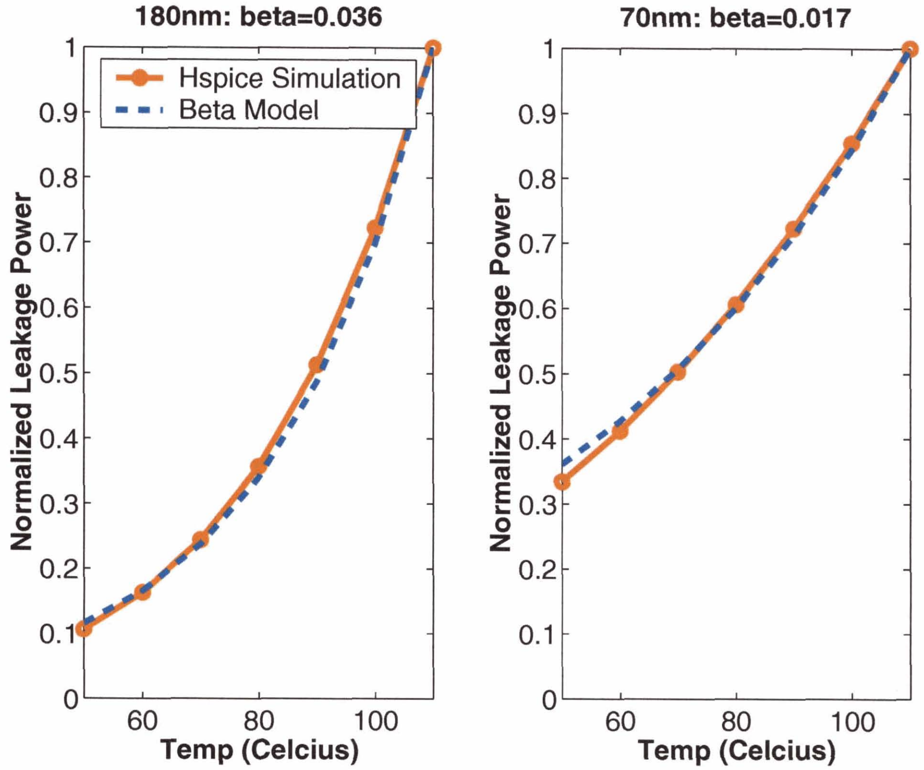


Figure 8-2: Leakage power versus temperature.

In Figure 8-3 (a), we can see that the constant leakage model overestimates the junction temperature. Figure 8-3 (b) shows the notorious thermal runaway phenomenon. The thermal runaway phenomenon happens due to exponential dependency of leakage power on temperature. Increased temperature increases leakage power and increased total power (switching power + leakage power) increases temperature. This positive feedback can bring exponential temperature increase, which means circuit failure. We found that this phenomenon only happens when certain conditions between switching power and leakage power at certain temperature and β are met. The constant leakage model cannot predict the thermal runaway phenomenon.

8.3 Activity Migration

AM can be done through software or hardware. However, we found that the OS timescale will be too slow for AM especially for the future processes. For our limited and first-cut evaluation to show the benefit of AM, we restrict our analysis to a simple hardware policy, where activity alternates between a block and the duplicated block with a fixed period. Using more than two sites would lead

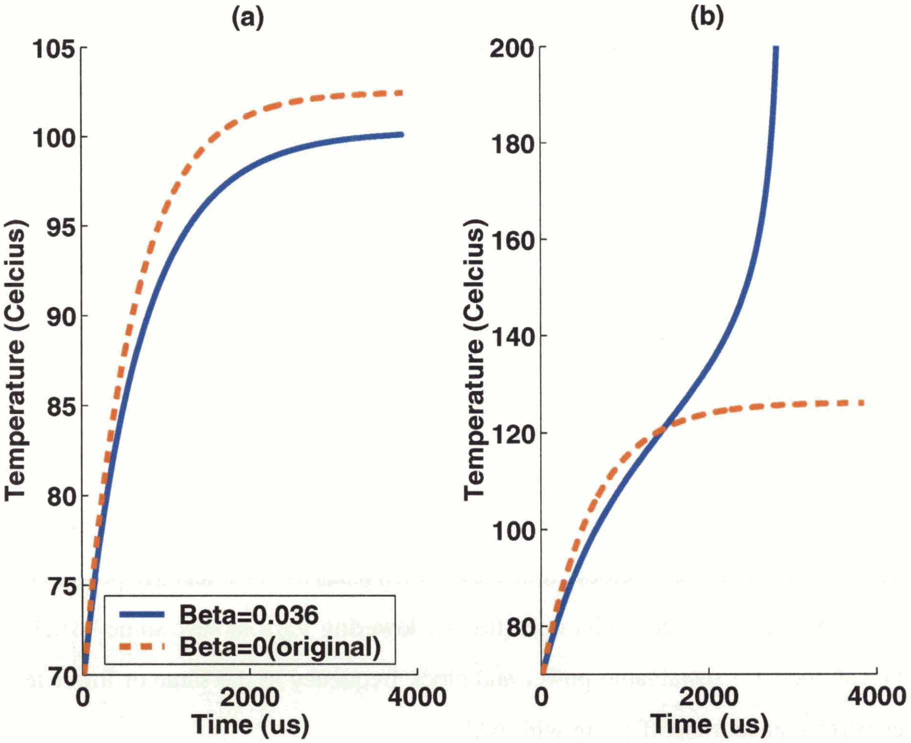


Figure 8-3: Comparison of thermal models. Simulation is based on Current case and we assume that T_j starts from T. Different switching and leakage power conditions are assumed for simulation (a) and (b).

to greater power density reduction but would require even greater area.

Figure 8-4 shows the equivalent RC circuit for our AM technique. The migration duty cycle is fixed at 50%. The lateral resistance and capacitance between the original and duplicated blocks are not modeled assuming the worst case that the two sites are spatially and thermally separated. If the two sites are next to each other, and the lateral resistance between them is comparable to the vertical resistances, then we will get lower temperature increase since the second site which is not burning power can help cool the first. In the extreme, if the lateral resistance between them is zero, the effective thermal resistance becomes half of the baseline.

By alternating between two sites, the ideal activity for each site is halved, which means half the total power density and half the temperature increase from the isothermal point. The same benefit could be achieved without AM if the computational load could be parallelized across the two sites. We are primarily interested in improving single thread performance and so do not consider parallel execution further here.

Reducing the power density could be used in two ways to increase performance or reduce en-

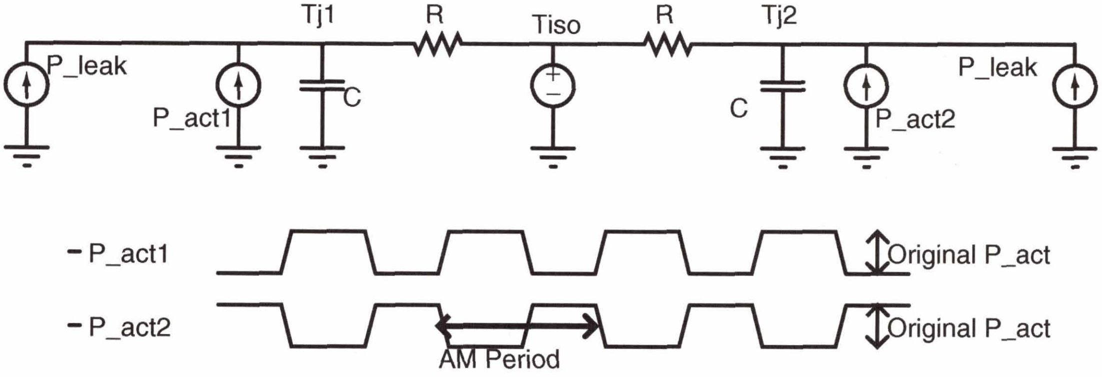


Figure 8-4: AM thermal model : an equivalent RC circuit.

ergy. Figure 8-5 shows the qualitative benefit of AM in terms of new operating points. First, we can lower die temperature at the same clock frequency, which leads to lower leakage power. We can also exploit the time slack due to the cooler operation by lowering V_{DD} to save some switching power. Second, we can increase sustainable power and clock frequency at the same or lower temperature with power-performance tradeoff along with AM.

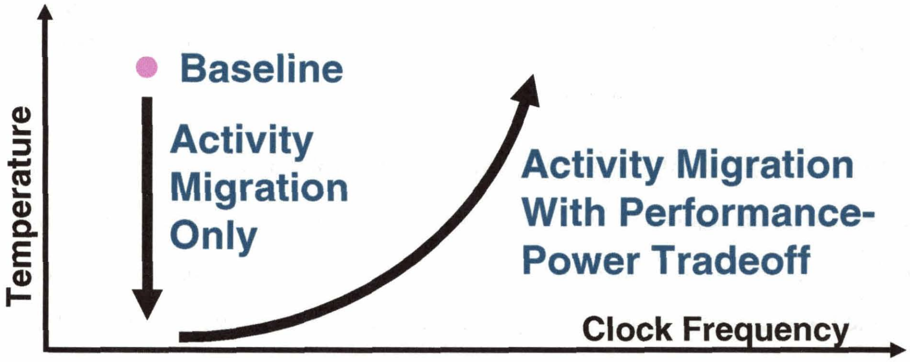


Figure 8-5: Conceptual benefits of AM: reduced temperature or increased frequency due to possibility to perform dynamic voltage scaling.

As an example of how this can be used, consider a laptop with limited heat removal capability. When plugged into a wall socket, we can use AM to burn more power to raise performance without raising die temperature. When running the system from battery power, we can use AM to lower die temperature, which lowers leakage and allows more energy-efficient execution.

We restrict ourselves to using dynamic voltage scaling (DVS) to trade off power and performance. To get the maximum benefit from the frequency-power tradeoff, supply voltage and threshold voltage should be scaled simultaneously, but for circuits with high activity factor, dynamic supply voltage scaling alone is adequate so we assumed a fixed threshold voltage. Many other power-performance tradeoff schemes such as dynamic cache configuration modification, fetch/decode throttling, or speculation control [BM01] could be used with AM, but DVS was just the simplest to evaluate and one that is likely to be supported in the system in any case.

8.3.1 Activity Migration: Analytical Model

We calculate the benefits of AM technique analytically (Figure 8-6) (Here, β is assumed as zero for the simplicity of computation.):

$$T_{high} = (T_{low} - T_{base})e^{-\frac{Period}{2\tau}} + T_{base} \tag{8.6}$$

$$T_{low} = (T_{high} - T)e^{-\frac{Period}{2\tau}} + T \tag{8.7}$$

τ is the time constant of the equivalent RC circuit. Solving the above equations, we get

$$T_{high} = \frac{T_{base} - T}{1 + e^{-\frac{Period}{2\tau}}} + T \tag{8.8}$$

The temperature increase from the isothermal point is decreased by $1 + e^{-\frac{Period}{2\tau}}$ through AM. With very small $Period$, the temperature increase will be halved. On the other hand, we can increase the power by $1 + e^{-\frac{Period}{2\tau}}$ times, which makes T_{high} the same as T_{base}. If $Period$ is considerably smaller than τ, we can double power at the same temperature as the baseline, which means increasing the clock frequency by a factor of $2^{\frac{1}{3}}$ (=1.26) with DVS since switching power is approximately proportional to Vdd^3.

8.3.2 Activity Migration: Simulation Results

This section shows the simulation results of AM alone and AM with DVS.

AM Alone

Table 8.2 and Table 8.3 show the Hspice simulation results of the AM technique for various AM periods.

AM drops temperature more than 12 and 7^oC for Current case and Future case respectively at the same clock frequency. In addition, AM saves power consumption in two ways. First, reduced temperature leads to the reduced leakage power, more than 37% and 12% for Current case

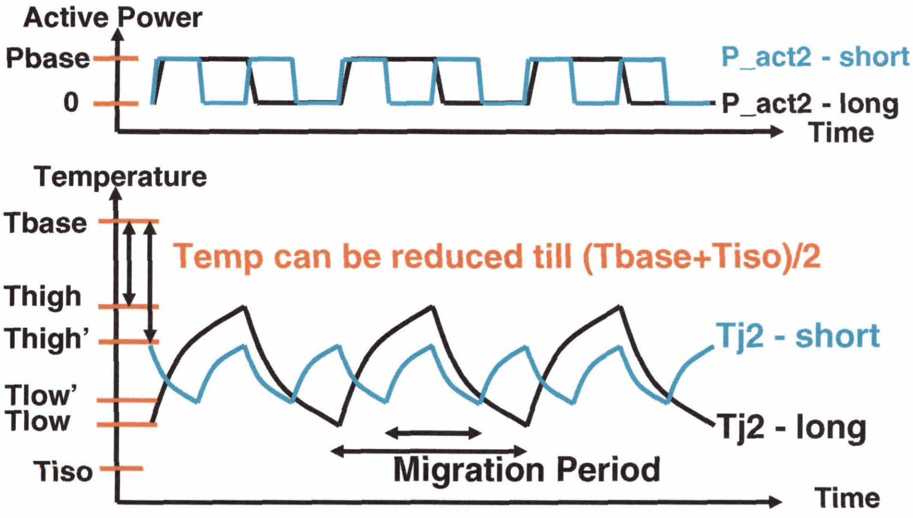


Figure 8-6: Analytical calculation of benefits of AM. T_{base} is the baseline temperature, T_{iso} is the isothermal point, T_{high} and T_{low} are the highest and lowest temperatures for AM technique, and $Period$ is AM period.

AM period (μs)	1800	600	200
Temperature drop (K)	9.2	11.5	12.4
Leakage power reduction (%)	29.6	35.3	37.6
Switching power reduction (%)	3.7	7.6	9.7

Table 8.2: Simulation measurements of AM (Current case).

and Future case respectively. Second, since the on-state drain current of MOSFETs has negative temperature dependence (even though at a very low supply voltage, it is positive [Kan01]), we gain some time slack after AM and we can exploit the time slack by scaling down V_{DD} to save switching power, which is the other way of using DVS. Almost 10% reduction was attained for both cases.

The smaller AM period gives the greater temperature drop; however, it results in a bigger cycles-per-instruction (CPI) penalty. Section 8.5 will examine the impact of short AM periods on CPI.

AM with DVS

Figure 8-8 and Figure 8-9 show the simulation results for the AM technique with dynamic voltage scaling.

DVS was modeled based on the simulation data of a 15 stage ring-oscillator. We assumed 30 times FO1 delay, the period of the ring-oscillator, as the clock period of the digital system. Switching power, leakage power, and period were measured for various V_{DD} levels in both technology

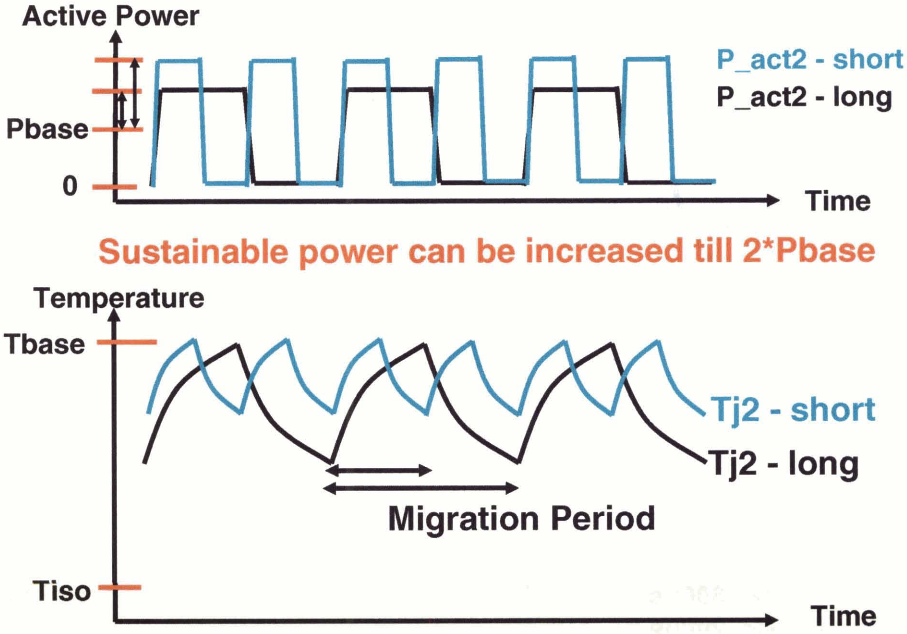


Figure 8-7: Analytical calculation of benefits of AM and a power-performance tradeoff.

AM period (μs)	600	200	60
Temperature drop (K)	3.4	6.4	7.5
Leakage power reduction (%)	5.9	10.8	12.6
Switching power reduction (%)	3.3	9.5	9.7

Table 8.3: Simulation measurements of AM (Future case).

cases. Activity factor was assumed to be as high as 0.2 since usually high activity areas cause hot spots.

DVS exploits the temperature drops by the AM and brings the frequency increase at the same or lower temperature. Table 8.4 and Table 8.5 show the clock frequency and power increase at the same temperature as the baseline through AM with DVS. Around 16% and 6% clock frequency increase could be attained.

AM period (μs)	1800	600	200
Clock frequency increase at same temperature (%)	10.5	14.1	15.9
Total power increase at same temperature (%)	56.8	79.5	90.9

Table 8.4: Summary of effects of AM(upper section) and AM+Dynamic V_{DD} scaling(lower section) (Current case).

According to our thermal model, the Future case has a time constant more than 6 times smaller

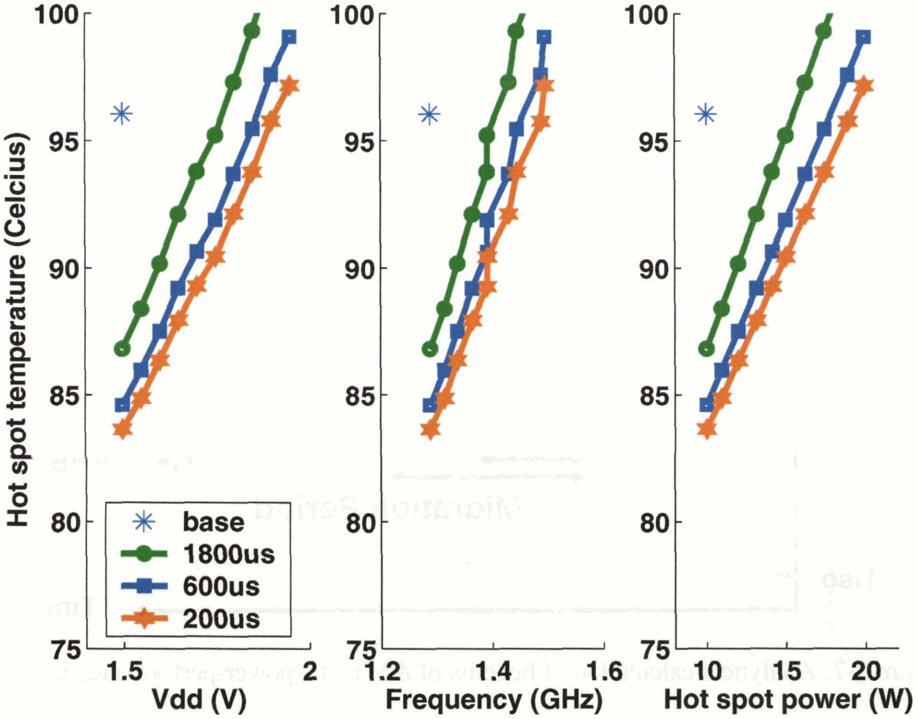


Figure 8-8: Simulation results of AM and DVS for various AM periods (Current case)

AM period (μs)	600	200	60
Clock frequency increase at same temperature (%)	2.3	5.0	5.9
Total power increase at same temperature (%)	25.0	61.4	79.6

Table 8.5: Summary of effects of AM(upper section) and AM+Dynamic V_{DD} scaling(lower section) (Future case).

than that of the Current case since RC is proportional to the square of die thickness, and therefore it requires a faster migration period than the Current case. For example, a 600 μs period does not work well for the Future case though it does for the Current case. Also, it is clear that the typical OS timescale, 1 to 10 μs, is too large to control migration efficiently.

8.4 AM Architectures

If we perform AM between multiple cores on a die, then long AM periods would be easy to handle in software as part of a regularly scheduled OS context swap, but Table 8.4 and Table 8.5 show that small AM periods are necessary to realize performance gains. To investigate the impact of fine-grain AM, we used the SimpleScalar simulator version 3.0b [BA97] to model AM between two

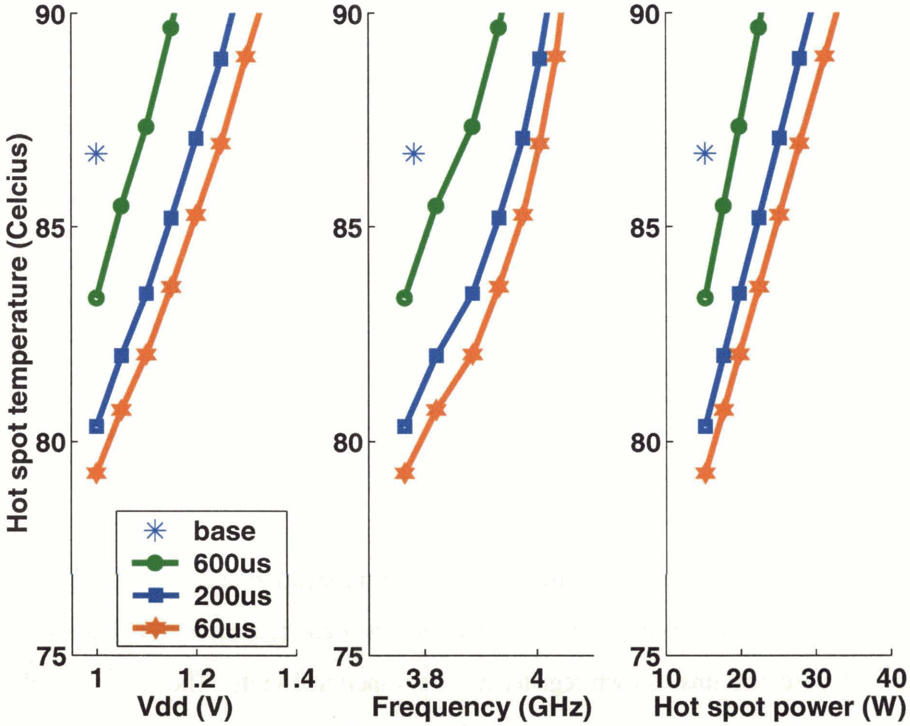


Figure 8-9: Simulation results of AM and DVS for various AM periods (Future case)

four-wide, out-of-order, superscalar processor cores.

We modeled the five different architectural configurations shown in Figure 8-10, using two different cache sizes: a single-cycle 8 KB cache and a two-cycle 32 KB cache. The size of each block in the figure roughly corresponds to the size of the same structure on the Alpha 21264 as measured from a published floorplan [Gie97]. The Simplescalar simulation parameters are given in Table 8.6.

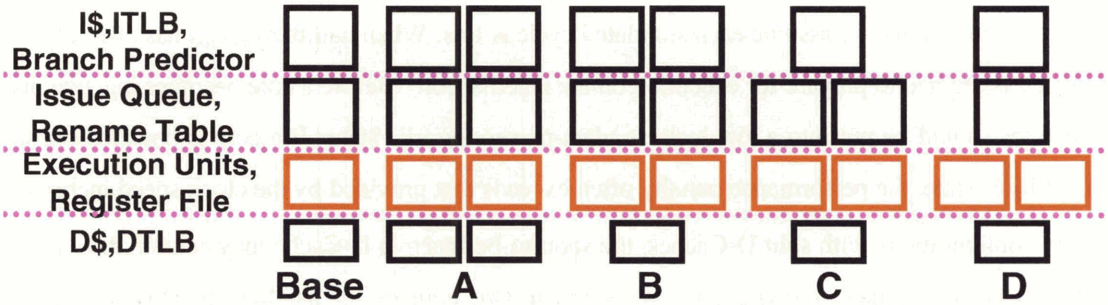


Figure 8-10: AM processor configurations.

AM implies the use of additional area to hold the replicated units, and so the five configurations

Pipeline width	4
# of RUUs	64
Load/Store queue depth	64
# of integer ALUs (multiplier/divider)	5 (1)
# of floating point ALUs (multiplier/divider)	3 (2)
# of load/store units	2
Instruction length	32 bits
L2 (64B blocks)	2MB/7cycles
L1 I$/D$ (64B blocks)	8KB/2way/1cycle or 32KB/4way/2cycles
Branch prediction	1024 entries
Misprediction latency	3 cycles
Memory latency	first 100 cycles, Next 2 cycles

Table 8.6: Simulated processor configuration

represent different design points with different degrees of replication. The baseline configuration is a single complete processor core and primary caches. Each configuration A–C has split execution cores, and each core contains its own register file and functional units. Though a structure may be shared by two execution cores, its size remains the same. As only one core can access it at a time, no additional ports are needed. We pessimistically add an extra cycle to the cache access times to account for driving a longer bus between two cores. Machine A contains duplicates of all structures in the baseline machine. Machine B consists of two cores sharing the same data cache (D-Cache) while Machine C cores share both instruction cache (I-Cache) and D-Cache. Machine D is similar to the baseline except that it shares an issue queue and rename table so the pipe need not be drained upon thermal interrupts. In this configuration, only the contents of the physical register file need to be copied, for which a 32 cycle penalty is assessed every T_{period} (16 cycles of pipeline drain plus 16 cycle copying four values per cycle).

To simulate AM, we assume each simulated cycle is 1 ns. When half the T_{period} has elapsed, the pipeline is drained to prepare for execution on the inactive core. Before a core becomes inactive, its structures should be put into a low-leakage, data-preserving idle state. If a core is powered-off so that it loses state, the performance penalty often exceeds that provided by the clock speed increase.

In configurations with split D-Caches, the soon-to-be-inactive D-Cache may contain dirty lines which need to be taken into account. We simulate two policies for handling this condition. In the *naive* policy, all dirty lines are written back before execution resumes. We assume this takes a constant time of 2 cycles per line as the duplicate cores share the same inclusive level 2 cache. Alternatively, a symmetric multiprocessing (SMP) protocol can be used to transfer lines on demand.

The inactive cache participates in the cache coherence protocol to move lines over.

SimPoints [SPHC02] were used to fast forward past benchmark initialization phase and choose a representative 100 million-cycle section of each program. To further reduce simulation time, a subset of the SPEC2000 benchmarks [Sta00] was chosen to find the upper and lower bound on performance change. The integer benchmarks (`vpr-route`, `mcf`, `perkbmk-perfect`, and `gap`) include those with the highest and lowest miss rates for reasonably-sized L1 I-Cache and L1 D-Cache as reported by [SC00]. All the SPEC floating point benchmarks tend to have low instruction cache miss rates; the floating point benchmarks `art` and `lucas` are two which reach their SimPoint relatively quickly and have contrasting D-Cache miss rates. Details of each benchmark's performance on our baseline machine are noted in Table 8.7. The benchmarks were compiled with optimization on an Alpha 21264 with Digital's C compiler (V5.9-008) and Compaq Fortran (V5.3-915) [Wea].

	8KB Caches			32KB Caches		
	CPI	L1 I$ miss rate	L1 D$ miss rate	CPI	L1 I$ miss rate	L1 D$ miss rate
art-110	1.04	0.00	0.33	1.04	0.00	0.32
lucas	1.08	0.00	0.11	1.15	0.00	0.10
gap	0.51	0.01	0.01	0.47	0.00	0.00
p-p	0.58	0.01	0.04	0.55	0.00	0.00
mcf	3.51	0.00	0.22	3.52	0.00	0.21
vpr-route	1.10	0.00	0.11	1.08	0.00	0.06

Table 8.7: Benchmark characteristics. (p-p:perlbmk-perfect)

8.5 Results and Discussion

AM allows one to remove hotspots and cool the die. Alternatively, AM allows the increased sustainable power to be supplied to the chip permitting dynamic voltage scaling and corresponding increase in clock frequency. However, the clock frequency gain is offset by the architectural migration overhead. Depending on the particular AM configuration, this overhead comes in the form of register file copies, additional pipeline drains, new sets of cache write-backs, or stale cache, TLB, and branch predictor data.

A long AM period favors performance as there are fewer write-backs and pipeline drains. It is unnecessary to simulate performance with an AM period greater than 200,000 cycles as any overhead is amortized over the large number of cycles. A short AM period favors thermal considerations, but increases the frequency of overhead-causing events.

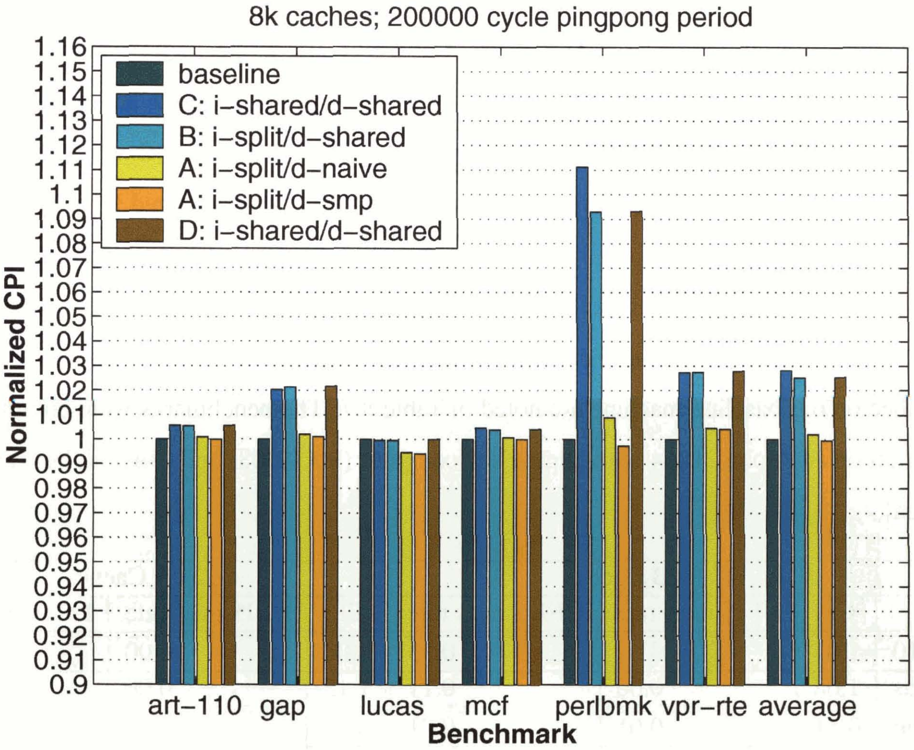


Figure 8-11: CPI with 8KB Caches (1 Cycle Hit Latency)

Figures 8-11 and 8-12 show the architectural performance effect of AM on the two simulated machines. The policies referred to in Section 8.4 are denoted by shaded bars and labeled in the format "Machine: I-Cache/D-Cache" (for example, A: i-split/d-naive describes the split I-Cache of machine A with the *naive* write-back of the entire data cache upon migration). Shorter periods were considered (not shown) but have very similar performance characteristics. Architectural performance is measured in CPI normalized to an architecture without AM.

Although we model a fixed interval, in practice, migration will likely be triggered by thermal sensors. This should alleviate any problems caused by an application having execution phases whose period matches that of AM.

The main penalty of AM technique is area increase. Table 8.8 and Table 8.9 show the tradeoff between area and performance with AM and DVS. Net performance is represented by $\frac{clock\ frequency}{CPI}$.

In summary, for the Current case, configuration D is the best choice. With only 18% area increase, we can increase performance by 14% and maintain the same temperature. Alternatively, we can cool down the hot spots significantly and save some power at slightly reduced performance

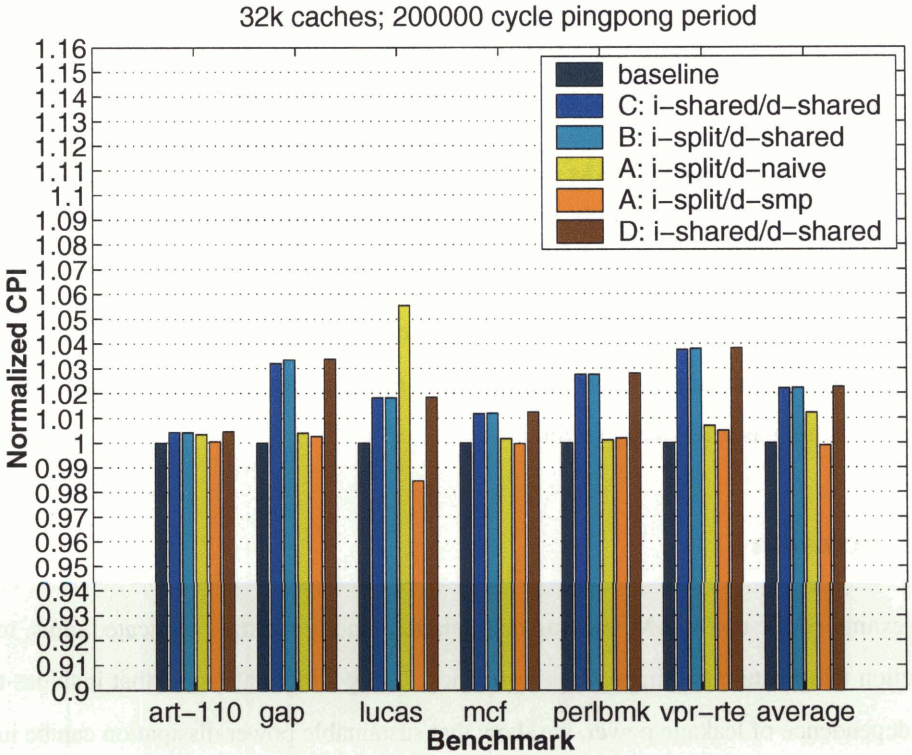


Figure 8-12: CPI with 32KB Caches (2 Cycles Hit Latency)

Configurations	A	B	C	D
Normalized area	2.00	1.76	1.41	1.18
Normalized net performance	1.16	1.16	1.14	1.14

Table 8.8: Current case: Effects of AM for area and performance normalized to baseline. 200,000 cycle period for CPI simulation (Figure 8-11 and 8-12) and 200 μs period for clock frequency (Table 8.2) were assumed considering clock frequency is around 1GHz.

due to CPI increase. As for the Future case, configuration D is again the best choice. However, the performance gain by AM and dynamic voltage scaling is small, so we might prefer to use AM only to cool down hot spots.

Another way of trading chip area for performance is increasing the size of caches. However, the IPC gain from a small increase in cache size is relatively tiny if the baseline cache size is already chosen for near-optimal performance. Furthermore, increasing the cache size can lengthen the clock period.

The extra power caused by the increased wire length can be another important penalty of AM. For example, for configuration B, the wire between D-Cache and an execution unit is longer than that

Configurations	A	B	C	D
Normalized Area	2.00	1.76	1.41	1.18
Normalized Net Performance	1.06	1.04	1.04	1.04

Table 8.9: Future Case: Effects of AM for area and performance normalized to baseline. 200,000 cycle period for CPI simulation (Figure 8-11 and 8-12) and 60 μs period for clock frequency (Table 8.3) were assumed considering clock frequency is around 3.3GHz.

of the baseline. However, we found that the extra power is negligible, in fact, less than 50 mW even under the worst-case condition: 4 mm increase of 64 top-layer metal wires, as high as 300 fF/mm wire cap, and as high as 0.2 activity factor.

8.6 Conclusion

We have examined the use of AM, moving computation among multiple replicated units, to reduce peak junction temperature and improve performance. Using a thermal model that includes the temperature dependence of leakage power, we show that sustainable power dissipation can be increased by nearly a factor of two for a given junction temperature limit. This increased power capacity can permit DVS to achieve nearly a 16% clock frequency increase at a 180 nm technology. Cycle loss to support AM causes only a 2% CPI increase on average. Net performance gain is 14%. We estimate the minimum area increase necessary for such a change to be 18% to accommodate an additional register file and execution units. Alternatively, peak die temperature can be reduced by 12.4 °C at reduced switching and leakage power dissipation. The model predicts that migration intervals of around 20–200 μs are required to achieve the maximum sustainable power increase.

Chapter 9

Conclusions and Future Work

9.1 Summary of Contributions

The main contributions presented in this thesis are:

- Optimal digital system design principles. We claim that there are two basic principles for digital system optimization in deep submicron technology: energy waste reduction and energy-delay tradeoff.

- Categorizations of circuit- and architectural-level innovations from the perspective of energy waste reduction and energy-delay tradeoff. We find that the circuit innovations either tune transistors' dimensions, structures, or voltage levels and the architectural innovations are based on three critical techniques (exploiting parallelism, utilizing predictability, and reducing energy waste).

- Categorization of leakage reduction techniques. We divide previous approaches to reducing leakage power into two categories: static and dynamic techniques. We find that critical paths' leakage dominates after static techniques, which trade delay for lower leakage are applied to non-critical paths. Dynamic techniques, which dynamically deactivate fast transistors, are necessary for effective leakage reduction.

- Fine-grain dynamic leakage reduction. We find that leakage energy waste is the most critical target for energy waste reduction and claim that fine-grain dynamic leakage reduction (FG-DLR), turning off small pieces for short idle intervals, is the key for successful leakage reduction. We also find that primary SRAM arrays, fast functional logics, and complex logic arrays are candidate functional units for FG-DLR and that transition energy and delay and break-even points are important parameters when we compare FG-DLR techniques.

- *Leakage Biasing*. We introduce a FG-DLR technique, *Leakage Biasing* (LB) which uses leakage currents themselves to bias the circuit into the minimum leakage state and has low transition overheads. We successfully apply LB to the bitline leakage reduction in primary SRAM arrays (*Leakage-Biased Bitlines*) with three microarchitectural deactivation policies and to the domino logic, presenting *Leakage-Biased Domino*.

- *Dynamic Resizing*. We introduce an FG-DLR technique, *Dynamically Resizing* (DR) which dynamically downsizes transistors on idle paths in a circuit. We apply DR to static CMOS logics and present *Dynamically Resizable CMOS* (DRCMOS) logic for FG-DLR.

- Energy reduction through pipelining. We show that energy reduction for high-performance ASIC systems can be achieved with the same computation throughput and communication bandwidth by first pipelining logic gates and global wires and then employing other energy-delay tradeoffs. We find that power reductions from pipelining datapaths are eventually limited by latch energy overheads.

- Power analysis of on-chip networks. We show that structuring global wires into a network provides a better environment for global wire optimization (e.g. pipelining). We find that the energy-efficiency improvement from a dynamically packet-routed network is bounded by router energy overheads.

- *Activity Migration*. We present a power density reduction technique, *activity migration* (AM), for hot spot removal. With AM, we spread heat by transporting computation to different locations on the die. We show how AM can be used either to increase the power that can be dissipated by a given package, or to lower the operating temperature and hence the operating energy.

9.2 Future Work

The future work that leads on from this work is summarized in three sections: energy waste reduction (Section 9.2.1), energy-delay tradeoff (Section 9.2.2), and overcoming thermal limits (Section 9.2.3).

9.2.1 Energy Waste Reduction

Effects of Process Variation on Leakage Reduction

Process variation is an important non-scalability in deep submicron technology and is getting worse as technology advances. It leads to large leakage spread as well as delay spread. Leakage reduction techniques that consider the possible yield loss due to the process variation will be indispensable [Aga05].

Gate and BTBT Leakage Reduction

The low oxide thickness gives rise to high electric field, resulting in significant gate oxide tunneling current. On the other hand, higher doping leads to high electric field across source and drain junctions, resulting in significant junction BTBT current. In the near future, the gate and junction BTBT leakages are expected to grow to be comparable to subthreshold leakage [Aga05]. Therefore, leakage reduction techniques considering all these components will be indispensable.

9.2.2 Energy-Delay Tradeoff

Measuring Energy-Delay Trading Curves of Existing and New Innovations

The energy-delay tradeoffs of many innovations, particularly architectural innovations, are not well studied yet. Theoretical analysis and measurements through simulating with prototype systems are necessary to obtain the trading slopes, overheads, effective ranges, and operating regimes of existing and new innovations. The obtained curves will be used as building blocks for optimal digital system design.

Optimal-Pipelining for Subthreshold CMOS Circuits

Many applications including medical and wireless applications, require ultra low power dissipation with low-to-moderate performance (10kHz-100MHz), where subthreshold CMOS circuits are found the most helpful [Wan02a]. The general concepts of optimal pipelining in this thesis can be easily applied. The sensitivity to process variation is a substantial pipelining overhead for the subthreshold CMOS logic in addition to the stage latch overhead [Zha05].

Effects of Wire Parameter Variation on Wire Optimization

Wire geometry and material parameter variations and their impacts on long wires' energy and delay are increasing rapidly as feature size is shrinking. Wire optimization without considering the effects of wire parameter variations only results in a suboptimal design.

Low-Power Router Circuit and Architecture

Though packet-routed on-chip networks have many advantages such as scalability and small area over wire-routed circuits, the fast-growing router power overhead is the major drawback (Chapter 6). Low-latency and low-power router design is necessary for optimal on-chip packet-routed network design.

9.2.3 Overcoming Thermal Limit

Activity Migration with Help of Thermal Sensors and OS

For a better utilization of activity migration (AM), the co-operation of temperature sensors and operating system is crucial. The sensors are responsible for identifying the troubled hot spots quickly and accurately, and the operating system is responsible for migrating computation with minimum power and performance overheads. AM can be combined with dynamic leakage reduction at the core-level in a multi-cored architecture and can provide a simple but effective solution for a unified temperature and leakage management.

Bibliography

[AAE00] M. W. Allam, M. H. Anis, and M. I. Elmasry. High-speed dynamic logic styles for scaled-down CMOS and MTCMOS technologies. In *ISLPED*, pages 155–160, 2000.

[ACG03] J. Abella, R. Canal, and A. Gonzalez. Power- and complexity-aware issue queue designs. *IEEE Micro*, 23:50–58, Sept/Oct 2003.

[Aga05] A. Agarwal et al. Effectiveness of low power dual-Vt designs in nano-scale technologies under process parameter variations. In *ISLPED*, 2005.

[And01] M. Anders et al. Robustness of sub-70nm dynamic circuits: analytical techniques and scaling trends. In *Symp. on VLSI Circuits*, pages 23–24, 2001.

[And03] H. Ando et al. A 1.3-GHz fifth-generation SPARC64 microprocessor. *IEEE JSSC*, 38(11):1896–1905, November 2003.

[Asa98] K. Asanović. *Vector Microprocessors*. PhD thesis, University of California, Berkeley, 1998.

[BA97] D.C. Burger and T.M. Austin. The simplescalar tool set, version 2.0. Technical Report CS-TR-97-1342, University of Wisconsin, Madison, June 1997.

[Bak90] H. B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI*. Addision-Wesley, 1990.

[Bar02] M. Barcella et al. Architecture-level compact thermal R-C modeling. Technical Report CS-2002-20, Univ. of Virginia Comp. Sci. Dept., Jul. 2002.

[BDH03] L.A. Barroso, J. Dean, and U. Holzle. Web search for a planet: The Google cluster architecture. *IEEE Micro*, 23:22–28, Mar/Apr 2003.

[BM01] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *HPCA*, Jan. 2001.

[BM02] K. Banerjee and A. Mehrotra. Power dissipation issues in interconnect performance optimization for sub-180-nm designs. In *Symposium on VLSI circuits*, pages 12–15, June 2002.

[BN71] C. G. Bell and A. Newell. *Computer Structures: Readings and Examples*. McGraw-Hill, 1971.

[Bro00] D. Brooks et al. Wattch: A framework for architectural-level power analysis and optimizations. In *ISCA*, pages 83–94, 2000.

[BS00] J. A. Butts and G. S. Sohi. A static power model for architects. In *MICRO-33*, pages 191–201, December 2000.

[Cal04] B. Calhoun et al. A leakage reduction methodology for distributed MTCMOS. *IEEE JSSC*, 39(5):818–826, May 2004.

[CBF00] A. Chandrakasan, W. J. Bowhill, and F. Fox. *Design of High Performance Microprocessor Circuits*. IEEE Press, 2000.

[Cha92] A. Chandrakasan et al. Low-power CMOS digital design. *IEEE JSSC*, 27(4):473–484, Apr. 1992.

[Coc02] P. Cocchini. Concurrent flip-flop and repeater insertion for high performance integrated circuits. In *ICCAD*, pages 268–273, Nov 2002.

[CP03] X. Chen and L. Peh. Leakage power modeling and optimization in interconnection networks. In *ISLPED*, pages 90–95, 2003.

[CS00] P. Christie and D. Stroobandt. The interpretation and application of Rent's rule. *IEEE TVLSI*, 8(6):639–648, Dec. 200.

[DB99] V. De and S. Borkar. Technology and design challenges for low power and high performance. In *ISLPED*, pages 163–168, 1999.

[Dee02] J. Deeney. Thermal modeling and measurement of large high-power silicon devices with asymmetric power distribution. In *Int'l Symposium on Microelectronics*, Sep. 2002.

[Dev01] Device Group at UC Berkeley. Predictive technology model. Technical report, PTM, 2001.

[Dho00] A. Dhodapkar et al. TEMPEST: A Thermal Enabled Multi-model Power/performance ESTimator. In *HPCA*, Nov. 2000.

[Dob96] D. Dobberpuhl. The design of a high performance low power microprocessor. In *ISLPED*, pages 11–16, 1996.

[DT01] W. Dally and B. Towles. Route packets, not wires: On-chip interconection networks. In *DAC*, pages 684–689, 2001.

[EP04] N. Eisley and L. Peh. High-level power analysis for on-chip networks. In *CASES*, Sept. 2004.

[Esp02] R. Espasa et al. Tarantula: A vector extension to the Alpha architecture. In *ISCA 29*, pages 281–292, May 2002.

[Fla02] K. Flautner et al. Drowsy caches: Simple techniques for reducing leakage power. In *ISCA 29*, pages 148–157, June 2002.

[Gei02] S. Geissler et al. A low-power RISC microprocessor using dual PLLs in a 0.13 μm SOI technology with copper interconnect and low-k BEOL dielectric. In *ISSCC*, February 2002.

[Gie97] B. A. Gieseke et al. A 600-MHz superscalar RISC microprocessor with out-of-order execution. *ISSCC*, pages 176–177, Feb. 1997.

[Gro96] P. E. Gronowski et al. A 433-MHz 64-b quad-issue RISC microprocessor. *IEEE JSSC*, 31(11):1687–1696, November 1996.

[Gun01] S. Gunther et al. Managing the impact of increasing microprocessor power consumption. *Intel Journal*, 2001.

[Gup03] P. Gupta et al. A high-level interconnect power model for design space exploration. In *ICCAD*, pages 551–558, Nov 2003.

[HA02] S. Heo and K. Asanović. Leakage-biased domino circuits for dynamic fine-grain leakage reduction. In *Symp. on VLSI Circuits*, pages 316–319, 2002.

[HA04a] S. Heo and K. Asanović. Dynamically resizable static CMOS logic for fine-grain leakage reduction. Technical report, MIT LCS Technical Report, 2004.

[HA04b] S. Heo and K. Asanović. Power-optimal pipelining in deep submicron technology. In *ISLPED*, pages 218–223, 2004.

[HA05] S. Heo and K. Asanović. Replacing global wires with an on-chip network: A power analysis. In *ISLPED*, 2005.

[Ham00] F. Hamzaoglu et al. Dual-vt SRAM cells with full-swing single-ended bit line sensing for high-performance on-chip cache in 0.13 μm technology generation. In *ISLPED*, pages 15 – 19, 2000.

[HBA03] S. Heo, K. Barr, and K. Asanović. Reducing power density through activity migration. In *ISLPED*, pages 217–222, 2003.

[HBHA02] S. Heo, K. Barr, M. Hampton, and K. Asanović. Dynamic fine-grain leakage reduction using leakage-biased bitlines. In *ISCA*, pages 137–147, 2002.

[Hin01a] G. Hinton et al. The microarchitecture of the Pentium 4 processor. *Intel Journal*, 2001.

[Hin01b] G. Hinton et al. A 0.18 μm CMOS IA-32 processor with a 4-GHz integer execution unit. *IEEE JSSC*, 36(11):1617–1627, Nov. 2001.

[HKA01] S. Heo, R. Krashinsky, and K. Asanović. Activity-sensitive flip-flop and latch selection for reduced energy. In *19th Conference on Advanced Research in VLSI*, Salt Lake City, UT USA, March 2001.

[HN97] J. P. Halter and F. Najm. A gate-level leakage power reduction method for ultra-low-power CMOS circuits. In *CICC*, pages 457–478, 1997.

[Ho 01] R. Ho et al. The future of wires. *Proceedings of the IEEE*, 89(4):490–504, Apr. 2001.

[HP96] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 1996.

[HP02] A. Hartstein and T. Puzak. The optimum pipeline depth for a microprocessor. In *ISCA 29*, pages 7–13, May 2002.

[HP03] A. Hartstein and T. Puzak. Optimum power/performance pipeline depth. In *MICRO*, Dec. 2003.

[Hri02] M. Hrishikesh et al. The optimal logic depth per pipeline stage is 6 to 8 FO4 inverter delays. In *ISCA 29*, pages 14–24, May 2002.

[Hua00] W. Huang et al. A framework for dynamic energy efficiency and temperature management. In *MICRO*, Dec. 2000.

[Int00] International Technology Roadmap for Semiconductors. 2000 update, process integration, devices, and structures. Technical report, ITRS, 2000.

[Int02] Intel Corp. Intel to introduce new technologies to reduce power consumption of MPUs, Aug. 2002. http://www.esi-online.com.sg/news/view/default.asp?newId=10.

[Int04] International Technology Roadmap for Semiconductors. 2004 update. Technical report, ITRS, 2004.

[Inu00] T. Inukai. Boosted Gate MOS (BGMOS): Device/circuit cooperation scheme to achieve leakage-free giga-scale integration. In *CICC*, pages 409–412, 2000.

[JSCR02] M. C. Johnson, D. Somasekhar, L. Chiou, and K. Roy. Leakage control with efficient use of transistor stacks in single threshold CMOS. In *IEEE Transactions on VLSI Systems*, February 2002.

[Kan01] K. Kanda et al. Design impact of positive temperature dependence on drain current in sub-1-V CMOS VLSIs. *IEEE JSSC*, 36(10):1559–1564, Oct. 2001.

[Kap02] P. Kapur et al. Power estimation in global interconnects and its reduction using a novel repeater optimization methodology. In *DAC*, pages 461–466, 2002.

[KC00] J. T. Kao and A. P. Chandrakasan. Dual-threshold voltage techniques for low-power digital circuits. *IEEE JSSC*, 35(7):1009–1018, July 2000.

[Kes01} A. Keshavarzi et al. Effectiveness of reverse body bias for leakage control in scaled dual Vt CMOS ICs. In *ISLPED*, pages 207–212, August 2001.

[Kha01] B. Khailany et al. Imagine: Media processing with streams. *IEEE Micro*, 21:35–46, Mar/Apr 2001.

[KHM01] S. Kaxiras, Z. Hu, and M. Martonosi. Cache decay: exploiting generational behavior to reduce cache leakage power. In *ISCA 28*, pages 240–251, May 2001.

[Kos01] S. V. Kosonocky et al. Enhanced multi-threshold (MTCMOS) circuits using variable well bias. In *ISLPED*, pages 165–169, August 2001.

[KS86] S. R. Kunkel and J. E. Smith. Optimal pipelining in supercomputers. In *Proceedings 13th Symposium on Computer Architecture*, pages 404–414, Tokyo, Japan, June 1986.

[Kur96] T. Kuroda et al. A 0.9-V, 150-MHz, 10-mW, 4 mm^2, 2-D discrete cosine transform core processor with variable threshold-voltage (VT) scheme. *IEEE JSSC*, 31(11):1770–1779, November 1996.

[Kur98] T. Kuroda et al. Variable supply-voltage scheme for low-power high-speed CMOS digital design. *IEEE JSSC*, 33(3):454–462, March 1998.

[Lee97] W. Lee et al. A 1-V programmable DSP for wireless communications. *IEEE JSSC*, 32(11):1766–1776, November 1997.

[LH03] W. Liao and L. He. Full-chip interconnect power estimation and simulation considering concurrent repeater and flip-flop insertion. In *ICCAD*, pages 574–580, Nov 2003.

[Lim02] C. Lim et al. A thermal-aware superscalar microprocessor. In *ISQED*, pages 517 – 522, Mar. 2002.

[LS96] M. H. Lipasti and J. P. Shen. Exceeding the dataflow limit via value prediction. In *MICRO-29*, pages 226–237, Dec. 1996.

[Mak98] H. Makino et al. An auto-backgate-controlled MT-CMOS circuit. In *Symp. on VLSI Circuits*, pages 42–43, 1998.

[Mar01] D. Marr et al. Hyper-threading technology architecture and microarchitecture. *Intel Journal*, 2001.

[Mar04] D. Markovic et al. Methods for true energy-performance optimization. *IEEE JSSC*, 39(8):1282–1293, Nov. 2004.

[Mat01] S. K. Mathew et al. Sub-500-ps 64-b ALUs in 0.18 μm SOI/bulk CMOS: Design and scaling trends. *IEEE JSSC*, 36(11):1636–1646, November 2001.

[McP00] T. McPherson et al. 760 MHz G6 S/390 microprocessor exploiting multiple Vt and copper interconnects. In *ISSCC*, pages 96–97, 2000.

[Miy00] M. Miyazaki et al. A 1000-MIPS/W microprocessor using speed-adaptive threshold-voltage CMOS with forward bias. In *ISSCC Digest*, pages 420–421, 2000.

[Mon96] J. Montanaro et al. A 160-MHz, 32-b, 0.5-W CMOS RISC microprocessor. *IEEE JSSC*, 31(11):1703–1714, November 1996.

[Mul04] R. Mullins et al. Low-latency virtual-channel routers for on-chip networks. In *ISCA 31*, pages 188–197, June 2004.

[Mut95] S. Mutoh et al. 1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS. *IEEE JSSC*, 30(8):847–854, August 1995.

[Nar01] S. Narendra et al. Scaling of stack effect and its application for leakage reduction. In *ISLPED*, pages 195–200, August 2001.

[Nar02] S. Narendra et al. 1.1V 1GHz communications router with on-chip body bias in 150nm CMOS. In *ISSCC*, pages 270–271, 2002.

[Nar03] S. Narendra et al. Forward body bias for microprocessors in 130-nm technology generation and beyond. *IEEE JSSC*, 38(5):696–701, May 2003.

[NS00] K. Nose and T. Sakurai. Optimization of Vdd and Vth for low-power and high-speed applications. In *Asia and South Pacific DAC*, Jan. 2000.

[Pal97] S. Palacharla et al. Complexity-effective superscalar processors. In *ISCA*, pages 206–218, 1997.

[Pie96] R. F. Pierret. *Semiconductor Device Fundamentals*. Addison Wesley, 1996.

[Pow00] M. Powell et al. Gated Vdd: A circuit technique to reduce leakage in deep-submicron cache memories. In *ISLPED*, 2000.

[Pre02] R. P. Preston et al. Design of an 8-wide superscalar RISC microprocessor with simultaneous multithreading. In *ISSCC Digest and Visuals Supplement*, February 2002.

[Rab96] J. M. Rabaey. *Digital Integrated Circuits: A Design Perspective*. Prentice Hall, 1996.

[San97] H. Sanchez et al. Thermal management system for high performance PowerPC microprocessors. In *IEEE Computer Society Int'l Conference*, Feb. 1997.

[San03] K. Sankaralingam et al. Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture. In *ISCA 30*, pages 422–433, June 2003.

[SC00] S. Sair and M. Charney. Memory behavior of the SPEC2000 benchmark suite. Technical Report RC 21852, IBM, Oct. 2000.

[SC02] E. Sprangle and D. Carmean. Increasing processor performance by implementing deeper pipelines. In *ISCA 29*, pages 25–36, May 2002.

[Set95] K. Seta et al. 50% active-power saving without speed degradation using standby power reduction (SPR) circuit. In *ISSCC*, pages 318–319, 1995.

[SG83] J. Smith and J. Goodman. A study of instruction cache organizations and replacement policies. In *ISCA 10*, pages 132–137, 1983.

[Sgr01] M. Sgroi et al. Addressing the system-on-a-chip interconnect woes through communication-based design. In *DAC*, 2001.

[Shi97] S. Shigematsu et al. A 1-V high-speed MTCMOS circuit scheme for power-down application circuits. *IEEE JSSC*, 32(6):861–869, June 1997.

[Ska02a] K. Skadron et al. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management. In *HPCA*, Feb. 2002.

[Ska02b] K. Skadron et al. HotSpot: Techniques for modeling thermal effects at the processor-architecture level. In *Int'l Workshop on Thermal Investigations of ICs and Systems*, Oct. 2002.

[Ska03] K. Skadron et al. Temperature-aware microarchitecture. In *ISCA 30*, pages 2–13, June 2003.

[SPHC02] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *ASPLOS*, Oct. 2002.

[Sri02] V. Srinivasan et al. Optimizing pipelines for power and performance. In *MICRO*, Nov. 2002.

[SSH99] I. Sutherland, R. F. Sproull, and D. Harris. *Logical Effort: Designing Fast CMOS Circuits.* Morgan Kaufmann, 1999.

[Sta00] Standard Performance Evaluation Corporation. CPU2000, 2000.

[Sto95] V. Stojanovic et al. Energy-delay tradeoffs in combinational logic using gate sizing and supply voltage optimization. In *ISSCC*, pages 318–319, 1995.

[TA00] J. Tseng and K. Asanović. Energy-efficient register access. In *Proc. of the 13th Symposium on Integrated Circuits and Systems Design*, Manaus, Brazil, September 2000.

[TA03] J. H. Tseng and K. Asanović. Banked multiported register files for high-frequency superscalar microprocessors. In *ISCA 30*, June 2003.

[Tak98] M. Takahasi et al. A 60-mW MPEG4 video codec using clustered voltage scaling with variable supply-voltage scheme. *IEEE JSSC*, 33(11):1772–1778, November 1998.

[Tsc03] J. W. Tschanz et al. Dynamic sleep transistor and body bias for active leakage power control of microprocessors. *IEEE JSSC*, 38(11):1838–1845, November 2003.

[Usa98] K. Usami et al. Automated low-power technique exploiting multiple supply voltages applied to a media processor. *IEEE JSSC*, 33(3):463–471, March 1998.

[VZA00] L. Villa, M. Zhang, and K. Asanović. Dynamic zero compression for cache energy reduction. In *MICRO-33*, 2000.

[Wan02a] A. Wang et al. Optimal supply and threshold scaling for subthreshold CMOS circuits. In *International Symposium on VLSI*, pages 5–9, 2002.

[Wan02b] H. Wang et al. Orion: A power-performance simulator for interconnection networks. In *MICRO*, pages 294–305, Nov. 2002.

[Wan02c] H. Wang et al. A power model for routers: Modeling Alpha 21364 and infiniband routers. *IEEE Micro*, 23(1):26–35, Jan/Feb 2002.

[Wea] C. Weaver. SPEC2000 Alpha binaries (little endian). http://www.eecs.umich.edu/˜chriswea/benchmarks/spec2000.html.

[Wei98] L. Wei et al. Design and optimization of low voltage high performance dual threshold CMOS circuits. In *DAC*, pages 489–494, 1998.

[Wit01] E. Witchel et al. Direct addressed caches for reduced power consumption. In *MICRO-34*, pages 124–133, Dec. 2001.

[Yan98] S. Yang et al. Scaling and integration of high performance interconnects. In *MRS Symposium on Advanced Interconnect*, Apr. 1998.

[YBD98] Y. Ye, S. Borkar, and V. De. A technique for standby leakage reduction in high-performance circuits. In *Symp. on VLSI Circuits*, pages 40–41, 1998.

[Ye 02] T. Ye et al. Analysis of power consumption on switch fabrics in network routers. In *DAC*, pages 524–529, 2002.

[ZA05] M. Zhang and K. Asanović. Victim replication: Maximizing capacity while hiding wire delay in tiled chip microprocessors. In *ISCA 32*, June 2005.

[Zha05] B. Zhai et al. Analysis and mitigation of variability in subthreshold design. In *ISLPED*, 2005.

[Zyu00] V. Zyuban. *Inherently Lower-Power High-Performance Superscalar Architecture*. PhD thesis, Notre Dame, 2000.