

# **Thermal Modeling and Management of Microprocessors**

A Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Doctor of Philosophy

Computer Science

by

**Karthik Sankaranarayanan**

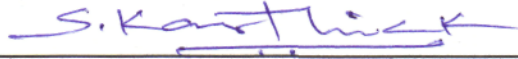
May 2009

## Approvals

This dissertation is submitted in partial fulfillment of the requirements for the degree of

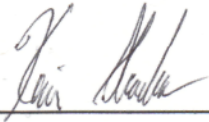
Doctor of Philosophy

Computer Science

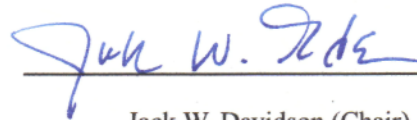


Karthik Sankaranarayanan

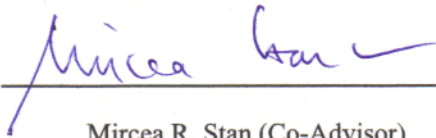
Approved:



Kevin Skadron (Advisor)



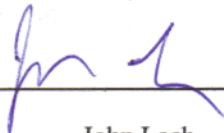
Jack W. Davidson (Chair)



Mircea R. Stan (Co-Advisor)

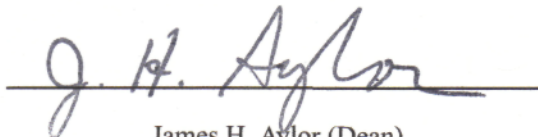


James P. Cohoon



John Lach

Accepted by the School of Engineering and Applied Science:



James H. Aylor (Dean)

April 2009

## Abstract

---

The most recent, and arguably one of the most difficult obstacles to the exponential growth in transistor density predicted by Moore's Law is that of removing the large amount of heat generated within the tiny area of a microprocessor. The exponential increase in power density and its direct relation to on-chip temperature have, in recent processors, led to very high cooling costs. Since temperature also has an exponential effect on lifetime reliability and leakage power, it has become a first-class design constraint in microprocessor development akin to performance.

This dissertation describes work to address the temperature challenge from the perspective of the architecture of the microprocessor. It proposes both the infrastructure to model the problem and several mechanisms that form part of the solution. This research describes HotSpot, an efficient and extensible microarchitectural thermal modeling tool that is used to guide the design and evaluation of various thermal management techniques. It presents several Dynamic Thermal Management (DTM) schemes that distribute heat both over time and space by controlling the level of computational activity. Processor temperature is not only a function of the power density but also the placement and adjacency of hot and cold functional blocks, determined by the floorplan of the microprocessor. Hence, this dissertation also explores various thermally mitigating placement choices available within a single core and across multiple cores of a microprocessor. It does so through the development of HotFloorplan, a thermally-aware microarchitectural floorplanner. Finally, through an analytical framework, this research also focuses on the spatial (size) granularity at which thermal management is important. If regions of very high power density are small enough, they do not cause hot spots. The granularity study quantifies this relationship and illustrates it using three different microarchitectural examples.

## Acknowledgments

---

This dissertation would not have been possible without the help of so many people. I am greatly indebted to my advisor Professor Kevin Skadron for his guidance. His approach of giving me complete freedom has made me grow as an individual and take ownership of the research problem. He pushed me when he needed to and that is the reason I have reached this point of completion. A significant portion of the text of the second and third chapters of this dissertation arose out of a paper in which my advisor played the primary role in writing. I am also thankful to him for his flexibility in letting me work remotely so I could be with my wife and for his continued financial support all these years. This work was supported in part by the National Science Foundation under grant nos. CCR-0105626, CCR-0133634, EIA-0224434, CCF-0429765, CSR-0509245, CRI-0551630 and MIP-9703440, an Army Research Office grant no. W911NF-04-1-0288, an Excellence Award from the University of Virginia Fund for Excellence in Science and Technology, a grant from Intel MRL and an IBM Research Faculty Partnership Award.

I am deeply grateful to my co-advisor Professor Mircea Stan for the countless hours of patient brainstorming. He helped me in grasping the circuit-level concepts right from the days when I was a student in his VLSI class. My sincere thanks to the members of my committee — apart from my advisor and co-advisor, Professors Jack Davidson, John Lach and James Cohoon, for their patient reading of my dissertation, considerate understanding of the change of scope of this work, willingness to give four crucial extra days for the submission of the dissertation and their constructive feedback that has helped in the shaping of this document. I am also thankful to Professor Hossein Haj-Hariri for teaching me the basics of analytical solutions of the heat equation and Professor Robert Ribando for directing me to the right person to ask such questions. I would also like to

express my thanks to Dr. Norman Jouppi for the discussion during his visit to UVA, which formed the origin of the L2 cache-bank insertion study. I have learnt much from my mentors during the two summer internships — Dr. Luiz Barroso and Dr. Scott McFarling and would like to thank them for the opportunity. I would also like to express my sincere thanks to all my professors and teachers, present and past, who have imparted the gift of knowledge in me. Many thanks to Intel and Dr. Daryl Nelson for the job offer and the patient wait. In these times of economic uncertainty, this security has helped me focus on my dissertation.

I cannot thank my colleague Wei Huang enough for all his help. We have collaborated very closely and it is impossible to list the number of times I have benefited from Wei's help in the form of explanations, experiments, feedback, reviews, code, text, figures *etc.* He is the primary contributor to the HotSpot thermal model. I would also like to convey my earnest appreciation for the contributions from Siva Velusamy, David Tarjan and Yingmin Li to the third chapter of this dissertation. Siva made important contributions to the implementation and evaluation of the DTM schemes. David calibrated the power model and Yingmin modified the performance model to mimic an Alpha-like architecture. I would also like to acknowledge the inputs of Siva to the modeling of the extra wire delay in the fourth chapter. I am thankful to Michele Co and Jeremy Sheaffer for their proposal and dissertation documents and presentations which served as samples to base mine upon. Many thanks to David Coppit and John Regehr for putting together a latex dissertation template conforming to the standards of UVA SEAS. I would like to express my thanks to my colleagues Michael Boyer, David Tarjan, Jiyuan Meng, Shuai Che, Eric Humenay and Yingmin Li for helping me work remotely by switching on my computer when it went down. Michael and David especially helped me by taking care of my office space and by plugging in my computer to the UPS.

As an international student in a foreign country, my graduate life in UVA has benefited immensely from the help of many graduate students who have helped me in getting established during my initial years in Charlottesville. I would like to thank them all for all their help including the many rides and errands.

I have been fortunate to have made and drawn support from many close friends formed during

my stay at UVA, during the CEG days and during the high school days at Madurai. Many of them have become my extended family and I am extremely grateful for all their wishes and support. Among them, as my co-travellers in the Ph.D. journey, Ram, Sundar, Easwar and Lakshmi have been my sounding boards offering encouragement and sharing insights innumerable times. I am deeply indebted to them.

I would not be here without my parents. Their constant belief in me, complete independence for my actions and respect for my choices have made me who I am. My heartfelt thanks to them for their love, blessings and prayers. I am thankful for my two little sisters' wishes and untiring confidence in me. Preethi has been constantly cheering me since my middle school. Deepu has brought many laughs by sharing her youthful exploits.

Many friends of our family have practically become part of our own over the years. I am thankful for all their wishes and blessings. I would also like to thank my relatives who took a lot of interest in my progress for their support.

I have received invaluable guidance from two mentors in my personal life — my undergraduate advisor Professor Ranjani Parthasarathi and my uncle Mr. R. Mahendavel. *RP ma'am*, as we affectionately call her, has been an inspiration and an unending source of positive energy. My uncle has been a hero to me and has offered timely advice that has directed me in the right path many times in my life. I would like to express my deep gratitude for both of them.

The last six months of dissertation research is similar to the last mile of a marathon. During this crucial time, my friend and brother, Ram, gave up many comforts in hosting us in his apartment. He would meditate for hours alongside of me when I was working on my dissertation. I am thankful beyond words for all his help, which has enabled my reaching this point.

I am blessed to have a wonderful friend and wife in Meena. She has put her career on hold for a year just to be with me during this critical period. She has laughed with me during the successes and cried with me during the failures. She has been a provider of nourishment both in spirit and as actual food. She has been a part of me in this whole process. It is not possible to do justice by expressing my thanks to her in words.

The entire process of dissertation research has been both an exercise in confidence and an exercise in humility. I thank the experience itself, for it was built of many life lessons. I realize that whatever is being presented in the forthcoming pages, I have only been an instrument of. Hence, this work is dedicated to the source of it all.

**Dedication**

*To where it all came from*



# Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Thermal Modeling</b>	<b>7</b>
2.1	Introduction . . . . .	7
2.2	Related Work . . . . .	7
2.3	Importance of Directly Modeling Temperature . . . . .	9
2.4	The HotSpot Thermal Model . . . . .	13
2.5	Analytical Choice of Parameters . . . . .	18
2.6	Empirical Leakage Model . . . . .	23
<b>3</b>	<b>Dynamic Thermal Management of a Single Core</b>	<b>26</b>
3.1	Introduction . . . . .	26
3.2	Related Work . . . . .	29
3.3	Techniques for Architectural DTM . . . . .	31
3.4	Simulation Setup . . . . .	42
3.5	Results for DTM . . . . .	48
3.6	Conclusions and Future Work . . . . .	58
<b>4</b>	<b>Static Thermal Management through Core-Level Floorplanning</b>	<b>60</b>
4.1	Introduction . . . . .	60
4.2	Related Work . . . . .	62
4.3	Potential in Lateral Spreading . . . . .	64

<i>Contents</i>	x
4.4 Methodology . . . . .	65
4.5 Results . . . . .	73
4.6 Conclusions and Future Work . . . . .	78
<b>5 Thermal Benefit of Multicore Floorplanning</b>	<b>80</b>
5.1 Introduction . . . . .	80
5.2 Related Work . . . . .	82
5.3 Methodology . . . . .	83
5.4 Results . . . . .	89
5.5 Conclusion . . . . .	95
<b>6 Granularity of Thermal Management</b>	<b>96</b>
6.1 Introduction . . . . .	96
6.2 Related Work . . . . .	99
6.3 An Analytical Approach . . . . .	101
6.4 Microarchitectural Examples . . . . .	110
6.5 Conclusions and Future Work . . . . .	133
<b>7 Conclusion</b>	<b>135</b>
<b>A Solution of the Heat Equation</b>	<b>139</b>
<b>Bibliography</b>	<b>145</b>

## List of Figures

---

2.1	Relationship between power density and temperature for different moving average intervals [104]. . . . .	10
2.2	Power density and temperature profiles of the load-store queue. . . . .	11
2.3	Side view of a typical package. [104] . . . . .	15
2.4	A conceptual view of the nodes of the thermal circuit in HotSpot [50] . . . . .	16
2.5	Illustration of the RC networks corresponding to the Elmore delay model and the equivalent circuit for thermal conduction. . . . .	18
2.6	A transient thermal conduction problem equivalent to HotSpot's RC network with a single node. . . . .	19
2.7	Distributed vs. lumped thermal response. . . . .	21
3.1	(a): Die photo of the Compaq Alpha 21364 [26]. (b): Floorplan corresponding to the 21364 that is used in our experiments. (c): Closeup of 21364 core. [104] . . . . .	32
3.2	Floorplan with spare integer register file for migrating computation. [104] . . . . .	36
3.3	Simulated and calculated operating frequency for various values of $V_{dd}$ . The nominal operating point of our simulated processor is 3 GHz at 1.3V. [104] . . . . .	38
3.4	Operating temperature as a function of time (in terms of number of clock cycles) for various warm and hot benchmarks. [104] . . . . .	49
3.5	Slowdown for DTM. Bars: better techniques. Lines: weaker techniques. [104] . . . . .	49
3.6	Slowdown for DTM from eliminating sensor noise, and from the consequent increase in trigger threshold to 82.8°. [104] . . . . .	53

3.7	Slowdown for MC with 1- and 2-cycle penalties for accessing the spare register file. [104] . . . . .	54
3.8	Two floorplans used to study effects of lateral thermal diffusion. [104] . . . . .	56
3.9	Temperature as a function of time for the integer register file with the mesa benchmark, for two different floorplans (tight and loose) and a simulation with lateral thermal resistance omitted (simple). [104] . . . . .	57
3.10	Percent error in execution time when DTM techniques are modeled using no-DTM heat sink temperatures. [104] . . . . .	58
4.1	(a) The basic floorplan corresponding to the 21364 that is used in our experiments. (b) Close-up of the core area. [91] . . . . .	69
4.2	Performance impact of varying the delay on critical wires. [91] . . . . .	71
4.3	Floorplans generated by (a) <i>flp-basic</i> and (b) <i>flp-advanced</i> schemes. [91] . . . . .	74
4.4	Impact of floorplanning on peak temperature. [91] . . . . .	76
4.5	Performance slowdown of the various thermal management schemes. [91] . . . . .	77
5.1	Illustration of different core arrangements for a four-way CMP with each core resembling an Alpha 21364. (a) shows a typical floorplan with hot units adjacent to each other. (b) shows a floorplan with alternative orientations of the cores. (c) shows a checkerboard-like arrangement with the use of L2 cache banks as cooling buffers between the cores. (d) shows an arrangement used for a potential study with functional blocks scattered amidst the cache banks. The adjacency of the blocks in the original floorplan is maintained through the scattering. [89] . . . . .	84
5.2	Floorplans used for the <i>altflp</i> and <i>mingled</i> studies [89] . . . . .	90
5.3	The peak temperature of the different floorplan configurations for the SPEC2000 benchmarks. [89] . . . . .	91
5.4	Effect of varying the number of cores. In scaling the cores, the power density of the functional blocks is kept constant. [89] . . . . .	93

5.5 Impact of the ratio of core area to chip area. The size of each core remains the same while that of the L2 cache is increased, keeping its power density constant. [89] . . . 94

5.6 Thermal performance of the floorplanning schemes on doubling the L2 cache power density. [89] . . . . . 95

6.1 Illustration of the spatial thermal filtering. (a) Two blocks of different sizes having the same power density. (b) low-pass filter “Bode plot” frequency response. [88] . . . 97

6.2 Simplified heat conduction problem to mimic a silicon die of thickness  $l$ . A power density of  $q$  is applied at the bottom in a region of size  $2a$ . Insulated boundaries are marked by slanted lines. Serrated lines indicate that the boundaries extend to infinity in that direction. (a) shows the original infinite version of the problem and (b) shows the equivalent semi-infinite version that (a) reduces to because of symmetry. (b) is essentially the right half of (a) [88] . . . . . 103

6.3 Comparison of the analytical model and FEM simulation. (a) shows the impact of power dissipator size and (b) shows the effect of aspect ratio. Note that the *center-4layers-spread* and *corner-4layers-spread* curves are coincident on top of each other. [88] . . . . . 108

6.4 Illustration of the several checkerboard configurations studied. Darkly shaded areas denote the cores and the unshaded areas denote the lower-level cache banks. The numeric suffix attached to the name of each configuration indicates the percentage of the die area occupied by the cores. Each figure above shows a manycore die with 32 cores. [88] . . . . . 112

6.5 Results of the checkerboard experiments. Note the overlap of the curves *rotated-25* and *regular-25* (*rotated-75/regular-75*) in (c) and (d). *alt-25* and *regular-25* are also coincident in (c). [88] . . . . . 113

6.6 Illustration of the two different manycore scenarios simulated. The unshaded areas indicate second-level cache. The shaded areas denote cores resembling Alpha 21364. [88] . . . . . 117

6.7 Results of the local vs. global thermal management study. [88] . . . . . 119

6.8 Two different placements studied for the data cache. The unshaded areas denote the SRAM sub-arrays while the shaded portions indicate the periphery and routing. [88] 123

6.9 Results of the data cache thermal experiments. (a) and (b) show the case when all lines are passive. (c) plots the peak temperature of the active lines as a function of the number of lines that are accessed contiguously. [88] . . . . . 125

6.10 Relationship between sensor error and inter-sensor distance. For an Alpha 21264-like core, (a) shows a sample temperature profile and (b) shows what is seen by the sensors. The black dots denote the positions of the sensors. (c) plots the average maximum sensor error of the SPEC2000 benchmark suite for two different types of package. [88] . . . . . 128

6.11 Results of the interpolation study. (a) and (b) illustrate the behaviour of the bilinear and bicubic spline algorithms for the thermal profile and sensor readings as in figures 6.10(a) and (b) respectively. (c) presents the experimental results of comparing the algorithms against the nearest-neighbour interpolation scheme. [88] . . . . . 131

## List of Tables

---

2.1	Correlation of average power vs. temperature for power averaging windows of 10K–1B cycles. [104] . . . . .	9
2.2	Analogy between thermal and electrical quantities [104] . . . . .	14
3.1	Configuration of simulated processor microarchitecture. [104] . . . . .	44
3.2	Benchmark summary. “I” = integer, “F” = floating-point. Fast-forward distance (FF) represents the point, in billions of instructions, at which warmup starts (see Sec. 3.4.5). [104] . . . . .	46
3.3	Fraction of cycles in thermal violation (no DTM modeled) for the two different floorplans (loose and tight) with lateral thermal diffusion properly modeled (correct), and with lateral resistances omitted (simple). [104] . . . . .	55
3.4	Fraction of cycles above the thermal trigger point (no DTM modeled) for the two different floorplans. [104] . . . . .	55
4.1	Peak steady-state temperature for different levels of lateral heat spreading ( $^{\circ}\text{C}$ ) [91]	65
4.2	Benchmark characteristics [91] . . . . .	70
6.1	Area, power and power density distribution within the data cache for the <i>bzip2</i> benchmark. [88] . . . . .	122

# Chapter 1

## Introduction

---

Since the advent of the very first microprocessor, the transistors that make them have continued to shrink in size exponentially. This scaling has consistently kept up with or even surpassed the rate of growth predicted by Gordon Moore's empirical law stated in 1965 [73]. In comparison with the early microprocessors, today's processors are faster and cheaper by several orders of magnitude owing to this exponential growth in transistor density. However, as process technology scales into the nanometer region, several hindrances to its continuation emerge. Low-level effects which were previously considered second-order and have traditionally been largely invisible to computer architects have surfaced to become primary concerns. Processor temperature is one such concern that has arguably become one of the hardest obstacles to continued technology scaling.

Most of the energy consumed by a microprocessor is dissipated as heat due to the resistive behaviour of the processor circuits. The temperature of a chip, which is a measure of the amount of heat energy stored in it, is directly related to the power density (*i.e.*, the power consumed per unit area of the chip). In order to see the impact of scaling on temperature, it is illuminating to study the relationship between scaling and power density. Scaling theory [29] provides a simple way to reason about such a relationship between the device parameters (such as transistor length, supply voltage *etc.*) of successive technology generations. Every successive process generation shrinks the length of the transistor by a constant fraction of the previous length. This fraction is by a called the scaling factor (say  $k$ ) and is typically  $\approx 1/\sqrt{2}$ . Hence, the area of the transistor scales proportional to  $k^2$ , *i.e.*, approximately halving every successive generation. Assuming ideal scaling, supply



voltage ( $V$ ) scales down and frequency ( $f$ ) scales up linearly. Assuming that the microarchitecture remains the same, the scaling of the intrinsic capacitance ( $C$ ) of the transistor is also linear in this factor. Hence, the power consumption of the transistor, given by the formula  $CV^2f$ , scales down by a factor  $= k \times k^2/k = k^2$ . In other words, since the area scales down by  $k^2$ , the power density remains constant under ideal scaling. However, in reality, supply voltage has not been able to scale as well as the transistor dimensions. In fact, more recently, in order to maintain the performance under increasing leakage, it has not been able to scale at all! This has resulted in processor power almost remaining constant in moving to a newer process generation. More interestingly, the power density increases approximately by a factor of  $k^2$  every generation! This exponential increase in power density, if left unmanaged, would result in an exponential increase of temperature every successive generation. Since that cannot be allowed (otherwise, the chip would melt!), huge efforts have been put into the removal of heat away from the die-surface of a microprocessor. This has led to expensive packages (heat sinks, heat pipes *etc.*) and as a corollary, exponential increase in the cost of such cooling solutions.

Increase in die temperature is undesirable for a variety of reasons. Catastrophic failure such as the melting of the chip is a possibility, albeit a distant one. In reality, more pertinent reasons are in the form of increased leakage power and accelerated aging that reduces lifetime reliability.

Transistors consume power even when they are idle and not switching. This is called static power or leakage power. Even under nominal conditions, it can be a significant fraction ( $> 30\%$ ) [99] of the total power consumption at current feature sizes. It varies exponentially with respect to temperature. As temperature itself depends on the power consumption, there is a circular dependence between them. In extreme cases, this can result in a self-reinforcing positive feedback loop that leads to thermal runaway.

Temperature also has an exponentially adverse effect on the expected lifetime of a microprocessor. The Mean Time To Failure (MTTF) of a chip can be empirically described using the Arrhenius Equation given by:

$$MTTF = Ae^{-\frac{E_a}{kT}}$$

Here,  $T$  is the temperature,  $A$  is an empirical constant and  $E_a$  is the activation energy of the failure mechanism. It is to be noted that this equation does not capture all the effects of temperature (like thermal cycling, thermal gradients *etc.*) on reliability. However, it is a useful expression for first-order estimation. It actually comes from Chemistry where the rate of a chemical reaction is expressed as an exponential function of the temperature of operation. The rate of reaction or equivalently, the rate of failure in case of silicon, is highly accelerated at higher temperatures. In fact, the same principle is used in the testing of chips by accelerating their ageing process from several years to laboratory time scales through the artificial elevation of temperatures.

Apart from high performance microprocessors, temperature is also a matter of concern for mobile processors. Though such chips tend to be low in power consumption, their power density can be quite high because of their form factor (*e.g.* System-On-a-Chip (SoC) architectures) and the high-performance nature of the real-time applications that run on them. Usability considerations like fan noise, wearability *etc.*, also dictate a lower operating temperature budget for these chips. Thus, with the increase in cooling costs, adverse effect on static power, lifetime reliability and usability of processor chips, temperature has become a first-class design constraint in microprocessor development akin to performance.

The thermal problem described above can be approached from different perspectives. For instance, from a mechanical engineering perspective, one can design better, more efficient and cheaper heat removal systems. This is a significant challenge since we are already at the limits of affordable air cooling. For instance, the ITRS [99] projects a very flat trend for air cooling capacity even in the long term. From an operating systems perspective, one can employ better scheduling algorithms that distribute heat evenly across the entire chip by interleaving hot and cold processes in both time and space. With the potential availability of spare cores to run the application, multi-core processors offer extra flexibility in this regard. From a compiler design perspective, one can generate code that distributes computation in a manner which minimizes power density. Instead, the scope of this dissertation is to study the possibilities of thermal alleviation from the perspective of the *architecture of the microprocessor*. The microarchitecture is the final determinant of how much computation happens where and when. Hence, it is unique in its ability to accurately identify

the sources of thermal inefficiency and either curb them at the origin or manage them effectively within a desired cost budget. This dissertation is an exercise in offering sufficient research evidence for such an assertion. It should be noted that although microarchitectural power management might help with temperature by limiting wasteful power dissipation, it does so mainly by managing activity in the temporal dimension. Thermal management extends such control to the spatial dimension as well, offering additional levers in the temperature *vs.* performance trade-off.

The first step in addressing the thermal management problem at the microarchitectural level is the ability to model the temperature of a microprocessor accurately and efficiently. Hence, this dissertation first presents a microarchitecture-level thermal modeling tool called HotSpot [50] that models the heat conduction as an equivalent electrical circuit constructed from the layout information of the microprocessor. Given the per-unit power consumption, it then uses standard circuit solving techniques to solve for temperature. HotSpot is a computationally-efficient infrastructure that has been widely used in the computer architecture research community. It has been downloaded by more than one thousand users. This dissertation guides the design of certain important parameters of HotSpot through approximations derived using analytical foundations. Furthermore, since there is a circular and exponential relationship between leakage power and temperature, this dissertation also comes up with an empirical leakage power model by reverse engineering data from the reports of the International Technology Roadmap for Semiconductors (ITRS) [99].

In order to make a case for the argument that microarchitectural thermal management can be effective, the dissertation then presents and evaluates several Dynamic Thermal Management (DTM) schemes for a single-core microprocessor. They distribute heat both over time and space by controlling the level of computational activity in the individual functional blocks of the core. These techniques can manage worst-case thermal situations efficiently, thereby assisting the external cooling solutions to be built for the average-case rather than the worst-case.

Orthogonal to the individual activity of the functional blocks in a microprocessor, another important factor that affects the temperature distribution of a chip is the lateral coupling between adjacent blocks due to the spreading of heat in silicon. This is determined by the floorplan of the microprocessor. Hence, this dissertation also investigates whether floorplanning at the microar-

chitectural level can be applied viably towards thermal management by placing hot units close to cold ones. It develops a thermally-aware microarchitectural floorplanning tool called HotFloorplan and studies the various thermally mitigating placement choices available within a single core and across multiple cores of a microprocessor. For a single core, this work employs a simulated annealing-based [60] scheme that includes both performance and temperature in its cost function. For multiple cores, it experiments with the orientation of the cores so as to keep the hottest units of adjacent cores as far apart as possible and uses the relatively cooler L2 banks as thermal buffers between the much hotter cores.

The final piece of this dissertation focuses on the size granularity at which thermal management is important. Regions of high power density do not cause hot spots if they are small enough in size (*i.e.*, if their lateral dimensions are much smaller than the chip thickness). In other words, silicon acts as a spatial low-pass filter for temperature. This dissertation explains this phenomenon with an analytical formulation derived by solving the steady-state two-dimensional differential equation of thermal conduction for a geometry similar to microprocessor chips. It also illustrates this with three different microarchitectural examples: a study of the thermal efficiency of small cores *vs.* large cores in a manycore processor, an investigation of whether high aspect ratio sub-blocks like cache lines can become hot spots due to malicious code behaviour and an exploration of thermal sensor accuracy as a function of the distance between sensors with an examination of the effectiveness of sensor interpolation schemes.

With this overview, following are the contributions of this work:

- The design of an accurate and efficient thermal model guided by analytical foundations from conduction heat transfer and an empirical model for leakage power that accounts for dependence on its temperature and supply voltage. [50, 52, 103]
- New microarchitectural thermal management schemes for a single-core microprocessor: temperature-tracking Dynamic Frequency Scaling (DFS) and migration of computation to a spare register file. [101, 102, 103]
- A thermally-aware floorplanning scheme for a single-core microprocessor that incorporates

the key insight of accounting for the amount of communication on the wires and weighting them according to that. [91]

- Multicore floorplanning techniques that experiment with the orientation of the cores and insert second-level cache banks in between the cores in order to reduce the peak temperature. [89]
- An analytical framework that explains the spatial filtering behaviour of lateral thermal conduction, which is then used to understand several interesting questions on microarchitectural thermal granularity including the thermal efficiency of manycores, cache lines and sensor interpolation. [88]

The remainder of the dissertation is organized as follows: Chapter 2 describes the design of the thermal and empirical leakage power models, throwing light on the analytical basis of the thermal model's parametric choices; Chapter 3 presents and evaluates the various single-core thermal management schemes; Chapter 4 explains the thermally-aware single-core floorplanning algorithm and discusses its thermal and performance impact; Chapter 5 studies the various multicore floorplanning options including core orientation and cache bank insertion; Chapter 6 discusses the investigation on the granularity of thermal management; and Chapter 7 concludes this dissertation by summarizing the most important lessons and providing suggestions for potential directions for the future.

# Chapter 2

## Thermal Modeling

---

### 2.1 Introduction

This chapter discusses the design of the HotSpot thermal model [50, 52, 103] and the empirical leakage power model [103], providing insights into the analytical foundations of some of the parametric choices in HotSpot. Section 2.2 describes the work done by others closely related to the area of microarchitectural thermal modeling. Section 2.3 discusses the need for directly modeling temperature at the microarchitectural level. Section 2.4 explains the construction of the HotSpot model. Section 2.5 presents the analytical rationale behind the choice of a couple of important parameters of HotSpot. Section 2.6 details the empirical leakage power model.

### 2.2 Related Work

Thermal models developed prior to this work are typically at the circuit-level and the few microarchitectural models ignore the lateral heat spreading in the chip. An excellent survey of the circuit-level techniques is given by Sabry in [86] and more recently, by Pedram and Nazarian [79]. Batty *et al.* [10], Cheng and Kang [22], Koval and Farmaga [62], Székely *et al.* [83, 109], Torki and Ciontu [111] all describe techniques for modeling localized heating within a chip due to different power densities of various blocks, but none of these tools are easily adapted to architectural exploration for a variety of reasons. Microarchitectural models are employed very early in the de-

sign stage, at a *pre-RTL* level, when detailed layout and power consumption information may not be available. Architectural modeling typically precludes direct thermal-response measurements, *e.g.* [109, 111]. Similarly, the use of analytical power models obviates the need for joint electro-thermal modeling, *e.g.* [109]. Furthermore, most of the prior models are either at a lower *post-RTL* level depending on VLSI netlists and structural implementation details, or only give steady-state solutions.

Two microarchitectural thermal models we are aware of prior to this work are from Dhodapkar *et al.* [30] and Skadron *et al.* [100]. TEMPEST [30] models temperature directly using an equivalent RC circuit, but contains only a single RC pair for the entire chip, giving no localized information. [100] proposed a simple model for tracking temperature on a per-unit level but ignored the effect of lateral heat diffusion. Since this work, a few more higher-level thermal models have been developed. ATMI [72] is an analytical model based on an explicit solution to the heat equation. While the accuracy of an analytical model is attractive, it also implies that the expression for the solution has to be evaluated at every point of interest on the chip. This can be prohibitive when one is looking for temperature not at a particular point but with a particular property (*e.g.* maximum temperature of each functional block). Moreover, it is not very flexible in terms of the package configuration and number of layers, which could be an obstacle in its use for exploration at the microarchitectural level. For instance, adding support for an extra layer of Thermal Interface Material (TIM) would involve significant effort in recomputing the analytical solutions. The Mercury [46] and Thermostat [24], models that are intended to be used at the full-system-level, are also worth mentioning here. Mercury is a system-level temperature emulation suite that uses offline calibration and online update of per-component utilization information to compute temperature of systems (including disks, processors, air inlets *etc.*). Thermostat employs Computational Fluid Dynamic (CFD) modeling of rack-mounted servers. Although both tools model the entire system including convective heat transfer, they are much more coarse-grained by design. The microprocessor itself is only a component within the system they model and hence no localized information is obtainable, which is essential for architectural studies.

Since this work, there have been efforts in applying state-of-the-art numerical techniques such

as multigrid methods [67], Discrete Cosine Transforms (DCT) [118], moment matching [69] and adaptive temporal adaptation [117] to thermal circuit solvers. There have also been improvements to the thermal solver of HotSpot suggested by others [42]. HotSpot has benefited from some of these efforts and I have since incorporated a multigrid method in HotSpot’s steady-state thermal solver. However, these models do not provide significant extra detail and lack the flexibility of HotSpot. Since microarchitectural simulations tend to be long, the simulation speed of HotSpot remains quite sufficient for most architectural studies.

### 2.3 Importance of Directly Modeling Temperature

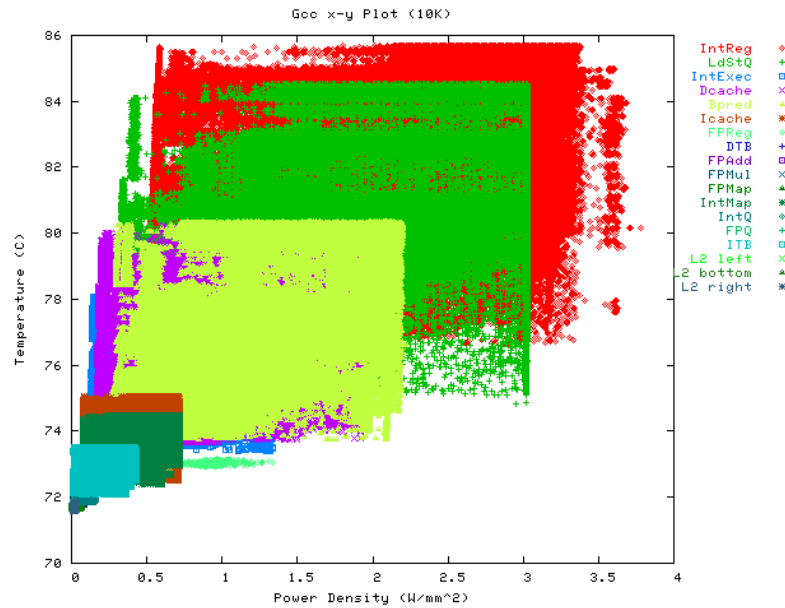
Units	Avg. Temp. (°C)	$R^2$ (%)					
		10K	100K	1M	10M	100M	1B
Icache	74.4	43.9	51.1	55.8	73.8	78.5	10.5
ITB	73.2	35.3	42.2	46.8	64.0	75.0	10.6
Bpred	76.2	54.0	71.5	77.6	88.7	91.0	5.3
IntReg	83.5	44.2	51.9	57.0	76.4	71.0	8.0
IntExec	76.7	46.3	53.3	57.9	75.7	76.6	8.3
IntMap	73.9	41.7	49.6	54.8	73.5	76.8	8.0
IntQ	72.4	31.5	36.4	39.6	53.9	80.7	13.0
LdStQ	79.2	47.9	63.4	69.0	83.6	83.2	6.6
Dcache	77.3	46.8	60.5	65.9	81.2	82.8	10.8
DTB	72.0	29.6	38.2	41.7	53.4	87.5	16.4
FPReg	73.0	26.0	29.6	38.8	64.6	84.8	21.1
FPAdd	72.6	49.7	51.1	54.9	66.5	86.4	24.9
FPMul	72.6	53.9	54.1	54.9	62.1	84.8	29.6
FPMap	71.7	16.8	20.2	22.3	26.9	0.5	3.2
FPQ	71.8	28.0	30.0	35.2	49.4	78.0	30.7
L2	71.7	14.2	19.7	21.8	26.6	49.9	3.3

Table 2.1: Correlation of average power vs. temperature for power averaging windows of 10K–1B cycles. [104]

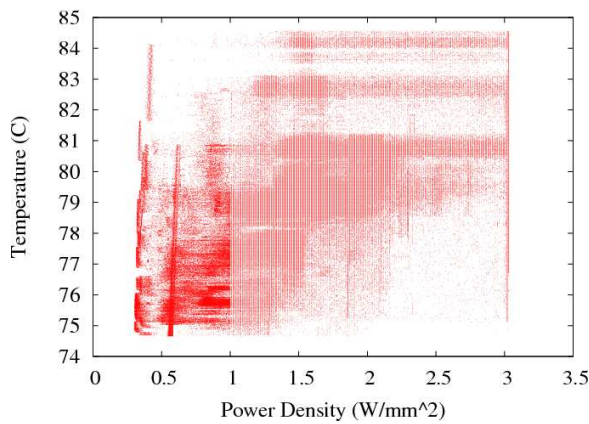
Temperature of a functional block of a microprocessor depends upon the power consumed by it per unit area (its power density). In fact, the thermal capacity of silicon acts as a temporal low pass filter and smooths out the spikes of power density. This behaviour has prompted a few prior studies to model temperature by averaging power dissipation over a window of time. This section will show the fallacy in using such a proxy for temperature. While the moving average helps in



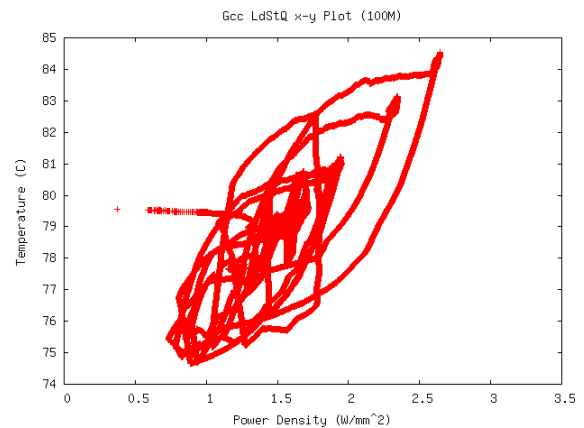
temporal smoothing, the temperature of a block depends not only on its power density but also on the temperatures of the other blocks nearby. Hence, any amount of temporal averaging without considering this lateral coupling of heat will not track its temperature reliably.



(a) 10 K, all blocks

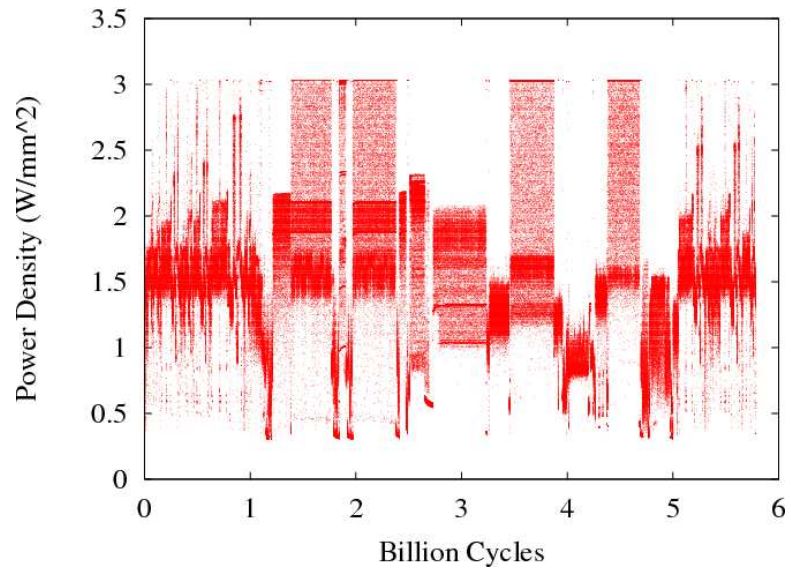


(b) 10 K, LdStQ

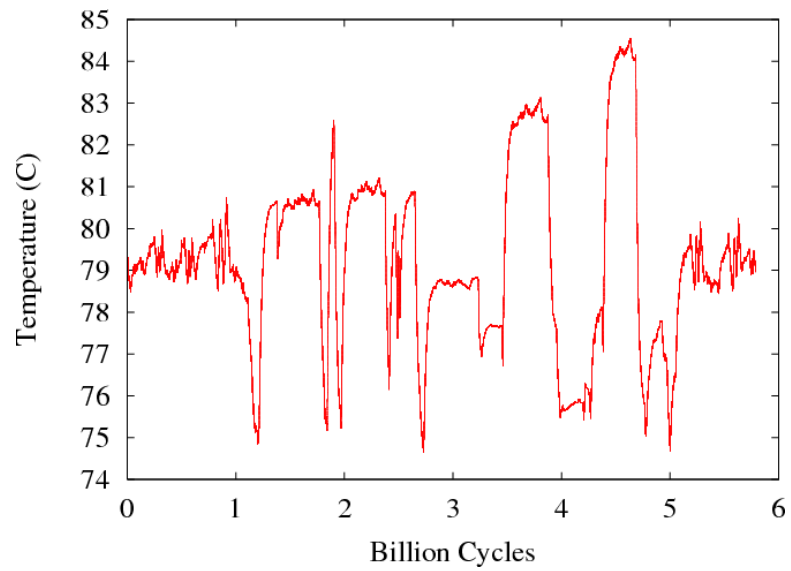


(c) 100 M, LdStQ

Figure 2.1: Relationship between power density and temperature for different moving average intervals [104].



(a) Power Density



(b) Temperature

Figure 2.2: Power density and temperature profiles of the load-store queue.

To show the importance of such a thermal model with lateral diffusion capability as opposed to a power metric, Table 2.1 shows the  $R^2$  value for correlation between temperature and a moving average of power dissipation for different averaging intervals. Each row of the table denotes an architectural functional block of an out-of-order processor similar to the Alpha 21364 [103]. The  $R^2$  value gives the percentage of variance that is common between two sets of data; values closer to 100% indicate better correlation. The data in Table 2.1 come from *gcc*, a representative benchmark. It can be seen that overall, there is poor correlation between temperature and moving average of the power dissipation. Even with the best-correlated averaging interval (100 million cycles), power still cannot track operating temperature effectively. This is further shown in figures 2.1 and 2.2. Figure 2.1 presents the relationship between power density and temperature for two different averaging intervals. Figure 2.1(a) shows the scatter plots of power density vs. temperature for all the major architectural blocks for an averaging interval of 10K cycles. For clarity, Figure 2.1(b) shows the same data only for the load-store queue. Figure 2.1(c) shows a similar scatter plot but for a higher averaging interval of 100M cycles. Figures 2.2(a) and (b) show the same power density and temperature data for the LdStQ block individually against time (in cycles). Any linear relationship between power density and temperature will show up as an approximate straight line in Figure 2.1. A temperature metric calibrated from a simple average of the power dissipation has such a linear relationship. However, what is seen in the graphs is an absence of any such linear relationship even at the higher averaging interval of 100M cycles (2.1(c)). It can be seen in Figure 2.1 that, for a given power density, the temperature varies by a large margin and *vice-versa*. Actually, for the LdStQ block, in 2.1(b), there is a clear range in the power density (around 0.35 to 3  $W/mm^2$ ) and temperature (around 74.5 to 84.5°C). On examining the graphs 2.2(a) and (b), it can be observed that this range corresponds to their maximum and minimum values over time. For power density, the maximum and minimum values also correspond to maximum and zero utilization of the LdStQ respectively. Also, the power density graph (2.2(a)) shows a pattern of bands while the temperature graph shows a pattern of lines. This means, for almost any given temperature, the instantaneous power density at that time could be all over the board. This uncorrelated behaviour is the reason for the rectangular shape in the first two graphs and the diverging shape in the third graph of Fig-

ure 2.1. It is also the reason why power density is not a good proxy for temperature. Hence, for reliable temperature estimation, it is important to model temperature directly.

## 2.4 The HotSpot Thermal Model

As a solution to the modeling problem, this dissertation presents my contributions to HotSpot [50, 52, 103], an efficient, extensible microarchitectural thermal model. HotSpot is available for free public download in the source form from <http://lava.cs.virginia.edu/HotSpot>. It is based on the well-known analogy [48, 64] that exists between the physical laws that govern current flow and heat flow, summarized in Table 2.2. Temperature is akin to “electric potential”. Heat flow can be described as a “current” passing through a thermal “resistance” ( $R$ ), leading to a temperature difference analogous to a “potential difference” or “voltage”. Thermal “capacitance” ( $C$ ) measures the amount of heat energy stored in a body — specifically, the product of its volume and specific heat capacity — and hence its rise in temperature. It is necessary for modeling transient behaviour, to capture the delay before a change in power results in the temperature’s reaching steady state. Lumped values of thermal  $R$  and  $C$  can be computed to represent the heat flow among units and from each unit to the thermal package. The thermal  $R$ s and  $C$ s together lead to exponential rise and fall times characterized by thermal  $RC$  time constants analogous to the electrical  $RC$  time constants. The rationale behind this analogy is that current and heat flow are described by exactly the same differential equations that describe the flow, *i.e.*, Ohm’s law for electrical conduction and Fourier’s law for heat conduction. Together with the principle of conservation of energy, these differential equations completely determine the electrical and thermal conduction in a body. HotSpot utilizes this principle in that it computes the thermal resistances and capacitances of a chip based on its geometry and floorplan. Then, for a power density map provided by a power-performance simulator (*e.g.* SimpleScalar [5] and Wattch [13]), it solves for the temperatures using standard circuit solving techniques. Early releases of HotSpot [103] incorporated a single circuit node for each functional block of a floorplan. Such a *block-based* model has the advantage of high computational efficiency. However, feedback from the use of HotSpot by the research community suggested that additional

modeling capability for high accuracy even at much finer levels of granularity is desirable. To this end, later releases of HotSpot [50, 52] also include support for high resolution through a three-dimensional, uniform grid-like structure of its thermal Rs and Cs. In such a case, each functional block is modeled as a grid of thermal Rs, each with its own capacitance to ground. With the presence of both the *block-based* and the *grid-based* models in HotSpot, the user is offered a choice between two levels of modeling granularity as a trade-off between accuracy and efficiency.

Thermal quantity	unit	Electrical quantity	unit
$Q$ , Heat	$J$	$q$ , Charge	$C$
$T$ , Temperature	$K$	$\phi$ , Potential	$V$
Temperature difference	$K$	$V$ , Voltage	$V$
$P$ , Heat flow, power	$W$	$I$ , Current flow	$A$
$\kappa$ , Thermal conductivity	$W/(m \cdot K)$	$\sigma$ , Electrical conductivity	$1/(m \cdot \Omega)$
$q$ , Heat flux, power density $= -\kappa \nabla T$ (Fourier's law)	$W/m^2$	$j$ , Current density $= -\sigma \nabla \phi$ (Ohm's law)	$A/m^2$
$R_{th}$ , Thermal resistance	$K/W$	$R$ , Electrical resistance	$\Omega = V/A$
$C_{th}$ , Thermal mass, capacitance	$J/K$	$C$ , Electrical capacitance	$F = C/V$
$\tau_{th} = R_{th} \cdot C_{th}$ , Thermal RC constant	$s$	$\tau = R \cdot C$ , Electrical RC constant	$s$

Table 2.2: Analogy between thermal and electrical quantities [104]

Chips today are typically packaged with the die placed against a spreader plate, often made of aluminum, copper, or some other highly conductive material, which is in turn attached to a heat sink of aluminum or copper through a Thermal Interface Material (TIM). The heat sink is cooled by a fan. HotSpot takes this into account and models the heat flow through such a typical cooling solution. An example of the configuration modeled by HotSpot is shown in Figure 2.3. Low-power/low-cost chips often omit the heat spreader and sometimes even the heat sink; and mobile devices often use heat pipes and other packaging that avoid the weight and size of a heat sink.

Given the floorplan of a processor and a set of power consumption values for each of its functional blocks, HotSpot first forms a thermal equivalent circuit by discretizing the geometry of the chip in all three dimensions. The topology of the resulting thermal circuit is very similar to the geometry of the chip and the thermal solution that it attempts to model. Each vertical layer of heat conduction (chip, heat sink, spreader, TIM *etc.*) is accounted for by a corresponding layer of circuit nodes connected to the layers immediately above and below it through a set of vertical thermal

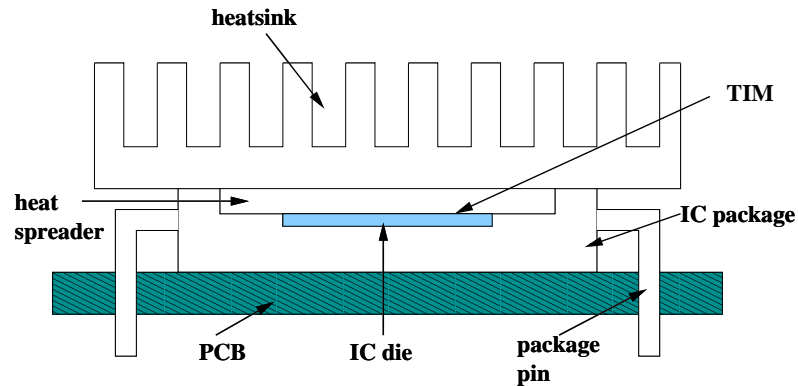


Figure 2.3: Side view of a typical package. [104]

resistances. Similarly, within a given layer, the lateral heat conduction is modeled through a network of lateral thermal resistances in both the  $x$  and  $y$  directions that connect regions of physical adjacency. For instance, in case of the *block-based* thermal model, this lateral discretization of the silicon layer results in one circuit node per functional block. Blocks adjacent to each other in the floorplan are also correspondingly connected by a lateral thermal resistance in the equivalent thermal circuit. Similarly, for the *grid-based* thermal model, the lateral discretization is uniform, with the silicon die being partitioned into a regular mesh of fixed number rectangular regions. Hence, the number of circuit nodes per layer is equal to the number of partitions. Also, partitions that are physically adjacent in the chip have their corresponding circuit nodes connected to each other in the equivalent circuit.

Since most of the vertical heat conduction through the different layers occurs through the regions directly beneath the die, the regions outside the die in the heat sink and spreader layers are discretized at a much coarser granularity. Specifically, the region of the heat spreader that lies outside the die is divided into four (north, east, south and west) with only one circuit node per region. Similarly, the region of the heat sink outside both the spreader area and the die area is also divided into four. The same holds true for the region of the heat sink that lies beneath the heat spreader but outside the coverage of the die as well. Figure 2.4 illustrates this arrangement through a conceptual view of a thermal solution in HotSpot consisting of TIM, spreader and sink. Each filled circle in the figure denotes a node in the thermal circuit. The power dissipation in silicon is modeled in the

thermal circuit as current sources connected to the circuit nodes corresponding to the silicon layer. Similarly, the heat removal at the fins of the heat sink is modeled as a convection resistance computed from the heat transfer coefficient of the fins. This is shown in the figure through the parallel arrows.

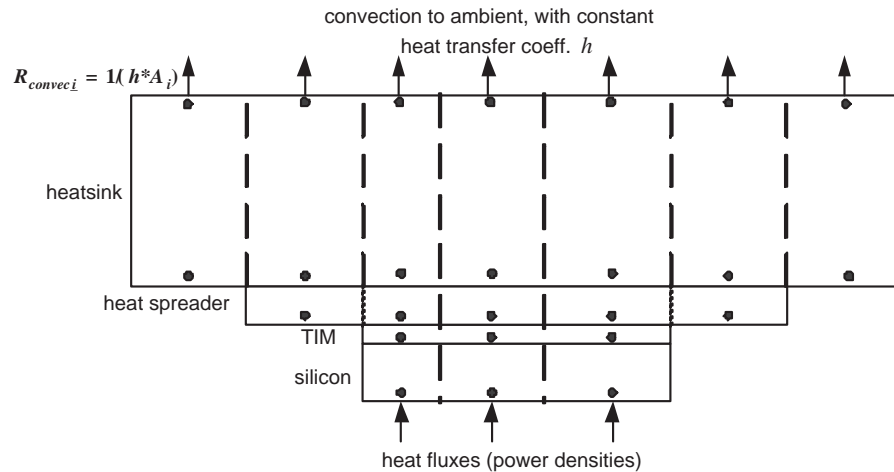


Figure 2.4: A conceptual view of the nodes of the thermal circuit in HotSpot [50]

The vertical and lateral thermal resistances described above only capture the heat conduction behaviour in the steady state. In order to model the transient thermal behaviour, each circuit node is also connected to ground through a thermal capacitance.

With this overview of the finite-element discretization in HotSpot, the final piece that remains in the construction of the equivalent thermal circuit is the computation of the thermal resistance and capacitance values themselves. The vertical and lateral discretizations in HotSpot reduce the heat conduction problem from three dimensions to a single dimension. Finally, in the single-dimensional case, the thermal resistance and capacitance values should be computed in such a manner that the response of the equivalent RC circuit matches closely with the true response of the one-dimensional transient thermal conduction problem. It turns out that for a given one-dimensional block of length  $t$ , cross-sectional area  $A$ , thermal conductivity  $\kappa$ , specific heat capacity  $c$  and density  $\rho$ , the thermal resistance  $R_{th}$  and capacitance  $C_{th}$  values are given by:

$$\begin{aligned}
 R_{th} &= \frac{t}{\kappa \cdot A} \\
 C_{th} &= c \cdot \rho \cdot t \cdot A
 \end{aligned}
 \tag{2.1}$$

The thermal equivalent circuit thus constructed is solved by HotSpot using standard circuit solving techniques that are simple yet fast. Since the number of nodes in the *block-based* model tends to be small, HotSpot uses a direct matrix inversion method to solve for its steady-state temperature. However, since the number of nodes in the *grid-based* model is usually large, it employs a more sophisticated multigrid technique [81] which in turn uses an iterative Gauss-Siedel method for steady-state solution. For transient calculation, HotSpot uses an adaptive step-size, fourth order Runge-Kutta method [81]. The reason for the choice of these solvers is that they are simple enough to be implemented and modified easily and efficient enough to be used in real-time microarchitectural simulations. Integration of more efficient off-the-shelf solvers into HotSpot is an area of future work not part of this dissertation.

HotSpot has been validated against two different finite-element solvers, Floworks [37, 103] and ANSYS [4, 50]. It has also been validated against actual measurements from a test chip [52, 108] and a Field-Programmable Gate Array (FPGA) [112].

For the kind of studies in this dissertation, the thermal model must have the following properties. It must track temperatures at the granularity of individual microarchitectural units. It must be parameterized, in the sense that a new network of Rs and Cs is automatically generated for different microarchitectures. It must be portable, making it easy to use with a range of power/performance simulators. It must be able to solve the RC-circuit's differential equations quickly. It must be calibrated so that simulated temperatures can be expected to correspond to what would be observed in real hardware. The HotSpot model meets all these conditions. It is a software that provides an interface for specifying some basic information about the thermal solution and for specifying any floorplan that corresponds to the architectural blocks' layout. HotSpot then generates the equivalent RC circuit automatically, and supplied with power dissipation over any chosen time step, computes



temperatures of each block of interest. It is efficient and extensible, making it a correct fit for the microarchitectural thermal studies of this dissertation.

HotSpot is a group effort that has involved many people. Apart from our professors (Skadron and Stan), acknowledgements are due to Wei Huang, Shougata Ghosh, Siva Velusamy and David Tarjan. Particularly, a majority of the modeling work and model validation are contributions of Wei Huang. My role, as one of the two primary contributors, has been in the design and implementation of the software, including feature enhancements (*e.g.* the ability to model a configuration without the heat sink or heat spreader), accuracy enhancements (of both the *block-based* and *grid-based* models), performance enhancements of the solvers (*e.g.* adaptive step sizing of the transient solver and the multigrid technique for the steady-state solver) and usability improvements (*e.g.* support for integration with other simulators, cleaner interfaces *etc.*). Furthermore, my contributions also include analytical reasoning of modeling granularity and capacitance fitting factors and some model validation using the ANSYS tool.

## 2.5 Analytical Choice of Parameters

### 2.5.1 Lumped Capacitance Scaling Factor

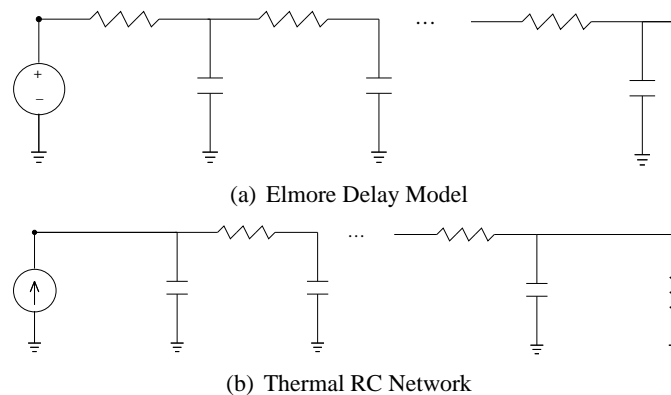


Figure 2.5: Illustration of the RC networks corresponding to the Elmore delay model and the equivalent circuit for thermal conduction.

It was mentioned in the last section that the three-dimensional discretization in HotSpot reduces the heat conduction problem to a single dimension. However, care has to be taken to ensure that

the equivalent RC approach in one dimension matches closely with the true thermal response. As a distributed RC network, the transient solution of the true thermal response is an infinite series of exponentials while that of a lumped RC is a single exponential. The common practice of calculating the lumped RC values is to use the Elmore delay model [34], as a result of which, a scaling factor of 50% is typically used. However, the Elmore delay model assumes a distributed RC network that is slightly different from the equivalent thermal RC network. Figure 2.5 illustrates both these networks. Figure 2.5(a) shows a network corresponding to the Elmore delay model while Figure 2.5(b) shows a network equivalent to thermal conduction. The Elmore delay model has a voltage source at its input while the thermal model has a current source. Moreover, the output terminal of the Elmore delay model is open while that of the thermal model is connected to ground. Hence, the 50% scaling factor from the Elmore delay model is not directly applicable in the thermal case. Instead, the principle of its derivation should be applied to the problem at hand (in this case, one-dimensional transient thermal conduction) in order to calculate the appropriate scaling factor. One of my contributions to the HotSpot model is the derivation of such an approach for the thermal conduction problem.

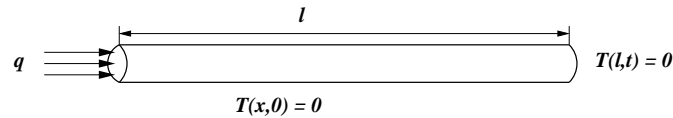


Figure 2.6: A transient thermal conduction problem equivalent to HotSpot's RC network with a single node.

Let us consider the transient thermal conduction problem in one dimension with boundary conditions similar to those found in HotSpot. We are interested in the case similar to Figure 2.5(b) where the input node of a thermal circuit is supplied by a current source and is connected to ground through a thermal distributed RC line. Figure 2.6 shows an equivalent thermal situation. A thin rod of length  $l$ , which is insulated on the sides, has one end at zero temperature while the other end is supplied with a heat flux (power density)  $q$ . Let the thermal conductivity of the material of the rod be  $\kappa$ , its thermal capacitance be  $c$  and its density be  $\rho$ . Then, if the transient temperature of the rod at distance  $x$  from the left end at time  $t$  is denoted by  $T(x,t)$ , then the one-dimensional transient

thermal conduction equation that describes this system is given by [17]:

$$\frac{\partial T}{\partial t} = \frac{\kappa}{c\rho} \frac{\partial^2 T}{\partial x^2} \quad (2.2)$$

subject to:

$$T(l, t) = 0 \quad (2.3)$$

$$T(x, 0) = 0 \quad (2.4)$$

$$\left. \frac{\partial T}{\partial x} \right|_{x=0} = -\frac{q}{\kappa} \quad (2.5)$$

The solution for the above equation and boundary conditions can be obtained from [17]. Note that we are only interested in the temperature at  $x = 0$  or  $T(0, t)$ . For simplicity of notation, let us define the normalized temperature  $T_{norm}$  to be  $\frac{T(0, t)}{T(0, \infty)}$ . Then, the expression for  $T_{norm}$  can be calculated as:

$$T_{norm} = 1 - \frac{8}{\pi^2} \sum_{n=1,3,5,\dots} \frac{e^{-n^2 \frac{\pi^2}{4} \frac{t}{RC}}}{n^2} \quad (2.6)$$

where,  $RC = \frac{c\rho l^2}{\kappa}$  or

$$R = \frac{1}{\kappa} \cdot \frac{l}{A} \quad \text{and} \quad C = c \cdot \rho \cdot l \cdot A \quad (2.7)$$

*i.e.*  $R$  and  $C$  are the same as equivalent thermal resistances  $R_{th}$  and  $C_{th}$  mentioned in equation 2.1. Now, we seek to approximate  $T_{norm}$  by an equivalent RC pair. Let us say that the time constant of that RC pair be  $\tau$ . The response for an RC pair is given by  $1 - e^{-\frac{t}{\tau}}$ . Clearly,  $T_{norm}$  is an infinite sum of exponentials while the response of an RC pair is a single exponential. Hence, an exact match is not possible. However, taking cue from the Elmore delay model, since both expressions are exponentials going from 0 to 1, we can seek that the area *above* their curves match (since the area *under* them is infinity). In other words, we could seek that the integrals of  $1 - T_{norm}$  and

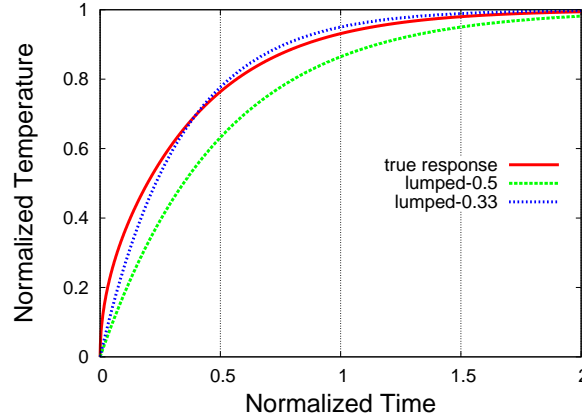


Figure 2.7: Distributed vs. lumped thermal response.

$1 - (1 - e^{-\frac{t}{\tau}}) = e^{-\frac{t}{\tau}}$  match. It can be seen that the latter integral (from 0 to  $\infty$ ) is equal to  $\tau$ . So, we get

$$\begin{aligned}
 \tau &= \frac{8}{\pi^2} \int_0^{\infty} \sum_{n=1,3,5,\dots} \frac{e^{-n^2 \frac{\pi^2}{4} \frac{t}{RC}}}{n^2} dt \\
 &= \frac{32 \cdot RC}{\pi^4} \sum_{n=1,3,5,\dots} \frac{1}{n^4} \\
 &= \frac{RC}{3}
 \end{aligned} \tag{2.8}$$

since  $\sum_{n=1,3,5,\dots} \frac{1}{n^4} = \frac{\pi^4}{96}$  from [1].

Thus, the scaling factor for the lumped capacitance turns out to be  $\frac{1}{3}$ . Figure 2.7 shows this experimentally. It plots the true thermal response ( $T_{norm}$ ) and the lumped responses with scaling factors of 0.5 and  $\frac{1}{3}$  respectively. It is evident that the scaling factor of 0.33 offers a better match.

## 2.5.2 Convection Capacitance

We saw in Section 2.4 from Figure 2.4 that HotSpot models the convection heat removal at the fins of the heat sink using a convection resistance. For a convection heat transfer co-efficient  $h$  and area of convection  $A$  (which is nothing but the area of the top face of the heat sink), the resistance  $R_{convec}$  is calculated as  $\frac{1}{h \cdot A}$ . In order to model the transient behaviour of the heat removal due to convection,

HotSpot also includes a convection capacitance  $C_{convec}$ . This section throws more light on how the value of  $C_{convec}$  is computed in HotSpot.

The rate of heat transfer due to convection from a given surface to an ambient temperature  $T_{amb}$  is directly proportional to the difference in temperature between the surface and  $T_{amb}$  [56]. The constant of proportionality is called the heat transfer co-efficient  $h$ . To compute the lumped capacitance  $C_{convec}$  at the heat sink, we can use this principle in conjunction with the law of conservation of energy. Assuming the average temperature of the surface is  $T$ , the rate of heat lost due to convection is the same as the rate of decrease in temperature multiplied by the bulk thermal mass  $m$  and the specific heat capacity  $c$  *i.e.*,

$$mc \frac{dT}{dt} = -hA(T - T_{amb}) \quad (2.9)$$

Solving this differential equation for an initial condition of  $T = T_0$  at  $t = 0$ , we get

$$T = T_{amb} + (T_0 - T_{amb}) \cdot e^{-\frac{hAt}{mc}} \quad (2.10)$$

Hence, the thermal time constant  $R_{convec} \cdot C_{convec}$  is  $\frac{mc}{hA}$ . In other words, if  $R_{convec}$  is  $\frac{1}{hA}$ , then  $C_{convec} = mc$ . For a typical configuration in HotSpot, most of the bulk thermal mass is from the heat sink. The mass of the heat spreader is about  $\frac{1}{28}^{th}$  of that of the heat sink and the mass of the silicon die is about  $\frac{1}{48}^{th}$  of that of the heat spreader. Hence, we can safely assume that the bulk thermal mass  $m$  is the mass of copper (heat sink and heat spreader). Hence, for the default configuration of HotSpot, with  $c$  denoting the specific heat capacity of copper, we get  $C_{convec} = mc = 91.377 \frac{J}{K}$ . However, since all capacitances are scaled by the scaling factor derived in the previous section (0.33), this value has to be divided by the same value to compensate for scaling. Hence, the default  $C_{convec}$  value for HotSpot is  $274.131 \frac{J}{K}$ .

## 2.6 Empirical Leakage Model

The leakage power consumed even when circuits are idle and not switching has an important relationship with temperature. Firstly, it increases exponentially with temperature. Secondly, its increase causes higher power density, resulting in even higher temperature and so on. Such a reinforcing feedback loop could, in extreme conditions, lead to catastrophic failure due to thermal runaway. Hence, in a study such as this dissertation involving the thermal management of microprocessors, it is important to consider the relationship of leakage power to temperature accurately. Hence, this section presents an empirical model that captures the most important factors in the relationship between leakage and temperature.

We begin with the BSIM3 v3.2 device model [61] for the drain current of a MOSFET transistor. Ignoring the effects of Drain-Induced Barrier Lowering (DIBL), assuming that the supply voltage is much greater than the threshold voltage and grouping parameters that remain constant for a given technology [119], we get,

$$I_D = K_1 \cdot T^2 e^{\frac{q}{k\eta} \left( \frac{V_{GS} - V_T - V_{off}}{T} \right)} \quad (2.11)$$

Where  $I_D$  is the drain current,  $V_{GS}$  is the gate voltage,  $V_T$  is the threshold voltage,  $T$  is the operating temperature,  $k$  is the Boltzmann constant,  $\eta$  is the sub-threshold swing co-efficient,  $q$  is the charge of an electron,  $V_{off}$  is an empirical BSIM3 parameter and  $K_1$  is an empirical constant dependent on the technology.

For two different values of the gate voltage  $V_{GS_1}$  and  $V_{GS_2}$ , if the corresponding drain currents are  $I_{D_1}$  and  $I_{D_2}$ , then using equation 2.11, we get (for  $\Delta = V_{GS_2} - V_{GS_1}$ ),

$$\Delta = \eta \frac{kT}{q} \ln\left(\frac{I_{D_2}}{I_{D_1}}\right) \quad (2.12)$$

The transistor is said to be in sub-threshold conduction if the gate voltage is less than the threshold voltage. So, under sub-threshold conduction, for  $\frac{I_{D_2}}{I_{D_1}} = 10$  ( $V_{GS} < V_T$ ), this value  $\Delta$  is called the

sub-threshold slope  $S$ . Therefore, the sub-threshold swing co-efficient  $\eta$  can be written as:

$$\eta = S \cdot \frac{q}{K \cdot T \cdot \ln(10)} \quad (2.13)$$

Now, from equation 2.12, when  $V_{GS} = V_T$ , the drain current is called the saturation drive current  $J_0$ . Similarly, when  $V_{GS} = 0$ , the drain current is called the sub-threshold leakage current  $I_0$ . It is this leakage current that determines the leakage power consumption. Hence, when  $\Delta = V_T - 0 = V_T$ , we get

$$V_T = \eta \frac{kT}{q} \ln\left(\frac{J_0}{I_0}\right) \quad (2.14)$$

Now, from equation 2.11, if we assume that the transistor is off (hence  $V_{GS} = 0$ ) and if we combine the technology-dependent parameters  $V_T$  and  $V_{off}$  into an empirical constant  $K_2$ , we get the following expression for the leakage power of a transistor assuming the supply voltage is  $V$ :

$$K_1 \cdot V \cdot T^2 \cdot e^{-\frac{K_2}{T}} \quad (2.15)$$

where,

$$K_2 = \frac{q}{k\eta} (V_T + V_{off}) \quad (2.16)$$

Assuming that the dynamic power consumed remains constant with respect to temperature, the ratio of the static to dynamic power also has a form similar to equation 2.15. Thus, the ratio  $R_T$  of leakage power to dynamic power as a function of temperature  $T$  is given by :

$$R_T = \frac{R_0}{V_0 T_0^2} e^{\frac{K_2}{T_0}} \cdot V T^2 \cdot e^{-\frac{K_2}{T}} \quad (2.17)$$

where  $T_0$  is the reference temperature and  $R_0$  is the ratio at  $T_0$  and nominal voltage  $V_0$ .

It is this equation 2.17 that is employed in this dissertation to model the relationship between leakage power and temperature/operating voltage. The value  $R_0$  is obtained from ITRS [99] reports.

In order to compute  $K_2$ , we first assume a constant sub-threshold slope of  $85 \frac{mV}{dec}$  and find out the sub-threshold swing co-efficient  $\eta$  using equation 2.13. Then, we obtain the sub-threshold current ( $I_0$ ) and saturation drive current ( $J_0$ ) values from the ITRS reports. Using these, we compute the threshold voltage  $V_T$  using equation 2.14. Finally, we use the value of the  $V_{off}$  parameter from the BSIM3 device models and plug in  $V_T$ ,  $V_{off}$  and  $\eta$  in equation 2.16 to obtain  $K_2$ . Thus, the leakage-to-dynamic ratio is computed using equation 2.17.

This chapter described the thermal and leakage power models that are used to evaluate the microarchitectural techniques of the remainder of the dissertation. Specifically, it

- Described my contributions to the HotSpot [50, 52, 103], microarchitectural thermal model.
- Presented the analytical rationale behind the choice of a couple of important parameters of the thermal model and
- Explained the construction of an empirical leakage power model [103] that models the relationship between leakage power and temperature.

With this background, this chapter sets the stage for the different microarchitectural thermal management techniques (both static and dynamic) that are to be presented in the remainder of the dissertation.



## Chapter 3

### Dynamic Thermal Management of a Single Core

---

#### 3.1 Introduction

Thermal solutions for microprocessors used to be designed for the worst-case on-chip power dissipation. However, such worst-case situations occur very rarely because the majority of applications, especially for the desktop, do not induce sufficient power dissipation to produce the worst-case temperatures. Thus, a thermal solution designed for the worst case is excessive. As cooling costs increase exponentially, such a large design margin becomes unaffordable. A solution to this problem is to decouple thermal management from the package. If the microprocessor can manage its own temperature by slowing down its execution or even halting completely when junction temperatures increase to unsafe levels, then the thermal solution need not be designed for the *absolute* worst-case. It can be designed with the worst *typical* application in mind and leave the *pathological* applications to the processor's self-management. Since typical high-power applications still operate 20% or more below the worst case [40], this can lead to dramatic savings. Even marginal savings in the cost of the thermal solution leads to large dollar savings because of the extremely high volumes involved. Hence, almost all current-day high performance processors employ runtime self-management of temperature, called Dynamic Thermal Management (DTM) [12]. In fact, this is the philosophy behind the thermal design of the Intel Pentium 4 [40]. It uses a thermal solution designed for a *typical* high-power application, reducing the package's cooling requirement by 20% and its cost accordingly. Should operating temperature ever exceed a safe temperature,

the clock is stopped (this is referred to as *global clock gating*) until the temperature returns to a safe zone. This protects against both timing errors and physical damage that might result from sustained high-power operation, from operation at higher-than-expected ambient temperatures, or from some failure in the package. As long as the threshold temperature that stops the clock (the *trigger threshold*) is based on the hottest temperature in the system, this approach successfully regulates temperature. This technique is similar to the “fetch toggling” technique proposed by Brooks and Martonosi [12], in which instruction fetch is halted when the trigger threshold is exceeded.

### 3.1.1 The Need for Architecture-Level Thermal Management

These *chip-level* hardware techniques illustrate both the benefits and challenges of runtime thermal management: while they can substantially reduce cooling costs and still allow typical applications to run at peak performance, these techniques also reduce performance for any applications that exceed the thermal design point. Such performance losses can be substantial with chip-wide techniques like global clock gating, with a 27% slowdown for the hottest application in this chapter, *art*.

Instead of using chip-level thermal-management techniques, this work argues that the microarchitecture has an essential role to play. *The microarchitecture has the ability to use runtime knowledge of application behaviour and the current thermal status of different units of the chip to adjust execution and distribute the workload in order to control thermal behaviour.* This chapter shows that architecture-level thermal management techniques regulate temperature with lower performance cost than chip-wide techniques by exploiting Instruction-Level Parallelism (ILP). For example, one of the better techniques in this chapter—with only an 8% slowdown—was a “local toggling” scheme that varies the rate at which only the hot unit (typically the integer register file) can be accessed. ILP helps mitigate the impact of reduced bandwidth to that unit while other units continue at full speed.

Architectural solutions do not of course preclude software or chip-level thermal-management techniques. Temperature-aware task scheduling, like that proposed by Rohou and Smith [85], can certainly reduce the need to engage any kind of runtime hardware technique, but there will always exist workloads whose operating temperature cannot successfully be managed by software. Chip-

level fail-safe techniques will probably remain the best way to manage temperature when thermal stress becomes extreme, for example when the ambient temperature rises above specifications or when some part of the package fails (for example, the heat sink falls off). But all these techniques are synergistic, and only architectural techniques have detailed temperature information about hot spots and temperature gradients that can be combined with dynamic information about ILP in order to precisely regulate temperature while minimizing performance loss.

### 3.1.2 Contributions

This chapter evaluates a variety of DTM techniques, proposing three new schemes: “Temperature-Tracking” Dynamic Frequency Scaling (TTDFS), migration of computation to a spare register file, and local toggling. These schemes are compared against previously proposed schemes such as Dynamic Voltage Scaling (DVS) [12, 36, 49], global clock gating [40] / fetch toggling [12] and a low-power secondary pipeline [68]. The most effective technique is TTDFS: timing errors due to hot spots can be eliminated with an average slowdown of 2%, and, if frequency can be changed without stalling computation, less than 1%. For temperature thresholds where preventing physical damage is also a concern, using a spare register file and migrating computation between the register files in response to heating is the best, with an average slowdown of 5–7.5%. Local toggling and an overhead-free voltage scaling technique performed almost as well, both with slowdowns of about 8%. All our experiments include the effects of sensor imprecision, which significantly handicaps runtime thermal management.

The experiments in this chapter also involved group effort with contributions from Siva Velusamy, David Tarjan and Yingmin Li. Hence, I would like to acknowledge their inputs. Among the newly proposed schemes, my contributions have been mainly in TTDFS and migration of computation. Siva Velusamy took the lead in modeling the local toggling. In the evaluation of the previously-proposed schemes, my contributions to DVS include the cost-benefit approach of determining when to change the voltage and frequency. This approach led to the avoidance of excessive voltage changes. I was also responsible for the modeling of the relationship between voltage and frequency. Furthermore, I modeled the low-power secondary pipeline while Siva contributed to the

global clock gating/fetch toggling. David Tarjan worked on the power model and its calibration against the Alpha 21364. Yingmin Li adapted SimpleScalar to an Alpha-like microarchitecture. Finally, I also contributed extensively to the extended studies about the roles of sensor error, heatsink temperature, lateral thermal diffusion and wire delay of the spare register file.

The rest of this chapter is organized as follows. The next section describes previous work related to DTM. Section 3.3 presents the three new DTM schemes and compares them against the previously-proposed techniques. It also explores the role of thermal-sensor non-idealities on thermal management. Section 3.4 describes our experimental setup, issues concerning initial temperatures, and the time-varying behaviour of some programs. Section 3.5 compares the various thermal-management techniques' ability to regulate temperature and discusses some of the results in further detail and Section 3.6 concludes the chapter.

## **3.2 Related Work**

### **3.2.1 Non-Architectural Techniques**

A wealth of work has been conducted to design new packages that provide greater heat-removal capacity, to arrange circuit boards to improve airflow, and to model heating at the circuit and board (but not architecture) levels. In addition to the design of new, higher-capacity packages, quiet fans, and the choice of materials for circuit boards and other components, recent work at the packaging level has given a great deal of consideration to liquid cooling to achieve greater thermal conductivity (but cost and reliability are concerns) [6]; heat pipes to spread or conduct the heat to a location with better airflow (especially attractive for small form factors like laptops), *e.g.* [114]; and to high-thermal-mass packages that can absorb large quantities of heat without raising the temperature of the chip—this heat can then be removed during periods of low computation activity or DVS can be engaged if necessary, *e.g.* [16].

### 3.2.2 Architectural Techniques

Despite the long-standing concern about thermal effects, only a few studies have been published in the architecture field, presumably because power itself has only become a major concern to architects within the past five years or so, and because no good models existed that architects could use to evaluate thermal-management techniques. Gunther *et al.* [40] describe the thermal design approach for the Pentium 4, where thermal management is accomplished via global clock gating. Lim *et al.* [68] propose a heterogeneous dual-pipeline processor for mobile devices in which the standard execution core is augmented by a low-power, single-issue, in-order pipeline that shares the fetch engine, register files, and execution units but deactivates out-of-order components like the renamer and issue queues. The low-power pipeline is primarily intended for applications that can tolerate low performance and hence is very effective at saving energy, but this technique is also potentially effective whenever the primary pipeline overheats. This work used Tempest [30], which does model temperature directly, but only at the chip level, and no sensor effects are modeled. Performance degradation is not reported, only energy-delay product.

Huang *et al.* [49] deploy a sequence of four power-reducing techniques—a filter instruction cache, DVS, sub-banking for the data cache, and if necessary, global clock gating—to produce an increasingly strong response as temperature approaches the maximum allowed limit. Brooks and Martonosi [12] compared several stand-alone techniques for thermal management: frequency scaling, voltage and frequency scaling, fetch toggling (halting fetch for some period of time, which is similar to the Pentium 4's global clock gating), decode throttling (varying the number of instructions that can be decoded per cycle [87]), and speculation control (varying the number of in-flight branches to reduce wasteful mis-speculated execution [70]). They also point out the value of having a direct microarchitectural thermal trigger that does not require a trap to the operating system and its associated latency. They find that only fetch toggling and aggressive DVS are effective, and they report performance penalties in the same range as found by the Huang group. Unfortunately, while these papers stimulated much interest, no temperature models of any kind were available at the time these papers were written, so both use chip-wide power dissipation averaged over a moving window as a proxy for temperature. As it was shown in Section 2.3, this value does not track temperature

reliably. A further problem is that, because no model of localized heating was available at the time, it was unknown which units on the processor ran the hottest, so some of the proposed techniques do not reduce power density in those areas which industry feedback and our own simulations suggest to be the main hot spots, namely the register files, load-store queue, and execution units. For example, we found that the low-power cache techniques are not effective, because they do not reduce power density in these other hot units.

Skadron *et. al.* [100] proposed feedback control to modify the Brooks fetch-toggling algorithm to respond gradually, showing a 65% reduction in performance penalty compared to the all-or-nothing approach. However, it used a chip-wide thermal model giving no localized information. A chip-wide model allows some exploration of chip-wide techniques like DVS, fetch toggling, and the Pentium 4's global clock gating, but not more localized techniques, and does not capture the effects of hot spots or changing chip layout. No prior work in the architecture field accounts for imprecision due to sensor noise and placement.

### 3.3 Techniques for Architectural DTM

This section describes the various architectural mechanisms for dynamic thermal management that are evaluated in this chapter, including both extant techniques and those that we introduce, and discusses how sensor imprecision affects thermal management. It is convenient to define several terms: the *emergency threshold* is the temperature above which the chip is in *thermal violation*; for 85°, violation may result in timing errors, while for lower-performance chips with higher emergency thresholds, violation results in higher error rates and reduced operating lifetime. In either case, we assume that the chip should *never* violate the emergency threshold. This is probably overly strict, since error rates and aging are probabilistic phenomena, and sufficiently brief violations may be harmless, but no good architecture-level models yet exist for a more nuanced treatment of these thresholds. Finally, the *trigger threshold* is the temperature above which runtime thermal management begins to operate; obviously,  $\text{trigger} < \text{emergency}$ .

Before describing the runtime DTM techniques, it is also useful to show the floorplan of the

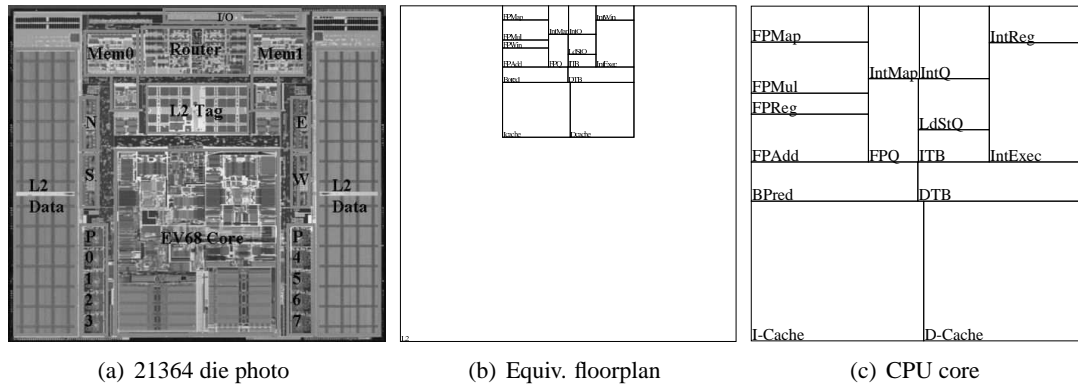


Figure 3.1: (a): Die photo of the Compaq Alpha 21364 [26]. (b): Floorplan corresponding to the 21364 that is used in our experiments. (c): Closeup of 21364 core. [104]

processor architecture we model. In all of our simulations so far, we have used a floorplan (and also approximate microarchitecture and power model) corresponding to that of the Alpha 21364. This floorplan is shown in Figure 3.1. Like the 21364, it places the CPU core at the center of one edge of the die, with the surrounding area consisting of L2 cache, multiprocessor-interface logic, etc. Since we model no multiprocessor workloads, we omit the multiprocessor interface logic and treat the entire periphery of the die as second-level (L2) cache. The area of this cache seems disproportionately large compared to the 21364 die photo in Figure 3.1(a), because we have scaled the CPU to 130nm while keeping the overall die size constant.

When we vary the microarchitecture, we currently obtain the areas for any new blocks by taking the areas of 21264 units and scaling as necessary. When scaling cache-like structures, we use CACTI 3.0 [97], which uses analytic models to derive area. Since CACTI’s a-priori predictions vary somewhat from the areas observed in the 21264, we use known areas as a starting point and only use CACTI to obtain scaling factors.

### 3.3.1 Runtime Mechanisms

This chapter proposes three new architecture techniques for DTM: “temperature-tracking” frequency scaling, local toggling, and migrating computation. They are evaluated in conjunction with four techniques that have previously been proposed, namely DVS (but unlike prior work, we add feedback control), global clock gating (where we also add feedback control), feedback-controlled

fetch toggling, and a low-power secondary pipeline. Each of the techniques is described below.

For techniques which offer multiple possible settings, we use formal feedback control to choose the setting. Feedback control allows the design of simple but robust controllers that adapt behaviour to changing conditions. Following [113], we use PI (proportional-integral) controllers, comparing the hottest observed temperature during each sample against the setpoint. The difference  $e$  is multiplied by the gain  $K_c$  to determine by how much the controller output  $u$  should change, *i.e.*:

$$u[k] = u[k - 1] + K_c \cdot e[k - 1] \quad (3.1)$$

This output is then translated proportionally into a setting for the mechanism being controlled. The hardware to implement this controller is minimal. A few registers, an adder, and a multiplier are needed, along with a state machine to drive them. But single-cycle response is not needed, so the controller can be made with minimum-sized circuitry. The datapath width in this circuit can also be fairly narrow, since only limited precision is needed.

As mentioned earlier, Brooks and Martonosi [12] pointed out that for fast DTM response, interrupts are too costly. We adopt their suggestion of on-chip circuitry that directly translates any signal of thermal stress into actuating the thermal response. We assume that it simply consists of a comparator for each digitized sensor reading, and if the comparator finds that the temperature exceeds the trigger, it asserts a signal. If any trigger signal is asserted, the appropriate DTM technique is engaged.

Next we describe the new techniques introduced in this chapter, followed by the other techniques we evaluate.

### 3.3.1.1 Temperature-Tracking Frequency Scaling

Dynamic voltage scaling (DVS) is typically preferred for power and energy conservation over dynamic frequency scaling (DFS), because DVS gives cubic savings in power density relative to frequency. However, *independently* of the relationship between frequency and voltage, the temperature-dependence of carrier mobility means that frequency is also linearly dependent on



the operating *temperature*. Garrett and Stan [38] report an 18% variation over the range 0-100°.

This suggests that the standard practice of designing the nominal operating frequency for the maximum allowed operating temperature is too conservative. When applications exceed the temperature specification, they can simply scale frequency down in response to the rising temperature. Because this temperature dependence is mild within the interesting operating region, the performance penalty of doing so is also mild—indeed, negligible.

For each change in setting, DVS schemes must stall for anywhere from 10–50  $\mu$ s to accommodate resynchronization of the clock’s phase-locked loop (PLL), but if the transition is gradual enough, the processor can execute through the change without stalling, as the Xscale is believed to do [93].

We examine a discretized frequency scaling with 10 MHz steps and 10 $\mu$ s stall time for every change in the operating frequency; and an ideal version that does not incur this stall but where the change in frequency does not take effect until after 10 $\mu$ s has elapsed. We call these “TT-DFS” and “TT-DFS-i(deal)”. Larger step sizes do not offer enough opportunity to adapt, and smaller step sizes create too much adaptation and invoke too many stalls.

This technique is unique among our other techniques in that the operating temperature may legitimately exceed the 85° threshold that other techniques must maintain. As long as frequency is adjusted before temperature rises to the level where timing errors might occur, there is no violation.

No feedback control is needed for TT-DFS, since the frequency is simply a linear function of the current operating temperature. It might seem odd, given the statement that DFS is inferior to DVS, that we only scale frequency. The reason is that the dependence of frequency on temperature is independent of its dependence on voltage: any change in voltage requires an additional reduction in frequency. This means that, unlike traditional DFS, TT-DFS does not allow reductions in voltage without further reductions in frequency.

### 3.3.1.2 Local Feedback-Controlled Fetch Toggling

A natural extension of the feedback-controlled fetch toggling proposed in [100] is to toggle individual domains of the processor at the gentlest duty cycle that successfully regulates temperature:

“PI-LTOG”. Only units in thermal stress are toggled. By toggling a unit like the integer-execution engine at some duty cycle of  $x/y$ , we mean that the unit operates at full capacity for  $x$  cycles and then stalls for  $y - x$  cycles. The choice of duty cycle is a feedback-control problem for which we use the PI controller with a gain of 1 (except the integer domain which uses a gain of 3) and a setpoint of  $81.8^\circ$ .

In our scheme, we break the processors into the following domains, each of which can be independently toggled:

- *Fetch engine*: I-cache, I-TLB, branch prediction, and decode.
- *Integer engine*: Issue queue, register file, and execution units.
- *FP engine*: Issue queue, register file, and execution units.
- *Load-store engine*: Load-store ordering queue, D-cache, D-TLB, and L2-cache.

Note that decoupling buffers between the domains, like the issue queues, will still dissipate some power even when toggled off in order to allow neighboring domains to continue operating; for example, allowing the data cache to write back results even though the integer engine is stalled that cycle.

Depending on the nature of the workload’s ILP and the degree of toggling, localization may reduce the performance penalties associated with toggling or GCG, but when the hot unit is also on the critical execution path, toggling that unit off will tend to slow the entire processor by a corresponding amount.

### 3.3.1.3 Migrating Computation

Two units that run hot by themselves will tend to run even hotter when adjacent. On the other hand, separating them will introduce additional communication latency that is incurred regardless of operating temperature. This suggests the use of spare units located in cold areas of the chip, to which computation can *migrate* only when the primary units overheat.

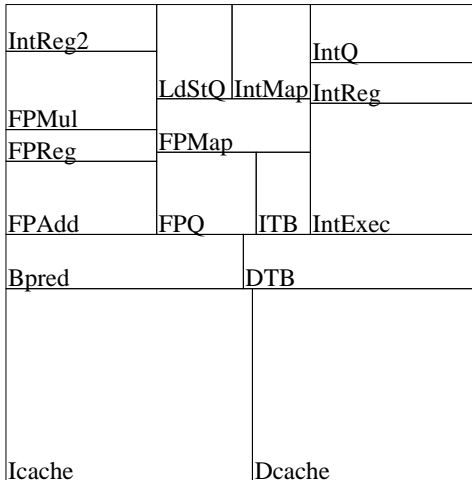


Figure 3.2: Floorplan with spare integer register file for migrating computation. [104]

We developed a new floorplan that includes an extra copy of the integer register file, as shown in Figure 3.2. When the primary register file reaches  $81.6^\circ$ , issue is stalled, instructions ready to write back are allowed to complete, and the register file is copied, four values at a time. Then all integer instructions use the secondary register file, allowing the primary register file to cool down while computation continues unhindered except for the extra computational latency incurred by the greater communication distance. The extra distance is accounted for by charging one extra cycle for every register-file access. (For simplicity in our simulator, we approximate this by simply increasing the latency of every functional unit by one cycle, even though this yields pessimistic results.) When the primary register file returns to  $81.5^\circ$ , the process is reversed and computation resumes using the primary register file. We call this scheme “MC”. Note that, because there is no way to guarantee that MC will prevent thermal violations, a failsafe mechanism is needed, for which we use PI-LTOG.

It is also important to note that the different floorplan will have some direct impact on thermal behaviour even without the use of any DTM technique. The entire integer engine runs hot, and even if the spare register file is never used, the MC floorplan spreads out the hot units, especially by moving the load-store queue (typically the second- or third-hottest block) farther away.

Another important factor to point out is that driving the signals over the longer distance to the secondary register file will require extra power that we currently do not account for, something that may reduce MC’s effectiveness, especially if the drivers are close to another hot area.

The design space here is very rich, but we were limited in the number of floorplans that we could explore, because developing new floorplans that fit in a rectangle without adding whitespace is a laborious process. However, later work detailed in the next chapter, automates this process through simulated annealing.

The dual-pipeline scheme proposed by Lim *et al.* [68] could actually be considered another example of migrating computation. Because the secondary, scalar pipeline was designed mainly for energy efficiency rather than performance, the dual-pipeline scheme incurred the largest slowdowns of any scheme we studied, and we compare this separately to our other schemes. MC could also be considered a limited form of multi-clustered architecture [15].

#### 3.3.1.4 Dynamic Voltage Scaling

DVS has long been regarded as a solution for reducing energy consumption, has recently been proposed as one solution for thermal management [12, 49], and is used for this purpose in Transmeta’s Crusoe processors [36]. The frequency must be reduced in conjunction with voltage since circuits switch more slowly as the operating voltage approaches the threshold voltage. This reduction in frequency slows execution time, especially for CPU-bound applications, but DVS provides a cubic reduction in power density relative to frequency.

We model two scenarios that we believe represent the range of what will likely be available in the near future. In the first (“PI-DVS”), there are ten possible discrete DVS settings ranging from 100% of the nominal voltage to 50% in equal steps. The penalty to change the DVS setting is  $10\mu s$ , during which the pipeline is stalled. In the second (“PI-DVS-i(deal)”), the processor may continue to execute through the change but the change does not take effect until after  $10\mu s$  have elapsed, just as with TT-DFS-i.

Because the relationship between voltage and frequency is not linear but rather is given by [82]

$$\frac{k(V_{dd} - V_t)^a}{V_{dd}} \quad (3.2)$$

voltage reductions below about 25% of the nominal value will start to yield disproportionate re-

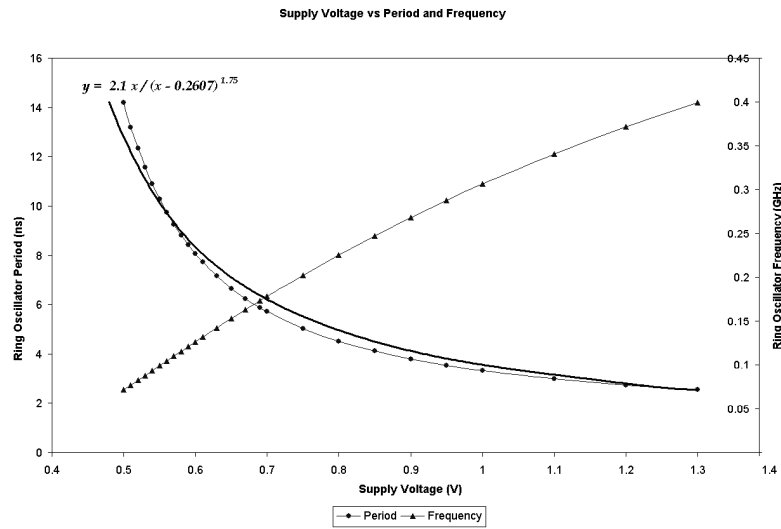


Figure 3.3: Simulated and calculated operating frequency for various values of  $V_{dd}$ . The nominal operating point of our simulated processor is 3 GHz at 1.3V. [104]

ductions in frequency and hence performance. We used Cadence with BSIM 100nm low-leakage models to simulate the period of a 101-stage ring oscillator under various voltages to determine the frequency for each voltage step (see Figure 3.3). Fitting this to a curve, we determined that  $k = 2.1$  and  $a = 1.75$ , which matches values reported elsewhere, *e.g.*, [58]. The appropriate values were then placed in a lookup table in the simulator. For continuous DVS, we perform linear interpolation between the table entries to find the frequency for our chosen voltage setting.

To set the voltage, we use a PI controller with a gain of 10 and a setpoint of  $81.8^\circ$ . A problem arises when the controller is near a boundary between DVS settings, because small fluctuations in temperature can produce too many changes in setting and a  $10\mu s$  cost each time that the controller does not take into account. To prevent this, we apply a low-pass filter to the controller output when voltage is to be scaled up. The filter compares the performance cost of the voltage change to the performance benefit of increasing the voltage and frequency and makes the change only when

profitable. Both these cost and benefit measures are percentages of change in delay. Computation of the benefit is straightforward: current delay is just the reciprocal of the current clock frequency. Similarly, future delay is also computed and the percent change is obtained from these numbers. However, in order to compute the cost, knowledge of how long the program will run before incurring another voltage switch is necessary, because the switch time is amortized across that duration. We take a simple prediction approach here, assuming the past duration to be indicative of the future and using it in the computation instead of the future duration. So, the ratio of the switch time and the past duration gives the cost. Note that this filter cannot be used when the voltage is to be scaled down because scaling down is mandatory to prevent thermal emergency.

### 3.3.1.5 Global Clock Gating and Fetch Toggling

As a baseline, we consider global clock gating (“GCG”) similar to what the Pentium 4 employs [40], in which the clock is gated when the temperature exceeds the trigger of  $81.8^\circ$  and ungated when the temperature falls back below that threshold. We also consider a version in which the duty cycle on the clock gating is determined by a PI controller with a gain of 1 (“PI-GCG”), similar to the way PI-LTOG is controlled. We recognize that gating the entire chip’s clock at fine duty cycles may cause voltage-stability problems, but it is moot for this experiment. We only seek to determine whether PI-GCG can outperform PI-LTOG, and find that it cannot because it slows down the entire chip while PI-LTOG exploits ILP.

We also evaluated fetch toggling [12], and a feedback controlled version in which fetching rather than the clock is gated until the temperature reaches an adequate level. Overall, fetch toggling and global clock gating are quite similar. We model global clock gating because it also cuts power in the clock tree and has immediate effect.

While the clock signal is gated, power dissipation within the chip is eliminated except for leakage power. Global clock gating is therefore a “duty-cycle based technique” for approximating traditional DFS, but without any latency to change the “frequency”. The Pentium 4 uses a duty cycle of  $1/2$ , where the clock is enabled for  $2\mu\text{s}$  and disabled for  $2\mu\text{s}$ , and once triggered, the temperature must drop below the trigger threshold by one degree before normal operation resumes [57]. In the

Pentium 4, each change in the clock frequency requires a high-priority interrupt, which Gunther *et al.* [40] report takes approximately  $1\mu\text{s}$  (but Lim *et al.* [68] report 1 ms). Brooks and Martonosi [12] instead proposed fetch *toggl*ing, in which fetch is simply halted until the temperature reaches an adequate level—they called this “toggle1.” This has two minor drawbacks compared to clock gating, in that power dissipation takes a few cycles to drop (as the pipeline drains) and the power dissipation in the clock tree (15% or more [70]) is not reduced. Brooks and Martonosi also considered setting the duty cycle on fetch to  $1/2$  (“toggle2”), but they and also Skadron *et al.* [100] found that this did not always prevent thermal violations. We believe the reason that the P4 succeeds with a duty cycle of  $1/2$  is that each phase is so long—microseconds rather than nanoseconds—that the chip can cool down sufficiently well. On the other hand, the penalty can be excessive when only minimal cooling is required.

Overall, fetch toggling and global clock gating are quite similar. We model global clock gating (“GCG”) separately because it also cuts power in the clock tree and has immediate effect. For the feedback-controlled versions of both schemes, we use a gain of 1.

### 3.3.1.6 Low-Power Secondary Pipeline

Lim, Daasch, and Cai [68] proposed, instead of migrating accesses to individual units, to use a secondary pipeline with very low-power dissipation. We refer to this technique as “2pipe.” Whenever the superscalar core overheats anywhere, the pipeline is drained, and then an alternate scalar pipeline is engaged. This pipeline shares the fetch engine, register file, and execution units of the superscalar pipeline; because they are now accessed with at most one instruction per cycle, their power dissipation will fall, but it is only the out-of-order structures whose active power dissipation is completely reduced. This scheme is essentially an aggressive version of computation migration, but we find that it penalizes performance more than necessary.

In [68], they do not model the extra latency that may be associated with accessing the now-disparate units, so we neglect this factor as well, even though we account for such latency in our “MC” technique. We also make the optimistic assumption here that when the low-power secondary pipeline is engaged, zero power is dissipated in the out-of-order units after they drain. We charge

1/4 the power dissipation to the integer-execution unit to account for the single-issue activity. These idealized assumptions are acceptable because they favor this scheme, and we still conclude that it is inferior to simpler alternatives like our floorplan-based techniques or even DVS alone. Of course, it is important to repeat that this technique was not optimized for thermal management but rather for energy efficiency.

### 3.3.2 Sensors

Runtime thermal management requires real-time temperature sensing. So far, all prior published work of which we are aware has assumed omniscient sensors, which we show in Section 3.5 can produce overly optimistic results. Sensors that can be used on chip for the type of localized thermal response we contemplate are typically based on analog CMOS circuits using a current reference. An excellent reference is [7]. The output current is digitized using a ring oscillator or some other type of delay element to produce a square wave that can be fed to a counter. Although these circuits produce nicely linear output across the temperature range of interest, and respond rapidly to changes in temperature, they unfortunately are sensitive to lithographic variations and supply-current variations. These sources of imprecision can be reduced by making the sensor circuit larger, at the cost of increased area and power. Another constraint that is not easily solved by up-sizing is that of sensor bandwidth—the maximum sampling rate of the sensor.

Our industry contacts tell us that CMOS sensors which would be reasonable to use in moderate quantity of say 10–20 sensors would have at best a precision of  $\pm 2^\circ\text{C}$  and sampling rate of 10 microseconds. This matches the results in [7]. We place one sensor per architectural block.

We model the imprecision by randomizing at each node the true temperature reading over the specified range  $\pm 2^\circ$ . We assume that the hardware reduces the sensor noises at runtime by using a moving average of the last ten measurements. Averaging reduces the error as the square root of the number of samples, if the measured value is stationary. However, the sensor measurement is not stationary for any meaningful averaging window. Hence, we must also account for the potential change in temperature over the averaging window, which we estimate to be potentially as much as  $0.4^\circ$  if temperatures can rise  $0.1^\circ$  per  $30\mu\text{s}$ . For  $\pm 2^\circ$ , we are therefore able to reduce the uncertainty



to  $S = \frac{2}{\sqrt{10}} + 0.4 = \pm 1^\circ$ . An averaging window of ten samples was chosen because the improved error reduction with a larger window is offset by the larger change in the underlying value.

There is one additional non-ideality that must be accounted for when modeling sensors and cannot be reduced by averaging. If a sensor cannot be located exactly coincident with every possible hot spot, the temperature observed by the sensor may be cooler by some spatial-gradient factor  $G$  than at the hot spot. If, in addition to the random error discussed above, there is also a systematic or offset error in the sensor that cannot be canceled, this increases the magnitude of the fixed error  $G$ . Based on simulations in our finite-element model and the assumption that sensors can be located near but not exactly coincident with hot spots, we choose  $G = 2^\circ$ .

It can therefore be seen that for any runtime thermal-management technique, the use of sensors lowers the emergency threshold by  $G + S$  ( $3^\circ$  in our case). This must be considered when comparing to other low-power design techniques or more aggressive and costly packaging choices. It is also strong motivation for finding temperature-sensing techniques that avoid this overhead, perhaps based on clever data fusion among sensors, or the combination of sensors and performance counters.

## 3.4 Simulation Setup

In this section, we describe the various aspects of our simulation framework and how they are used to monitor runtime temperatures for the SPEC2000 benchmarks [106].

### 3.4.1 Integration with HotSpot

HotSpot is completely independent of the choice of power/performance simulator. Adding HotSpot to a power/performance model merely consists of two steps. First, initialization information must be passed to HotSpot. This consists of an adjacency matrix describing the floorplan (the floorplan used for the experiments in this chapter is included in the HotSpot release) and an array giving the initial temperatures for each architectural block. Then at runtime, the power dissipated in each block is averaged over a user-specified interval and passed to HotSpot's RC solver, which returns

the newly computed temperatures. A time step must also be passed to indicate the length of the interval over the which power data is averaged. We chose a sampling rate of 10K cycles as the best tradeoff between precision and overhead.

The Runge-Kutta solver employs an adaptive step size algorithm. Given an interval for transient simulation, the solver breaks it into multiple steps, computing temperatures iteratively in each step. The size of each step is determined by the adaptive step sizing. If the slope of the temperature rise/fall is steep, the algorithm chooses small steps for accuracy. On the other hand, when the slope of the temperature curve is shallow, the algorithm rushes through the flat terrain using large steps. This improves the performance of the solver by many times. In fact, the extra simulation time for thermal modeling is less than 1%.

### 3.4.2 Power-Performance Simulator

We use a power model based on power data for the Alpha 21364 [9]. The 21364 consists of a processor core identical to the 21264, with a large L2 cache and (not modeled) glueless multiprocessor logic added around the periphery. An image of the chip is shown in Figure 3.1, along with the floorplan schematic that shows the units and adjacencies that HotSpot models. Because we study microarchitectural techniques, we use Wattch version 1.02 [13] to provide a framework for integrating our power data with the underlying SimpleScalar [14] architectural model. Our power data was for 1.6 V at 1 GHz in a  $0.18\mu$  process, so we used Wattch's linear scaling to obtain power for  $0.13\mu$ ,  $V_{dd}=1.3V$ , and a clock speed of 3 GHz. These values correspond to the recently-announced operating voltage and clock speed that for the Pentium 4 [84]. We assume a die thickness of 0.5mm. Our spreader and sink are both made of copper. The spreader is 1mm thick and  $3cm \times 3cm$ , and the sink has a base that is 7mm thick and  $6cm \times 6cm$ . Power dissipated in the per-block temperature sensors is not modeled.

The biggest difficulty in using SimpleScalar is that the underlying *sim-outorder* microarchitecture model is no longer terribly representative of contemporary processors, so we augmented it to model an Alpha 21364 as closely as possible. We extended both the microarchitecture and corresponding Wattch power interface; extending the pipeline and breaking the centralized RUU

into four-wide integer and two-wide floating-point issue queues, 80-entry integer and floating-point merged physical/architectural register file, and 80-entry active list. First-level caches are 64 KB, 2-way, write-back, with 64B lines and a 2-cycle latency; the second-level is 4 MB, 8-way, with 128B lines and a 12-cycle latency; and main memory has a 225-cycle latency. The branch predictor is similar to the 21364’s hybrid predictor, and we improve the performance simulator by updating the fetch model to count only one access (of fetch-width granularity) per cycle. The only features of the 21364 that we do not model are the register-cluster aspect of the integer box, way prediction in the I-cache, and speculative load-use issue with replay traps (which may increase power density in blocks that are already quite hot). The microarchitecture model is summarized in Table 3.1. Finally, we augmented SimpleScalar/Wattch to account for dynamic frequency and voltage scaling and to report execution time in seconds rather than cycles as the metric of performance.

Processor Core		Other Operating Parameters	
Active List	80 entries	Nominal frequency	3 GHz
Physical registers	80	Nominal $V_{dd}$	1.3 V
LSQ	64 entries	Ambient air temperature	45°C
Issue width	6 instructions per cycle (4 Int, 2 FP)	Package thermal resistance	0.8 K/W
Functional Units	4 IntALU, 1 IntMult/Div, 2 FPALU, 1 FPMult/Div, 2 mem ports	Die	0.5mm thick, 15.9mm × 15.9mm
		Heat spreader	Copper, 1mm thick, 3cm × 3cm
		Heat sink	Copper, 7mm thick, 6cm × 6cm
Memory Hierarchy		Branch Predictor	
L1 D-cache Size	64 KB, 2-way LRU, 64 B blocks, writeback	Branch predictor	Hybrid PAg/GAg with GAg chooser
L1 I-cache Size	64 KB, 2-way LRU, 64 B blocks both 2-cycle latency	Branch target buffer	2 K-entry, 2-way
L2	Unified, 4 MB, 8-way LRU, 128B blocks, 12-cycle latency, writeback	Return-address-stack	32-entry
Memory	225 cycles (75ns)		
TLB Size	128-entry, fully assoc., 30-cycle miss penalty		

Table 3.1: Configuration of simulated processor microarchitecture. [104]

### 3.4.3 Modeling the Temperature-Dependence of Leakage

Because leakage power is an exponential function of temperature, these power contributions may be large enough to affect the temperature distribution and the effectiveness of different DTM techniques. Furthermore, leakage is present regardless of activity, and leakage at higher temperatures may affect the efficacy of thermal-management techniques that reduce only activity rates. Hence, to make sure that leakage effects are modeled in a reasonable way, we use a simpler model: like Wattch, leakage in each unit is simply treated as a percentage of its power when active, but this percentage is now determined based on the empirical model discussed in Section 2.6.

We retain Wattach’s notion that power dissipated by a block during a cycle in which it is idle can be represented as a percentage of its active power. The only difference is that this percentage should be a function of temperature. This condition is achieved by employing the empirical leakage model from Section 2.6.

### 3.4.4 Benchmarks

We evaluate our results using benchmarks from the SPEC CPU2000 suite. The benchmarks are compiled and statically linked for the Alpha instruction set using the Compaq Alpha compiler with SPEC *peak* settings and include all linked libraries but no operating-system or multiprogrammed behaviour. For each program, we fast-forward to a single representative sample of 500 million instructions. The location of this sample is chosen using the data provided by Sherwood *et al.* [95]. Simulation is conducted using SimpleScalar’s EIO traces<sup>1</sup> to ensure reproducible results for each benchmark across multiple simulations.

Due to the extensive number of simulations required for this study and the fact that many did not run hot enough to be interesting thermally, we used only 11 of the total 26 SPEC2k benchmarks. A mixture of integer and floating-point programs with low, intermediate, and extreme thermal demands were chosen; all those we omitted operate well below the 81.8° trigger threshold. Table 3.2 provides a list of the benchmarks we study along with their basic performance, power, and thermal characteristics. It can be seen that IPC and peak operating temperature are only loosely correlated with average power dissipation. For most SPEC benchmarks, and all those in Table 3.2, the hottest unit is the integer register file—interestingly, this is even true for most floating-point and memory-bound benchmarks. It is not clear how true this will be for other benchmark sets.

For the benchmarks that have multiple reference inputs, we chose one. For *perlbmk*, we used *splitmail.pl* with arguments “957 12 23 26 1014”; *gzip* - graphic; *bzip2* - graphic; *eon* - rushmeier; *vortex* - lendian3; *gcc* - expr; and *art* - the first reference input with “-startx 110”.

---

<sup>1</sup>External Input-Output (EIO) traces are replayable records of external I/O activity in SimpleScalar

	IPC	Average Power (W)	FF (bil.)	% Cycles in Thermal Viol.	Dynamic Max Temp. (°C)	Steady-State Temp. (°C)	Sink Temp. (no DTM) (°C)	Sink Temp. (w/ DTM) (°C)
Low Thermal Stress (cold)								
parser (I)	1.8	27.2	183.8	0.0	79.0	77.8	66.8	66.8
facerec (F)	2.5	29.0	189.3	0.0	80.6	79.0	68.3	68.3
Severe Thermal Stress (medium)								
mesa (F)	2.7	31.5	208.8	40.6	83.4	82.6	70.3	70.3
perlbmk (I)	2.3	30.4	62.8	31.1	83.5	81.6	69.4	69.4
gzip (I)	2.3	31.0	77.3	66.9	84.0	83.1	69.8	69.6
bzip2 (I)	2.3	31.7	49.8	67.1	86.3	83.3	70.4	69.8
Extreme Thermal Stress (hot)								
eon (I)	2.3	33.2	36.3	100.0	84.1	84.0	71.6	69.8
crafty (I)	2.5	31.8	72.8	100.0	84.1	84.1	70.5	68.5
vortex (I)	2.6	32.1	28.3	100.0	84.5	84.4	70.8	68.3
gcc (I)	2.2	32.2	1.3	100.0	85.5	84.5	70.8	68.1
art (F)	2.4	38.1	6.3	100.0	87.3	87.1	75.5	68.1

Table 3.2: Benchmark summary. “I” = integer, “F” = floating-point. Fast-forward distance (FF) represents the point, in billions of instructions, at which warmup starts (see Sec. 3.4.5). [104]

### 3.4.5 Package, Warmup, and Initial Temperatures

The correct choice of convection resistance and heat-sink starting temperature are two of the most important determinants of thermal behaviour over the relatively short time scales than can be tractably simulated using SimpleScalar.

To obtain a useful range of benchmark behaviours for studying dynamic thermal management, we set the convection resistance manually. We empirically determined a value of 0.8 K/W that yields the most interesting mix of behaviours. This represents a medium-cost heat sink, with a modest savings of probably less than \$10 [114] compared to the 0.7 K/W convection resistance that would be needed without DTM. Larger resistances, *e.g.* 0.85 K/W, save more money but give hotter maximum temperatures and less variety of thermal behaviour, with all benchmarks either hot or cold. Smaller resistances save less money and bring the maximum temperature too close to 85° to be of interest for this study.

The initial temperatures that are set at the beginning of simulation also play a large role in thermal behaviour. The most important temperature is that of the heat sink. Its time constant is on the order of several minutes, so its temperature barely changes and certainly does not reach steady-state in our simulations. This means simulations must begin with the correct heat-sink temperature, otherwise dramatic errors occur. For experiments with DTM (except TT-DFS), the heat-sink temperature should be set to a value commensurate with the maximum tolerated die temperature (81.8° with our sensor architecture): the DTM response ensures that chip temperatures never exceed this

threshold, and heat sink temperatures are correspondingly lower than with no DTM. If the much hotter no-DTM heat-sink temperatures are used by mistake, we have observed dramatic slowdowns as high as 4.5X for simulations of up to one billion cycles, compared to maximum slowdowns of about 1.5X with the correct DTM heat-sink temperatures. The difference between the two heat-sink temperatures can be seen in Table 3.2. All our simulations use the appropriate values from this table.

Another factor that we have not accounted for is multi-programmed behaviour. A “hot” application that begins executing when the heat sink is cool may not generate thermal stress before its time slice expires. Rohou and Smith [85] used this to guide processor scheduling and reduce maximum operating temperature.

Other structures will reach correct operating temperatures in simulations of reasonable length, but correct starting temperatures for all structures ensure that simulations are not influenced by such transient artifacts. This means that after loading the SimpleScalar EIO checkpoint at the start of our desired sample, it is necessary to warm up the state of large structures like caches and branch predictors, and then to literally warm up HotSpot. When we start simulations, we first run the simulations in full-detail cycle-accurate mode (but without statistics-gathering) for 100 million cycles to train the caches—including the L2 cache—and the branch predictor. This interval was found to be sufficient using the MRRL technique proposed by Haskins and Skadron [44], although a more precise use of this technique would have yielded specific warmup intervals for each benchmark. With the microarchitecture in a representative state, we deal with temperatures. These two issues must be treated sequentially, because otherwise cold-start cache effects would idle the processor and affect temperatures. To warm up the temperatures, we first set the blocks’ initial temperatures to the steady-state temperatures calculated using the per-block average power dissipation for each benchmark. This accelerates thermal warmup, but a dynamic warmup phase is still needed because the sample we are at probably does not exhibit average behaviour in all the units, and because this is the easiest way to incorporate the role of the temperature dependence of leakage on warmup. We therefore allow the simulation to continue in full-detail cycle-accurate mode for another 200 million cycles to allow temperatures to reach truly representative values. Only

after these two warmup phases have completed do we begin to track any experimental statistics.

Note that, in order to have the statistics come from the program region that matches the Sherwood simulation points, the checkpoints must actually correspond to a point 300 million instructions prior to the desired simulation point. The “FF” column in Table 3.2 therefore shows where are checkpoints are captured, namely the fast-forward distance to reach the point where are warmup process begins.

### 3.4.6 Time Plots

To more clearly illustrate the time-varying nature of programs’ thermal behaviour, in Figure 3.4 we present a few plots of programs’ operating temperature (with no DTM) in each unit as a function of time. In each plot, the vertical line toward the left side of the plot indicates when the warmup period ends.

*Mesa* (Figure 3.4(a)) deserves special comment because it shows clear program phases. At each drop in its sawtooth curve, we found (not shown) a matching sharp rise in L1 and L2 data misses and a sharp drop in branch mispredictions. The rate of rise and fall exactly matches what we calculate by hand from the RC time constants. The temperatures are only varying by a small amount near the top of their range. So the increase in temperature occurs slowly, like a capacitor that is already close to fully charged, and the decrease in temperature is quite sharp, like a full capacitor being discharged.

At the other end of the spectrum is *art*, which has steady behaviour and therefore a flat temperature profile.

## 3.5 Results for DTM

In this section, we use the HotSpot thermal model to evaluate the performance of the various techniques described in Section 3.3. First we assume realistic, noisy sensors, and then consider how much the noise degrades DTM performance. The remainder of the section then presents results exploring the MC technique, lateral thermal diffusion and the role of initial heat-sink temperatures.

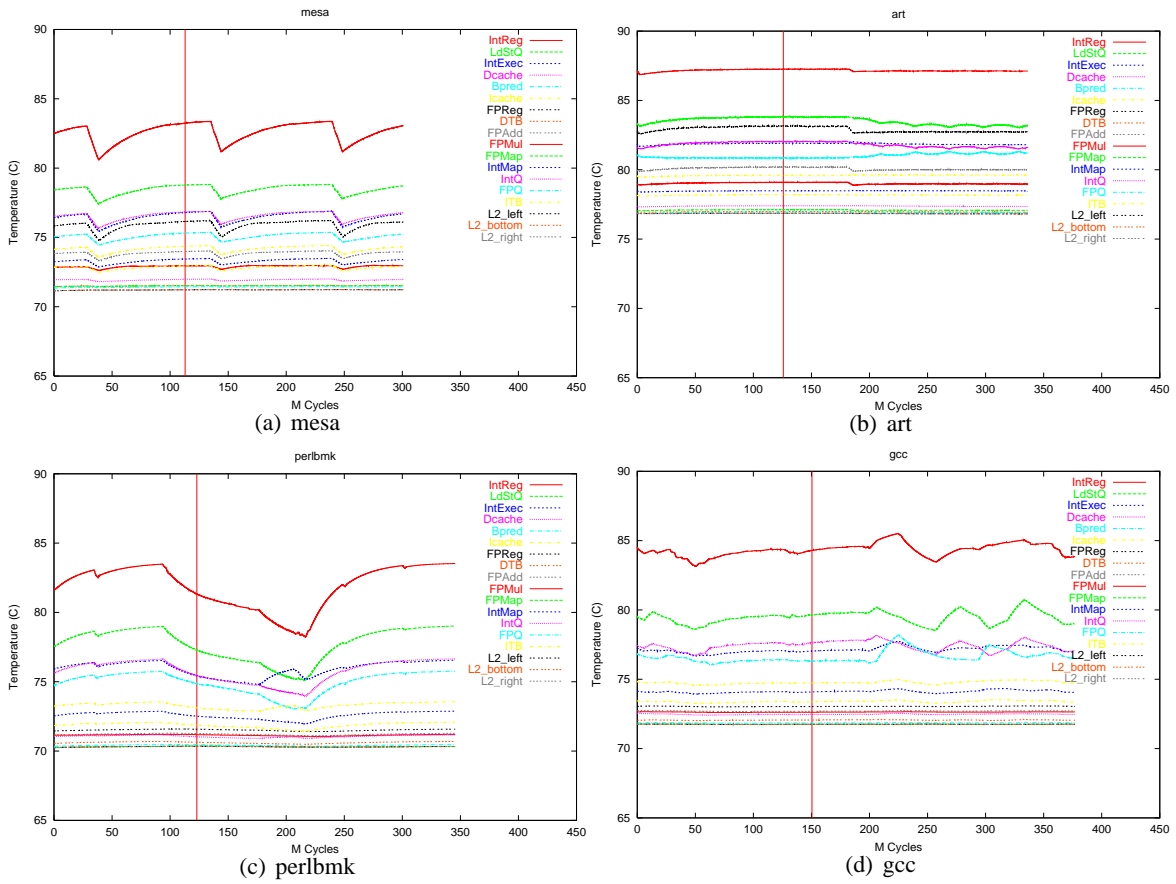


Figure 3.4: Operating temperature as a function of time (in terms of number of clock cycles) for various warm and hot benchmarks. [104]

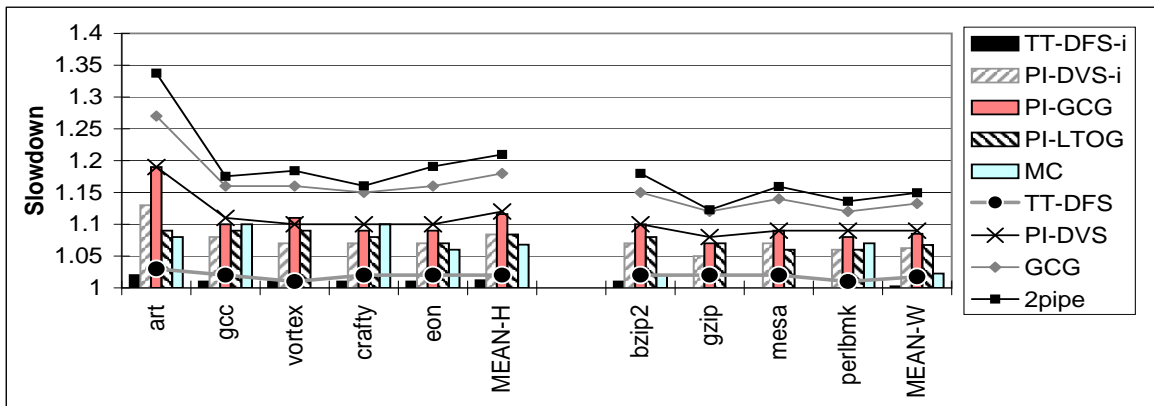


Figure 3.5: Slowdown for DTM. Bars: better techniques. Lines: weaker techniques. [104]



### 3.5.1 Results with Sensor Noise Present

Figure 3.5 presents the slowdown (execution time with thermal management divided by original execution time) for the “hot” and “warm” benchmarks for each of the thermal management techniques. The bars are the main focus: they give results for the better techniques: “ideal” for TT-DFS and PI-DVS, the PI-controller version of GCG, PI local toggling, and MC. The lines give results for non-ideal TT-DFS and the weaker techniques: non-ideal PI-DVS, GCG with no controller (*i.e.*, all-or-nothing), and 2pipe. None of the techniques incur thermal violations. Only the hot and warm benchmarks are shown; the two cold benchmarks are unaffected by DTM, except for mild effects with TT-DFS (see below).

The best technique for thermal management by far is TT-DFS, with the TT-DFS-i version being slightly better. The performance penalty for even the hottest benchmarks is small; the worst is *art* with only a 2% slowdown for TT-DFS-i and a 3% slowdown for TT-DFS. The change in operating frequency also reduces power dissipation and hence slightly reduces the maximum temperature, bringing *art* down to 87.0°. If the maximum junction temperature of 85° is strictly based on timing concerns, and slightly higher temperatures can be tolerated without unduly reducing operating lifetime, then TT-DFS is vastly superior because its impact is so gentle.

It might seem there should be some benefit with TT-DFS from *increasing* frequency when below the trigger threshold, but we did not observe any noteworthy speedups—even for TT-DFS-i with the coldest benchmark, *mcf*, we observed only a 2% speedup, and the highest speedup we observed was 3%. With TT-DFS, a few benchmarks actually experienced a 1% slowdown, and the highest speedup we observed was 2%. The reason for the lack of speedup with DFS is partly that the slope is so small—this helps minimize the slowdown for TT-DFS with warm and hot benchmarks, but minimizes the benefit for cold ones. In addition, for higher frequency to provide significant speedup, the application must be CPU-bound, but then it will usually be hot and frequency cannot be increased.

If the junction temperature of 85° is dictated not only by timing but also physical reliability, then TT-DFS is not a viable approach. Of the remaining techniques, MC, idealized DVS, and PI-LTOG are the best. MC with a one-cycle penalty is best for all but three applications, *gcc*, *crafty*,

and *perlbmk*, and the average slowdown for MC is 4.8% compared to 7.4% for DVS-i and 7.7% for PI-LTOG. Naturally, MC performs better if the extra communication latency to the spare register file is smaller: if that penalty is two cycles instead of one, MC's average slowdown is 7.5%. It is interesting to note that MC alone is not able to prevent all thermal violations; for two benchmarks, our MC technique engaged the fallback technique, PI-LTOG, and for those benchmarks spent 20-37% of the time using the fallback technique. This means that the choice of fallback technique can be important to performance. Results for these two benchmarks are much worse, for example, if we use DVS or GCG as the fallback.

Migrating computation and localized toggling outperform global toggling and non-idealized DVS, and provide similar performance as idealized DVS, even though DVS obtains a cubic reduction in power density relative to the reduction in frequency. The reason is primarily that GCG and DVS slow down the entire chip, and non-ideal DVS also suffers a great deal from the stalls associated with changing settings. In contrast, MC and PI-LTOG are able to exploit ILP.

A very interesting observation is that with MC, two benchmarks, *gzip* and *mesa*, never use the spare unit and suffer no slowdown, and *vortex* uses it only rarely and suffers almost no slowdown. The new floorplan by itself is sufficient to reduce thermal coupling among the various hot units in the integer engine and therefore prevents many thermal violations.

Although we were not able to explore a wider variety of floorplans, the success of these floorplan-based techniques suggests an appealing way to manage heat. And once alternate floorplans and extra computation units are contemplated, the interaction of performance and temperature for microarchitectural clusters [15] becomes an interesting area for further investigation. Our MC results also suggest the importance of modeling lateral thermal diffusion.

These results also suggest that a profitable direction for future work is to re-consider the tradeoff between latency and heat when designing floorplans, and that a hierarchy of techniques from gentle to strict—as suggested by Huang *et al.* [49]—is most likely to give the best results. A thermal management scheme might be based on TT-DFS until temperature reaches a dangerous threshold, then engage some form of migration, and finally fall back to DVS.

### 3.5.2 Role of Sensor Error

Sensor noise hurts in two ways; it generates spurious triggers when the temperature is actually not near violation, and it forces a lower trigger threshold. Both reduce performance. Figure 3.6 shows the impact of both these effects for our DTM techniques (for TT-DFS and DVS, we look at the non-ideal versions). The total height of each bar represents the slowdown with respect to DTM simulations with noise-free sensors and a trigger threshold of  $82.8^\circ$ . The bottom portion of each bar shows the slowdown from reducing the trigger by one degree while keeping the sensors noise-free, and the top portion shows the subsequent slowdown from introducing sensor noise of  $\pm 1^\circ$ . (For the warm and hot benchmarks, the impact of both these sensor-related effects was fairly similar.)

For TT-DFS the role of the different threshold was negligible. That is because the TT-DFS change in frequency for one degree is negligible. MC also experiences less impact from the different threshold. We attribute this to the fact that the floorplan for MC itself has a cooling effect and reduces the need for DTM triggers. Otherwise, lowering the trigger threshold from  $82.8^\circ$  (which would be appropriate if noise were not present) reduces performance by 1–3% for the other major techniques. 2pipe experiences a larger impact—4%—because it is so inefficient at cooling that it must work harder to achieve each degree of cooling.

The spurious triggers further reduce performance by 0–2% for TT-DFS; 3–6% for PI-GCG; by 6–8% for PI-DVS, with *art* an exception for PI-DVS at 11%; by 2–5% for LTOG; 0–4% for MC; and 4–9% for 2pipe. The higher impact of noise for DVS is due to the high cost of stalling each time a spurious trigger is invoked, and similarly, the higher impact of noise for 2pipe is due to the cost of draining the pipeline each time a spurious trigger is invoked.

Sensor error clearly has a significant impact on the effectiveness of thermal management. With no sensor noise and a higher trigger, DTM overhead could be substantially reduced: TT-DFS’s slowdown for the hot benchmarks moves from 1.8% to 0.8%, PI-DVS’s slowdown from 10.7% to 2.5%, PI-GCG’s slowdown from 11.6% to 3.6%, GCG’s slowdown from 17.9% to 6.6%, PI-LTOG’s slowdown from 8.4% to 5.2%, MC’s slowdown from 7.0% to 4.2%, and 2pipe’s slowdown from 21.0% to 9.5%. These results only considered the impact of sensor noise, the “S” factor. Reducing sensor offset—the “G” factor—due to manufacturing variations and sensor placement

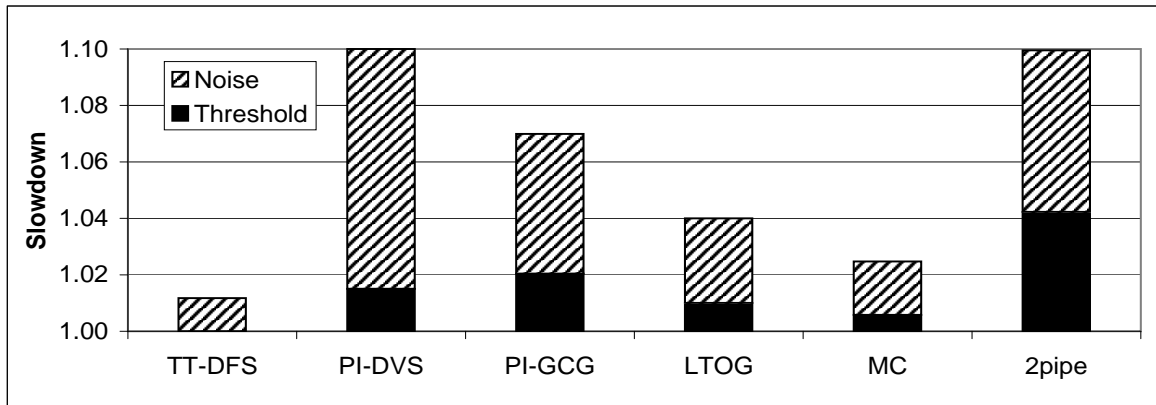


Figure 3.6: Slowdown for DTM from eliminating sensor noise, and from the consequent increase in trigger threshold to  $82.8^\circ$ . [104]

would provide substantial further improvements commensurate with the impact of the  $1^\circ$  threshold difference seen in the black portion of the bars.

Overall, our results also indicate not only the importance of modeling temperature in thermal studies, but also the importance of modeling realistic sensor behaviour. And finding new ways to determine on-chip temperatures more precisely can yield substantial benefits.

### 3.5.3 Further Analysis of MC

The MC technique, with local toggling as a fallback, merits further discussion in order to clarify the respective roles of floorplanning, migration, and the fallback technique.

If the MC floorplan is used without enabling the actual use of the migration, the spare register file is unused and has a mild cooling effect. The permutation of the floorplan also changes some of the thermal diffusion behaviour. This has negligible effect for most benchmarks, and actually mildly exacerbates hot spots for *perlbmk*, but actually is enough to eliminate thermal violations for *gzip*, *mesa*, and *vortex*.

When the MC technique is enabled, it is able to eliminate thermal violations without falling back to local toggling in all but two benchmarks, *gcc* and *perlbmk*.

Figure 3.5 reported the results for a single extra cycle of latency in accessing the spare register file. Figure 3.7 shows the effect of increasing this penalty to two cycles. It can be seen that a

two-cycle latency significantly increases the cost of MC, from an average of 4.9% to 7.5%. On the other hand, we do not fully model the issue logic, which should factor in this latency and wake instructions up early enough to read the register file by the time other operands are available on the bypass network. This makes both sets of results (one and two cycle penalties) pessimistic.

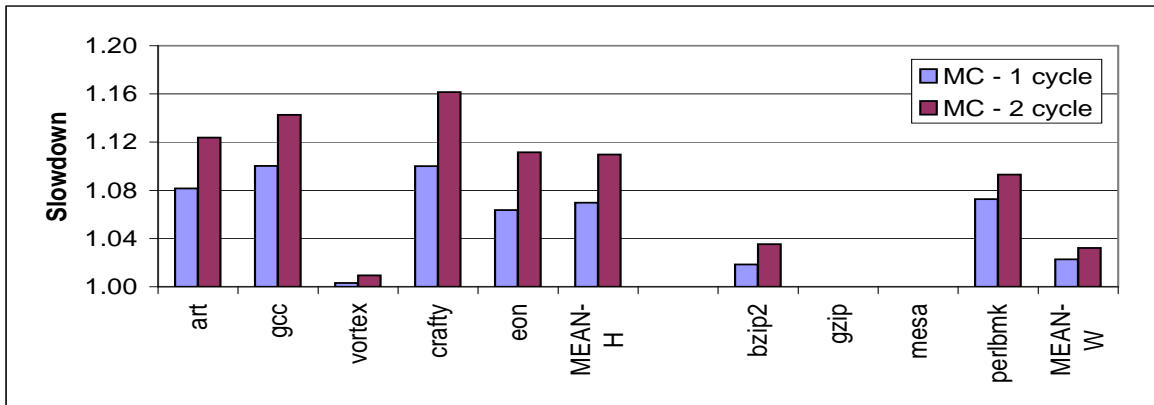


Figure 3.7: Slowdown for MC with 1- and 2-cycle penalties for accessing the spare register file. [104]

Other floorplans that accommodate the spare register file may give different results, and spare copies of other units may be useful as well, especially for programs that cause other hot spots. We have not yet had a chance to explore these issues. Another study we have not had a chance to perform is the cost-benefit analysis of whether the extra die area for the spare register file would be better used for some other structure, with purely local toggling as the DTM mechanism.

### 3.5.4 Importance of Modeling Lateral Thermal Diffusion

Lateral thermal diffusion is important for three reasons. First, it can influence the choice of a floorplan and the placement of spare units or clusters for techniques like MC or multi-clustered architectures. Second, it can have a substantial impact on the thermal behaviour of individual units. When a consistently hot unit is adjacent to units that are consistently colder, the colder units help to draw heat away from the hot unit. Failing to model lateral heat flow in situations like these can make hot units look hotter than they really are, overestimating thermal triggers and emergencies and potentially distorting conclusions that might be drawn about temperature-aware design. Third,

	loose-correct	tight-correct	loose-simple	tight-simple
art	1.00	1.00	1.00	1.00
gcc	1.00	1.00	1.00	1.00
vortex	1.00	1.00	1.00	1.00
crafty	1.00	1.00	1.00	1.00
eon	1.00	1.00	1.00	1.00
bzip2	0.68	0.76	0.90	0.90
gzip	0.67	0.72	0.91	0.91
mesa	0.42	0.58	0.71	0.71
perlbnk	0.31	0.36	0.39	0.39
facerec	0.00	0.00	0.00	0.00
parser	0.00	0.00	0.00	0.00

Table 3.3: Fraction of cycles in thermal violation (no DTM modeled) for the two different floorplans (loose and tight) with lateral thermal diffusion properly modeled (correct), and with lateral resistances omitted (simple). [104]

	loose-correct	tight-correct	loose-simple	tight-simple
art	1.00	1.00	1.00	1.00
gcc	1.00	1.00	1.00	1.00
vortex	1.00	1.00	1.00	1.00
crafty	1.00	1.00	1.00	1.00
eon	1.00	1.00	1.00	1.00
bzip2	1.00	1.00	1.00	1.00
gzip	1.00	1.00	1.00	1.00
mesa	0.87	0.94	1.00	1.00
perlbnk	0.45	0.47	0.52	0.52
facerec	0.00	0.00	0.00	0.00
parser	0.00	0.00	0.00	0.00

Table 3.4: Fraction of cycles above the thermal trigger point (no DTM modeled) for the two different floorplans. [104]

it turns out that failing to model lateral heat flow also produces artificially fast thermal rise and fall times, contributing to the overestimation of thermal triggers but also making DTM techniques seem to cool the hot spots faster than would really occur.

As a preliminary investigation of these issues, we compared the two floorplans shown in Figure 3.8. The “tight” floorplan in (a) places several hot units like the integer register file, integer functional units, and load-store queue near each other, while the “loose” one in (b) places the hottest units far from each other. In our experiments, we did not change any access latencies to account for distance between units in the two floorplans. This isolates thermal effects that are due to thermal diffusion rather than differences in access latency. We compared the thermal behaviour of

these floorplans using our full proposed model and also a modified version in which lateral thermal resistances have been removed.

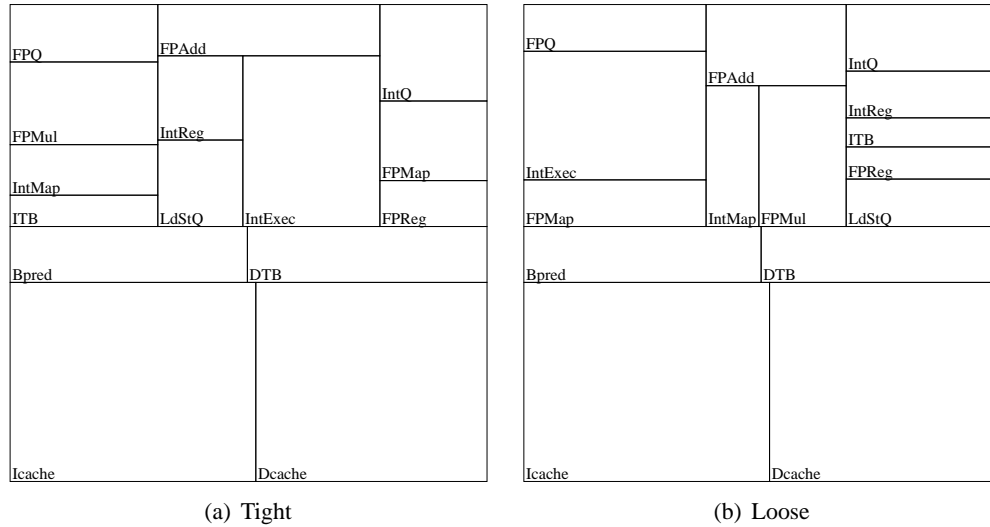


Figure 3.8: Two floorplans used to study effects of lateral thermal diffusion. [104]

Table 3.3 presents, for each floorplan, the fraction of cycles spent in thermal violation (no DTM was used for these experiments). The left-hand pair of columns present data obtained with the full model (“correct”), and the right-hand pair of columns present data obtained with the lateral resistances omitted (“simple”). Table 3.4 presents the fraction of cycles spent above the thermal trigger temperature.

Looking at the correct data, the distinction between the two floorplans is clear, with the tight floorplan spending more time at higher temperatures due to the co-location of several hot blocks. The tight floorplan will engage DTM more. A time plot for the integer register file of *mesa* is given in Figure 3.9.

The simplified model, on the other hand, fails in two regards. First, it predicts higher temperatures and higher frequencies of thermal violation, higher even than what is observed with the tight floorplan. This error happens because even the tight floorplan is able to diffuse away some of the heat in the hot blocks to neighboring blocks. This difference is largest for *gzip* and *mesa*. The artificially high temperatures mean that simulations of DTM will generate spurious thermal triggers and predict larger performance losses for DTM than would really be expected. Second, the failure

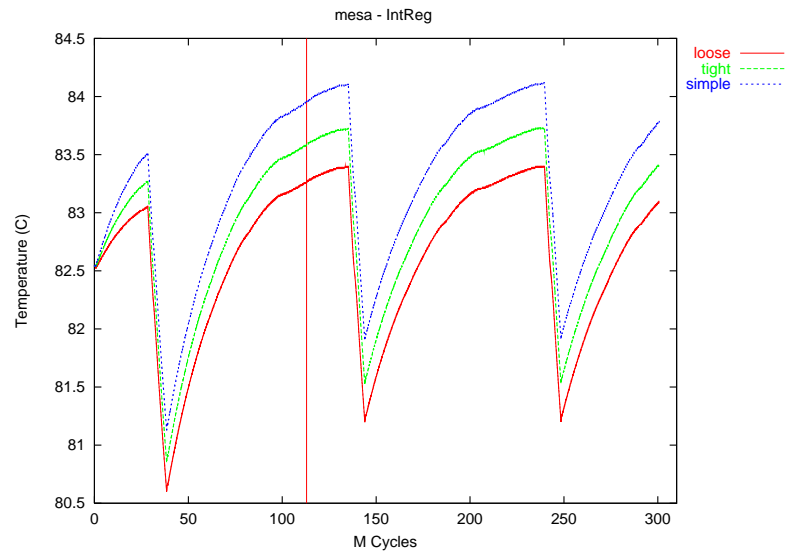


Figure 3.9: Temperature as a function of time for the integer register file with the mesa benchmark, for two different floorplans (tight and loose) and a simulation with lateral thermal resistance omitted (simple). [104]

to model lateral heat flow means that issues related to floorplan simply cannot be modeled, as seen by the fact that the two floorplans give identical results.

Without modeling lateral thermal diffusion, tradeoffs between thermal management and latency cannot be explored, and studies of dynamic thermal management may give incorrect results.

### 3.5.5 Role of Initial Heat-Sink Temperature

Finally, we wish to follow up on the point made in Section 3.4.5 that the choice of initial heat-sink temperature plays a major role, and the use of incorrect or unrealistic temperatures can yield dramatically different simulation results. Figure 3.10 plots the percentage error in execution time for various DTM techniques when the no-DTM heat-sink temperatures are used instead of the proper DTM heat-sink temperatures. The error only grows as the heat sink temperature increases. When we tried heat-sink temperatures in the 90°s, we observed slowdowns of as much as 4.5X.

The main reason this error is an issue is that microarchitecture power/performance simulators have difficulty simulating a benchmark long enough to allow the heat sink to change temperature and settle at a proper steady-state temperature, because the time constant for the heat sink is so large. Over short time periods, changes in heat-sink temperature effectively act as an offset to the



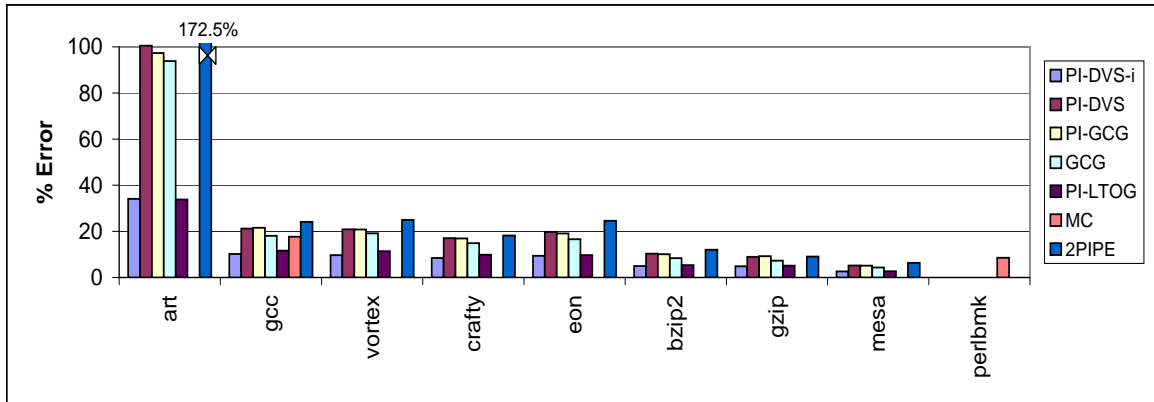


Figure 3.10: Percent error in execution time when DTM techniques are modeled using no-DTM heat sink temperatures. [104]

chip surface temperatures. That is why the correct steady-state temperature must be obtained before simulation begins. Thus for each DTM technique, floorplan, or trigger temperature, new initial temperatures must be determined. Developing simulation techniques or figures of merit to avoid this tedious task is an important area for future work. Fortunately, for all of our DTM techniques except MC, we found that the same initial “with-DTM” temperatures given in Table 3.2 were fine. MC’s use of a different floorplan requires a separate set of initial temperatures.

Because time slices are much smaller than the time constant for the thermal package, a multiprogrammed workload will tend to operate with a heat-sink temperature that is some kind of average of the natural per-benchmark heat-sink temperatures, possibly reducing the operating temperature observed with the hottest benchmarks and conversely requiring DTM for cold benchmarks. Thermal behaviour and the need for DTM will therefore depend on the CPU scheduling policy, which Rohou and Smith used to help regulate temperature in [85]. Combining architecture-level and system-level thermal management techniques in the presence of context switching is another interesting area for future work.

### 3.6 Conclusions and Future Work

Using HotSpot, this chapter evaluated a number of techniques for regulating on-chip temperature. When the maximum operating temperature is dictated by timing and not physical reliability concerns, “temperature-tracking” frequency scaling lowers the frequency when the trigger temperature

is exceeded, with average slowdown of only 2%, and only 1% if the processor need not stall during frequency changes. When physical reliability concerns require that the temperature never exceed the specification—85° in our studies—the best solutions we found were an idealized form of DVS that incurs no stalls when changing settings or a feedback-controlled localized toggling scheme (average slowdowns 7.4 and 7.7% respectively), and a computation-migration scheme that uses a spare integer register file (average slowdown 5–7.5% depending on access time to the spare register file). These schemes perform better than global clock gating, and as well as or better than the non-ideal feedback-controlled DVS, because the localized toggling exploits instruction-level parallelism while GCG and DVS slow down the entire processor.

A significant portion of the performance loss of all these schemes is due to sensor error, which invokes thermal management unnecessarily. Even with a mere  $\pm 1^\circ$  margin, sensor error introduced as much as 11% additional slowdowns, which accounted in some cases for as much as 80% of the total performance loss we observed.

These results make a strong case that runtime thermal management is an effective tool in managing the growing heat dissipation of processors, and that microarchitecture DTM techniques must be part of any temperature-aware system. But to obtain reliable results, architectural thermal studies must evaluate their techniques based on *temperature* and must include the effects of sensor noise as well as lateral thermal diffusion.

Future work in this area could explore new workloads and DTM techniques; a better understanding is needed for how programs' execution characteristics and microarchitectural behavior determine their thermal behaviour; and clever data-fusion techniques for sensor readings are needed to allow more precise temperature measurement and reduce sensor-induced performance loss. The sensor interpolation approaches of Chapter 6 are a step in this direction. Another important problem is to understand the interactions among dynamic management techniques for active power, leakage power, current variability and thermal effects, which together present a rich but poorly understood design space where the same technique may possibly be used for multiple purposes but at different settings. Finally, thermal adjacency was shown to be important, making a case for temperature-aware floorplanning and setting the stage for the next chapter.

# Chapter 4

## Static Thermal Management through Core-Level Floorplanning

---

### 4.1 Introduction

Since most DTM schemes involve stopping the processor clock or reducing its supply voltage, they have certain implications for a high-performance microprocessor. Firstly, in multi-processor server-based systems, this results in problems with clock synchronization and accurate time-keeping. Secondly, high performance, power-hungry, hot applications causing the DTM to be enabled are slowed down. This slowdown impacts systems offering real-time guarantees negatively as the slowdowns caused are unpredictable and could potentially lead to failures in meeting the computational deadlines. DTM schemes are designed as solutions to deal with the worst-case applications where the thermal package deals with the average case. However, as processors become hotter across technology generations, this average-case application behaviour itself tends to grow hotter causing reliability lapses and higher leakage. Hence, static microarchitectural techniques for managing temperature can complement what DTM is trying to achieve.

Orthogonal to the power density of the functional blocks, another important factor that affects the temperature distribution of a chip is the lateral spreading of heat in silicon. This spreading depends on the functional unit adjacency determined by the floorplan of the microprocessor. Traditionally, floorplanning has been dealt with at a level closer to circuits than to microarchitecture.

One of the reasons for this treatment is the level of detailed information floorplanning depends on, which is only available at the circuit level. However, with wire delays dominating logic delays and temperature becoming a first-class design constraint, floorplanning has started to be looked at even at the microarchitecture level. In this work, we investigate the question of whether floorplanning at the microarchitectural level can be applied viably towards thermal management. The question and the associated trade-off between performance and temperature are examined at a fairly high-level of abstraction. In spite of using models that are not necessarily very detailed, this chapter hopes to at least point out the potential of microarchitectural floorplanning in reducing peak processor temperature and the possibility of its complementing DTM schemes. It should be noted that floorplanning does not reduce the average temperature of the entire chip very much. It just evens out the temperatures of the functional units through better spreading. Therefore, the hottest units become cooler while the temperature of a few of the colder blocks increases accordingly. This aspect of floorplanning is particularly attractive in comparison with static external cooling. While cooling reduces the ambient temperature and hence the peak temperature on the chip, it does not reduce the temperature gradient across the chip. This can be bad from a reliability standpoint.

**Contributions** This chapter specifically makes the following contributions:

1. It presents a microarchitecture level thermal-aware floorplanning tool, HotFloorplan, that extends the classic simulated annealing algorithm for slicing floorplans [115], to account for temperature in its cost function using HotSpot [52]—a fast and accurate model for processor temperature at the microarchitecture level. HotFloorplan is a part of the HotSpot software release version 3.0 and can be downloaded from the HotSpot download site. The URL is: <http://lava.cs.virginia.edu/HotSpot>.
2. It makes a case for managing the trade-off between performance and temperature at the microarchitectural level. It does so by employing a profile-driven approach of evaluating temperature and performance respectively by using previously proposed thermal [52] and wire delay [3, 8, 78] models.

3. It finds that thermal-aware floorplanning reduces the hottest temperatures on the chip by a significant amount (about 20 degrees on the average and up to 35 degrees) with minimal performance loss. In fact, floorplanning is so effective that it eliminates all the thermal emergencies (the periods of thermal stress where temperature rises above a safety threshold) in the applications without the engagement of DTM.

The remainder of this chapter is organized as follows: Section 2 discusses the previous work in the area closely related to this chapter. Section 3 investigates the cooling potential of lateral spreading and presents it as the motivation for this work. Section 4 describes the thermal-aware floorplanning algorithm, the microarchitectural performance model used to study the delay impact of floorplanning and the simulation setup used in the evaluation of this work. Section 5 presents the findings of our research. Section 6 concludes the chapter and discusses possible future work.

## **4.2 Related Work**

Previous work related to this chapter falls into three broad categories—first is the wealth of classical algorithms available for floorplanning, second is the addressing of floorplanning at the architecture level for performance and the third is floorplanning for even chip-wide thermal distribution.

Since the research in classical floorplanning is vast and it is impossible to provide an exhaustive overview of the contributions in the field, we only mention a very small sample of the work related to our thermal-aware floorplanning algorithm. A more thorough listing can be found in VLSI CAD texts like [39, 92]. Many floorplan-related problems for general floorplans have been shown to be intractable. Even when the modules are rectangular, the general floorplanning problem is shown to be NP-complete [107, 77]. Hence, ‘sliceable floorplans’ or floorplans that can be obtained by recursively sub-dividing the chip into two rectangles, are most popular. Most problems related to them have exact solutions without resorting to heuristics. While more general and complex floorplan algorithms are available, this chapter restricts itself to sliceable floorplans because of their simplicity. For our work which is at the architectural level, since the number of functional blocks is quite small, sliceable floorplans are almost as good as other complex floorplans. The most widely

used technique in handling sliceable floorplans is Wong et al.'s simulated annealing [115]. It is easy to implement, is versatile in handling any arbitrary objective function and has been implemented in commercial design automation tools.

In the area of floorplanning at the architectural level for performance, Ekpanyapong et al.'s work on profile-guided floorplanning [33] and Reinman et al.'s MEVA [27], are works that we are aware of. Profile-guided floorplanning uses microarchitectural profile information about the communication patterns between the functional blocks of a microprocessor to optimize the floorplan for better performance. MEVA evaluates various user-specified microarchitectural alternatives on the basis of their IPC vs. cycle time trade-off and performs floorplanning to optimize the performance. In spite of dealing with architectural issues, it does so at a level close the circuit by specifying architectural template in structural verilog and architectural alternatives in a Synopsis-like '.lib' format. Both of these do not deal with temperature.

Thermal placement for standard cell ASIC designs is also a well researched area in the VLSI CAD community. [21, 25] is a sample of the work from that area. However, they target the "post-RTL" design stage and hence cannot be applied at early ("pre-RTL") design stages. Hung et al.'s work on thermal-aware placement using genetic algorithms [54] and Ekpanyapong et al.'s work on microarchitectural floorplanning for 3-D chips [32] are also close to the area of this work. Although genetic algorithms are a heuristic search technique that can be employed as an alternative to simulated annealing used in this chapter, [54] deals with a Network-on-Chip architecture (not microarchitectural floorplanning) and hence assumes the availability of more detailed, circuit-level information. [32] uses a Mixed Integer Linear Programming (MILP) approach and its objective function is not flexible enough to be easily adapted for improving the solution quality.

Parallel to our work [90], recently, Han et al. also published research on thermal-aware floorplanning at the microarchitectural level [43] in which they use the previously published Parquet floorplanner [2]. While their work also deals with the same topic as ours, it does not consider performance directly and uses the total wire-length of a chip as the proxy for performance. As we show in our results, this proxy could be misleading and could potentially result in a floorplan with shorter wire-length but worse performance. Also, as [43] does not compare the performance impact

of floorplanning with that of DTM, it does not answer the crucial question of why it is necessary to address temperature at the floorplanning stage even in the presence of DTM.

Thermal-aware floorplanning is also likely to be a well-researched topic in the industry, especially at the post-RTL design stage. However, we are unaware of any previously published work at the *pre-RTL* or microarchitectural level design stage.

Apart from the above-mentioned research, we would also like to mention the wire delay model and parameters from Brayton et al. [78] and Banerjee et al. [8] and the wire capacitance values from Burger et al [3]’s work exploring the effect of technology scaling on the access times of microarchitectural structures. We use these models and parameters in the evaluation of our floorplanning algorithm for calculating the wire delay between functional blocks.

### 4.3 Potential in Lateral Spreading

Before the description of the thermal-aware floorplanner, it is important to perform a potential study that gives an idea about the gains one can expect due to floorplanning. Since the cooling due to floorplanning arises due to lateral spreading of heat, we study the maximum level of lateral heat spreading possible. This is done using the HotSpot thermal model which models heat transfer through an equivalent circuit made of thermal resistances and capacitances corresponding to the package characteristics and to the functional blocks of the floorplan. In the terminology of the thermal model, maximum heat spreading occurs when all the lateral thermal resistances of the floorplan are shorted. Such shorting is equivalent to averaging out the power densities of the individual functional blocks. That is, instead of the default floorplan and non-uniform power densities, we use a floorplan with a single functional block that equals the size of the entire chip and has a uniform power density equal to the average power density of the default case. On the other extreme, we also make the thermal resistances corresponding to the lateral heat spreading to be equal to infinity. This gives us an idea of the extent of temperature rise possible just due to the insulation of lateral heat flow. The table below presents the results of the study for a subset of SPEC2000 benchmarks [106]. The ‘Min’ and ‘Max’ columns correspond to the case when the lateral thermal resistances are zero

and infinity respectively, while the ‘Norm’ column shows the peak steady-state temperature of the chip when the thermal resistances have the normal correct values.

Table 4.1: Peak steady-state temperature for different levels of lateral heat spreading ( $^{\circ}\text{C}$ ) [91]

Bench	Min	Norm	Max
bzip2	56	123	222
gcc	55	120	220
crafty	54	120	217
gzip	54	120	215
perlbmk	54	114	201
mesa	54	114	203
eon	54	113	201
art	55	109	188
facerec	52	104	183
twolf	51	98	168
mgrid	47	75	126
swim	44	59	84

Clearly, lateral heat spreading has a large impact on processor temperature. Though the ideal spreading forms an upper bound on the amount of achievable thermal gain, realistic spreading due to floorplanning might only have a much lesser impact because, the functional blocks of a processor have a sizeable, finite area and cannot be broken down into arbitrarily small sub-blocks that can be moved around independently. Hence, the maximum attainable thermal gain is constrained by the functional unit granularity of the floorplan. In spite of the impracticality of implementation, this experiment gauges the potential available to be tapped. Conversely, if this experiment indicated very little impact on temperature, then the need for the rest of this chapter would be obviated.

## 4.4 Methodology

### 4.4.1 HotFloorplan Scheme

The broad approach we take in this work is to use the classic simulated annealing based floorplanning algorithm [115]. The only difference is that the cost function here involves peak steady-state temperature, which comes from a previously proposed microarchitectural thermal model,



HotSpot [52]. Just like [115], HotFloorplan uses Normalized Polish Expressions (NPE) to represent the solution space of sliceable floorplans and uses three different types of random perturbation *moves* to navigate through them. The aspect ratio constraints for each functional block are represented as piecewise linear shape curves. For each slicing structure corresponding to an NPE, the minimum-area sizing of the individual blocks is done by a bottom-up, polynomial-time addition of the shape curves at each level of the slicing tree [92]. Once the sizing is done, the placement is then passed onto HotSpot for steady-state temperature calculation. It uses the profile-generated power dissipation values of each functional block and the placement generated by the current step of HotFloorplan to return the corresponding peak steady-state temperature. HotFloorplan then continues through the use of simulated annealing as the search scheme through the solution space.

This work uses a cost function of the form  $(A + \lambda W)T$  where  $A$  is the area corresponding to the minimum-area sizing of the current slicing structure,  $T$  is the peak steady-state temperature,  $W$  is the wire-length metric given by  $\sum c_{ij}d_{ij}$ ,  $1 \leq i, j \leq n$ , where  $n$  is the number of functional blocks,  $c_{ij}$  is the wire density of the interconnection between blocks  $i$  and  $j$ , and  $d_{ij}$  is the Manhattan distance between their centers.  $\lambda$  is a control parameter that controls the relative importance of  $A$  and  $W$ . As the units of measurement of  $A$  and  $W$  differ,  $\lambda$  is also used to match up their magnitudes to the same order. In our work,  $A$  is in the order of hundreds of square millimeters.  $W$  is in the order of tens of millimeters. So, just to match up the units,  $\lambda$  should be in the order of 0.01. We also find that assigning a relative importance of about 10% to the wire-length in general produced good floorplans. Hence, the value of  $\lambda$  we use is  $0.01 \times 10\% = 0.001$ . It is to be noted that the cost function is a product of two terms. The  $(A + \lambda W)$  term is the same as in the original floorplanning algorithm [115]. The new  $T$  term has been included as a product term instead of a sum term because we found the cost function to have a better gradation in value for different floorplans when  $T$  was included in the product term than when it was included in the sum term. This enabled the floorplanner to find more optimized solutions.

There are two floorplanning schemes that we evaluate. The first, called as *flp-basic*, is a scheme where all the microarchitectural wires modeled are given equal weightage, i.e.,  $c_{ij} = 1, \forall i, j$ . In the second, called as *flp-advanced*, the weights  $c_{ij}$  are computed in such a way that  $W = \sum c_{ij}d_{ij}$  is

proportional to an estimate of the slowdown in the execution time of the application when run on the floorplan being evaluated, in comparison to one with a default floorplan.

For the simulated annealing, we use a fixed ratio temperature schedule such that the annealing temperature of a successive iteration is 99% of the previous one. Initial annealing temperature is set such that the initial move acceptance probability is 0.99. The annealing process is terminated after 1000 iterations or after the annealing temperature becomes lesser than a threshold, whichever is earlier. The threshold is computed such that the move rejection probability at that temperature is 99%.

#### 4.4.2 Wire Delay Model

Thermal-aware floorplanning algorithms are faced with an interesting temperature-performance trade-off. While separating two hot functional blocks is good for thermal gradient, it is bad for performance. To manage this trade-off during the design stage at the microarchitectural level, it is essential to have a wire delay model that is detailed enough to accurately indicate the trend of important effects and at the same time, simple enough to be handled at the architecture level. In our work, such a model is essential for evaluating the performance trade-off of the thermal-aware floorplans generated. In the *flp-advanced* scheme mentioned above, such model is also necessary during the profile-driven floorplanning phase to convert the critical wire-lengths of the floorplan into actual performance estimates. Hence, we use a previously proposed, simple, first-order model for wire delay [8, 78]. We assume optimal repeater placement and hence, wire delay becomes a linear function of wire-length. The equation from [78] that gives the wire delay for an interconnect of length  $l$  segmented optimally into segments each of size  $l_{opt}$ , is given by

$$T(l) = 2l\sqrt{rcr_o c_o} \left( b + \sqrt{ab \left( 1 + \frac{c_p}{c_o} \right)} \right) \quad (4.1)$$

where  $r_o$ ,  $c_o$  and  $c_p$  are the resistance, input and parasitic output capacitances of a minimum-sized inverter respectively.  $a = 0.7$ ,  $b = 0.4$  and  $r$  and  $c$  are the resistance and capacitance of the wire per unit length respectively. We use the equation for  $l_{opt}$  and its measured values for a global

130 nm wire (2.4 mm) from [8] and also assume that  $c_o = c_p$ . We then obtain the values of  $r$  and  $c$  for global and intermediate level wires at the 130 nm technology node from [3]. Using these and the equation for  $l_{opt}$ , we obtain the  $l_{opt}$  value for intermediate level wires also (since  $l_{opt}$  only depends on  $\sqrt{rc}$  of that metal layer), which is found to be 1.41 mm. Using the above mentioned equation and constants derived from previously published works, we compute the delay of a global or intermediate level wire, given its length for the 130 nm technology node. Assuming a clock frequency of 3 GHz, using this model, the delay of a 5 mm wire amounts to 1.69 cycles at the global layer and 2.88 cycles at the intermediate layer.

#### 4.4.3 Simulation Setup and Evaluation

The microarchitectural performance model we use is a derivative of the SimpleScalar [14] simulator, the power model is a derivative of Wattch [13] and the thermal model used is the *block model* of HotSpot version 3.0 [52]. The basic processor architecture and floorplan modeled is similar to [103], i.e., closely resembling the Alpha 21364 processor. The leakage power model is also similar to [103] which uses ITRS [98] projections to derive the empirical constants. The differences are mentioned here. This chapter uses a later version of HotSpot which additionally models an interface material of thickness  $75\mu$  between the die and the heat spreader. Further, the package thermal resistance is 0.1 K/W and the ambient temperature is at  $40^\circ$  C. The threshold at which the thermal sensor of the processor engages DTM (called the trigger threshold) is  $111.8^\circ$  C while the absolute maximum junction temperature that the processor is allowed to reach with DTM (called the emergency threshold) is  $115^\circ$  C. The floorplan similar to Alpha 21364 processor core is scaled to 130 nm and is located in the middle of one edge of the die. Figure 4.1 shows this base processor floorplan. The entire die size is 15.9 mm x 15.9 mm while the core size is 6.2 mm x 6.2 mm. In other words, the Manhattan distance between diagonally opposite corners of the core is 4.21 cycles if a signal travels by a global wire while 7.16 cycles when it travels by an intermediate level wire. The floorplanning schemes mentioned above operate on the set of blocks shown in Figure 4.1. For the purposes of the floorplanning algorithm, all the core blocks are allowed to be rotated and the maximum allowed aspect ratio is 1:3 except when the aspect ratio of a block in the base processor

floorplan is itself greater than that. In that case, the aspect ratio of the block in the basic floorplan, rounded to the nearest integer, forms the upper limit on the allowable aspect ratio. Moreover, for this chapter, HotFloorplan operates only upon the core functional blocks. Once the core floorplan has been computed, the L2 cache is just wrapped around it so as to make the entire die a square.

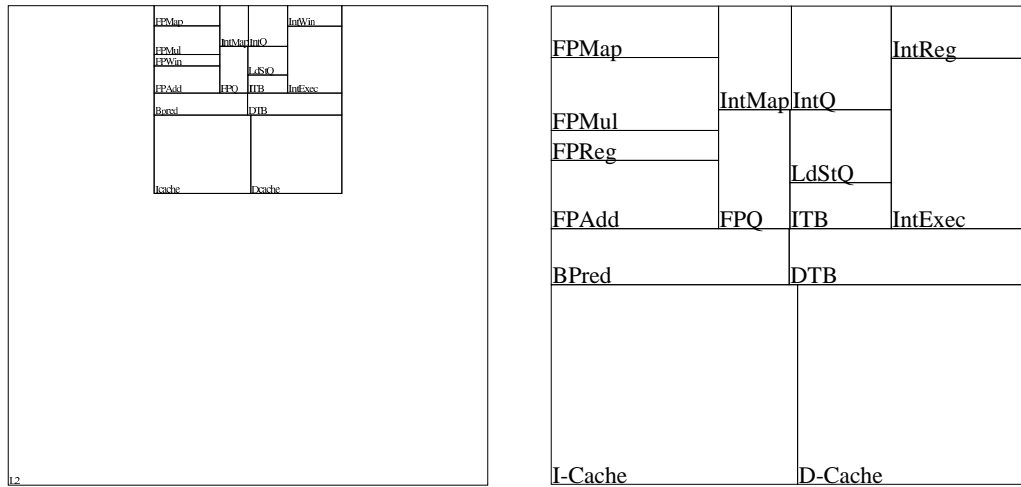


Figure 4.1: (a) The basic floorplan corresponding to the 21364 that is used in our experiments. (b) Close-up of the core area. [91]

The first step of our thermal-aware floorplanning is profiling to obtain the average power dissipation value for each of the functional blocks. Hence, we use a set of twelve benchmarks from the SPEC2000 [106] benchmark suite and obtain the power values through simulation. These values are then averaged across all benchmarks and a single set of power dissipation numbers (one value corresponding to each functional block) is passed onto the floorplanner. The floorplanner uses it to find a thermally-optimized floorplan. During this profiling phase, the benchmarks are run with the *train* input set. After the floorplanning phase, the floorplans generated are evaluated in the evaluation phase. These evaluation runs also use the same set of twelve benchmarks but the *reference* input set is used as opposed to the *train* input set. A subset of benchmarks from the SPEC2000 suite was chosen so as to minimize the simulation time. However, the choice was made carefully to exclude any bias. Of the 12 benchmarks, 7 are integer benchmarks and 5 are from the floating-point suite. They form a mixture of hot and cold, power hungry and idle benchmarks. The list of benchmarks and their characteristics is shown in Table 4.2. Whether it is from the integer or floating-point

suite is indicated alongside the benchmark name. This data is gathered using the reference input set and the benchmarks use the default Alpha-like floorplan. The temperatures shown are transient values across the entire run of the benchmark. In these reference runs and also in the evaluation runs, all the benchmarks are simulated for 500 Million instructions after an architectural warmup of 100 Million instructions and a thermal warmup of 200 Million instructions. It is to be noted that like [103], this thermal warmup is after setting the initial temperatures of the chip and package to the per-benchmark steady-state values. Hence, any possibility of a thermal cold-start is eliminated. Also, like [103], the simulation points for the reference runs are chosen using the SimPoint [96] tool. It is also worth mentioning that the profile runs are mainly to obtain the average power dissipation values and as will be explained below, the performance impact of the wires. The thermal modeling in those runs is only to ensure accurate calculation of the leakage power dissipation. Furthermore, these profile runs do not use simulation points from the SimPoint tool but are just simulated after fast-forwarding for 2 Billion instructions to remove unrepresentative startup behaviour. Like the evaluation runs, these simulations are also run for 500 Million instructions after an architectural warmup of 100 Million instructions and a thermal warmup of 200 Million instructions.

Table 4.2: Benchmark characteristics [91]

Bench.	IPC	Average Power (W)	Average Temp. (°C)	Peak Temp. (°C)
bzip2 (I)	2.6	42.2	81.7	127.1
gcc (I)	2.2	39.8	79.3	121.4
gzip (I)	2.3	39.3	79.1	122.1
crafty(I)	2.5	39.3	79.0	120.0
eon(I)	2.3	38.6	79.0	113.5
art(F)	2.4	41.9	78.7	109.9
mesa(F)	2.7	37.4	78.2	114.6
perlbnk(I)	2.3	37.1	76.9	117.3
facerec(F)	2.5	33.6	74.4	107.5
twolf(I)	1.7	28.8	68.6	98.6
mgrid(F)	1.3	19.6	61.2	77.6
swim(F)	0.7	11.2	51.6	59.8

In order to model the performance impact of floorplanning, this work models in SimpleScalar,

the delay impact of 13 major architecture level wires that connect the blocks of the floorplan shown in Figure 4.1. These are by no means exhaustive but attempt to capture the most important wire delay effects, especially with very little connectivity information available at the architectural level. Such performance modeling of wire delay is used not only during the evaluation runs (wherein, it is used to measure the IPC cost of the floorplans being evaluated) but also during the profiling phase of the *flp-advanced* scheme. For each of the major architectural wires considered, the extra delay on them is varied from 0 to 8 cycles. The train input set is used. The average of this data across all benchmarks is presented in Figure 4.2. The x-axis of the figure shows extra delay incurred due to a particular wire and the y-axis shows the resulting average performance slowdown. The slowdown is computed in comparison to the base Alpha 21364-like microarchitectural model. The *flp-advanced* scheme makes use of the summary information from this data to quantify the relative importance of the different wires and floorplans accordingly.

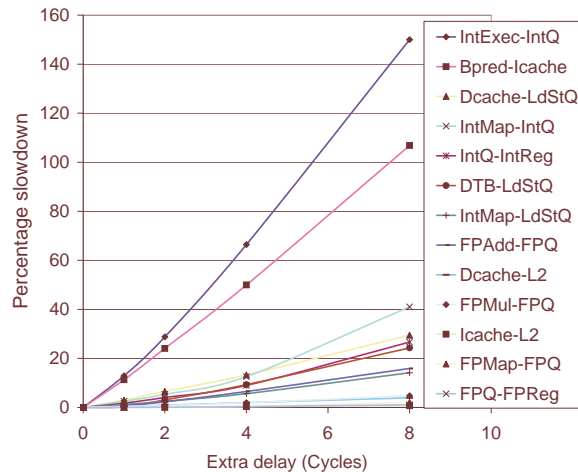


Figure 4.2: Performance impact of varying the delay on critical wires. [91]

Figure 4.2 clearly shows that some wires are more critical than others for performance. The *flp-advanced* scheme is designed to exploit this. Given a floorplan, its cost function tries to estimate its performance in comparison with the base Alpha-like floorplan. We do this normalization as a sanity check for our wire delay model. While the wire delay model we use might be accurate in terms of the time it takes for a signal to propagate through a wire of given length, it ignores routing information and metal layer assignment because of such details not being available at the

architectural level. So, we use the microarchitectural model itself as a sanity check for the wire delay model. For instance, in the base floorplan, assuming global wires, the wire delay model indicates that the delay between the FPMap and FPQ units is 1.13 cycles (2 cycles when rounded to the next higher integer). However, these units microarchitecturally correspond to the dispatch stage and it just takes 1 cycle in the Alpha 21364 pipeline for dispatch. Hence, when comparing performance of a floorplan with the base floorplan, for each of the 13 wires, we find the difference in the corresponding wire delays between the given floorplan and the base case. Only this difference, and not the actual wire delay indicated by the model, is rounded to the nearest higher integer cycle boundary and used in our performance model as the extra delay incurred. If the new floorplan has a wire shorter than the base case, it is ignored. This style of performance modeling is advantageous to the base floorplan but is also justifiably so because the base floorplan is derived from an actual processor and hence is most likely to be optimized in the best possible manner for performance. If a wire is longer in the base floorplan, it is still probably the optimal point for performance. Hence, we do not count the wires shorter in the floorplans generated by our schemes. Also, in order to deal with the issue of metal layer assignment, we take the approach of doing a sensitivity study with two extremes—all wires being global vs. all wires being intermediate. Studying these two extremes will show the best- and worst-case gains achievable by floorplanning.

The *flp-advanced* scheme uses the slowdown data from Figure 4.2 for its cost function. The performance slowdown is averaged across all the benchmarks. A simple linear regression analysis is performed on the averaged data and a straight line fit is made between the extra wire delay (in cycles) and the average performance slowdown for each wire. The slope of this regression line gives the average performance slowdown per cycle of extra delay on a wire. Assuming that the performance impact of different wires add up, a summation of these individual slowdowns can be used to obtain an estimate of the overall slowdown for the entire floorplan. That is, if  $s_k \forall k$  are the slopes of the regression lines of the wires under consideration, the performance slowdown for the entire floorplan is given by  $\sum s_k n_k$  where  $n_k$  is the number of cycles it takes for a signal to propagate through the particular wire. Since  $n_k$  varies linearly with the  $d_{ij}$  term (from the wire-length term of the cost function  $W = \sum c_{ij} d_{ij}$ ), it can be seen that assigning the weights  $c_{ij}$  proportional to  $s_k$  makes

$W$  proportional to an estimate of the performance slowdown due to the IPC loss of a floorplan.

The profile data about the power dissipation and the performance impact of wire delay are used by the floorplan schemes and floorplans optimized for their respective objective functions are produced. These floorplans are then evaluated for their thermal characteristics and performance. For the former, the HotSpot model is used and for the latter, the performance model developed in this work is used. In evaluating the performance impact of the floorplanning schemes, this chapter compares them with control theoretic DVS [103] as the DTM technique where the voltage is varied from 100% to 50% in ten discrete steps. The frequency is also changed accordingly. The time taken for changing the voltage and re-synchronizing the PLL is assumed to be  $10 \mu s$ . Two versions of DVS are modeled. In the first, called *dvs*, the processor stalls during the  $10 \mu s$  interval when the voltage is being changed. In the second, called *dvs-i* (for ‘ideal dvs’), the processor continues to execute albeit the new voltage becomes effective only after the  $10 \mu s$  period. Finally, we would like to mention that while the thermal-aware floorplanning is designed to reduce temperature, still a DTM scheme is required as a fall-back in case the temperature rises beyond the threshold even with floorplanning. This is also a reason why floorplanning is not a replacement for DTM but a complement to it.

## 4.5 Results

First, we present the floorplans selected by the *flp-basic* and *flp-advanced* schemes. Figure 4.3 shows the core floorplans. The dead space in the floorplans is 1.14% for *flp-basic* and 5.24% for *flp-advanced*, computed as ratios to the base core area. In case of the latter, the extra space is chosen by the floorplanner as a trade-off for maintaining both good thermal behaviour and performance. With current day microprocessors being more limited by thermal considerations than by area, we feel that a 5% overhead could be tolerated. There is a possibility that this extra area could be used for better performance. However, due to diminishing ILP, as processors are moving away from increasing single processor resources to more throughput-oriented designs, area is becoming less critical than the other variables. Also, since clock frequencies are limited today by the cooling



capacity of a processor, if floorplanning reduces the peak temperature significantly, then similar to the temperature-tracking dynamic frequency scaling scheme of [103], the clock frequency could probably be increased to compensate for, or even enhance the performance.

The aspect ratios of the entire core and the data cache is also interesting. Right now, the aspect ratio of the core is not constrained by any upper bound while that of the data cache is limited by a bound of 1:3. The fact that the floorplanner chooses such aspect ratios as shown in Figure 4.3 is interesting and suggests future work, both to explore the pros and cons of such aspect ratios from an implementation and performance perspective, and to continue refinement of the floorplanner.

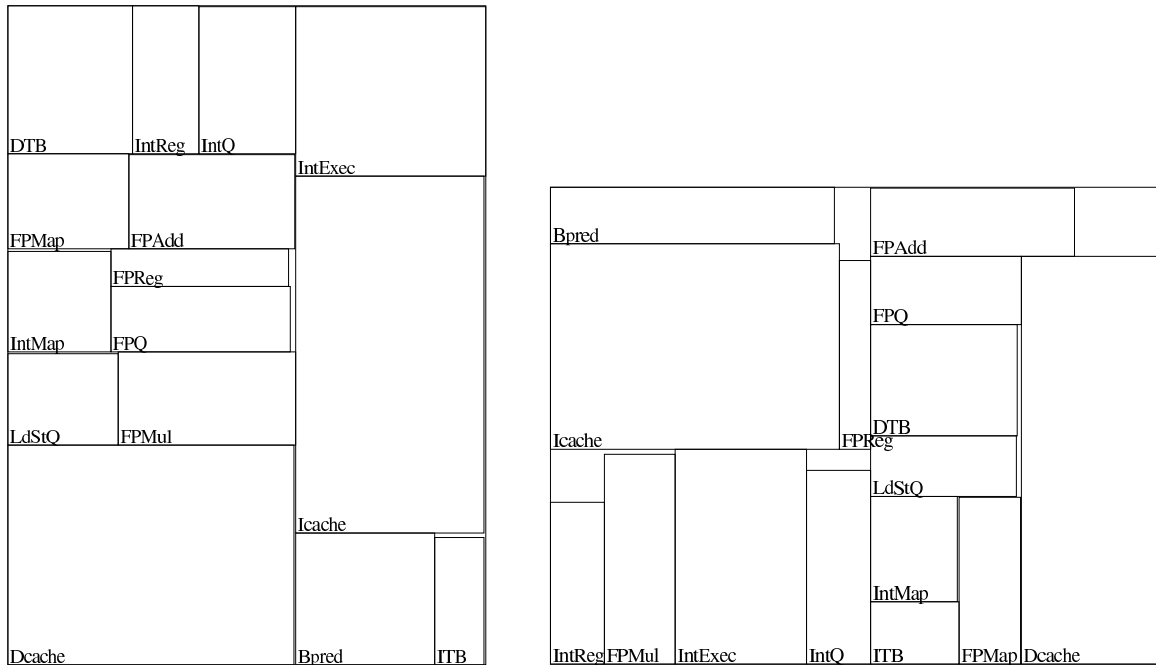


Figure 4.3: Floorplans generated by (a) *flp-basic* and (b) *flp-advanced* schemes. [91]

These floorplans are then analyzed using the wire model in comparison with the base floorplan. For the *flp-basic* floorplan, its weighing all the 13 wires equally has resulted in most wires being shorter than the base floorplan. The only longer wires are Bpred-Icache, DTB-LdStQ and IntMap-IntQ. The first two are longer by 1 cycle while the last is longer by 2 cycles irrespective of the assumption about the metal level of the wires (global vs. intermediate). The total wire-length of this floorplan is better than that of the base case by 12.7%. However, the longer wires are those critical to performance. In case of the *flp-advanced* scheme, five of the 13 wires are longer than the

base floorplan. They are: IntQ-IntReg, Dcache-L2, FPMul-FPQ, Icache-L2 and FPMaP-FPQ. All except the FPMul-FPQ interconnect are longer by 1 cycle, which is longer by 2 cycles. While the total wire-length of this floorplan is longer than the base floorplan by 13.7%, it can be seen from Figure 4.2 that these wires are less critical to performance than the wires longer in the *flp-basic* floorplan. This can be seen better when the performance results are shown.

Figure 4.4 shows the impact of our floorplan schemes on peak temperature. The leftmost bar in each group shows the base case. The middle bar shows the data for *flp-basic* and the rightmost one is for *flp-advanced*. The temperature reduction due to floorplanning occurs because of three main reasons. One is the lateral spreading of heat in the silicon. The second is the reduction of power density due to performance slowdown and the third is the reduction of leakage power due to the lowering of temperature. In order to decouple the effect of the later two from the first, each bar in the figure shows two portions stacked on top of each other. The bottom portions, called *basic* and *advanced* respectively, show the combined effect of all the three factors mentioned above. The top portions, called *basic-spread* and *advanced-spread*, show only the effect of spreading. This data is obtained by setting the power density of the new floorplans to be equal to that of the base case and observing the steady-state temperature for each benchmark. This data does not involve the performance model and hence the effects of slowdown and reduced leakage. It is to be noted that in the *basic* and *advanced* portions of the graph, we assume zero power density for the white spaces generated by our floorplanner. However, the results do not vary much when the white spaces are assigned a power density equal to the minimum power density on the chip (which is usually in the L2 cache). In fact, in the *basic-spread* and *advanced-spread* portions of the graph shown, we actually do assign power densities in such a manner.

It can be seen from the graph that with 115° C emergency threshold, all thermal emergencies have been eliminated by floorplanning itself, even if not accounting for power reduction due to performance slowdown and leakage reduction. Also, between *flp-basic* and *flp-advanced*, the latter shows better temperature reduction. This improvement is because of its increased area due to white spaces, which absorb the heat from hotter units. Also, it can be observed that a significant portion of the temperature reduction comes from spreading. *flp-basic* shows a larger reduction

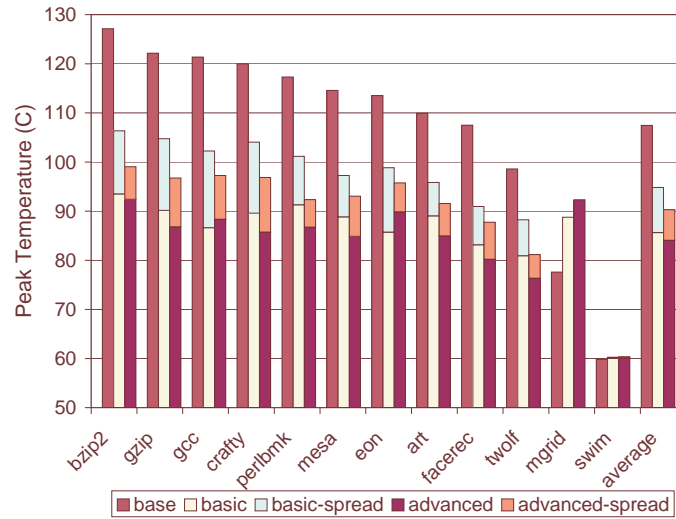


Figure 4.4: Impact of floorplanning on peak temperature. [91]

in temperature due to performance and leakage effects when compared to *flp-advanced*. As it will be shown later, this reduction is because the slowdown in performance itself is larger for that scheme. On the average, *flp-basic* and *flp-advanced* reduce peak temperature by 21.9 and 23.5 degrees respectively with 12.6 and 17.2 degrees respectively being just because of spreading. Since the peak temperatures with floorplanning are much lower than the emergency threshold, a careful increase in the processor clock frequency could compensate for the performance loss, still keeping the peak temperature within desired limits.

Figure 4.5 shows the performance impact of the schemes. The *flp-basic* and *flp-advanced* schemes are compared against *dvs* and *dvs-i*. The *advanced-g* and *advanced-i* bars correspond to the *flp-advanced* floorplan with global and intermediate metal layer assumptions respectively. The *basic* bar corresponds to the *flp-basic* scheme. There are no separate bars for different metal layer assumptions for *flp-basic* because the wire model's extra delay predictions fall into the same cycle boundary in both cases. The graph also shows a few other DVS schemes named in the format '*scheme-threshold*' where '*scheme*' is either *dvs* or *dvs-i* and the '*threshold*' is the thermal emergency threshold for the DTM scheme. While the normal emergency threshold is 115° C for our experiments, we show these additional data as a sensitivity study with respect to the threshold.

It can be seen from the graph that *flp-advanced* performs better than *flp-basic*. It should be

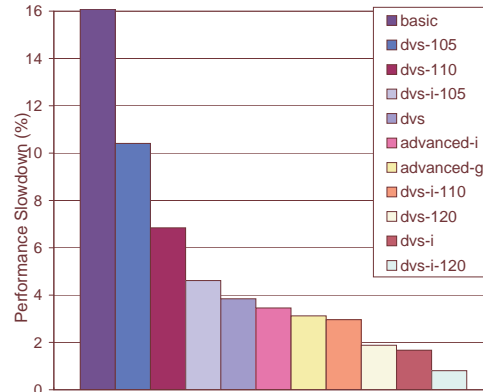


Figure 4.5: Performance slowdown of the various thermal management schemes. [91]

noted that its total wire-length is 30.3% worse than *flp-basic*. Still, its performance is better than *flp-basic* because of its high weightage for the critical wires. This fact also shows that a simple wire-length metric cannot be used as a proxy for performance when optimizing for temperature at the microarchitectural level. It could lead to an erroneous conclusion projecting a slower floorplan as the better one because of its shorter wire-length. *flp-advanced* is also quite competitive with the DTM schemes. It is slightly better than regular DVS and while worse than ideal DVS, is comparable to it. Even when the emergency threshold is reduced to 105° C, the performance of the floorplan schemes does not change because the peak temperature with floorplanning is well below that and the fall-back DTM is never engaged. However, changing the threshold affects the DTM schemes adversely. At a threshold of 105° C, even ideal DVS is worse than *flp-advanced*. In real processors, the threshold temperature is set based on considerations like the capability of the cooling solution, leakage, lifetime and reliability. It is designed to be well above the average-case peak temperature. As technology scales and as this average-case temperature itself increases, the gap between the threshold and the peak temperature gets smaller. The data shown in Figure 4.5 with threshold temperatures lower than 115° C aim to model this narrowing gap.

#### 4.5.1 Comparison against Static Voltage Scaling

While the above results compare the floorplanning schemes with a *DTM scheme* (feedback-controlled DVS), they do not explicitly compare them against *static* Voltage Scaling (VS). The

performance of the *advanced* floorplanning scheme was comparable to that of the idealized DVS DTM scheme. Hence, one would expect that it performs better than static VS since DVS adjusts the voltage at run-time while static VS is an offline setting.

As verification, let us consider the following ideal case: for the applications requiring DTM (7 out of the 12 benchmarks), let us scale down the voltage of the base-case floorplan. In this experiment, the per-benchmark scale down factor is assumed to be known *a priori* in an omniscient fashion. Comparing the percentage slowdown of such an experiment with the *advanced* floorplanning scheme, following are the results:

When the scaling performed is just enough to eliminate thermal emergencies, the slowdown is 3.39%. When the scaling performed is to match the peak temperature of the thermally-aware floorplan, the slowdown is 18.45%, while the slowdown due to floorplanning itself is 3.41%. This means that only under omniscient assumptions about the per-benchmark scaling factors to eliminate the thermal emergencies exactly, static VS matches the thermally-aware floorplan. Simply scaling the voltage to obtain the thermal benefits of floorplanning comes with a much higher performance cost. Thus, clearly, floorplanning performs better than static VS.

## 4.6 Conclusions and Future Work

This chapter presented a case for considering microarchitectural floorplanning for thermal management. It described HotFloorplan, a microarchitectural floorplanning tool that incorporates profile information to evaluate the temperature-performance trade-off early in the design stage. Results of this work show that there is a significant peak temperature reduction potential in floorplanning. In our experiments, all the thermal emergencies were removed by just floorplanning alone. A major part of this reduction comes from lateral spreading while a minor portion also comes from reduced leakage and slowed down execution. In comparison with a simple performance metric like the sum of the lengths of all wires, a profile-driven metric that takes into account the amount of communication and the relative importance of the wires reduces temperature better without losing much performance. In fact, the simple scheme results in a floorplan better in terms of total wire length (and temperature) but significantly worse in terms of performance. In order to optimize perfor-

mance and temperature, the profile-driven scheme trades off a third variable—area. A tolerable area overhead is used in reducing temperature significantly without compromising performance. In comparison with DVS DTM scheme, the profile-based floorplanning scheme performed competitively. The floorplanning scheme also performed much better than an offline static voltage setting.

As the gap between the average-case peak temperature and the thermal envelope is narrowing down, the performance impact of DTM is on the rise. A combination of floorplanning and DTM could address this issue effectively. By reducing the peak temperature, floorplanning can reduce the amount of time DTM is engaged, thereby also reducing the undesirable clock and real-time effects of DTM. Furthermore, since the peak temperature with floorplanning is significantly less than the emergency threshold, the small performance impact of floorplanning could possibly be compensated by an increase in processor frequency, still staying within the desired thermal limits. While floorplanning reduces temperature, it does not eliminate the need for DTM. Even with floorplanning, DTM is necessary as a failsafe option. Moreover, both DTM and floorplanning address two orthogonal issues of power density and lateral spreading. Hence, they can complement each other in achieving the same objective.

In our immediate future work, we would like to investigate the effect of constraining the aspect ratio of the entire core area in our floorplanning schemes and its impact on the magnitude of white space. However, as a more general future direction of research, we would like to study the effects of thermal-aware floorplanning in multicore architectures, which is the topic to be considered in the next chapter. This work has given an architectural framework to treat the area variable quantitatively. Such a treatment opens up many interesting venues of future exploration. One could research efficient ways of trading off area and more precisely, white space, against the design constraints of temperature, power and performance. Combining such research with existing DTM schemes or coming up with new DTM schemes that work collaboratively, taking into account the area variable, could be further fruitful directions of research.

# Chapter 5

## Thermal Benefit of Multicore Floorplanning

---

### 5.1 Introduction

Early approaches to the thermal management problem involved designing the thermal solution (heatsink, fan *etc.*) for the absolute worst-case application behaviour. These approaches have later been complemented by circuit and microarchitectural techniques that adaptively trade-off the performance of applications to suit the thermal needs of the microprocessor. Such Dynamic Thermal Management (DTM) techniques (*e.g.* [40,49,12,68,103,47]) allow for the thermal solution to be designed for the average-case rather than the worst-case, thereby saving cooling costs. Circuit-based DTM techniques involve either the scaling of the voltage and frequency of the microprocessor or the stopping of the processor clock. Although effective in dealing with temperature, such alterations to the clock are undesirable in server environments as they lead to problems in clock synchronization and accurate time-keeping. Moreover, with non-ideal threshold voltage scaling, such an ability to reduce the voltage might not be easily available. Furthermore, microarchitectural DTM techniques that delay an application in response to a thermal overshoot are problematic in real-time systems as they lead to unpredictable slowdowns and hence could lead to applications missing their deadlines. Hence, there have been research efforts to examine microarchitectural thermal management schemes that do not compromise the latency of applications unpredictably.

Apart from controlling the level of computational activity of an application, another way to handle the thermal management problem is through better distribution of heat in *space*. In a multi-

threaded environment, this can be accomplished by the scheduling of threads on the hardware substrate in a thermally-aware fashion to distribute heat evenly across the hardware. With the advent of multicore and multithreaded processors, this approach has received research interest [85, 80, 19, 23]. However, orthogonal to both these dynamic methods of thermal management (performance trade-off and scheduling), a static technique to distribute heat spatially is thermally-aware floorplanning at the microarchitectural level. For a single-core microprocessor, such thermally-aware floorplanning was investigated in the previous chapter. It is not only attractive because of its predictability (which is relevant for real-time applications), but also for its ability to complement the dynamic schemes since it is orthogonal to them.

Thermally-aware microarchitectural floorplanning has been studied for single-core processors [91, 43, 116, 18, 76]. However, multicore processors have become ubiquitous and they offer a spectrum of placement choices from the functional block level to the core level. Exploiting these choices for thermal benefit is the focus of this chapter. Apart from Healy *et. al.*'s research [45] that occurred parallel to this work and Donald and Martonosi's paper [31] that tried out alternative placement strategies in the context of thermal efficiency of Simultaneous Multithreading (SMT) and Chip Multiprocessing (CMP) architectures, we are not aware of previous work that addressed thermally-aware multicore floorplanning at the microarchitectural level. Specifically, this chapter makes the following contributions:

- It examines the thermal benefit in changing the relative orientation of cores in a homogeneous multicore chip so as to keep the hottest units of adjacent cores as far apart from each other as possible.
- Since second-level caches have much lower computational activity than the cores, they are among the coolest units in a processor. Hence, this work studies the placement of L2 banks between adjacent cores so that they can function as cooling buffers that absorb the heat from the cores.
- As an ideal case-study, it investigates the temperature reduction potential of multicore floorplanning by relaxing the requirements that functional blocks should stay within core bound-



aries and L2 cache banks should stay outside core boundaries.

The remainder of the chapter is organized as follows; Section 5.2 describes previous work related to this chapter. Section 5.3 explains our multicore floorplanning methodology. Section 5.4 presents the experimental results of our study and Section 5.5 concludes the chapter providing direction to possible future work.

## 5.2 Related Work

The work that is most related to this chapter is a parallel effort by Healy *et. al.* [45]. They also present a multicore floorplanner that optimizes the temperature profile of a microprocessor. They take a multi-granularity approach that considers both the within-core and across-core floorplans for thermal optimization. Their work employs a floorplanning algorithm that uses simulated annealing [60] based upon the sequence-pair representation [74] with a cost function incorporating both temperature and bus length. Although their work has the advantage of accounting for performance more accurately, we believe that our approach is complementary to it. The main innovation in their paper is to floorplan an individual core in a multicore-aware fashion with the functional blocks of the core moved inwards from the periphery of the core so that when the cores form a mosaic in the multicore chip, the tiling does not result in the hot blocks being adjacent to each other. On the other hand, our approach achieves the same end through different means: by changing the orientation of the cores and by placing second-level cache banks between them. Furthermore, as an academic exercise disregarding the feasibility of practical implementation, we perform a potential study of the achievable thermal benefit in multicore floorplanning by a) letting the functional blocks from different cores be close to each other crossing core boundaries and b) inserting L2 cache banks in between the functional blocks of a core. We believe that this is also a significant point of distinction. Moreover, our work also performs a sensitivity study on core size, core area as a fraction of chip area and L2 power density. Since these variables affect the thermal performance of a multicore floorplanner, it is important to consider them in the evaluation. Also, their work results in floorplans with dead spaces (which might be a consequence of using hard blocks), which is a costly

inefficiency in the area *vs.* temperature trade-off since silicon real estate is expensive. Finally, their work employs multiple floorplans for the same microarchitecture, which could lead to a significant replication of design effort for each kind of floorplan used.

Another paper that has considered multicore floorplanning in a limited fashion (with its primary focus being other issues) is from Donald and Martonosi [31]. When studying the thermal efficiency of SMT and CMP architectures, they try an alternative layout strategy to reduce temperature by moving the two cores of a CMP apart, from the center of the chip to its edge. This is similar to the use of L2 cache banks in our work as cooling buffers. However, they approach it as a one-off technique without any generic extensions or applicability to arbitrary floorplans. Since temperature is a serious concern for 3-D architectures, thermally-aware floorplanning for 3-D chips is relevant to our work [32, 55]. Similarly, the wealth of work in thermally-aware placement for ASICs and SoCs (*e.g.* [25, 21, 54, 41]) and microarchitectural floorplanning for thermal optimization [91, 43, 116, 76, 18] is also related to this work. However, none of these papers study the placement choices in multicore architectures.

## 5.3 Methodology

As multicore architectures have become the norm today, there are many levels of placement choices available for a designer—from the functional block level to the core level. These choices can be exploited to spatially distribute heat effectively. Specifically, following are some of the possibilities we consider for a homogeneous CMP:

### 5.3.1 Core Orientation

The advantage of having identical cores in a CMP is physical design re-use. A single core can be designed once and re-used many times. In such homogeneous cores seen today, the orientation of the cores is such that their floorplans are mirror images of each other. This arrangement typically leads to hot functional blocks being adjacent to each other and oriented towards the center of the core. Figure 5.1(a) illustrates the thermal profile of such an arrangement for a homogeneous four-

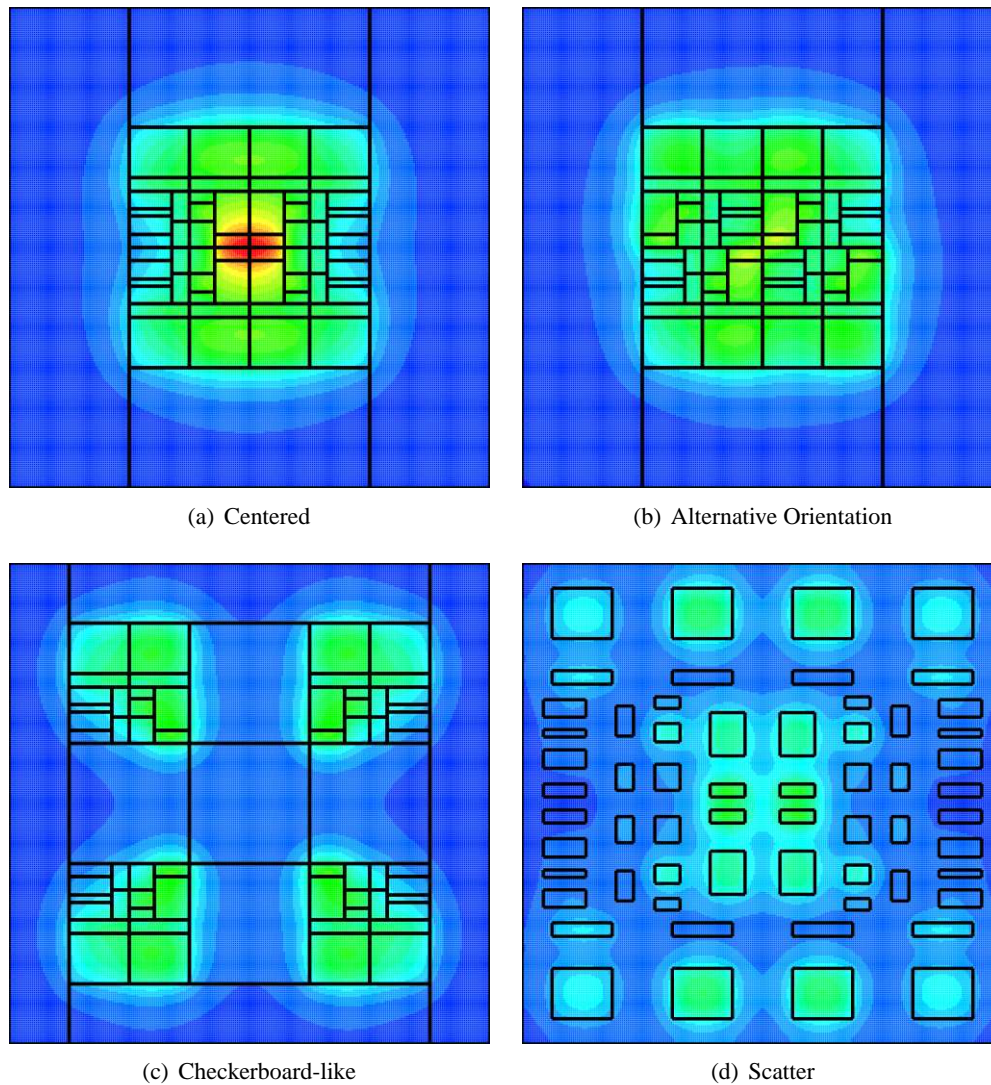


Figure 5.1: Illustration of different core arrangements for a four-way CMP with each core resembling an Alpha 21364. (a) shows a typical floorplan with hot units adjacent to each other. (b) shows a floorplan with alternative orientations of the cores. (c) shows a checkerboard-like arrangement with the use of L2 cache banks as cooling buffers between the cores. (d) shows an arrangement used for a potential study with functional blocks scattered amidst the cache banks. The adjacency of the blocks in the original floorplan is maintained through the scattering. [89]

way CMP with each core resembling that of an Alpha 21364 as in [103]. Without compromising the re-usability of the cores, a simple temperature-aware floorplanning scheme would be to experiment with the orientation of the cores for temperature reduction. Fig 5.1(b) illustrates such a floorplan with alternative orientations of the cores that result in reduced peak temperature. Specifically, the

cores on the bottom right and top left are flipped about the vertical axes passing through their respective centers.

Each core can have eight different orientations. They are the four rotational symmetries and their corresponding four reflections (mirror images). Hence, for a  $k$ -way CMP, there are a total of  $8^k$  different possible floorplans. For a small number of cores (*e.g.*  $k \leq 6$ ), this is still within the limits of a brute-force search. Given a set of representative power numbers (which can be obtained through application profiling), we use the HotSpot [103] thermal model to obtain the corresponding thermal profile and choose the floorplan which offers the lowest peak temperature. Once the number of cores crosses the limit of a brute-force search, we employ simulated annealing [60] to search through the vast solution space. Each floorplan is encoded as a  $k$ -digit octal number denoting the set of core orientations it is comprised of. The only type of move used in the annealing schedule is a random increment move where a random digit is chosen from the  $k$ -digit string and incremented (modulo 8) to the next numerical orientation.

### 5.3.2 L2 Cache Banks

As second-level caches are large, they are already partitioned into many banks. Furthermore, their power density is quite low because of relatively infrequent accesses. Hence, their temperatures are usually among the lowest in a chip. So, a possible strategy for temperature reduction is to use the L2 banks as cooling buffers between cores. However, in doing so, the performance cost of a longer L2 bus must be accounted for. Since we assume a traditional L2 cache with Uniform Cache Access (UCA), the L2 latency already includes the worst-case latency from a core to the farthest L2 bank. Thus, in placing the cache banks between cores, the latency increase is only proportional to the maximum extra distance a core moves within the chip due to the cache-bank insertion. For the floorplan configurations considered in this chapter, a part of the L2 always wraps around the periphery of the chip. In such a scenario, for the range of L2-area to chip-area ratios we consider (25-85%), a core can move an extra distance between 7 and 44%. Assuming a linear wire delay model similar to Chapter 4, this implies the same percentage increase in L2 latency too. Since L2 latency is tolerated well by the microarchitecture due to the presence of L1 caches that filter out

most of the accesses, this translates to less than 2% slowdown for SPEC2000 benchmarks [91, 18]. Hence, in this chapter, we consider the performance impact of distributing the L2 cache banks between the cores to be negligible. However, it is to be noted that our simulation setup involves running identical benchmarks on all the cores with no communication between them. Although this modeling methodology is a limitation of this work, we believe it is not a serious one because of two reasons. First, in comparison with an arrangement of the cores adjacent to each other, the cache insertion provides extra space for routing in the vicinity of the cores (over the sub-arrays of the cache). This space could be used to reduce the latency of the interconnection network by using thicker wires, thus minimizing the impact of the latency on coherence traffic. Second, for a large number of cores, Non-Uniform Cache Access (NUCA) is the likely choice and since it already envisions an interconnection network with a distributed arrangement of the cores and the cache banks [59], the performance of a cache-bank inserted layout as suggested in this work is not likely to be much different.

In exploring the placement of the cache banks, we first assume that their size and aspect ratio are flexible. Then, the processor cores and L2 blocks could be arranged to tile the entire silicon real estate in a checkerboard-like fashion to increase the lateral spreading of heat. Since silicon acts as a spatial low-pass filter for temperature [51], maximizing the spatial frequency of the power density distribution is beneficial for temperature reduction. For a checkerboard-like tiling of a multicore chip, this is accomplished by making the high power areas (cores) as small as possible (by separating the cores from one another) and the low power areas between them (caches) as large as possible. Furthermore, since the chip boundaries allow lateral heat conduction only in two or three directions (instead of four), this also means that the cores should be placed away from the chip boundaries to facilitate heat spreading. A sample of such an arrangement is shown in Figure 5.1(c). Please note that although we use the term *checkerboard-like*, the cache-bank insertion is in essence separating the cores in *both* the  $x$  and  $y$  directions with cache banks. In Figure 5.1(c), a true checkerboard arrangement would have had another core in the middle square. However, in this chapter, we use the term *checkerboard-like* for the lack of a better alternative. Also, for determining the positions of the cores and the cache banks, we assume (heuristically) that adjacent cores are

equidistant from each other and that the distance between a peripheral core and chip boundary is half the distance between adjacent cores.

### 5.3.3 Hierarchical Floorplanning

Since the checkerboard-like arrangement separates the cores from one another, it reduces the lateral thermal coupling between them. This affords us a possibility of floorplanning the functional blocks of the cores independent of their multicore arrangement. Hence, we apply a hierarchical floorplanning algorithm combining the single-core floorplanner from Chapter 4 with both of the above techniques (orientation and tiling). Given a set of functional blocks and their area and aspect ratio constraints, it first floorplans the core using the classic Wong and Liu [115] simulated annealing algorithm with a cost function that includes area, delay and temperature. In assigning the relative importance to architectural wires, we use the modifications suggested by [18] instead of the order proposed in Chapter 4. This single-core floorplan is then arranged in a checkerboard-like fashion with L2 cache banks arranged in between the cores as described in Section 5.3.2. Then, as a final step, the orientation space of the cores is searched using simulated annealing as described in Section 5.3.1.

### 5.3.4 Potential Study

Finally, as a theoretical exercise without much regard to its practical feasibility, we investigate a floorplanning strategy that allows for complete flexibility in the placement of functional blocks even disregarding the core boundaries. Since this strategy compromises design re-use and performance at all levels, it is not presented here as a practical technique. Such an experiment is useful just as a potential study to measure against the performance of the other techniques mentioned above. We look at two possibilities: in the first, we apply the single-core floorplanning algorithm from Chapter 4 to a combined list of all the functional blocks in the entire multicore chip. This scheme basically results in a medley of functional blocks from different cores occupying the center of the multicore chip and surrounded by the L2 cache banks. Compared against the alternative core orientation strategy mentioned in Section 5.3.1, it tells us how much thermal potential is available

in distributing the functional blocks within the cores. The second possibility we look at is to insert the L2 cache banks even between the functional blocks within a core. Such insertion results in the scattering of the functional blocks through out the entire chip. In the process of scattering, the adjacency of the functional blocks is retained as it was in the original core floorplan. This scheme is illustrated in Figure 5.1(d). It is useful to compare the figure against the non-scattered version in Figure 5.1(a) and the core-level cache inserted version in Figure 5.1(b). Such a scattering serves as a benchmark for the L2 cache insertion technique described in Section 5.3.2 to measure against. However, the performance of such a scattered floorplan is likely to be significantly worse than a floorplan in which closely communicating functional units are adjacent to each other. This is especially true if the delay on the architectural wires involved in critical loops increases due to the scattering [11]. Hence, we only consider this scheme to understand its potential for temperature reduction and not as a realistic approach.

### 5.3.5 Experimental Setup

In order to evaluate the floorplanning choices described above, we use a simulation infrastructure comprised of the HotSpot [103] thermal model, Wattch [13] power model and SimpleScalar [5] performance model. We use the SPEC2000 [106] benchmark suite and run each benchmark for an interval of 500 million instructions using the reference inputs. The interval that is most representative of the entire program is identified using the SimPoint [96] tool. We extend the HotFloorplan tool from Chapter 4 to implement the multicore floorplanning strategies described in Section 5.3. The floorplanner is fed with the floorplan of a single core and a representative set of power values for each of its functional blocks (which we compute as the average of the power values of all the benchmarks simulated as mentioned above). It uses this information and searches through the solution space to find a floorplan configuration that is thermally sound. In doing so, it uses the block-based thermal model of HotSpot to compute the temperature of the configurations at every step and chooses the floorplan with the lowest peak temperature. The block-based model is chosen because of its computational speed. However, in evaluating these floorplans, we employ the regular grid-based thermal model of HotSpot which is slower but more accurate. We use a grid resolution

of 256 x 256 and perform steady-state thermal simulations to obtain the peak temperature of the different floorplan configurations.

Although we model the dependence of leakage power on temperature using the empirical model in Section 2.6, the model is employed only during the computation of the input power densities of each benchmark. Once the peak steady-state temperature of a benchmark for a given floorplan is computed, the power densities are not re-evaluated by taking into account the reduced leakage due to temperature reduction. Hence, the temperature improvement because of floorplanning reported here does not include the benefit accrued from reduced leakage. Thus, it really forms a lower bound to what can be expected in practice and the actual enhancement is likely to be higher.

We model a microarchitecture similar to the Alpha 21364 as in [103] but scaled to 90 nm for the single-core case. In order to model a multicore configuration, we scale both the area of each core and its power consumption such that the power density remains constant. The thermal model parameters are set to the default values of HotSpot except the convection resistance and TIM thickness, which are assigned values of  $0.5 \frac{K}{W}$  and  $30\mu m$  respectively in order to model a moderate cooling solution. The default configuration modeled is a four-core processor with 75% of the die-area occupied by L2 cache. To reduce simulation complexity, each core is assumed to run the same benchmark.

## 5.4 Results

We will now describe the various placement choices evaluated. The first configuration is with the four cores arranged at the center of the chip wrapped around by the L2 cache. The cores are oriented in such a manner that their hottest units, the integer register files, are touching each other. This is similar to the illustration in Figure 5.1(a). Such a configuration is chosen to simulate the worst-case behaviour. We call this arrangement the *hot* scheme. Next is the configuration that forms the base-case of our evaluation. It is similar to *hot* in that the cores are arranged at the center of the chip but the orientation of all the cores is the same — pointing upwards. We call this the *base* scenario.

The next floorplan evaluated is the result of searching the orientation space as described in



Section 5.3.1. For four cores, a brute-force search is performed. The floorplan with the lowest peak temperature is considered for evaluation. This scheme is called *orient*. Next, the cache-bank insertion described in Section 5.3.2 is performed over the *base* configuration. This is called *base+l2*. When the same is done on top of the *orient* scheme, it is called *orient+l2*.

In order to perform the first of the two potential studies described in Section 5.3.4, we scatter the blocks in the *base* configuration in between the L2 cache banks as shown in Figure 5.1(d). This scheme is called *base+scatter*. Such a scatter performed for the *orient* configuration is called *orient+scatter*.

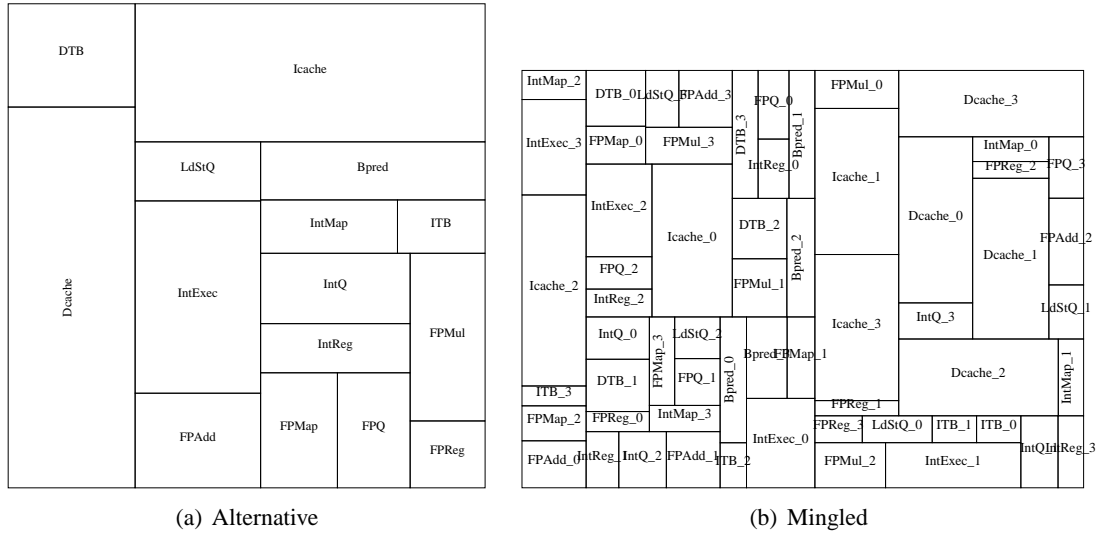


Figure 5.2: Floorplans used for the *altflp* and *mingled* studies [89]

In order to evaluate the hierarchical floorplanning algorithm described in Section 5.3.3, we first use the *base* configuration with the floorplan of each core derived from the single-core floorplanner in Chapter 4. Apart from assigning the relative weights of the architectural wires as per [18], we also incorporate the core aspect ratio into the cost function of simulated annealing. This method results in a floorplan with less than 0.05% dead space and a better wire length metric when compared to the base-case Alpha 21364-like floorplan. This alternative floorplan is shown in Figure 5.2(a). Its aspect ratio is close to 1. We call the four-way multicore scenario obtained by replicating this alternative floorplan as *altflp*. It is to be noted that the alternative floorplan is computed in isolation, without being mindful of the core’s location in a multicore processor. Hence, hot units are steered

away from the boundaries as much as possible to minimize the peak temperature. This floorplan is not necessarily beneficial in a multicore environment — especially with the L2 cache-bank insertion because units near the boundaries of a core are closer to the L2 cache, which is relatively cooler.

Retaining the nomenclature above, we call the cache-bank inserted version of *altflp* as *altflp+l2* and an orientation space search for *altflp+l2* as *altflp+orient+l2*.

Finally, we examine the potential of mingling the functional blocks from different cores as described in Section 5.3.4. The result of the simulated annealing performed on a single list of functional blocks from all the cores is shown in Figure 5.2(b). The amount of dead space for this floorplan is less than 0.1% of the total area of the cores. All the blocks are assumed to be soft with the constraints on their aspect ratio specified in the input file. Hence, the same functional blocks (*e.g.* L1 data cache) from different cores can have different aspect ratios. This scheme is called *mingled*. We also employ the insertion of cache banks in between the functional blocks of the *mingled* scheme. This scheme is called *mingled+scatter*.

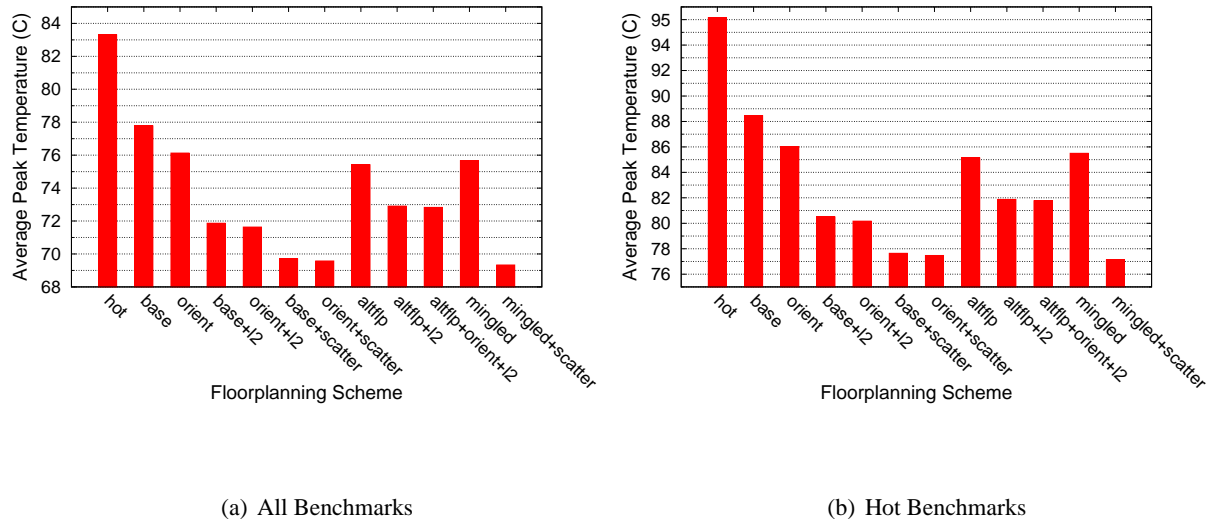


Figure 5.3: The peak temperature of the different floorplan configurations for the SPEC2000 benchmarks. [89]

Figure 5.3 shows the results of our evaluation. It plots the peak temperature of each floorplan scheme described above (and listed on the x-axis of the figure), averaged over all the 26 SPEC2000 benchmarks in the left side and over the eleven hottest benchmarks alone ((7 *int* (*bzip2*, *crafty*, *eon*,

*gcc*, *gzip*, *perlbmk*, *vortex*) and 4 *fp* (*art*, *galgel*, *mesa*, *sixtrack*)). Clearly, the *hot* configuration with the four hottest blocks adjacent to each other has the highest average peak temperature. Exploiting the orientation of the cores is beneficial when the cores are adjacent to one another as can be seen from the difference between the *base* and *orient* bars. However, when the cores are already separated by the L2 cache banks, the orientation of the cores matters to a much lesser degree. This is the reason the *base+l2* and *orient+l2* bars are pretty similar to each other.

The insertion of L2 cache banks between the cores reduces peak temperatures significantly. There is a 6.1 degree difference between the *base* and *orient+l2* bars, with a large portion of it coming from L2 insertion. For the 11 hottest benchmarks, this improvement is about 8.3 degrees on an average. For an ambient temperature of 45°C, this translates to about a 20.2% improvement over the temperature in excess of the ambient. The *base+scatter*, *orient+scatter* and *mingled+scatter* bars indicate the thermal spreading potential available in multicore floorplanning. Comparing the *orient+l2* bar against these, we can see that a combination of core orientation and L2 cache insertion is able to achieve a significant portion of that potential (about three-fourths).

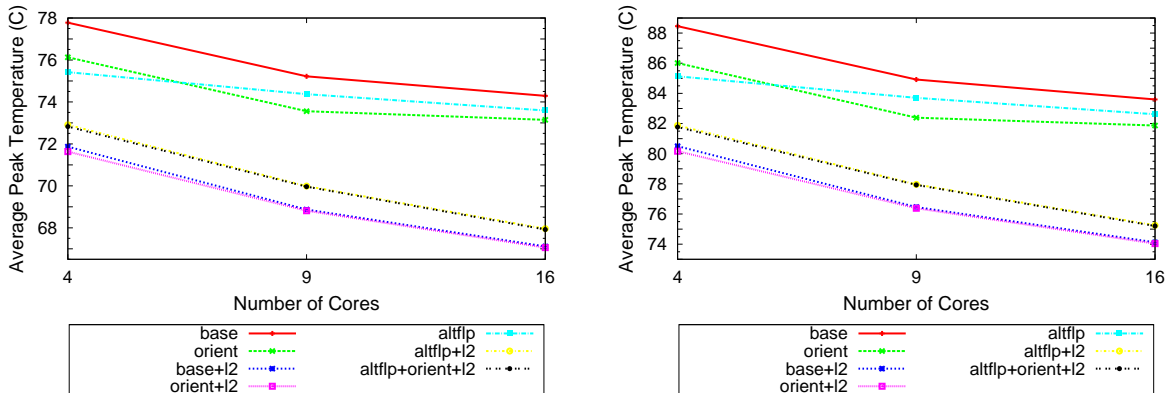
It can also be seen that although the alternative floorplan and the mingled floorplan are able to achieve temperature reduction, much of that reduction can be achieved by a simple orientation space search (*orient*). Furthermore, since the alternative floorplan has the hot functional blocks towards its center, the use of L2 cache banks as cooling buffers does not benefit it as much as it does the default floorplan. This is the reason *altflp+l2* and *altflp+orient* bars are higher than *base+l2* and *orient+l2*.

### 5.4.1 Sensitivity Studies

In this section, we investigate how the conclusions of the previous section are affected by our assumptions about the core size, occupancy and L2 power density respectively. This investigation is done through sensitivity studies that vary the above-mentioned parameters.

### 5.4.1.1 Effect of Core Size

Figure 5.4 plots the effect of varying the size of each core (and hence the total number of cores in a chip). It plots the average peak temperature of the practical (non-ideal) schemes from the previous section against the number of cores. It is to be noted that the power density of the functional blocks is maintained in changing the size of the cores. The lines shown are decreasing because silicon acts as a spatial low-pass filter for temperature [51]. Hence, for the same power density, smaller cores (high frequency) are cooler than larger cores (low frequency). It can be noted that the trends observed in the previous section still hold. Much of the thermal benefit comes from L2 cache insertion. The only reversal in trend is that *altflp* performs even worse than *orient* for higher number of cores.



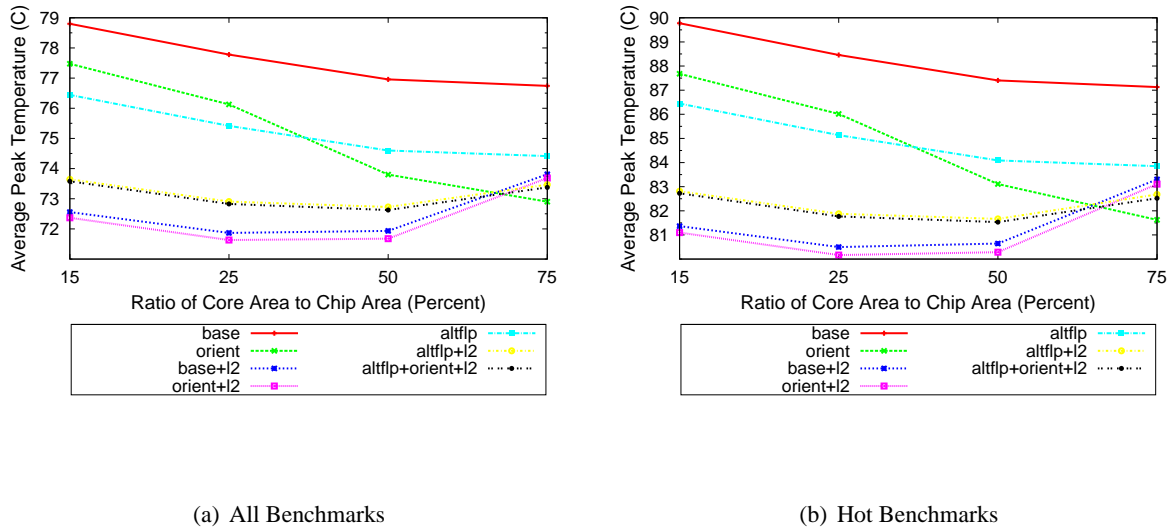
(a) All Benchmarks

(b) Hot Benchmarks

Figure 5.4: Effect of varying the number of cores. In scaling the cores, the power density of the functional blocks is kept constant. [89]

### 5.4.1.2 Effect of Core Occupancy

Another important parameter in our study is the ratio of core area to the total area of the chip. We call this ratio the core occupancy. Figure 5.5 plots the result of an experiment varying the core occupancy from 15% to 75%. Actually, two competing factors determine the temperature in this experiment. First is that as the core occupancy decreases, in order to keep the core and L2 power



(a) All Benchmarks

(b) Hot Benchmarks

Figure 5.5: Impact of the ratio of core area to chip area. The size of each core remains the same while that of the L2 cache is increased, keeping its power density constant. [89]

densities constant for an apples-to-apples comparison, the total chip area increases. Hence, the total power dissipated also increases with decreasing occupancy. The second factor is the reduced availability of the relatively cooler L2 space to act as a thermal buffer as occupancy increases. Depending on which factor predominates, sections of the curves decrease or increase. It is evident from the graph that as the occupancy increases, the importance of core orientation increases, which is the reason the *orient* line decreases quickly. At 16 cores, it even performs marginally better than the L2 cache insertion techniques.

#### 5.4.1.3 Effect of L2 Power Density

Since the floorplanning schemes presented in this chapter involve the L2 cache banks, the power density of L2 cache is an important factor to be considered. Hence, we perform an experiment replicating the results in Figure 5.3 with the power density of the L2 cache being double of what it was in that figure. The results of this are presented in Figure 5.6. Clearly, the trends remain the same with elevated temperatures due to the increased power density.

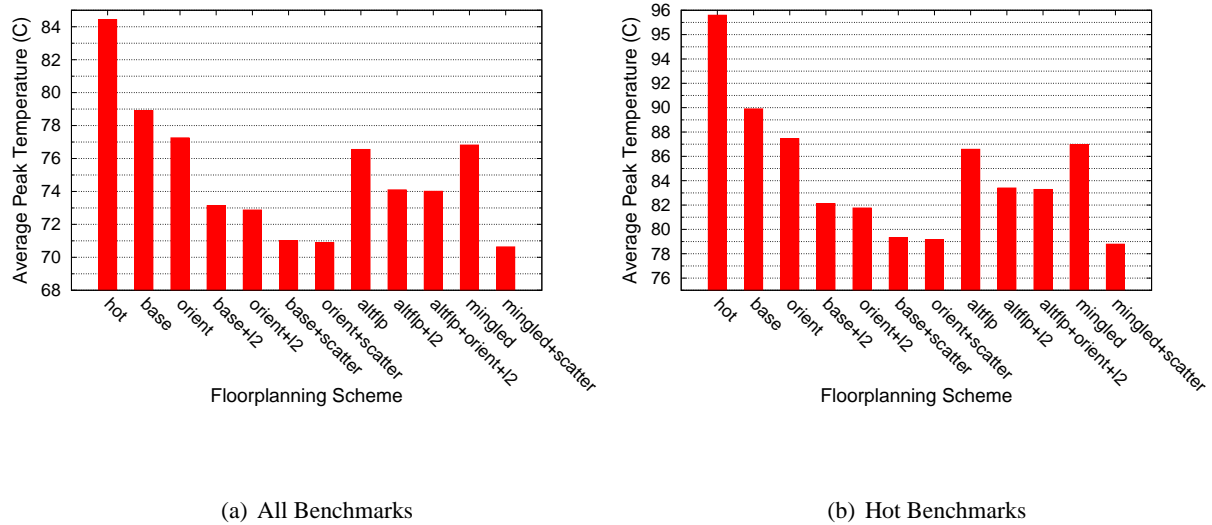


Figure 5.6: Thermal performance of the floorplanning schemes on doubling the L2 cache power density. [89]

## 5.5 Conclusion

This chapter investigated the temperature reduction potential of multicore floorplanning. It advocated the exploitation of various placement choices available in a multicore processor ranging from the functional block level to the core level. It proposed the exploration of alternative core orientations in order to separate hot units from being adjacent to each other in a multicore chip. It also presented the idea of inserting L2 cache banks between the cores as cooling buffers for better heat distribution. Furthermore, it studied the potential of multicore floorplanning by letting functional blocks and L2 cache banks cross core boundaries. The most important conclusion from this work is that L2 bank insertion achieves significant thermal benefit — about 20% of the temperature above the ambient on an average for SPEC2000 benchmarks. Furthermore, a combination of core orientation and L2 bank insertion is able to achieve about 75% of the temperature reduction achievable by an ideal floorplanning scheme that mingles functional blocks from multiple cores and disperses them amidst a sea of L2 cache banks. With the advent of SIMD processors including GPUs, future work in this direction could examine the applicability of floorplanning techniques in reducing their peak temperature. Furthermore, since heterogeneous multicore architectures offer additional levels of placement options, exploiting them for thermal benefit is another interesting possibility.

# Chapter 6

## Granularity of Thermal Management

---

### 6.1 Introduction

Two important recent trends have had a tremendous impact on the microprocessor industry: First, non-ideal semiconductor technology scaling has made physical effects that were previously abstracted away by computer architects to become first-order design constraints. A crucial example of this phenomenon is the exponential increase of power density as feature size shrinks. This increase has caused temperature to be a serious concern since power density is directly related to on-chip temperature. Second, multicore processors have become the norm because of the disproportionate scaling of wire and logic delays, diminishing Instruction-Level Parallelism (ILP) and the challenges of power and heat dissipation. In such a multicore era, an interesting question is the size granularity at which microarchitectural thermal management should be performed. That is, if thermal management is most relevant when performed at the level of a sub-block (*e.g.* cache line, register file entry *etc.*), functional block (*e.g.*, register file, cache, branch predictor *etc.*), architectural sub-domain within a core (*e.g.*, the integer pipeline, the floating-point pipeline, the memory pipeline *etc.*), at the level of a core, or globally for the entire chip.

As core sizes shrink with technology scaling, an argument for core-level Dynamic Thermal Management (DTM) is that a sub-domain within a core is affected more by its neighbour's temperature than its own computational activity. So, neighbouring sub-domains effectively get aggregated, resulting in the degeneration of any localized DTM scheme in to a global one. Investigation of this

phenomenon shows that the determining factor in it is the relationship between heat conduction in the vertical (through the die, heat spreader and heat sink) and lateral (along the die) directions. When the former is much greater than the latter, a block's temperature is mainly dependent on its own computational activity. Otherwise, it is mainly neighbour influence. Technology scaling results in lateral dimensions of silicon diminishing faster than the vertical thickness for reasons of mechanical strength. Moreover, vertical dimensions of the package layers (heat spreader, heat sink) do not scale much. This fact means lateral spreading grows quicker than vertical conduction. The implication is that the spatial granularity of thermal management has to become coarser with technology scaling. One would have to move from thermal management at the level of a functional block to the core-level to groups-of-cores. With the advent of multicores and manycores, how this happens is an interesting question for exploration.

### 6.1.1 Spatial Filtering

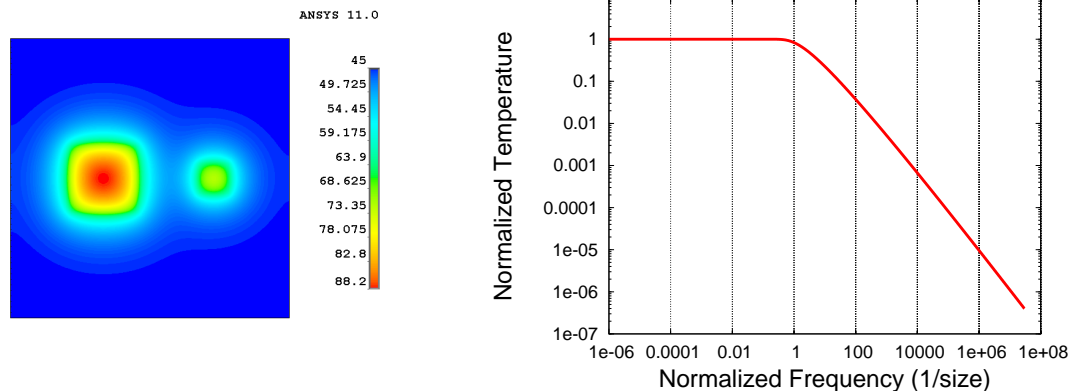


Figure 6.1: Illustration of the spatial thermal filtering. (a) Two blocks of different sizes having the same power density. (b) low-pass filter “Bode plot” frequency response. [88]

It turns out that silicon behaves like a spatial low-pass filter for temperature. That is, power dissipating blocks that are small enough do not cause hot spots. Silicon “filters out” these “spatial high frequencies” (blocks of small size). Figure 6.1 illustrates this aspect. The image on the left



(6.1(a)) shows a silicon die with two square power dissipating blocks. Both the blocks have the same power density ( $1 \frac{W}{mm^2}$ ) but the area of the block on the left is four times that of the one on the right. Although the power densities are the same, it can be seen that the larger block on the left is significantly hotter than the smaller block on the right. This relationship between block-size and peak temperature is illustrated in Figure 6.1(b). It is the classic “Bode plot” frequency response of a low-pass filter with the spatial frequency (1/block-size) plotted on the x-axis and the peak temperature on the y-axis. Both the axes are on a logarithmic scale. It shows a clear “cut-off frequency” beyond which the temperature falls rapidly.

A study of this spatial filtering effect from a microarchitectural perspective is useful since it throws light on several aspects of the size granularity of thermal management. Hence, this chapter explores this thermal spatial filtering effect in detail. Since the effect is nothing but a manifestation of the relationship between lateral and vertical heat conduction in silicon, it first takes an analytical approach and solves the two-dimensional steady-state heat equation from first principles for a sample geometry similar to that of microprocessor chips but in two dimensions. Then, in the light of the analytical equation, it explores how the spatial filtering appears in practice when the assumptions of the analytical model are removed. Finally, it illustrates the spatial filtering behaviour using three microarchitectural examples:

1. A manycore checkerboarding experiment where the die is tiled in a checkerboard-like fashion with alternating cores and L2 blocks. The objective is to study the temperature by varying the number of cores keeping the total area occupied by the cores, the power dissipated in them and their power density constant. It illustrates the potential increase in Thermal Design Power (TDP) achievable solely due to spatial filtering when all the other variables remain the same. Furthermore, it studies the relationship between the number of cores and the granularity of thermal management by exploring the effectiveness of local (within-core) thermal management as opposed to global (core-level) thermal management as a function of the number of cores.
2. The question of whether a high aspect ratio sub-block like a single cache line or a register

file entry can become the hot spot because of pathological code behaviour is examined from a spatial filtering perspective. This study illuminates the relationship between aspect ratio of a block and its peak temperature.

3. For a regular grid of thermal sensors, this chapter studies the relationship between the inter-sensor distance and the error in temperature measurement and explains it using spatial filtering. It also explores the effectiveness of interpolation schemes in mitigating sensor errors.

The remainder of this chapter is organized as follows. Section 6.2 discusses the previous work related to this chapter. Section 6.3 describes the analytical formulation and solution of the two-dimensional heat equation for boundary conditions similar to those in microprocessors. It also reconciles the analytical model with what is observed in practice. Section 6.4 details the microarchitectural examples of the spatial filtering effect and presents the results. Section 6.5 concludes the chapter providing pointers to interesting future directions.

## 6.2 Related Work

Three papers we know have examined the spatial filtering aspect from the viewpoint of thermal modeling accuracy. Etessam-Yazdani *et. al.* [35] study the temperature response of the chip to a white noise power distribution experimentally. They also perform a two-dimensional Fast Fourier Transform (FFT) analysis to determine the spatial cut-off frequency beyond which high power density has lesser impact on temperature. They conclude that for accurate thermal modeling, it is sufficient to model at a size granularity close to this cut-off frequency. Two previous papers from our group [53, 50] also deal with the correct spatial granularity for modeling temperature accurately. In studying the size granularity, the first [53] employs an equivalent electrical approximation through a resistor network analogy. The second [50] examines the impact of high aspect ratio functional blocks on thermal modeling accuracy and sub-divides high aspect ratio blocks into multiple sub-blocks with aspect ratios closer to unity. These papers mainly approach the size granularity from the standpoint of thermal *modeling* accuracy and not from a thermal *management* perspective. They do

not study the microarchitectural implications of thermal filtering. Furthermore, they do not include the effect of the package in detail, which, this chapter shows to be an important factor.

Another recent paper from our group that deals with the spatial filtering aspect is [51]. It discusses the thermal efficiency of manycore processors in the context of spatial thermal filtering. While it has a microarchitectural angle with the consideration of manycore design, its focus is narrow as it only approaches a single design aspect related to thermal filtering — core size and TDP benefit. In comparison, this chapter expands on it and considers broader implications of spatial filtering on microarchitectural thermal management including aspects such as granularity of thermal *control* and sensor accuracy. Moreover, even for manycore processors, the results presented in the current chapter are more detailed, taking into account the effect of the package, chip thickness, area of the L2 cache *etc.* Also, like our group’s earlier paper [53], the more recent work [51] reasons about spatial filtering using an equivalent RC approach. This chapter, on the other hand, studies the thermal conduction equation directly. Furthermore, none of the above previous papers except [50] consider the effect of aspect ratio, which is nothing but an alternative manifestation of the spatial filtering.

Since this chapter treats the accuracy of thermal sensors in the light of spatial filtering, research on sensor accuracy and fusion is relevant here. Lee *et. al.* [66] present an analytical model for determining the sensor accuracy as a function of the distance from the hot spot. However, they do not offer any techniques for improving sensor accuracy. Memik *et. al.*’s work on sensor allocation [71] discusses uniform and non-uniform placement of sensors. For uniform sensor allocation, it describes a simple interpolation scheme to reduce sensor errors. The interpolation schemes proposed in this chapter are more accurate than the simple scheme from [71] but without significantly higher overhead. Sharifi *et. al.* [94] employ Kalman filter for eliminating sensor errors. While their scheme is very accurate, it requires the knowledge of per-unit power consumption numbers, which might not be available easily. The usefulness of our work is in offering an option *between* a simple scheme such as [71] and a computationally intensive scheme such as Kalman filtering [94].

Dadvar and Skadron [28] raise concerns about the possible security risks involved in software-controlled thermal management. Since our work examines the question of pathological code heating

up individual cache lines, [28] is relevant here. Our finding is partly in agreement with [28] in that there is a possibility of cache lines heating up to high temperatures due to code activity. However, their high aspect ratio prevents the temperatures from reaching catastrophic limits. Ku *et. al.*'s research [65] on reducing the power density of caches by keeping the live lines as far apart from each other as possible is relevant to this work as well. In effect, they exploit the spatial filtering by increasing the spatial frequency. Our work offers a framework to reason about such thermal management schemes.

## 6.3 An Analytical Approach

As explained in Section 6.1, at the core of the thermal filtering is the relationship between heat conduction in the lateral direction along the die and the vertical direction through the die, heat spreader and heat sink. In order to understand this relationship better, an analytical look into a simplified version of the problem is beneficial. While the heat conduction, geometry and boundary conditions of an actual chip and package are quite complex, we make several simplifying assumptions in this section for the sake of mathematical ease. However, we do examine the effect of these assumptions later by comparing the trend predicted by the analytical model to experimental data obtained by simulating a realistic chip and package configuration in a Finite Element Model (FEM).

### 6.3.1 A Simplified Heat Conduction Problem

The assumptions made in this section are the following:

**Simplification of the geometry** Since the two lateral dimensions of heat conduction are not conceptually different from each other, we first collapse them into one and consider a two-dimensional heat conduction problem with one vertical and one lateral dimension. In fact, we actually examined a preliminary solution of the full three-dimensional case and observed that its analytical form is not qualitatively different from that of the two-dimensional case. Hence, in the interest of time and clarity, we present the two-dimensional solution here. Moreover, we only consider a single vertical layer of conduction (silicon) and ignore the other package and interface layers. Although this is the

most serious assumption, making it affords a much simpler treatment of the problem. Also, it is not far from reality for handheld processors where the form factor and cost preclude any cooling solution.

**Simplification of the boundary conditions** First, we assume an isothermal boundary at the top (the side exposed to the ambient temperature). In reality, this is convective due to the presence of a fan. Second, we assume that the bottom of the chip (the power dissipating end) is insulated and hence no heat flows through the I/O pads and the Printed Circuit Board (PCB). Third, for mathematical convenience, we assume the boundaries of the lateral dimension to extend indefinitely since doing so leads to simple closed-form solutions comprising of exponentials. Furthermore, we perform only a steady-state analysis here, since for peak power density, steady-state temperature provides an upper bound on the attainable temperature and hence is sufficient for our purpose.

Figure 6.2 represents the above-mentioned simplifications pictorially. Figure 6.2(a) can be thought of as representing a sheet of silicon with a thickness  $l$  (along the y-axis) that extends indefinitely along the positive and negative x-axes. Its depth (along the z-axis) is considered negligible, effectively reducing the problem to two dimensions. A power-dissipating block of size  $2a$  with a power density of  $q$  resides at the bottom surface of silicon. The top surface of silicon is isothermal with a temperature of 0 (when the ambient temperature is non-zero, it only shifts the solution by that constant value). Our task is to find the temperature distribution on this infinite sheet of silicon as a function of  $a$  and  $l$ . Note that the value  $a$  being half of the size of the dissipator  $2a$ , is called the “half-size”. This problem is a classic heat transfer problem that can be solved by using standard Partial Differential Equation (PDE) solving techniques [17, 75, 63].

Assuming that the origin is at the center of the power dissipator, it can be seen that the setup shown in Figure 6.2(a) is symmetric about the y-axis. Therefore, the left and right halves are indistinguishable with respect to temperature. Hence, the net heat flux (power density) from one side of the y-axis to the other must be zero. If not, that very difference constitutes an asymmetry through which the two halves can be distinguished. Thus, the y-axis behaves like it is insulated. This is shown in Figure 6.2(b), which is essentially the right half of 6.2(a) with slanted lines clearly

marking the insulated behaviour of the y-axis.

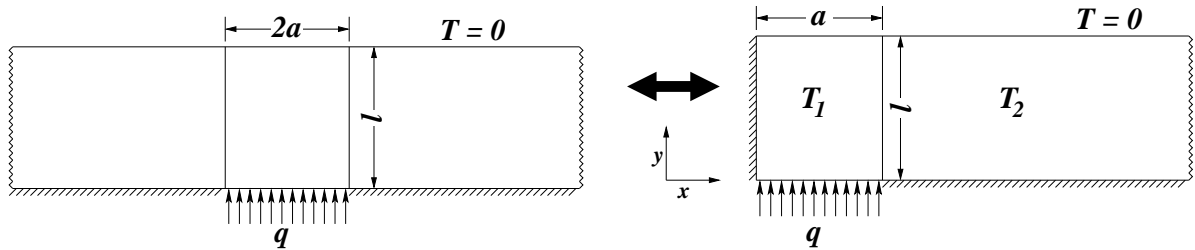


Figure 6.2: Simplified heat conduction problem to mimic a silicon die of thickness  $l$ . A power density of  $q$  is applied at the bottom in a region of size  $2a$ . Insulated boundaries are marked by slanted lines. Serrated lines indicate that the boundaries extend to infinity in that direction. (a) shows the original infinite version of the problem and (b) shows the equivalent semi-infinite version that (a) reduces to because of symmetry. (b) is essentially the right half of (a) [88]

### 6.3.2 Analytical Solution

For a moment, let us consider this heat conduction problem qualitatively. In steady state, the temperature distribution of this infinite block does not change over time. Thus, for any infinitesimal volume in it, there is no change in its internal energy. In other words, the rate at which heat energy enters it (input power) from its neighbours through conduction is the same as that of the heat energy leaving it (output power). Another way of saying this is that the *gradient* (derivative in space) of the power (and hence the power density, since area is constant) is zero. The relationship between this power entering and leaving the volume and its temperature is governed by Fourier's law of heat conduction [17] which is nothing but the "Ohm's law" for thermal conduction. It says that the power density is directly proportional to the gradient of the temperature and that the constant of proportionality is the thermal conductivity of the material (in our case, silicon). This is intuitive because the higher the gradient, the higher is the difference between the temperatures of this volume and its neighbours and hence, the higher is the rate of heat transfer. So, the fact that in steady state, the gradient of the power density is zero can be re-stated using Fourier's law that the *gradient of the gradient of temperature* is zero. This statement with a *second-order* derivative in it is called the steady-state heat equation, which we will be solving for the two-dimensional case.

If we denote the steady-state temperature functions of the two portions of Figure 6.2(b) by  $T_1(x, y)$  and  $T_2(x, y)$  and the gradient operator by  $\nabla$  (for two dimensions, this is nothing but  $\frac{\partial}{\partial x} + \frac{\partial}{\partial y}$ ), the steady-state heat equations we would like to solve are:

$$\begin{aligned} \nabla^2 T_1 = 0 \quad , \quad \nabla^2 T_2 = 0 & \quad (6.1) \\ (0 \leq x \leq a, 0 \leq y \leq l) \quad (a \leq x \leq \infty, 0 \leq y \leq l) \end{aligned}$$

Now let us list the boundary conditions for our problem. From the discussion above, if  $k$  is the thermal conductivity of silicon, Fourier's law is nothing but the fact that *power density* =  $-k\nabla T$ . Hence, for an insulated boundary, since the power density across the boundary is zero, the condition can be written as  $\nabla T = 0$ . Thus, the boundary conditions for our problem are:

$$T_1(x, l) = 0 \quad T_2(x, l) = 0 \quad (6.2a)$$

$$T_2(\infty, y) = 0 \quad (\text{isothermal boundaries}) \quad (6.2b)$$

$$\frac{\partial T_1}{\partial x} \Big|_{x=0} = 0 \quad \frac{\partial T_2}{\partial y} \Big|_{y=0} = 0 \quad (\text{symmetry, insulation}) \quad (6.2c)$$

$$\frac{\partial T_1}{\partial y} \Big|_{y=0} = -\frac{q}{k} \quad (\text{Fourier's law, power dissipator}) \quad (6.2d)$$

$$T_1(a, y) = T_2(a, y) \quad (6.2e)$$

$$\frac{\partial T_1}{\partial x} \Big|_{x=a} = \frac{\partial T_2}{\partial x} \Big|_{x=a} \quad (\text{continuity at } x = a) \quad (6.2f)$$

The analytical solution of this boundary value problem can be obtained using classic PDE-solving techniques [17, 75, 63]. We provide only the closed-form solutions of the functions  $T_1$  and  $T_2$  here and refer interested readers to appendix A for a detailed derivation.

$$T_1(x, y) = \frac{ql}{k} \left[ 1 - \frac{y}{l} - 2 \sum_{n=0}^{\infty} \frac{e^{-(\gamma_n \frac{a}{l})}}{\gamma_n^2} \cosh(\gamma_n \frac{x}{l}) \cos(\gamma_n \frac{y}{l}) \right] \quad (6.3)$$

$$T_2(x, y) = \frac{ql}{k} \left[ 2 \sum_{n=0}^{\infty} \frac{\sinh(\gamma_n \frac{a}{l})}{\gamma_n^2} e^{-(\gamma_n \frac{x}{l})} \cos(\gamma_n \frac{y}{l}) \right] \quad (6.4)$$

$$\text{where, } \gamma_n = (2n + 1) \frac{\pi}{2} \quad (n = 0, 1, 2, \dots)$$

It is useful to make a couple of observations about these solutions. It can be seen that all the variables of interest ( $x$ ,  $y$  and the half-size  $a$ ) appear in the equation as ratios with respect to  $l$ . This pattern indicates that the lateral dimensions matter only relative to the vertical thickness and not in the absolute. For our purposes, it is sufficient to restrict ourselves to the peak temperature on the infinite sheet. Actually, for a given  $a$ , the peak temperature  $T_{peak}$  occurs at  $(0, 0)$  *i.e.*, at the center of the power dissipator. Hence,  $T_{peak} = T_1(0, 0)$ . Also, the absolute maximum peak temperature (for all  $a$ ) occurs when  $a = \infty$  *i.e.*, when the power dissipating block is very large. Let us call this temperature  $T_{peak}^{\infty}$ . Then,  $T_{peak}^{\infty} = \frac{ql}{k}$ , where  $q$  is the power density and  $\frac{l}{k}$  divided by the area of the power dissipator gives the vertical *thermal resistance*. Now, defining the normalized peak temperature  $T_{peak}^{norm}$  as  $T_{peak}/T_{peak}^{\infty}$  and the normalized half-size  $a_{norm}$  as  $\frac{a}{l}$ , we get

$$\begin{aligned} T_{peak}^{norm} &= \frac{T_{peak}}{T_{peak}^{\infty}} = \frac{T_1(0, 0)}{\frac{ql}{k}} \\ &= 1 - \frac{8}{\pi^2} \sum_{n=1,3,5,\dots} \frac{e^{-n \frac{\pi}{2} a_{norm}}}{n^2} \end{aligned} \quad (6.5)$$

It is this  $T_{peak}^{norm}$ , whose value ranges between 0 and 1, that has been plotted against  $\frac{1}{a_{norm}}$  in Figure 6.1(b). One can clearly see the low-pass filter behaviour with a cut-off frequency around  $a_{norm} = 1$ . This means that when the half-size of the power dissipator is smaller than the vertical thickness, the peak temperature falls rapidly in relation to the half-size.

### 6.3.3 Spatial Filtering in Practice

Equation (6.5) and Figure 6.1(b) show the exponential relationship between the size of a power dissipator and its peak temperature. Similarly, (6.3) shows the linear relationship between the power



density  $q$  and peak temperature. This behaviour suggests an interesting trade-off for microarchitectural thermal management. A thermal management scheme can reduce the temperature of a hot block by reducing its computational activity, thereby reducing its power density. Alternatively, it can manipulate the size of the power dissipating block (*e.g.*, by partitioning the block and separating the partitions by a safe distance) and hence exploit this spatial filtering effect to reduce peak temperature. When the size of the power dissipating block is close to the cut-off frequency, this latter choice can have a significant benefit over the former because of its exponential relationship as opposed to the linear relationship in the former. At the same time, when the block is so large that even the partitions are much larger than the cut-off frequency, reducing the power density might be a better choice since the benefit from size reduction becomes negligible. To examine such microarchitectural implications, it is necessary to reconcile the theoretical model discussed in the previous section with what happens in practice. So, in this section, we relax the assumptions of the analytical model and examine the effects through experiments under a commercially available FEM solver, ANSYS 11.0 [4].

### 6.3.3.1 Impact of Size

We remove the assumptions of the last section in three steps. First, we consider the full three-dimensional problem with a convective boundary near the ambient temperature. Next, we include the effect of multiple package layers and finally, we investigate the impact of geometric extent. For the first step, we consider a 10 mm x 10 mm silicon die that is 1 mm thick. The top of the die is cooled by convection to an ambient at  $0^{\circ}\text{C}$  with a heat transfer co-efficient equivalent to a  $0.1 \frac{\text{K}}{\text{W}}$  thermal resistance (*i.e.*, for a  $100\text{mm}^2$  area, it is  $0.1 \frac{\text{W}}{\text{mm}^2\text{K}}$ ). At the center of the base of the die is a square power dissipator with a power density of  $1 \frac{\text{W}}{\text{mm}^2}$ . The size of this power dissipator is varied from 0.2 mm to the full 10 mm. The maximum peak temperature for this experiment occurs when the power dissipator occupies the entire silicon die and its value is 20 degrees more than the ambient. We call this experiment *center-Si* to denote the single layer of *silicon* with a power dissipator at its *center*.

Next, we repeat the same experiment with the die attached to three other layers — a layer of

Thermal Interface Material (TIM) followed by a layer of copper (representing the heat spreader), followed by a second layer of copper (denoting the heat sink). To isolate the effect of multiple layers from that of finite extent, we restrict the lateral dimensions of all the four layers to 10 mm x 10 mm. The vertical thickness of each layer is set in such a manner that the *proportion* of the thicknesses is similar to what is seen in a typical package and that the total *equivalent thickness* in silicon terms (*i.e.*, sum of the thicknesses of the individual layers weighted by the ratios of their thermal conductivities to that of silicon) is still 1 mm. So, the maximum peak temperature is still 20°C. The actual thicknesses of the silicon, TIM, spreader and heat sink layers are 0.0571 mm, 0.0076 mm, 0.3810 mm and 2.6286 mm respectively. The respective thermal conductivities are 100, 4, 400 and 400  $\frac{W}{mK}$ . The heat transfer co-efficient at the top surface remains the same as before and the size of the power dissipator is varied as before. We call this experiment *center-4layers-equal* to denote the equal lateral size of the four layers and the location of the power dissipator at the center of the base.

Finally, to examine the effect of the finite extent, we add a few more configurations. To model a normal package in which the heat spreader and heat sink are larger than the die, we extend the *center-4layers-equal* configuration to make the lateral sizes of the heat spreader and the heat sink layers to be 20 mm x 20 mm and 40 mm x 40 mm respectively. We call this *center-4layers-spread* since it models the effect of lateral spreading beyond the die area into the spreader and the sink. The heat transfer co-efficient remains the same as before. It is to be noted that the maximum peak temperature in this case will be lower than the 20°C seen in the previous cases because of the lateral spreading beyond the die area. Finally, to study the effect of the location of the power dissipator (and hence the effect of finite boundaries), we extend all of the above configurations by changing the location of the power dissipator from the center of the base to a corner. This restricts the directions in which heat spreads laterally in silicon to two from the four possibilities (north, south, east and west) at the center. We name these configurations in the form *corner-xxxx* where *xxxx* derives from the names of the previous configurations. For instance, *corner-Si* is basically the same as *center-Si* except for the location of the power dissipator and so on. A *center-xxxx* experiment and the corresponding *corner-xxxx* experiment can be thought of as forming the lower and upper bounds on

the attainable peak temperature in that particular configuration for a given power density.

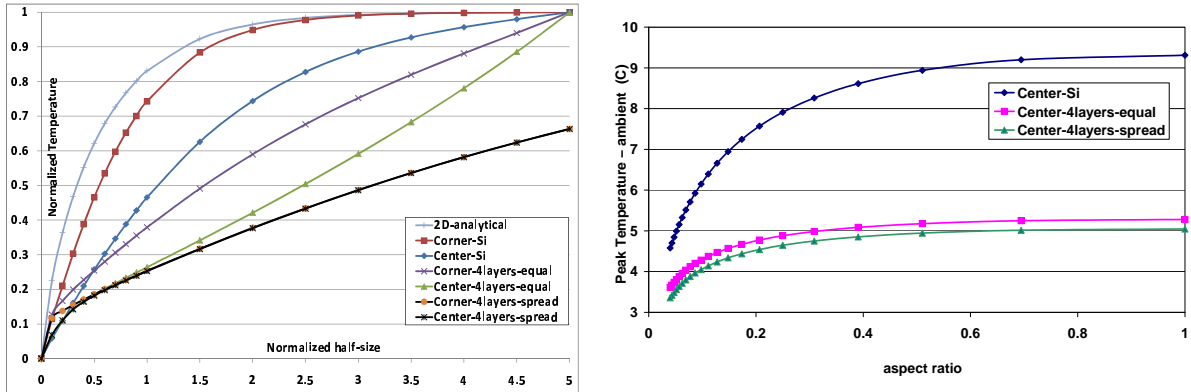


Figure 6.3: Comparison of the analytical model and FEM simulation. (a) shows the impact of power dissipator size and (b) shows the effect of aspect ratio. Note that the *center-4layers-spread* and *corner-4layers-spread* curves are coincident on top of each other. [88]

All the above experiments are run under ANSYS 11.0 with a lateral grid size of 100 x 100 on silicon (except *corner-4layers-spread*, which uses a 50 x 50 grid due to a software license restriction on the number of nodes). The lateral dimensions of the grid cells are the same in the other layers as well. Vertically, the silicon die is divided into three layers, spreader and sink are divided into four layers each and the TIM is modeled as a single layer. Figure 6.3(a) shows the results of these experiments. It plots the peak temperature of a configuration as a function of the size of its power dissipator. Both the axes are normalized. The temperature axis reports the ratio of the peak temperature of a configuration to the maximum peak temperature of *center-Si*. The size axis reports the ratio of the half-size of the power dissipator to the vertical thickness of the configuration (1 mm in silicon equivalent). For comparison, it also plots the peak temperature computed by the analytical equation (6.5) (*2d-analytical* in the graph).

The main conclusion one can draw from the graph is that the behaviour suggested by the analytical equation is not universal. There is a significant difference between the exponential relationship suggested by the analytical equation and the “straight line” plots of the *4layers-spread* configurations that model a desktop-type package. The configurations with only a single layer of silicon

(*corner-Si* and *center-Si*) on the other hand, are closer to the analytical equation showing that going from two to three dimensions or from an isothermal boundary to a convective boundary are not factors causing significant difference. Furthermore, it can be seen that the plot shows clear “bands” bounded from below by a *center-xxx* configuration and from above by the corresponding *corner-xxx* configuration. This pattern means that the location of the power dissipator matters to the degree indicated by the width of the “band”. Hence, the finite boundaries of the geometry affect different configurations differently. The location matters a lot for a mobile-type configuration without any spreader or sink (*\*-Si*) while it does not matter at all for a typical desktop-type package (*\*-4layers-spread*). This behaviour is because of the extra heat spreading offered by copper (as opposed to silicon) and the extra area available for spreading beyond the die. Since the entire die is roughly at the center of the heat spreader and the heat sink, the position of the power dissipator within the die becomes immaterial.

Thus, the lateral heat spreading in the heat spreader and the heat sink are the most significant factors determining the extent of spatial thermal filtering. The steep exponential nature of the spatial filtering indicated by the analytical equation is only valid for chips without a cooling solution (*e.g.* processors in hand-held devices). For a typical package found in desktop processors, the exponential is so shallow that it behaves like a straight line. Since previous work [35, 53] examined the spatial filtering from the standpoint of thermal *modeling* accuracy and necessary resolution, it was sufficient for them to consider the “worst-case” thermal gradients that occur in the absence of a heat spreader or a heat sink. However, for a thermal *management* angle, the distinction is indispensable.

### 6.3.3.2 Impact of Aspect Ratio

In the analytical simplification of the spatial filtering, a factor that was not considered due to the lack of three dimensions was the effect of the lateral aspect ratio of the power dissipator on its peak temperature. This relationship is important for understanding the thermal behaviour of high aspect ratio microarchitectural sub-blocks such as cache lines. Hence, Figure 6.3(b) plots this relationship for the three *center-\** configurations. The area of the power dissipator is fixed at  $4\text{mm}^2$  and its

aspect ratio is varied from 1:25 to 1:1. At an aspect ratio of 1:1 (2 mm x 2 mm block  $\Rightarrow$  half-size = 1 mm), this is nothing but the set of points corresponding to the normalized half-size of 1 in Figure 6.3(a). It can be seen from the plot that aspect ratio also has an exponential relationship with peak temperature. Also, similar to the dissipator size, the extent of spatial filtering is dependent on the additional layers of copper in the package. The extra heat spreading in copper essentially smoothes out the response of the thermal filter. Hence, the curves are shallow for the *4layers* configurations and steep for *Center-Si*.

## 6.4 Microarchitectural Examples

So far, we have examined the thermal spatial filtering phenomenon in isolation. In this section, we will study its microarchitectural implications through three microarchitectural examples. The first is a study of the thermal efficiency of manycore processors in the light of the granularity of thermal management. The second is an investigation of whether high aspect ratio sub-blocks like cache lines can become hot spots due to pathological code behaviour. The third is an exploration of thermal sensor accuracy as a function of the distance between sensors and an examination of the effectiveness of sensor interpolation schemes.

### 6.4.1 Many-Core Processors

#### 6.4.1.1 Thermal Benefit of Spatial Filtering

As it was mentioned in Section 6.1, thermal management, when considered distinctly from power management, is mainly the efficient distribution of heat within the available die area. One way to accomplish this is through a cooling system that spreads heat well. For instance, heat spreaders made out of artificial diamonds are available commercially [105], since diamond has the highest thermal conductivity among known materials. Heat pipes are also an example of such efficient spreading. On the other hand, considering the thermal spatial filtering discussed before, manycores provide an interesting alternative. In comparison with large monolithic cores, they provide an opportunity to distribute the *heat generation* better. Since the power density on the lower-level

cache banks is much lower compared to that on the cores, a possibility is to floorplan the die in the form of a checkerboard with cores forming the dark squares and lower-level cache banks forming the light squares (see Figure 6.4(a) for an example). This arrangement exploits the spatial filtering by maximizing the spatial frequency of the power density distribution. Hence, in this section, we study the effect of such checkerboarding of manycores in the light of spatial filtering.

Assuming that the silicon die is arranged in the form of a checkerboard as shown in Figure 6.4(a), we vary the number of cores by varying the size of the checkerboard (from  $2 \times 2$  to  $20 \times 20$ ) and study its effect on peak temperature. The power densities in the cores and the cache blocks are assumed to be constant nowadays through the scaling of the number of cores. This assumption is made to isolate the thermal effect of spatial filtering from other circuit and microarchitectural factors affecting temperature. The constant power density assumption is also reasonable under constant electric field scaling principles [29], since the microarchitecture of a core is assumed to remain constant while its size is scaled as permitted by technology. Hence, with the number of cores increasing, the total *power* dissipated remains constant across the configurations. Since the thickness of the die does not scale with the feature size for reasons of mechanical strength, it is assumed to be constant as well. Later, we examine the effect of this assumption by performing a sensitivity study on varying the die thickness.

We know from the discussion in the last section that the package is a significant determinant in the spatial filtering. Hence, we consider two different package configurations: one without any cooling system to mimic hand-held mobile processors and another with a typical package found in desktop processors with a heat spreader and a heat sink. We call the former *mobile-regular-50* with *regular-50* denoting a regular checkerboard as shown in Figure 6.4(a), with the cores occupying 50% of the total die area. In a similar vein, the latter is called *desktop-regular-50*.

The experimental setup is as follows: the silicon die is 16 mm x 16 mm x 0.15 mm in size for both cases. For *desktop-regular-50*, the additional layers include a TIM of thickness  $20\mu\text{m}$  and the same lateral dimensions as the die, a copper heat spreader of size 3 cm x 3 cm x 1 mm and a copper heat sink of size 6 cm x 6 cm x 6.9 mm. The heat sink is cooled by convection with a co-efficient of heat transfer equivalent to a  $0.1 \frac{\text{K}}{\text{W}}$  thermal resistance. For an apples-to-apples

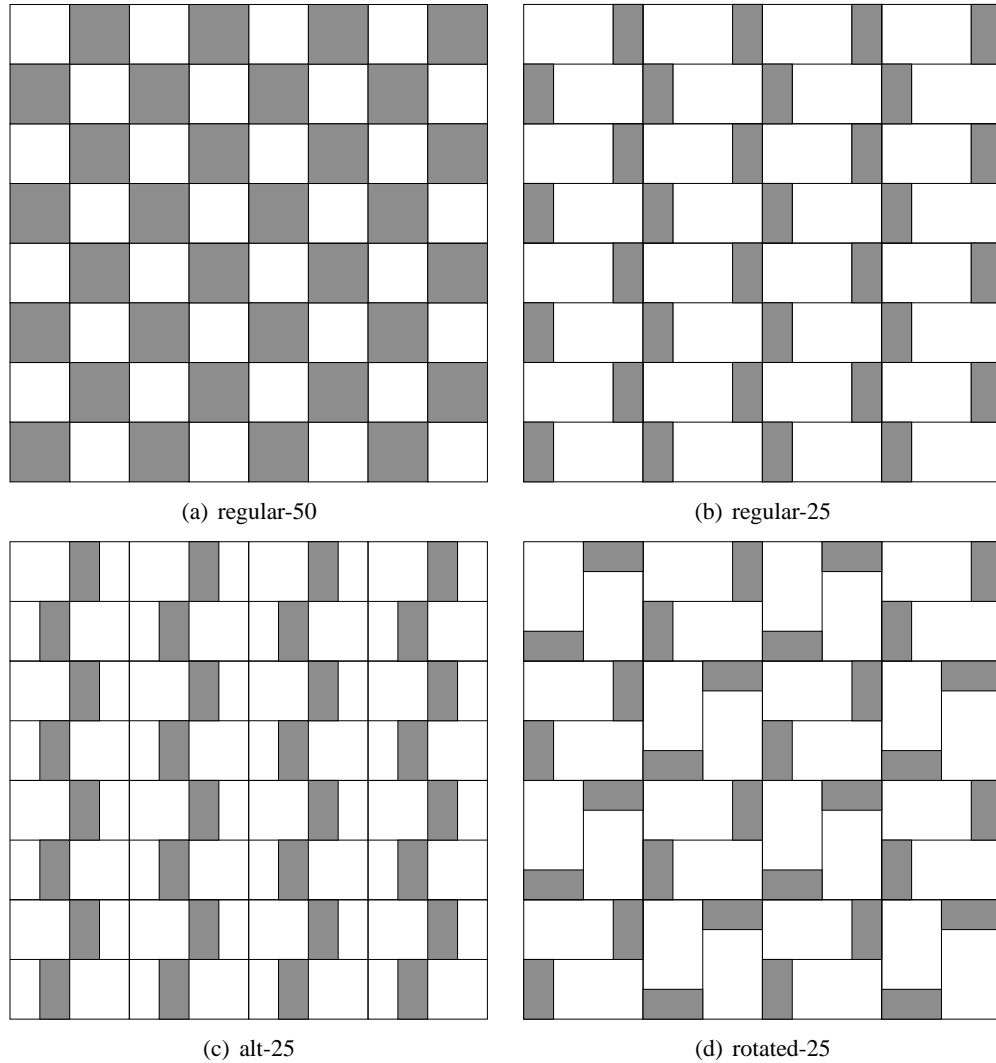


Figure 6.4: Illustration of the several checkerboard configurations studied. Darkly shaded areas denote the cores and the unshaded areas denote the lower-level cache banks. The numeric suffix attached to the name of each configuration indicates the percentage of the die area occupied by the cores. Each figure above shows a manycore die with 32 cores. [88]

comparison, we set the heat transfer co-efficient at the convective boundary of *mobile-regular-50* (top of the die) to be such that the peak temperature on the die matches that of *desktop-regular-50* when the power density on the entire die is uniform. The thermal conductivities of the materials are as in Section 6.3.3. The uniform power density on the cores is set to be  $1 \frac{W}{mm^2}$ , while that on the L2 cache banks is set to be  $0.1 \frac{W}{mm^2}$ . In both the desktop and mobile configurations, we also explore the case of infinite cores as a limit study. When the number of cores is infinity, the power density on

the entire die becomes uniform with a value equal to the average of the power density on the cores and that on the cache banks, weighted by their respective area occupancies (50% in this case).

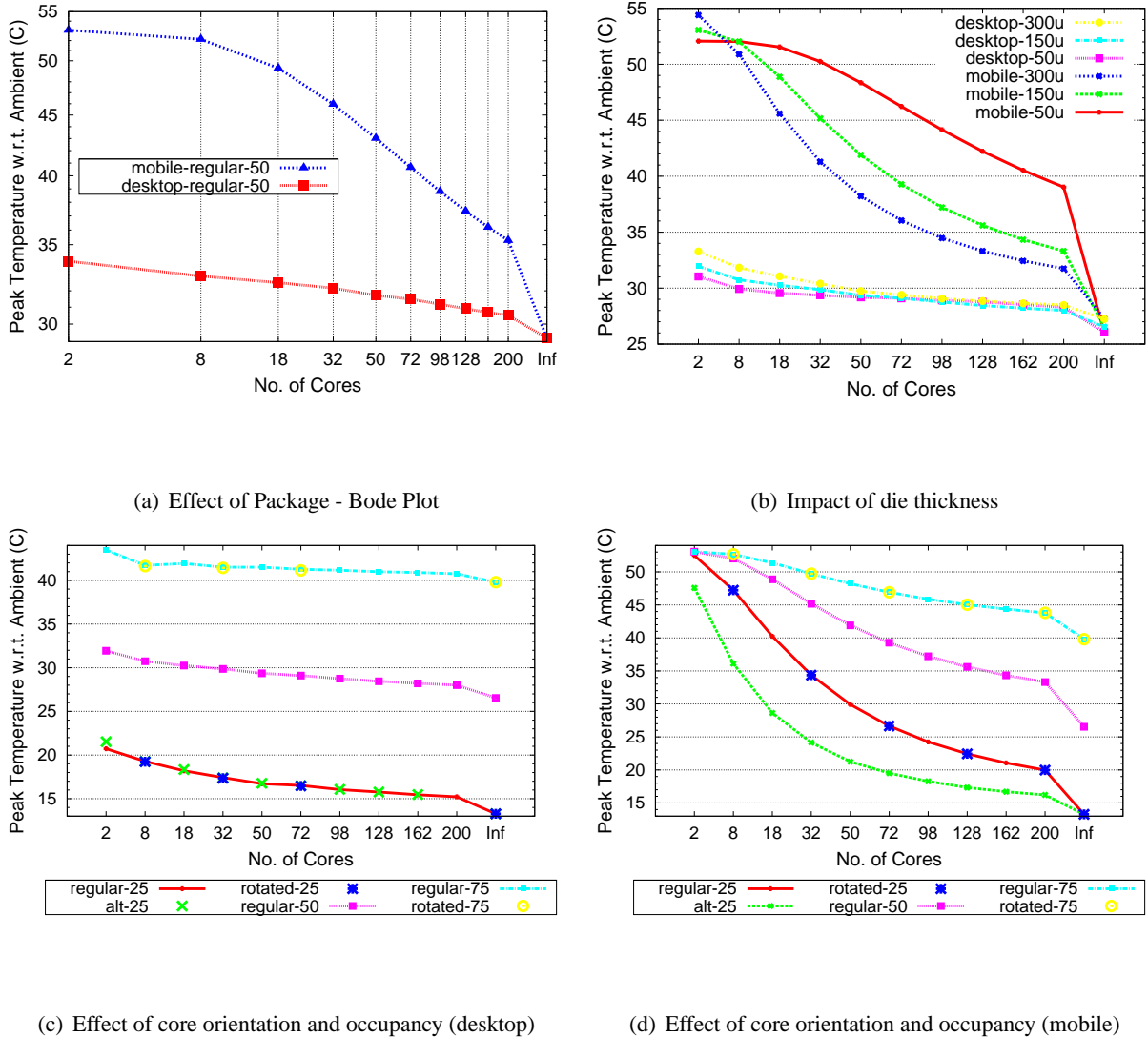


Figure 6.5: Results of the checkerboard experiments. Note the overlap of the curves *rotated-25* and *regular-25* (*rotated-75/regular-75*) in (c) and (d). *alt-25* and *regular-25* are also coincident in (c). [88]

A steady-state thermal analysis of this setup is performed using ANSYS 11.0 FEM solver. For *mobile-regular-50*, we use a lateral grid size of 96 x 96 on the die and for *desktop-regular-50*, a grid size of 48 x 48 is employed (due to software license restriction on the number of nodes). The lateral dimensions of the grid cells in the other layers are the same as in silicon. The vertical modeling



resolution is identical to that in Section 6.3.3. The difference between the peak temperature on the die and the ambient temperature for this study is plotted in Figure 6.5(a). Both the axes are logarithmic (except for the infinity point on the x-axis). Since the number of cores is inversely related to the size of a core, it is similar in sense to spatial frequency. Hence, with the number of cores on the x-axis, the graph is analogous to a Bode plot. The low-pass filter behaviour is clearly evident from the graph of *mobile-regular-50* with a cut-off of around tens of cores. On the other hand, the response is quite shallow for *desktop-regular-50*. Actually, in comparison to *mobile-regular-50*, the response has actually “shifted left” in *desktop-regular-50*. The total vertical thickness of *desktop-regular-50* (in silicon equivalents) is much greater than that of *mobile-regular-50* because of the additional layers. Hence, for a given core size in silicon, the *normalized size* (the ratio of the size to the vertical thickness) is much smaller for the desktop configuration than for the mobile configuration. In comparison to two cores, at infinite cores, there is a 14% reduction in peak temperature for *desktop-regular-50*, which translates into an allowable Thermal Design Power (TDP) increase of an equal amount. At 200 cores, this benefit drops to 10%. These results are similar to the numbers reported in [51]. Although significant, this is not as good as the benefit for *mobile-regular-50*, which is 45% going from two cores to infinite cores and 33.5% at 200 cores. This is because of the heat spreading in copper which smoothes out the response of the low-pass filter.

From the discussion in the previous section, we know that vertical thickness is an important factor to be considered in a study of spatial filtering. Hence, we explore the effect of the die thickness in both desktop and mobile configurations. Figure 6.5(b) displays the results of such an investigation. It shows the peak temperature for a checkerboard similar to Figure 6.4(a) for die thicknesses of  $50\mu\text{m}$ ,  $150\mu\text{m}$  and  $300\mu\text{m}$ . The experimental setup is similar to the one in Figure 6.5(a) except that the power density on the L2 caches is assumed to be zero. It is evident that die thickness has a great impact on the mobile configuration and a very small effect on the desktop case. This behaviour is expected because the die forms the entire heat conduction stack for the mobile case while it is only a small fraction of the total vertical thickness in the desktop case. Also, the trends in spatial filtering (steep response for mobile and shallow response for desktop) remain the same across different

die thicknesses. An interesting behaviour in the mobile configuration is that a thinner die does not necessarily mean lower peak temperature. In fact, it turns out to be the opposite for most core sizes because, the thicker the die, the greater is the lateral heat spreading. In the absence of a heat spreader, for a checkerboard arrangement, the reduction in temperature due to additional spreading in silicon (because of a thicker die) is greater than the increase in temperature due to higher vertical thermal resistance. The exceptions to this rule are the endpoints (two cores and infinite cores), although for different reasons. In the two-core case, the effect of lateral spreading is significantly diminished because all the cores occupy the corners of the die where spreading is possible in only two directions as opposed to four. In case of infinite cores, lateral spreading is made *irrelevant* because of the uniform power density on the entire die. When neighbours have the same temperature, there is no heat flow between them even if the opportunity is offered by the greater die thickness.

An assumption in the previous checkerboard experiments was that the cores occupy 50% of the total die area. In reality, core-to-cache ratio varies across different microarchitectures. Hence, we study the effect of die occupancy by cores here. When the die occupancy is different from 50%, alternative checkerboard configurations are possible. Figure 6.4 illustrates a few such configurations for an occupancy of 25%. *regular-25* is a derivative of *regular-50*. *alt-25* is very similar to *regular-25* except that the cores do not occupy the corners of the chip and the cache banks of a particular processor are placed on both sides of the core. This alternative floorplan is examined in relation to *regular-25* for studying the effect of chip boundaries. For a die occupancy of 25%, since the cores shown in Figure 6.4 have an aspect ratio other than one (1:2 in this case), we also study the effect of rotating the cores by  $90^\circ$  so as to nullify the effect of orientation. This arrangement shown in Figure 6.4(d) is called *rotated-25*. We also explore a die occupancy of 75% through arrangements similar to the 25% case. By the same convention, *regular-75* is nothing but the complement of *regular-25* where cores become cache banks and *vice versa*. *rotated-75* is the complement of *rotated-25*. We do not examine an *alt-75* configuration since that would mean sub-dividing a core into two portions and placing them on either side of a cache bank, which would be difficult in practice. The peak temperature for these arrangements in the desktop and the mobile configurations are presented in figures 6.5(c) and 6.5(d) respectively.

It can be observed that the die occupancy plays the most important role in determining the peak temperature. When the die occupancy is lower, the spatial “duty cycle” is small *i.e.*, two small high power density areas (cores) are separated by a large low power density area (cache banks). On the other hand, the duty cycle is higher for higher die occupancy. Moreover, in comparison with a low occupancy case, the total power dissipated in the corresponding high occupancy die is greater. This is the reason for at least three distinct lines (one for each occupancy level) in each of the graphs 6.5(c) and 6.5(d). Another interesting phenomenon is the impact of the different floorplan configurations. Clearly, core orientations do not matter at all for the desktop or the mobile configurations as can be seen from the overlapping lines and points of *regular/rotated-25* and *regular/rotated-75*. Similarly, the fact that *alt-25* is practically on top of *regular-25* for desktop indicates that placing cores in the corner of the chip does not matter in the presence of the heat spreader and the heat sink. This is consistent with our findings in Section 6.3.3 about the location of a power dissipator on the die (*center-4layers-spread vs. corner-4layers-spread*). This is again because of the lateral spreading in the copper layers. While core orientation did not matter for the mobile configuration, there is a clear difference in moving the cores away from the corners of the die. This behaviour is the reason the curve for *alt-25* is significantly lower than *regular-25* in Figure 6.5(d). The extra space available for lateral spreading cools down the cores in the corners significantly.

In addition to the above-mentioned parameters, we also vary the magnitude of the cache bank power density and the convection heat transfer coefficient. The effect of these parameters on peak temperature tracks the trends observed above with a steep response in the mobile configuration and a shallow pattern in the desktop configuration. In summary, there is a significant thermal benefit in the choice of many small cores as opposed to a few large cores due to spatial filtering.

#### 6.4.1.2 Local vs. Global Thermal Management

In the context of manycore processors, it is important to study the relationship between core size and the granularity of thermal management. For a given number of cores, it is useful to know the appropriate granularity at which DTM is most effective. Coarser granularity would incur excessive

performance cost to contain the temperature below the thermal emergency threshold, while finer granularity would incur superfluous hardware cost. Hence, in this section, we investigate the effectiveness of DTM at various levels of granularity: locally at the level of a functional block (*e.g.*, register file, cache, branch predictor *etc.*), an architectural sub-domain within a core (*e.g.*, the integer pipeline, the floating-point pipeline, the memory pipeline *etc.*) or globally at the level of an entire core. This investigation is accomplished through a microarchitectural simulation study of a manycore architecture with each core resembling that of an Alpha 21364 as in [103].

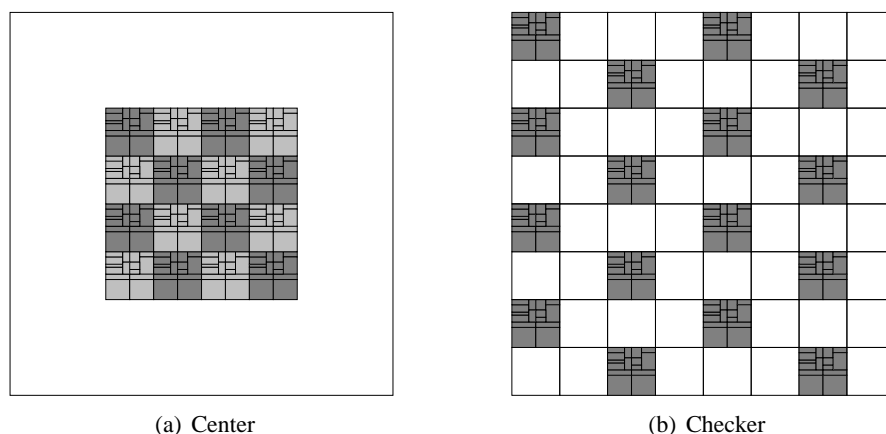


Figure 6.6: Illustration of the two different manycore scenarios simulated. The unshaded areas indicate second-level cache. The shaded areas denote cores resembling Alpha 21364. [88]

We simulate two different scenarios marking the two extremes of core arrangement with respect to spatial filtering. In the first, all the cores are arranged close to each other at the center of the die with the second-level cache surrounding them. The floorplan of such a setup with 16 cores is shown in Figure 6.6(a). The shaded areas denote the cores while the unshaded areas denote the L2 cache (the cores are shaded in alternating light and dark shades for better visibility). In the second, the cores and the L2 cache blocks are arranged in a checkerboard-like fashion similar to the previous section. The floorplan of such a setup is shown in Figure 6.6(b). The former scenario is called *center* and the latter as *checker*. In both cases, the L2 cache is assumed to occupy 75% of the total die area.

As in the previous section, we vary the number of cores, keeping the power density in each core (and hence the total power on the entire die) constant. In order to examine the effectiveness

of DTM, out of all the  $n$  cores, we select a single core that is closest to the center of the die and manage its temperature by turning off computational activity in regions of varying sizes. Since we are interested in the best-case DTM potential of a region, we only study the steady-state temperature due to its shutdown. If turning a region off does not help in the steady state, it is certainly not going to help in the transient.

The regions are turned off at three levels of granularity. First is at the functional unit level where each of the 15 architectural units are turned off separately, independent of each other. The unit that provides the greatest temperature reduction is chosen as the candidate for comparison against other levels of granularity. The second level of granularity is that of a sub-domain within a core. The core is subdivided into the following four non-overlapping sub-domains:

- *Fetch engine*: I-cache, I-TLB, branch predictor and decode logic.
- *Integer engine*: Issue queue, register file and execution units.
- *FP engine*: Issue queue, register file and execution units.
- *Load-store engine*: Load-store ordering queue, D-cache and D-TLB.

Similar to the functional unit level, each sub-domain is also turned off independent of each other and the best-case peak temperature reduction is chosen. The third and the coarsest level of granularity we consider is that of an entire core. We do not consider coarser levels of granularity (such as groups of cores) here. At the coarsest level, we only turn off a single core closest to the center of the die.

The experimental setup for this study involves the use of SPEC2000 benchmark suite [106] simulated on a tool set involving modified versions of the SimpleScalar performance model [5], Wattch power model [13] and HotSpot 4.1 thermal model [50]. The simulation run of each benchmark involves a subset of 500 Million instructions identified using the SimPoint [96] tool. The average power density values generated from these runs are for a single core at 130 nm. For these steady-state thermal simulations, we vary the number of cores by scaling the lateral dimensions of each core linearly. All the cores are assumed to run the same benchmark and hence the power numbers are replicated across them. The thermal model parameters are set to the defaults of HotSpot

4.1 except the die size which is set to 12.4 mm x 12.4 mm (to reflect the 75% L2 cache area) and the convection resistance of the package which is set to  $0.75 \frac{K}{W}$ . We use the grid-based model of HotSpot with a resolution of 256 x 256 grid cells on the die. Of the 26 SPEC2000 benchmarks, only 11 (7 *int* (*bzip2*, *crafty*, *eon*, *gcc*, *gzip*, *perlbmk*, *vortex*) and 4 *fp* (*art*, *galgel*, *mesa*, *sixtrack*)) have peak steady-state temperatures above the thermal emergency threshold of 85°C. Since only these benchmarks need DTM, the results presented here are averages over these 11 benchmarks.

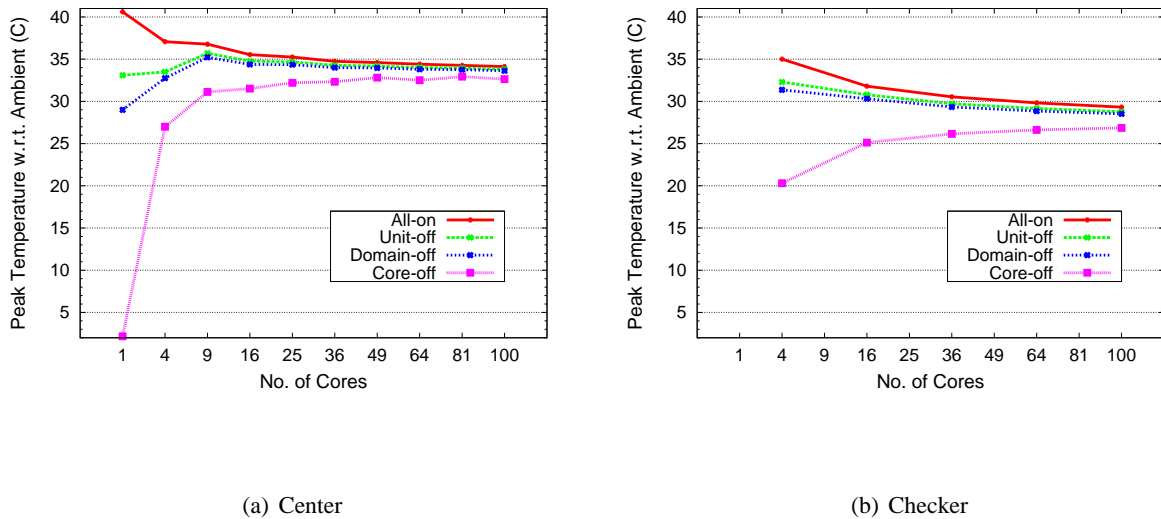


Figure 6.7: Results of the local vs. global thermal management study. [88]

Figure 6.7 presents the results of this study. It plots the difference between the peak and the ambient temperatures for the various levels of DTM granularity as a function of the number of cores averaged over the 11 benchmarks listed above. Figure 6.7(a) plots the results for the *center* scenario with the cores at the center while Figure 6.7(b) plots the same for the *checker* scenario. The *All-on* curves denote the case with no DTM. It can be seen that it is a decreasing curve with about 16% temperature reduction going from a single core to 100 cores for the *center* case and 16% going from 4 cores to 100 cores for the *checker* case. This temperature reduction is solely due to spatial filtering since the power density and total power remain constant across the number of cores. This is consistent with the findings in the previous section. Also, the curve for *checker* is lower than that for *center* because there is a much wider spatial distribution of heat in the latter because the cores are dispersed.

The *Unit-off*, *Domain-off* and *Core-off* curves denote the best-case (lowest) peak temperatures for the three levels of DTM granularity. Since the power density is kept constant across the number of cores, the total power dissipated in a single core decreases with increasing cores. Hence, the thermal benefit in shutting down a single core also decreases. This reduced benefit is the reason why the *Core-off* curves are mostly increasing. The *Unit-off* and *Domain-off* curves on the other hand, show both increasing and decreasing behaviours. This pattern is a function of two competing factors: on the one hand, the peak temperature (even with all the units turned on) diminishes due to spatial filtering and on the other, the power dissipated (and saved) per unit (or per sub-domain) decreases due to technology scaling. In the curves for the *center* scenario, the latter predominates the former up to 9 cores after which the former takes over.

The main conclusion one can draw from the graphs is that local thermal management (at the functional unit level or at the sub-domain level) ceases to be effective after 9-16 cores. This behaviour can be seen from the minimal difference between the *Unit-off* and *Domain-off* curves relative to the *All-on* curve after sixteen cores. In fact, this behaviour happens even earlier for the *center* scenario at nine cores. Also, functional unit level management degenerates into sub-domain level management even at four cores. It can also be seen that for 50+ cores, even turning off an entire core is not very different from *All-on*. This fact suggests that one might have to toggle groups of cores at that core size. Another interesting finding is that even though the integer register file is the hottest unit in most of our experiments, it is not the one that gives the most thermal benefit when shut down. This behaviour is a function of two factors: first, a higher amount of power is saved in turning off the data cache; and second, other units like the data cache and the integer ALU are hot as well - only marginally cooler than the register file. For these reasons, the load-store engine is the sub-domain that offers the best thermal benefit when shut down. Finally when the entire core is shut down, the hottest units are actually the ones in the periphery of the core and especially the ones adjoining the integer register file of the neighbouring core in the case of *center* scenario.

We also performed sensitivity studies varying the ratio of the L2 cache area and the package characteristics by removing the heat sink and spreader. The results of these studies were not much different from what has been presented above. Thus, to summarize, we can learn from this section

that within-core thermal management ceases to be effective beyond tens of cores.

#### 6.4.2 Sub-blocks with High Aspect Ratio

We saw in Section 6.3.3.2 that the aspect ratio of a block bears an exponential relationship with its peak temperature. Several microarchitectural features like register file entries and cache lines are of very high aspect ratios. Their access can be controlled by software either directly (as with registers) or indirectly (as with cache lines). Hence, in this context, it is interesting to examine the question of whether pathological code behaviour (intentional or not) can heat up such sub-blocks to undesirable levels by concentrating activity in them. In this section, we approach this question for the data cache from a spatial granularity angle.

Under typical program behaviour, the data array is usually not the hot spot in a cache because, over the time scales at which silicon heats up, the accesses to the data array are usually well-distributed in space. Also, on every cache access, only the line that is addressed dissipates dynamic energy. On the other hand, the periphery is usually the hot spot in the cache since the address/data drivers, sense amps, pre-decoders *etc.* dissipate dynamic energy every time the cache is accessed. However, under malicious program behaviour, a single cache line can potentially become the hot spot if the program directs all cache accesses to it. In this section, we perform a microarchitectural study to examine this possibility.

We model a processor similar to the Alpha 21364 as in [103] but scaled to 90 nm. We first collect a representative set of per-unit power consumption values. We do so by simulating the SPEC2000 benchmark suite over a modeling setup similar to that in Section 6.4.1.2. From these simulations, we select the benchmark *bzip2* as the candidate for further exploration and illustration, since it has the highest average data cache temperature.

Next, we subdivide the data cache power consumption into those of the individual sub-blocks within the data cache. This is accomplished using the Cacti 5.3 [110] tool that models the performance, dynamic power, leakage power and area of caches. We use Cacti to model an Alpha-like data cache (64 KB, 2-way, 64-byte lines) and separate the data array into the active lines that dissipate dynamic power and the remainder of passive lines that dissipate leakage. We then aggregate the



periphery of each sub-array into a single sub-block (including the row and column decoders, sense amps, muxes, comparators *etc.*). The address-input/data-output drivers and the request/reply networks are aggregated into the “outside mat” sub-block. For ease of modeling, the tag and the data portions of the sub-blocks are aggregated into one. Table 6.1 shows the area, power consumption and power density of each of these sub-blocks for the *bzip2* benchmark. The area and power numbers are expressed as a percentage of the total cache area and power respectively while the power density is represented as a ratio to the average cache power density. It is to be noted that among these

Region	Area %	Power %	Power Density Ratio
Outside Mat	56.2	61.4	1.09
Passive Lines	25.7	7.0	0.27
Sub-array Periphery	18.1	22.6	1.24
Active Line	0.03	9.0	358.4

Table 6.1: Area, power and power density distribution within the data cache for the *bzip2* benchmark. [88]

sub-blocks, the location and the power consumption of only the active lines (and hence the passive lines as well) can be easily controlled by program behaviour. The other sub-blocks are typically a lot less amenable to program control. Moreover, their thermal behaviour is fairly similar across all accesses, independent of the address/location of the line accessed. Table 6.1 shows that much of the cache area is occupied by the address/data drivers and request/reply networks. The SRAM array occupies only about a quarter of the cache area with most of it being passive lines at any given time. The power density in these passive lines (due to leakage) is only about a quarter of the average cache power density. On the other hand, the power density at the sub-array periphery is about 24% more than the average. On any given cache access, the power dissipated in the active line is about 9% of the total cache power. For the 64K cache with 64-byte lines, since there are 1024 lines, the area of an active line is a mere 0.03% of the total cache area. Hence, the power density of an active cache line is two orders of magnitude greater than the average. Such high power density however, is not sustained for the time scale at which silicon heats up (tens of thousands of cycles), since the accesses to the cache lines are usually distributed in space. However, pathological code behaviour can concentrate activity in a single line and sustain such a high power density, potentially rising

its temperature to undesirable levels. However, the small area of the line and its high aspect ratio (between two and three orders of magnitude greater compared to the cache itself) increase lateral heat transport. It is not immediately clear whether this spatial filtering counteracts the high power density that could be sustained by repeated accesses to a single line. In order to study the effect of cache layout on its thermal profile, we investigate two different cache arrangements. Figure 6.8 illustrates them. The first is a simplified arrangement with a single sub-array. The shaded portion includes both the “outside mat” sub-block and the sub-array periphery. The second is a placement optimized for performance with the sub-arrays sized in such a manner (using Cacti) that the delays along the two dimensions are balanced. It subdivides the SRAM array into 16 sub-arrays, each with 64 lines. The unshaded areas denote the SRAM sub-arrays. The lightly shaded areas denote the periphery of each sub-array and the darkly shaded area denotes the “outside mat” sub-block. It should be noted that the aspect ratio of a single cache line in *Placement 1* is higher than that of a line in *Placement 2*. Since the power densities of the sub-blocks are the same in both the layouts, in considering the two placements, we are actually investigating the impact of sub-block size and aspect ratio on temperature.

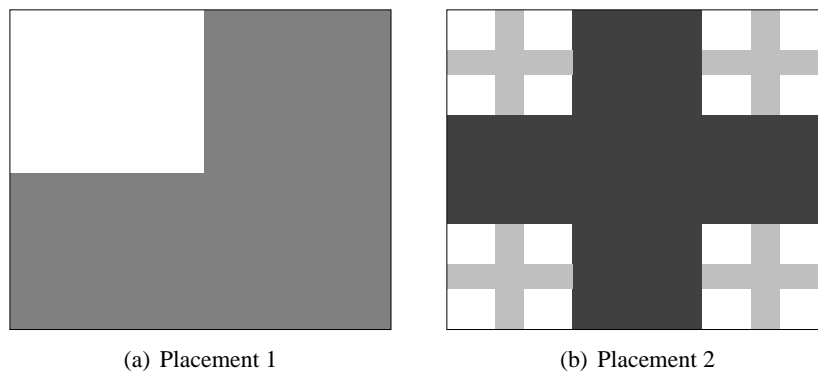


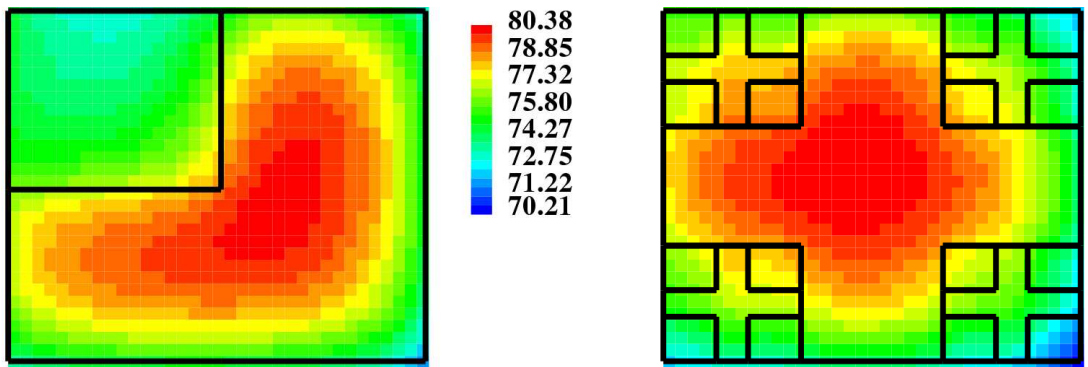
Figure 6.8: Two different placements studied for the data cache. The unshaded areas denote the SRAM sub-arrays while the shaded portions indicate the periphery and routing. [88]

In modeling the thermal distribution of these cache arrangements, a challenge arises because of the vastly differing sizes of the sub-blocks. While the other sub-blocks of the cache are comparable in size, the active cache line is different in size and aspect ratio by about three orders of magnitude. Modeling the entire die at the resolution of the single cache line is prohibitive. In an FEM parlance,

doing so would entail millions of FEM nodes. Hence, we perform thermal modeling at two distinct resolutions. First, we model the case with all the sub-blocks except the active line, *i.e.*, all other cache lines are passive, dissipating only leakage power. This modeling is done using the grid-based model of the HotSpot tool at a grid size of 256 x 256 for the entire die (The data cache alone occupies a sub-grid of about 30 x 35). The package parameters are set to their default values except the convection resistance of the package, which is set to  $0.75 \frac{K}{W}$ . Next, we model the active cache line alone (with the rest of the die dissipating zero power). This simulation is done using the ANSYS tool for a die and package configuration identical to the first step above. Due to the small size of the cache line, the thermal distribution of the die in this case is independent of the location of the line within the cache. Hence, we assume that the cache line is at the center of the die and exploit the symmetry of such a setup about the two axes, thereby reducing the number of required nodes by a factor of four. Furthermore, ANSYS employs a non-uniform mesh to model this setup and hence is able to model it with tens of thousands of FEM nodes. Finally, for combining these two steps, since thermal conduction is a linear phenomenon, we use the principle of superposition and sum up the temperatures. It should be noted that since we are interested in the worst-case, we only consider steady-state thermal behaviour here.

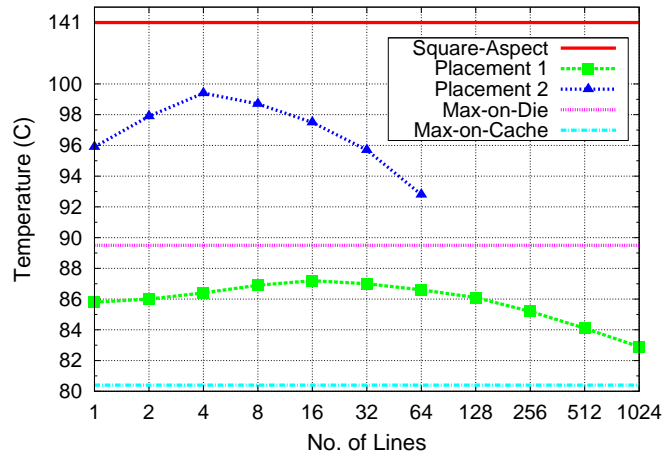
Figure 6.9 shows the results of these experiments. Figures 6.9(a) and 6.9(b) plot the spatial temperature distribution of the data cache for the case where all the lines are passive (the first step described above). It can be seen that the hottest region of the cache lies outside the SRAM array in both cases. This observation is consistent with the power density data since the passive cells have the lowest power density within the cache. It is also clear from the performance-optimized placement that the address/data drivers and the request/reply networks are the most significant contributors to the cache temperature (however, such a conclusion cannot be drawn from the naive placement as it joins the sub-array periphery and “outside mat” sub-blocks into one).

Figure 6.9(c) plots the results of including the active cache lines. It plots the peak temperature within the active lines for both the naive and performance-optimized cache placements. Malicious code intent upon heating up the cache can do so either by concentrating activity on a single cache line, or on  $n$  contiguous cache lines in a round-robin fashion to prevent spatial filtering due to



(a) Placement 1

(b) Placement 2



(c) Active Lines

Figure 6.9: Results of the data cache thermal experiments. (a) and (b) show the case when all lines are passive. (c) plots the peak temperature of the active lines as a function of the number of lines that are accessed contiguously. [88]

the small size and large aspect ratio of a single cache line. Increasing the size of the target region reduces average power in each of the targeted line since it accesses every cache line only once every  $n$  cycles. Thus the round-robin technique is actually a trade-off between power density and spatial filtering. Figure 6.9(c) plots this trade-off for both the cache placements. For the naive placement,

with 1024 lines in the sub-array, a round-robin technique can access up to 1024 contiguous lines. For the performance-optimized placement, it is restricted to 64 lines since each sub-array only has that many lines. Accessing the lines of another sub-array reduces the advantage of contiguity. The trade-off between power density and spatial filtering is evident in both the *Placement 1* and *Placement 2* curves. They both increase up to a point and then start falling off. The maximum point is when the reduction in power density starts to outweigh the benefit from size and aspect ratio.

As mentioned above, the thermal behaviour of an active cache line has two mitigating factors that counterbalance its high power density: its size and its aspect ratio. The *Square-Aspect* curve isolates the effect of the former. It shows the peak temperature of a single cache line assuming that it is a square. The area and the total power dissipated are assumed to be the same as a normal cache line. It can be seen from the graph that the *Placement \** curves are far below the *Square-Aspect* curve. This behaviour means that aspect ratio is a significant determinant in the peak temperature of a cache line. In fact, this is the reason why the *Placement 2* curve is higher than *Placement 1* (since the aspect ratio of a cache line in *Placement 1* is higher than that of *Placement 2*). Furthermore, the graph also plots the peak temperatures within the cache (*Max-on-Cache*) and across the entire die (*Max-on-Die*) as references. It is to be noted that these reference temperatures are assuming zero active lines.

It can be seen that for *Placement 1*, the steady-state temperature rise due to worst-case code behaviour is only 6.8 degrees above the peak temperature within the cache. This increase is not sufficient to make the cache lines the hottest spots within the die (as can be seen from *Max-on-Die*). On the other hand, for *Placement 2*, the maximum rise is 19 degrees, which is significant enough to make it 9.9 degrees hotter than the hottest region of the die. As we saw above, this difference is due to the aspect ratio of cache lines in *Placement 1* vs. *Placement 2*.

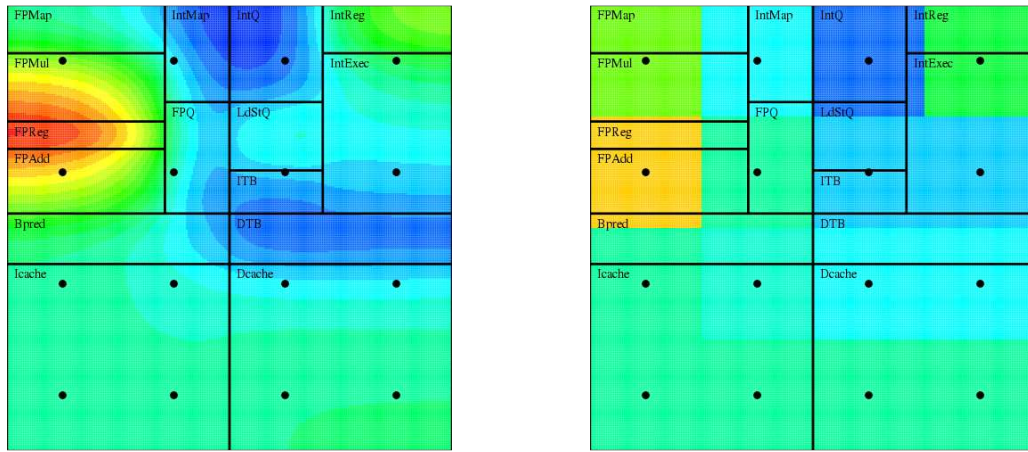
In conclusion, we have provided an example of a cache layout where the application behaviour can cause the lines to become a hot spot and another where this is not possible. However, in both cases, we have shown that aspect ratio plays a crucial role in reducing the peak temperature from an extraordinary value at a square aspect ratio (141°C in our examples) to a much more manageable level (99.4°C and 87.2°C) in spite of a power density that is orders of magnitude higher

than average.

### 6.4.3 Thermal Sensors

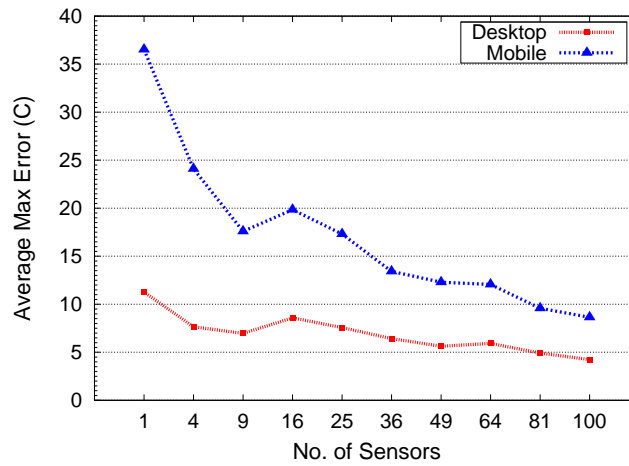
Thermal sensing is a necessary component of Dynamic Thermal Management (DTM), and sensor accuracy determines the performance of DTM [103]. There can be three major sources of errors in a sensor's thermal measurement. The first is calibration error where the sensor is off by a constant or a linear factor. This can usually be rectified by calibrating the sensor during "burn-in" or if that is not possible, by compensating for it by modifying the temperature at which DTM engages. The second is noise (*e.g.* modeled as following a particular probability distribution like Gaussian, uniform *etc.*). This noise is usually a temporal effect and the measurements vary around the mean of the distribution. Like calibration error, this noise can also be addressed easily (by averaging across many sample measurements because such error diminishes as the square-root of the number of samples [103]). Since the thermal time constant of silicon is many times greater than the typical thermal sensor bandwidth, multiple readings can be taken before the temperature rises significantly. The third source of sensor error is the spatial thermal gradient. The thermal sensor might not be co-located with a hot spot and hence, the temperature seen by it could be lower than the actual hot spot temperature. This error is the most difficult to address as it depends not only upon the temperature in the immediate vicinity of the sensor but also on its neighbours. It is also directly related to the spatial thermal filtering effect. The steeper the thermal gradient, the greater is this error. Similarly, the farther from the hot spot the sensor is, the greater the error. As Lee *et. al.* [66] point out, there is an exponential relationship between the distance from the hot spot and the sensor error. In other words, similar to the spatial thermal filtering we studied, silicon acts like a low-pass filter for the sensor error due to spatial thermal gradient (with the spatial frequency =  $1/\text{inter-sensor distance}$ ).

Figure 6.10 explains this aspect. Figure 6.10(a) shows the steady-state thermal profile of the *sixtrack* benchmark from the SPEC2000 benchmark suite for a 6.2 mm x 6.2 mm die with an Alpha 21264-like core in a "no L2" configuration with a regular grid of 16 thermal sensors marked by the black dots. Figure 6.10(b) is a picture of what is seen by the thermal sensors. Every point on the die is assigned the temperature of the sensor that is closest to it. This is the so-called "nearest



(a) Actual

(b) Sensed



(c) Sensor Error

Figure 6.10: Relationship between sensor error and inter-sensor distance. For an Alpha 21264-like core, (a) shows a sample temperature profile and (b) shows what is seen by the sensors. The black dots denote the positions of the sensors. (c) plots the average maximum sensor error of the SPEC2000 benchmark suite for two different types of package. [88]

neighbour” algorithm [81]. Figure 6.10(c) plots the result of an experiment varying the number of sensors and studying the worst-case transient error *i.e.*, the maximum difference between the actual temperature distribution of the die and the thermal profile as seen by the sensors using the

nearest neighbour algorithm. The experimental setup is similar to that in section 6.4.1.2 except that HotSpot 4.1 has been modified to include modeling of a configuration without the package as is the case in hand-held mobile processors. This configuration is denoted by the *mobile* curve. The *desktop* curve includes a typical desktop package with TIM, heat sink and spreader. Furthermore, these experiments are transient thermal simulations with a sampling interval of 0.33 ms. Also, for higher simulation speed, the thermal model resolution is set to a grid size of 64 x 64 as opposed to 256 x 256. At the end of each interval, the maximum error between the actual thermal profile and the interpolated profile across the entire die is computed. We call this error as the *spatial* error. The maximum of these spatial error values is then computed over the whole simulation of 500 million instructions. The maximum error thus computed also includes the *temporal* component of the error. The values plotted in the graph are averages of this maximum error for the entire SPEC2000 benchmark suite.

Clearly, the *desktop* and *mobile* packages behave differently. The former has a shallow curve while the latter has a steep one, similar to what we saw in Section 6.4.1.1. As before, this behaviour is due to the better lateral spreading in copper for the *desktop* configuration. Moreover, the sensor error is much higher for the *mobile* curve indicating the need for a higher number of sensors to compensate for the lack of lateral spreading. It can also be seen that the curves show diminishing returns around the mid-range of the curves (mid-tens of sensors) beyond which, increasing the number of sensors does not provide commensurate increase in accuracy. In fact, 36 sensors appears to be an optimal spot in the cost vs. accuracy trade-off. Furthermore, the graph shows a non-monotonic pattern. The error is higher for a grid of sixteen sensors than for a grid of nine sensors. This behaviour is because of the location of the sensors relative to the functional blocks. In the increasing portions of the curves shown in figure 6.10(c), although the grid resolution increases, the sensors move farther from the nearest hot spot, resulting in an increase in the sensor error.

#### 6.4.3.1 Sensor Interpolation

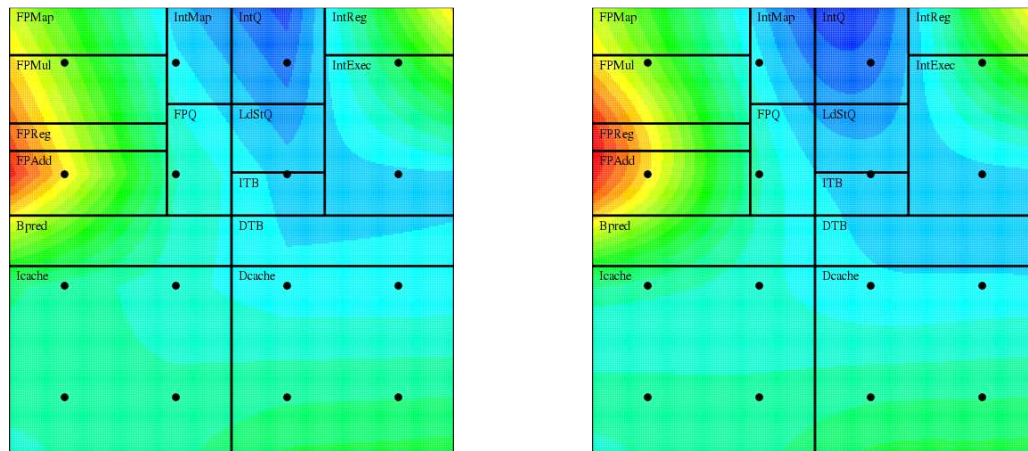
A natural means to improve the accuracy of temperature estimation using sensors is to employ spatial interpolation in the regions between them. The problem then becomes the estimation of the



temperature field on the die at certain desired points (*e.g.* centers of each functional block, borders between two hot units *etc.*), given the  $n$  sensor positions and their readings, where  $n$  is the number of sensors. Previous research has addressed this problem and solutions involve a heavyweight control theoretic filter (Kalman filter) [94] on the one end of the spectrum and a simple sub-linear interpolator [71] on the other. However, with the insight that spatial filtering behaves differently for different types of package, a spectrum of choices *between* becomes interesting in the cost *vs.* accuracy trade-off. The objective of this section is to evaluate such a trade-off and characterize a couple of interpolation schemes. Specifically, bilinear and bicubic spline interpolators [81] are two low-cost interpolation schemes that are very popular in the image processing community. In fact, the former is supported in hardware by many Graphic Processing Units (GPU) due to its low computational cost.

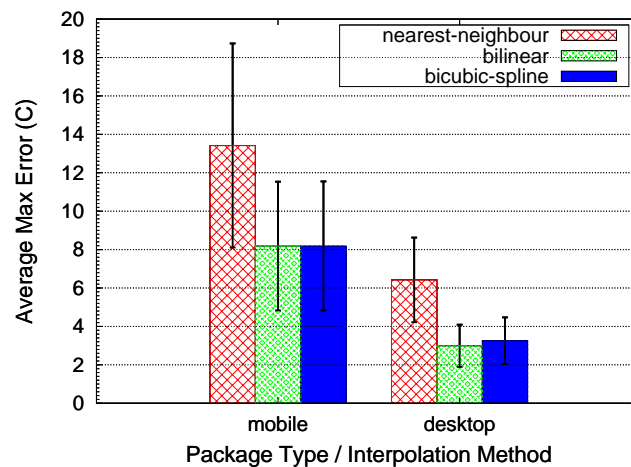
For each desired point at which temperature needs to be estimated, the nearest neighbour interpolation algorithm mentioned above is a constant-time algorithm. Similarly, the sub-linear interpolator in [71] and the bilinear algorithm are all constant-time schemes. At the other end, the steady-state Kalman filter is cubic in the number of sensors. An online version of it employed by [94] is quadratic in the number of sensors. The bicubic spline interpolation scheme occupies a position in the middle since it is linear in the number of sensors. It is worth mentioning that while we assume a regular grid of sensors for ease of modeling, the interpolation algorithms themselves are not constrained to a regular grid of thermal sensors. The above-mentioned time complexities of the interpolation schemes assume a regular grid of sensors. In case of non-uniform sensor arrangement, the input to the algorithms would also include the locations of the sensors and an additional step would be required in locating the desired point (where temperature needs to be estimated) relative to the sensors surrounding it. This step would increase the above-mentioned interpolation complexities by an additional  $\log n$  term. Furthermore, it should be noted that while Kalman filter is almost a zero-error scheme, its input includes per-unit power consumption estimates which might not be normally available or easily computable. Hence, we do not consider it in our evaluation. Also, the sub-linear interpolation scheme in [71] uses weights ( $= 0.5$ ) such that it is between a zeroth order nearest-neighbour interpolation scheme and a (bi)linear interpolator. Moreover, it employs interpo-

lation only in the vicinity of the hottest sensor, which might be inaccurate if none of the sensors are at the hottest spot. One might have to consider the top  $k$  hottest sensors instead of just *the* hottest sensor. Hence, we use the bilinear interpolation scheme as a proxy for the interpolation algorithm in [71].



(a) Bilinear

(b) Bicubic Spline



(c) Results

Figure 6.11: Results of the interpolation study. (a) and (b) illustrate the behaviour of the bilinear and bicubic spline algorithms for the thermal profile and sensor readings as in figures 6.10(a) and (b) respectively. (c) presents the experimental results of comparing the algorithms against the nearest-neighbour interpolation scheme. [88]

Figures 6.11(a) and 6.11(b) illustrate the operation of the bilinear and the bicubic interpolation schemes for the steady-state thermal profile of the *sixtrack* benchmark shown in Figure 6.10(a) and the sensor readings shown in Figure 6.10(b). It turns out that bilinear interpolation results in discontinuous first derivatives at the sensor locations. Bicubic spline interpolation solves this problem by using a third degree polynomial for curve fit (in each of x and y dimensions) and mandates that the derivatives match at the sensor positions. Hence, it is usually smoother than bilinear interpolation. This behaviour can be observed from the pictures, especially near the sensor inside the *FPAdd* unit. It should also be noted that in order to be able to interpolate in the region outside the peripheral sensors but within the chip boundaries, the temperatures of the peripheral sensors are linearly *extrapolated* onto the chip boundaries. The interpolation algorithms are run after this first extrapolation step. Figure 6.11(c) plots the result of the comparison between the interpolation algorithms for a regular 6 x 6 grid of sensors. The experimental setup is similar to that used for Figure 6.10(c). The graph plots the average maximum error for the entire SPEC2000 benchmark suite with the error bars marking one standard deviation above and below the mean.

The main conclusion is that interpolation reduces the sensor errors significantly (39% for the *mobile* case and 53% for the *desktop* case). However, while interpolation leads to acceptable sensor error in the *desktop* case (three degrees on average), the error is still quite high for the *mobile* case. Hence, the absence of lateral smoothing by copper leads to the necessity for a higher number of sensors even with interpolation. Another observation is that the difference between the performance of bicubic spline interpolation and bilinear interpolation is marginal. In fact, bicubic spline interpolation is even slightly worse than bilinear for the *desktop* case showing that, in this case, a straight line is a better approximation for its thermal distribution than a cubic polynomial. The minimal computational overhead of the bilinear scheme is an added advantage to its accuracy. We also studied the effect of the interpolation schemes for denser sensor grids (up to 10 x 10). The benefit due to interpolation improves with the number of sensors since there are many more points to interpolate from. However, the overall trend remains the same with much of the benefit due to interpolation coming from bilinear interpolation itself.

## 6.5 Conclusions and Future Work

This chapter presented an analysis of the role of heating granularity on microarchitectural thermal management. It identified spatial thermal filtering as a crucial factor in the study of thermal granularity and derived an analytical equation to model the same. It then explored the agreement of the analytical model with practice and found the behaviour to be dependent on the type of package (*i.e.*, whether a heat spreader and heat sink were present). Using this insight, it then provided three microarchitectural examples where spatial thermal granularity is important:

- It presented a thermal evaluation of a manycore architecture and demonstrated the thermal benefit in choosing many small cores as opposed to a few large cores when all other variables remain constant. It also studied the effect of core size on local *vs.* global thermal management and concluded that local thermal management ceases to be effective beyond low tens of cores.
- It examined the question of whether pathological code behaviour can cause the catastrophic heating of high aspect ratio sub-blocks such as cache lines. While the answer depended on the cache layout and dimensions, aspect ratio always plays a crucial role in mitigating high power density.
- It explained the relationship between inter-sensor distance and sensor accuracy from a spatial filtering standpoint and studied the impact of bilinear and bicubic spline interpolation techniques on sensor accuracy. Interpolation contributes to the reduction of sensor errors, but the magnitude of the benefit was dependent upon the package. Moreover, the bilinear algorithm was almost as accurate as bicubic spline and more efficient to compute.

In the manycore thermal study, this work identified the ratio of core area to that of the L2 cache to be a crucial determinant of peak temperature. This observation suggests that a thermally-aware multicore floorplanning scheme that has flexibility in the use of L2 cache blocks as thermal buffers can potentially reduce the peak temperature significantly. In fact, floorplanning the cores and L2 blocks in such a manner as to minimize the size of the cores and maximize the spacing between them (thereby increasing the spatial frequency to exploit the spatial filtering behaviour) was the

topic of the previous chapter. This work mainly focused on superscalar cores. With the advent of SIMD architectures like GPUs, the question of whether the different SIMD “lanes” offer a thermal granularity advantage due to their small size is an interesting area of future work.

# Chapter 7

## Conclusion

---

This dissertation described research in the area of microarchitectural thermal modeling and management. It presented my contributions to the design of HotSpot, a microarchitectural thermal modeling infrastructure. It also explained the analytical foundations in the choice of appropriate thermal model parameters. Furthermore, it offered evidence to the hypothesis that the microarchitecture is effective in addressing the temperature challenge through efficient management both in the temporal and the spatial dimensions. In the temporal dimension, it achieved thermal control through the judicious throttling of computational activity, thereby spreading the power consumption across time. In the spatial dimension, it distributed heat generation as uniformly as possible both through microarchitectural floorplanning (within a core and across multiple cores of a microprocessor) and migration of computational activity to spare functional blocks. Since the effectiveness of static and dynamic thermal management depends on the granularity at which high and low power density blocks are interleaved and controlled, this work also analytically examined the question of the proper granularity of thermal management. This study concluded that although within-core Dynamic Thermal Management (DTM) might become less effective at smaller core sizes, opportunities for static management arise in the form of multicore floorplanning using second-level cache banks. This observation should be seen in the context of the broader trend in computer architecture towards multicore and manycore processors. In the near term, as core sizes are still very large relative to the die thickness, the techniques of the earlier chapters in this dissertation (DTM and single-core floorplanning) are relevant possibilities. For the longer term, as the core sizes shrink

to become comparable to the die thickness, a synergy between the techniques presented in the earlier chapters and those discussed in the later ones (*e.g.* multicore floorplanning) becomes more applicable.

This work has resulted in the following specific contributions:

1. It has developed an accurate and efficient thermal model whose design is informed by analytical formulations from the equations of thermal conduction [50,52,103]. The HotSpot thermal model is widely used in the architecture research community and can be freely downloaded from the web. Future extensions to HotSpot can examine the application of state-of-the-art numerical techniques such as the Conjugate Gradient Method [81] or explore the integration of off-the-shelf solver libraries. With the rise of multicores and SIMD processors like GPUs, there is potential to obtain performance improvements of many orders of magnitude through a combination of algorithmic enhancements and parallelization [20]. Such speedups could then translate into more extensive modeling abilities.
2. It has created an empirical leakage power model that captures its relationship with temperature and supply voltage effectively [103].
3. It has proposed and evaluated new microarchitectural thermal management schemes that manage the temperature of a single-core microprocessor with little performance loss [101, 102, 103]. When the maximum operating temperature is dictated by timing and not physical reliability concerns, “temperature-tracking” frequency scaling, that lowers the frequency when the trigger temperature is exceeded (but does not stop the clock), performs the best. When physical reliability concerns require that the temperature never exceed the specification, the best solutions are an idealized form of DVS that incurs no stalls when changing the voltage/frequency or a feedback-controlled localized toggling scheme that toggled subdomains of the processor independently and a computation-migration scheme that uses a spare integer register file. The localized schemes perform better than global clock gating and non-ideal global DVS since they exploit instruction-level parallelism while the global schemes slow down the entire processor. A significant portion of the performance loss of

all these schemes is due to sensor error, underlining the importance of sensor accuracy in thermal management. These results show that microarchitectural DTM schemes are effective in managing the temperature of a single-core microprocessor.

Even though the performance losses due to the DTM schemes are minimal, the unpredictability in their engagement is undesirable for real-time applications. Hence, future work could examine techniques for collaborative thermal management by the operating system and the hardware. Real-time scheduling of processes under the thermal constraints posed by autonomous DTM, while still offering performance guarantees, is a valuable future direction to explore.

4. It has built a temperature-aware floorplanner that can be used at the microarchitectural level. Using this tool, it has made a case that static thermal management can complement DTM [91]. In this study, all the thermal emergencies were removed by just floorplanning alone. A major part of this reduction comes from lateral spreading while a minor portion also comes from reduced leakage and slowed down execution. In comparison with a simple performance metric such as the sum of the lengths of all wires, a profile-driven metric that takes into account the amount of communication and the relative importance of the wires reduces temperature more effectively without losing much performance. The profile-driven floorplanning scheme performed competitively against DVS while doing much better than a static voltage setting.
5. It has presented several temperature-aware placement schemes for multicore architectures, leveraging the orientation of individual cores and the availability of second-level cache banks as cooling buffers [89]. The most important conclusion from this work is that L2 bank insertion achieves significant thermal benefit—the maximum temperature difference from the ambient improves by about 20% for SPEC2000 benchmarks. This thermal benefit is achieved with negligible performance loss and omitting the benefit due to reduced leakage. Furthermore, a combination of core orientation and L2 bank insertion is able to achieve about three-fourths of the temperature reduction achievable by an ideal floorplanning scheme that mingles functional blocks from multiple cores and disperses them amidst a sea of L2 cache banks.



As an extension to the floorplanning efforts in this dissertation, the exploration of the placement choices available in heterogeneous and SIMD architectures is an interesting possibility. The availability of different types of cores augments the placement opportunities of a heterogeneous multicore architecture. Similar to the use of L2 cache banks as cooling buffers, cores with low power density can also be used as cooling buffers. Since SIMD architectures are made of separate small SIMD “lanes”, the spatial thermal filtering might reduce their peak temperature if the lanes could be placed sufficiently apart from one another using clever floorplanning. These are all potential future directions from this dissertation.

6. It has examined the question of the proper granularity of thermal management with the help of an analytical framework. This has led to several interesting insights regarding manycore thermal management, cache line thermal efficiency and sensor accuracy and interpolation [88].

The key conclusions are:

- Many small cores are better in thermal efficiency than a few large cores even when the power density in them remains the same.
- Localized thermal management ceases to be effective beyond low tens of cores.
- On the question of whether pathological code behaviour can cause the catastrophic heating of high aspect ratio sub-blocks such as cache lines, while the answer depends on the cache layout and dimensions, aspect ratio always plays a crucial role in mitigating high power density.
- Interpolation contributes to the reduction of sensor errors, but the magnitude of the benefit is dependent upon the package. Moreover, the bilinear algorithm is almost as accurate as bicubic spline and more efficient to compute.

Consistent with the trends towards multicore and manycore processors, these results show that for the long term future architectures, fine-grained throttling of activity alone is insufficient to address the thermal challenge. Hence, spatial phenomena that exploit lateral heat conduction play a crucial role in the successful continuation of technology scaling.

# Appendix A

## Solution of the Heat Equation

---

We re-state the boundary value problem from Section 6.3 (equations (6.1)-(6.2)) here.

$$\frac{\partial^2 T_1}{\partial x^2} + \frac{\partial^2 T_1}{\partial y^2} = 0 \quad (0 \leq x \leq a, 0 \leq y \leq l) \quad (\text{A.1a})$$

$$\frac{\partial^2 T_2}{\partial x^2} + \frac{\partial^2 T_2}{\partial y^2} = 0 \quad (a \leq x \leq \infty, 0 \leq y \leq l) \quad (\text{A.1b})$$

These equations are to be solved subject to the boundary conditions:

$$T_1(x, l) = 0 \quad T_2(x, l) = 0 \quad (\text{A.2a})$$

$$T_2(\infty, y) = 0 \quad (\text{A.2b})$$

$$\frac{\partial T_1}{\partial x} \Big|_{x=0} = 0 \quad \frac{\partial T_2}{\partial y} \Big|_{y=0} = 0 \quad (\text{A.2c})$$

$$\frac{\partial T_1}{\partial y} \Big|_{y=0} = -\frac{q}{k} \quad (\text{A.2d})$$

$$T_1(a, y) = T_2(a, y) \quad (\text{A.2e})$$

$$\frac{\partial T_1}{\partial x} \Big|_{x=a} = \frac{\partial T_2}{\partial x} \Big|_{x=a} \quad (\text{A.2f})$$

Of the two parts of the problem,  $T_2$  is the easier one to solve since it is a homogeneous problem (*i.e.*, its boundary conditions are all zero). The standard way to solve for such homogeneous

problems is to seek solutions of a *separable* form *i.e.*, assuming  $T(x, y) = X(x)Y(y)$ . In that case, the steady state heat equation  $\nabla^2 T = 0$  reduces to the form  $\frac{X''}{X} + \frac{Y''}{Y} = 0$ . In this equation, the term on the left side of the + symbol is a function of  $x$  alone and the term on the right is a function of  $y$  alone. If their sum has to be zero for all  $x$  and  $y$ , then each term has to separately be equal to a constant (*i.e.*, it cannot be a function of either  $x$  or  $y$ ). Since temperature is a real-valued function, these constants have to be real as well. So, one of them has to be positive and the other negative, so that their sum would be zero. Let us call the positive constant  $\lambda^2$  (the other would be  $-\lambda^2$ ). Then, depending on our boundary conditions, we typically have two types of equations to solve. The first case is of the form  $X'' = \lambda^2 X$  and the second is of the form  $X'' = -\lambda^2 X$ . The solution to the former is of the form  $C_1 e^{\lambda x} + C_2 e^{-\lambda x}$  where  $C_1$  and  $C_2$  are arbitrary constants. This can be seen from the fact that differentiating this expression twice with respect to  $x$  results in it being multiplied by  $\lambda^2$ , satisfying  $X'' = \lambda^2 X$ . Similarly, the solution to the latter case ( $X'' = -\lambda^2 X$ ) is of the form  $C_1 \cos(\lambda x) + C_2 \sin(\lambda x)$ . This can also be verified to satisfy  $X'' = -\lambda^2 X$  as before. Furthermore, since the above-mentioned constants ( $\lambda^2$  and  $-\lambda^2$ ) occur in positive-negative pairs, it should be noted that when  $X$  takes the former form (comprised of exponentials),  $Y$  takes the latter form (comprised of sines/cosines) and *vice versa*.

With this introduction to the method of *separation of variables*, let us now apply it to solve for  $T_2$ . Let  $T_2(x, y) = X_2(x)Y_2(y)$ . From the discussion above,  $X_2(x)$  can be of the form  $C_1 e^{\lambda x} + C_2 e^{-\lambda x}$  or  $C_1 \cos(\lambda x) + C_2 \sin(\lambda x)$ . Correspondingly,  $Y_2(y)$  can be of the form  $D_1 \cos(\lambda y) + D_2 \sin(\lambda y)$  or  $D_1 e^{\lambda y} + D_2 e^{-\lambda y}$  respectively. However, we know from boundary conditions (A.2a) and (A.2b) that  $T_2$  has to be zero at  $x = \infty$  and at  $y = l$ . This restricts  $X_2$  to the exponential form and  $Y_2$  to the sine/cosine form. Thus

$$\begin{aligned} X_2 &= C_1 e^{\lambda x} + C_2 e^{-\lambda x} \quad \text{and} \\ Y_2 &= D_1 \cos(\lambda y) + D_2 \sin(\lambda y) \end{aligned}$$

Now, applying (A.2b) to  $X_2$ , we get  $C_1 = 0$ . Also, applying (A.2c) to  $Y_2$  we get  $D_2 = 0$ . Therefore,  $T_2$  is of the form  $C_2 e^{-\lambda x} D_1 \cos(\lambda y)$ . However, when we apply (A.2a) to  $T_2$ , we get  $Y_2(l) = 0 \Rightarrow$

$\cos(\lambda l) = 0$ . Since there are an infinite number of  $\lambda$ 's that satisfy this equation, we can denote them by  $\lambda_n$  where  $n = 0, 1, 2, \dots$ . So,  $\lambda_n l = (2n + 1)\frac{\pi}{2}$ . For each  $\lambda_n$ , we get a different solution. Since the problem we are solving is a homogeneous one, the principle of superposition holds. That is, if there are two solutions satisfying the boundary conditions, their sum is also a valid solution. Hence, all solutions corresponding to the different  $\lambda_n$ 's can be added up to obtain  $T_2$ . Therefore,

$$T_2 = \sum_{n=0}^{\infty} B_n e^{-\lambda_n x} \cos(\lambda_n y) \quad (\text{A.3})$$

where  $B_n$ 's are arbitrary constants. Now, before unraveling  $T_2$  further, let us take a look at  $T_1$ . It can be seen that boundary condition (A.2d) is not homogeneous. In order to be able to apply the method of separation of variables as we did for  $T_2$ , we split  $T_1$  into two parts:  $T_1'$  and  $T_p$  where  $T_1'$  is the homogeneous part.  $T_p$  is still not homogeneous but is much simpler to construct a *particular solution* for. So,  $T_1 = T_1' + T_p$ . The original problem  $\nabla^2 T_1 = 0$  gets split into two sub-problems:

$$\nabla^2 T_1' = 0, \quad \nabla^2 T_p = 0 \quad (\text{A.4})$$

subject to the following boundary conditions:

$$T_1'(x, l) = 0 \quad T_p(x, l) = 0 \quad (\text{A.5a})$$

$$\frac{\partial T_1'}{\partial x} \Big|_{x=0} = 0 \quad \frac{\partial T_p}{\partial x} \Big|_{x=0} = 0 \quad (\text{A.5b})$$

$$\frac{\partial T_1'}{\partial y} \Big|_{y=0} = 0 \quad \frac{\partial T_p}{\partial y} \Big|_{y=0} = -\frac{q}{k} \quad (\text{A.5c})$$

Of the two problems  $T_1'$  and  $T_p$ , the latter is the simpler one in spite of the non-homogeneity because, unlike  $T_1'$  (and the previous case  $T_2$ ), we are not looking for *all* general solutions. Instead, we are looking for a *particular* solution that suits our problem and boundary conditions. Roughly speaking, we are looking for a function that vanishes when differentiated twice (A.4) and becomes a constant when differentiated once (A.5c). Thus, it can be seen that a linear function of  $y$  solves the

sub-problem (A.4) by construction, since its second derivative is zero. Hence,  $T_p = Py + Q$  where  $P$  and  $Q$  are arbitrary constants. Applying the boundary conditions (A.5b) and (A.5c), we get

$$T_p = \frac{q}{k}(l - y) \quad (\text{A.6})$$

Now, since  $T'_1$  is a homogeneous problem, it can be solved just as we did  $T_2$  by separating the variables, *i.e.*, with  $T'_1 = X_1(x)Y_1(y)$ . Then, by the same arguments as before, one of  $X_1$  and  $Y_1$  must comprise of exponentials and the other of sines/cosines. Applying the boundary conditions (A.5b) and (A.5c) to the possible cases narrows down the solution choices to two forms *viz.*  $A \cosh(\lambda x) \cos(\lambda y)$  or  $A \cos(\lambda x) \cosh(\lambda y)$  where,  $\cosh(\theta) = \frac{e^\theta + e^{-\theta}}{2}$  and  $A$  is an arbitrary constant. Now, for the boundary condition (A.5a) to hold, the latter choice is not possible because  $\cosh(\lambda l)$  can never be zero. Hence,  $T'_1$  can only be of the form  $A \cosh(\lambda x) \cos(\lambda y)$ . Applying (A.5a) to this  $\Rightarrow \cos(\lambda l) = 0$ , giving rise to infinite  $\lambda_n$ 's as before. So, superposing all such solutions, we get

$$T'_1 = \sum_{n=0}^{\infty} A_n \cosh(\lambda_n x) \cos(\lambda_n y) \quad (\text{A.7})$$

(or)

$$\begin{aligned} T_1 &= T'_1 + T_p \\ &= \frac{q}{k}(l - y) + \sum_{n=0}^{\infty} A_n \cosh(\lambda_n x) \cos(\lambda_n y) \end{aligned} \quad (\text{A.8})$$

$$\text{where, } \lambda_n l = (2n + 1) \frac{\pi}{2} \quad (n = 0, 1, 2, \dots)$$

From equations (A.8) and (A.3), we almost have the complete solutions for  $T_1$  and  $T_2$  except that we need to determine  $A_n$  and  $B_n$  for  $n = 0, 1, 2, \dots$ . This can be done by including the continuity conditions from (A.2e) and (A.2f). Substituting for  $T_1$  and  $T_2$  from equations (A.8) and (A.3) in the boundary condition (A.2f) and grouping like terms together, we get

$$B_n = -A_n e^{\lambda_n a} \sinh(\lambda_n a) \quad (\text{A.9})$$

where,  $\sinh(\theta) = \frac{e^\theta - e^{-\theta}}{2}$ . Now, substituting for  $B_n$  from (A.9) into (A.3), applying the boundary

condition (A.2e) and grouping like terms together, we get

$$\frac{q}{k}(l-y) = \sum_{n=0}^{\infty} -A_n e^{\lambda_n a} \cos(\lambda_n y) \quad (\text{A.10})$$

The right side of this equation is an infinite series made of cosines. Hence, it bears a strong resemblance to a Fourier cosine series expansion. Hence, let us expand the function  $\frac{q}{k}(l-y)$  using Fourier cosine series and compare with the equation above. In order to be able to expand a function using Fourier series, it has to be periodic. So, let us define a periodic function  $f(y)$  which is a periodic extension of  $\frac{q}{k}(l-y)$  with a period of  $4l$  *i.e.*,

$$f(y) = \begin{cases} -\frac{q}{k}(l-y) & -2l \leq y < 0 \\ \frac{q}{k}(l-y) & 0 \leq y < 2l \end{cases}$$

and  $f(y+4l) = f(y)$ . It is to be noted that for our problem, we are only interested in the interval  $0 \leq y \leq l$ . Now, we are seeking a Fourier cosine series expansion for  $f(y)$  *i.e.*, we are looking for constants  $a_m$  ( $m = 0, 1, 2, \dots$ ) such that  $f(y) = a_0 + \sum_{m=1}^{\infty} a_m \cos\left(\frac{m\pi y}{2l}\right)$ . From standard texts on Engineering Mathematics (for *e.g.*, [63]), these can be computed as below:

$$\begin{aligned} a_0 &= \frac{1}{2l} \int_0^{2l} \frac{q}{k}(l-y) dy = 0 \\ a_m &= \frac{1}{l} \int_0^{2l} \frac{q}{k}(l-y) \cos\left(\frac{m\pi y}{2l}\right) dy \\ &= \frac{1}{l} \left[ \frac{q}{k}(l-y) \sin\left(\frac{m\pi y}{2l}\right) \left(\frac{2l}{m\pi}\right) \right]_0^{2l} \\ &\quad - \frac{1}{l} \int_0^{2l} \sin\left(\frac{m\pi y}{2l}\right) \left(\frac{2l}{m\pi}\right) \left(-\frac{q}{k}\right) dy \quad (\text{integration by parts}) \\ &= \frac{ql}{k} \left[ \frac{1 - \cos(m\pi)}{\left(\frac{m\pi}{2}\right)^2} \right] \quad \text{for } m = (1, 2, 3, \dots) \end{aligned}$$

In other words,  $f(y) = \frac{q}{k}(l-y)$  can be written as

$$\frac{q}{k}(l-y) = \sum_{n=0}^{\infty} 2 \frac{ql}{k} \frac{1}{\lambda_n^2} \cos(\lambda_n y) \quad (\text{A.11})$$

where,  $\lambda_n l = (2n+1) \frac{\pi}{2} \quad (n = 0, 1, 2, \dots)$

Comparing this equation with (A.10), we can determine  $A_n$  as

$$A_n = -2 \frac{ql}{k} \frac{e^{-\lambda_n a}}{\lambda_n^2} \quad (\text{A.12})$$

Now, if we substitute  $A_n$  and  $B_n$  from (A.12) and (A.9) into (A.8) and (A.3) respectively, we have the full solution to our problem. Before we present the full solution, for notational convenience, let us define  $\gamma_n = \lambda_n l$  i.e.,  $\gamma_n = (2n+1) \frac{\pi}{2}$ . Then, the solution for the heat equation is given by:

$$T_1(x, y) = \frac{ql}{k} \left[ 1 - \frac{y}{l} - 2 \sum_{n=0}^{\infty} \frac{e^{-(\gamma_n \frac{a}{l})}}{\gamma_n^2} \cosh(\gamma_n \frac{x}{l}) \cos(\gamma_n \frac{y}{l}) \right] \quad (\text{A.13})$$

$$T_2(x, y) = \frac{ql}{k} \left[ 2 \sum_{n=0}^{\infty} \frac{\sinh(\gamma_n \frac{a}{l})}{\gamma_n^2} e^{-(\gamma_n \frac{x}{l})} \cos(\gamma_n \frac{y}{l}) \right] \quad (\text{A.14})$$

where,  $\gamma_n = (2n+1) \frac{\pi}{2} \quad (n = 0, 1, 2, \dots)$

## Bibliography

---

- [1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, 1964.
- [2] S. N. Adya and I. L. Markov. Fixed-outline floorplanning : Enabling hierarchical design. *IEEE Transactions on VLSI*, 11(6):1120–1135, December 2003.
- [3] V. Agarwal, S. W. Keckler, and D. Burger. The effect of technology scaling on microarchitectural structures. Technical Report TR-00-02, University of Texas at Austin Computer Sciences, May 2001.
- [4] Ansys 11.0. <http://www.ansys.com>.
- [5] T. Austin, E. Larson, and D. Ernst. SimpleScalar: An infrastructure for computer system modeling. *IEEE Computer*, 35(4):59–67, Feb. 2002.
- [6] K. Azar. Thermal design basics and calculation of air cooling limits, Oct. 2002. Tutorial at the 2002 International Workshop on THERMal Investigations of ICs and Systems (THERMINIC).
- [7] A. Bakker and J. Huijsing. *High-Accuracy CMOS Smart Temperature Sensors*. Kluwer Academic, Boston, 2000.
- [8] K. Banerjee and A. Mehrotra. Global (interconnect) warming. *IEEE Circuits and Devices Magazine*, 17(5):16–32, September 2001.
- [9] P. Bannon. Personal communication, Sep. 2002.



- [10] W. Batty et al. Global coupled EM-electrical-thermal simulation and experimental validation for a spatial power combining MMIC array. *IEEE Transactions on Microwave Theory and Techniques*, pages 2820–33, Dec. 2002.
- [11] E. Borch, E. Tune, S. Manne, and J. Emer. Loose loops sink chips. In *High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on*, pages 299–310, 2002.
- [12] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the Seventh IEEE International Symposium on High-Performance Computer Architecture*, pages 171–82, Jan. 2001.
- [13] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th Annual ACM/IEEE International Symposium on Computer Architecture*, pages 83–94, June 2000.
- [14] D. C. Burger and T. M. Austin. The SimpleScalar tool set, version 2.0. *Computer Architecture News*, 25(3):13–25, June 1997.
- [15] R. Canal, J-M. Parcerisa, and A. González. A cost-effective clustered architecture. In *Proceedings of the 1999 IEEE/ACM/IFIP International Conference on Parallel Architectures and Compilation Techniques*, pages 160–68, Oct. 1999.
- [16] L.P. Cao, J.P. Krusius, M.A. Korhonen, and T.S. Fisher. Transient thermal management of portable electronics using heat storage and dynamic power dissipation control. *IEEE Transactions on Components, Packaging, and Manufacturing Technology—Part A*, 21(1):113–23, Mar. 1998.
- [17] H. S. Carslaw and J. C. Jaeger. *Conduction of Heat in Solids: Second Edition*. Oxford University Press, Oxford, UK, 1986, c1959.

- [18] A. Chakravorty, A. Ranjan, and R. Balasubramonian. Re-visiting the performance impact of microarchitectural floorplanning. In *Third Workshop on Temperature-Aware Computer Systems(TACS-3), held in conjunction with ISCA-33*, June 2006.
- [19] P. Chaparro, J. González, and A. González. Thermal-aware clustered microarchitectures. In *Proceedings of the 2004 IEEE International Conference on Computer Design*, Oct. 2004.
- [20] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, and K. Skadron. A performance study of general-purpose applications on graphics processors using cuda. *Journal of Parallel and Distributed Computing*, 68(10):1370–1380, 2008.
- [21] G. Chen and S. Sapatnekar. Partition-driven standard cell thermal placement. In *ISPD '03: Proceedings of the 2003 international symposium on Physical design*, pages 75–80, 2003.
- [22] Y.-K. Cheng and S.-M. Kang. A temperature-aware simulation environment for reliable ULSI chip design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(10):1211–20, Oct. 2000.
- [23] J. Choi, C-Y. Cher, H. Franke, H. Hamann, A. Weger, and P. Bose. Thermal-aware task scheduling at the system software level. In *ISLPED '07: Proceedings of the 2007 international symposium on Low power electronics and design*, pages 213–218, 2007.
- [24] J. Choi, Y. Kim, A. Sivasubramaniam, J. Srebric, Q. Wang, and J. Lee. Modeling and managing thermal profiles of rack-mounted servers with thermostat. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2007.
- [25] C. N. Chu and D. F. Wong. A matrix synthesis approach to thermal placement. In *ISPD '97: Proceedings of the 1997 international symposium on Physical design*, pages 163–168, 1997.
- [26] Compaq 21364 die photo. From website: CPU Info Center. [http://bwrc.eecs.berkeley.edu/CIC/die\\_photos](http://bwrc.eecs.berkeley.edu/CIC/die_photos).
- [27] J. Cong, A. Jagannathan, G. Reinman, , and M. Romesis. Microarchitecture evaluation with physical planning. In *Proceedings of the 40th Design Automation Conference*, June 2003.

- [28] P. Dadvar and K. Skadron. Potential thermal security risks. In *Proceedings of the IEEE/ASME Semiconductor Thermal Measurement, Modeling, and Management Symposium (SEMI-THERM)*, Mar. 2005.
- [29] R.H. Dennard, F.H. Gaensslen, H.N. Yu, V.L. Rideout, E. Bassous, and A.R. LeBlanc. Design of ion implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, October 1974.
- [30] A. Dhodapkar, C. H. Lim, G. Cai, and W. R. Daasch. TEMPEST: A thermal enabled multi-model power/performance estimator. In *Proceedings of the Workshop on Power-Aware Computer Systems*, Nov. 2000.
- [31] J. Donald and M. Martonosi. Temperature-aware design issues for smt and cmp architectures. In *Proceedings of the 2004 Workshop on Complexity-Effective Design*, June 2004.
- [32] M. Ekpanyapong, M. B. Healy, C. S. Ballapuram, S. K. Lim, H. S. Lee, and G. H. Loh. Thermal-aware 3d microarchitectural floorplanning. Technical Report GIT-CERCS-04-37, Georgia Institute of Technology Center for Experimental Research in Computer Systems, 2004.
- [33] M. Ekpanyapong, J. R. Minz, T. Watwai, H. S. Lee, and S. K. Lim. Profile-guided microarchitectural floorplanning for deep submicron processor design. In *Proceedings of the 41th Design Automation Conference*, pages 634–639, 2004.
- [34] WC Elmore. The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of applied physics*, 19(1):55, 1948.
- [35] K. Etesam-Yazdani, H.F. Hamann, and M. Asheghi. Impact of power granularity on chip thermal modeling. In *ITHERM '06: The Tenth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronics Systems*, pages 666–670, May/June 2006.
- [36] M. Fleischmann. Crusoe power management: Cutting x86 operating power through LongRun. In *Embedded Processor Forum*, June 2000.

- [37] Floworks: Fluid Flow Analysis for SolidWorks. NIKA GmbH. Website. <http://www.floworks.com>.
- [38] J. Garrett and M. R. Stan. Active threshold compensation circuit for improved performance in cooled CMOS systems. In *Proceedings of the International Symposium on Circuits and Systems*, pages 410–413, May 2001.
- [39] S. H. Gerez. *Algorithms for VLSI Design Automation*. John Wiley & Sons, Inc., 1999.
- [40] S. Gunther, F. Binns, D. M. Carmean, and J. C. Hall. Managing the impact of increasing microprocessor power consumption. In *Intel Technology Journal*, Q1 2001.
- [41] A. Gupta, N. Dutt, F. Kurdahi, K. Khouri, and M. Abadir. Stefal: A system level temperature- and floorplan-aware leakage power estimator for socs. In *VLSID '07: Proceedings of the 20th International Conference on VLSI Design*, pages 559–564, 2007.
- [42] Y. Han, I. Koren, and C.M. Krishna. TILTS: A Fast Architectural-Level Transient Thermal Simulation Method. *Journal of Low Power Electronics*, 3(1):13, 2007.
- [43] Y. Han, I. Koren, and C. A. Moritz. Temperature aware floorplanning. In *Second Workshop on Temperature-Aware Computer Systems(TACS-2), held in conjunction with ISCA-32*, June 2005.
- [44] J. W. Haskins, Jr. and K. Skadron. Memory reference reuse latency: Accelerated sampled microarchitecture simulation. In *Proceedings of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 195–203, Mar. 2003.
- [45] M. B. Healy, H-H. S. Lee, G. H. Loh, and S. K. Lim. Thermal optimization in multi-granularity multi-core floorplanning. In *ASP-DAC '09: Proceedings of the 2009 Conference on Asia and South Pacific Design Automation*, pages 43–48, 2009.
- [46] T. Heath, A. P. Centeno, P. George, L. Ramos, and Y. Jaluria. Mercury and freon: temperature emulation and management for server systems. In *Proceedings of the Twelfth International*

- Conference on Architectural Support for Programming Languages and Operating Systems*, pages 106–116, 2006.
- [47] S. Heo, K. Barr, and K. Asanovic. Reducing power density through activity migration. In *Proceedings of the 2003 ACM/IEEE International Symposium on Low Power Electronics and Design*, Aug. 2003.
- [48] K. E. Holbert. Interdisciplinary electrical analogies. <http://www.fulton.asu.edu/~holbert/analogy.html>.
- [49] W. Huang, J. Renau, S.-M. Yoo, and J. Torellas. A framework for dynamic energy efficiency and temperature management. In *Proceedings of the 33rd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 202–13, Dec. 2000.
- [50] W. Huang, K. Sankaranarayanan, K. Skadron, R. J. Ribando, and M. R. Stan. Accurate, pre-rtl temperature-aware design using a parameterized, geometric thermal model. *IEEE Transactions on Computers*, 57(9):1277–1288, September 2008.
- [51] W. Huang, M. R. Stan, K. Sankaranarayanan, R. J. Ribando, and K. Skadron. Many-core design from a thermal perspective. In *DAC '08: Proceedings of the 45th annual conference on Design automation*, pages 746–749, June 2008.
- [52] W. Huang, M. R. Stan, K. Skadron, S. Ghosh, K. Sankaranarayanan, and S. Velusamy. Compact thermal modeling for temperature-aware design. In *Proceedings of the ACM/IEEE 41st Design Automation Conference*, pages 878–883, June 2004.
- [53] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, and S. Ghosh. Hotspot: A compact thermal modeling method for CMOS VLSI systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(5):501–513, May 2006.
- [54] W. Hung, Y. Xie, N. Vijaykrishnan, C. Addo-Quaye, T.Theocharides, and M. J. Irwin. Thermal-aware floorplanning using genetic algorithms. In *Sixth International Symposium on Quality of Electronic Design (ISQED'05)*, March 2005.

- [55] W.-L. Hung, G. M. Link, Y. Xie, N. Vijaykrishnan, and M. J. Irwin. Interconnect and thermal-aware floorplanning for 3d microprocessors. In *ISQED '06: Proceedings of the 7th International Symposium on Quality Electronic Design*, pages 98–104, 2006.
- [56] F.P. Incropera and D.P. DeWitt. Heat and mass transfer. *John Wiley & Sons, Inc*, 2001.
- [57] Intel Corp. *Intel Pentium 4 Processor In the 423-pin Package: Thermal Design Guidelines*, Nov. 2000. Order no. 249203-001.
- [58] A. Iyer and D. Marculescu. Power and performance evaluation of globally asynchronous locally synchronous processors. In *Proceedings of the 29th Annual ACM/IEEE International Symposium on Computer Architecture*, pages 158–68, May 2002.
- [59] C. Kim, D. Burger, and S.W. Keckler. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. In *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 211–222, 2002.
- [60] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
- [61] P. Ko, J. Huang, Z. Liu, and C. Hu. BSIM3 for analog and digital circuit simulation. In *Proceedings of the IEEE Symposium on VLSI Technology CAD*, pages 400–429, 1993.
- [62] V. Koval and I. W. Farmaga. MONSTR: A complete thermal simulator of electronic systems. In *Proceedings of the ACM/IEEE 31st Design Automation Conference*, June 1994.
- [63] E. Kreyszig. *Advanced Engineering Mathematics: 8th Edition*. John Wiley & Sons, Inc., New York, NY, USA, 1998.
- [64] A. Krum. Thermal management. In F. Kreith, editor, *The CRC handbook of thermal engineering*, pages 2.1–2.92. CRC Press, Boca Raton, FL, 2000.

- [65] J. C. Ku, S. Ozdemir, G. Memik, and Y. Ismail. Thermal management of on-chip caches through power density minimization. In *MICRO 38: Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture*, pages 283–293, November 2005.
- [66] K.-J. Lee and K. Skadron. Analytical model for sensor placement on microprocessors. In *Proceedings of the 2005 IEEE International Conference on Computer Design*, pages 24–27, Oct. 2005.
- [67] P. Li, LT Pileggi, M. Asheghi, and R. Chandra. Efficient full-chip thermal modeling and analysis. In *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004*, pages 319–326, 2004.
- [68] C.-H. Lim, W. Daasch, and G. Cai. A thermal-aware superscalar microprocessor. In *Proceedings of the International Symposium on Quality Electronic Design*, pages 517–22, Mar. 2002.
- [69] P. Liu, H. Li, L. Jin, W. Wu, SX Tan, and J. Yang. Fast Thermal Simulation for Runtime Temperature Tracking and Management. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 25(12):2882, 2006.
- [70] S. Manne, A. Klauser, and D. Grunwald. Pipeline gating: speculation control for energy reduction. In *Proceedings of the 25th Annual ACM/IEEE International Symposium on Computer Architecture*, pages 132–41, June 1998.
- [71] S. O. Memik, R. Mukherjee, M. Ni, and J. Long. Optimizing thermal sensor allocation for microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(3):516–527, March 2008.
- [72] P. Michaud, Y. Sazeides, A. Sez nec, T. Constantinou, and D. Fetis. An analytical model of temperature in microprocessors. Technical Report PI-1760/RR-5744, IRISA/INRIA, November 2005.

- [73] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, pages 114–17, Apr. 1965.
- [74] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. Vlsi module placement based on rectangle-packing by the sequence-pair. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 15(12):1518–1524, 1996.
- [75] G. E. Myers. *Analytical Methods in Conduction Heat Transfer*. Genium Pub. Corp., Schenectady, NY, USA, 1987.
- [76] V. Nookala, D. J. Lilja, and S. S. Sapatnekar. Temperature-aware floorplanning of microarchitecture blocks with ipc-power dependence modeling and transient analysis. In *ISLPED '06: Proceedings of the 2006 international symposium on Low power electronics and design*, pages 298–303, 2006.
- [77] R. H. J. M. Otten. Efficient floorplan optimization. In *Proceedings of the International Conference of Computer Design*, pages 499–502, 1983.
- [78] R. H. J. M. Otten and R. K. Brayton. Planning for performance. In *DAC '98: Proceedings of the 35th annual conference on Design automation*, pages 122–127, 1998.
- [79] M. Pedram and S. Nazarian. Thermal modeling, analysis, and management in vlsi circuits: principles and methods. *Proceedings of the IEEE*, 94(8):1487–1501, 2006.
- [80] M. D. Powell, M. Gomaa, and T. N. Vijaykumar. Heat-and-run: Leveraging SMT and CMP to manage power density through the operating system. In *Proceedings of the Eleventh International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 2004.
- [81] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1992.



- [82] Jan M. Rabaey. *Digital Integrated Circuits: A Design Perspective*. Prentice-Hall, Englewood Cliffs, NJ, 1995.
- [83] M. Rencz, V. Székely, A. Poppe, and B. Courtois. Friendly tools for the thermal simulation of power packages. In *Proceedings of the International Workshop On Integrated Power Packaging*, pages 51–54, July 2000.
- [84] J. Robertson. Intel hints of next-generation security technology for mpus. *EE Times*, Sept. 10 2002.
- [85] E. Rohou and M. Smith. Dynamically managing processor temperature and power. In *Proceedings of the 2nd Workshop on Feedback-Directed Optimization*, Nov. 1999.
- [86] M.-N. Sabry. Dynamic compact thermal models: An overview of current and potential advances. In *Proceedings of the 8th Int'l Workshop on THERMal INvestigations of ICs and Systems*, Oct. 2002. Invited paper.
- [87] H. Sanchez et al. Thermal management system for high-performance PowerPC microprocessors. In *COMPCON*, page 325, 1997.
- [88] K. Sankaranarayanan, W. Huang, M. R. Stan, H. Haj-Hariri, R. J. Ribando, and K. Skadron. Granularity of microprocessor thermal management: a technical report. Technical Report CS-2009-03, University of Virginia Department of Computer Science, April 2009.
- [89] K. Sankaranarayanan, M. R. Stan, and K. Skadron. A discussion on the thermal benefit of multicore floorplanning at the microarchitectural level. Technical Report CS-2009-04, University of Virginia Department of Computer Science, April 2009.
- [90] K. Sankaranarayanan, S. Velusamy, and K. Skadron. Microarchitectural floorplanning for thermal management: A technical report. Technical Report CS-2005-08, University of Virginia Department of Computer Science, May 2005.

- [91] K. Sankaranarayanan, S. Velusamy, M. R. Stan, and K. Skadron. A case for thermal-aware floorplanning at the microarchitectural level. *Journal of Instruction-Level Parallelism*, 7, 2005. (<http://www.jilp.org/vol7>).
- [92] M. Sarrafzadeh and C. K. Wong. *An Introduction to VLSI Physical Design*. McGraw-Hill Higher Education, 1996.
- [93] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott. Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In *Proceedings of the Eighth IEEE International Symposium on High-Performance Computer Architecture*, pages 29–40, Feb. 2002.
- [94] S. Sharifi, C. Liu, and T. S. Rosing. Accurate temperature estimation for efficient thermal management. In *9th International Symposium on Quality Electronic Design, 2008. ISQED 2008*, pages 137–142, March 2008.
- [95] T. Sherwood, E. Perelman, and B. Calder. Basic block distribution analysis to find periodic behavior and simulation points in applications. In *Proceedings of the 2001 IEEE/ACM/IFIP International Conference on Parallel Architectures and Compilation Techniques*, pages 3–14, Sept. 2001.
- [96] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *Proceedings of the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 45–57, October 2002.
- [97] P. Shivakumar and N. P. Jouppi. Cacti 3.0: An integrated cache timing, power and area model. Technical Report 2001/2, Compaq Western Research Laboratory, Aug. 2001.
- [98] SIA. *International Technology Roadmap for Semiconductors*, 2001.
- [99] SIA. *International Technology Roadmap for Semiconductors*, 2006. <http://www.itrs.net/links/2006Update/2006UpdateFinal.htm>.

- [100] K. Skadron, T. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management. In *Proceedings of the Eighth IEEE International Symposium on High-Performance Computer Architecture*, pages 17–28, Feb. 2002.
- [101] K. Skadron, K. Sankaranarayanan, S. Velusamy, D. Tarjan, M. R. Stan, and W. Huang. Temperature-aware microarchitecture: Modeling and implementation. *ACM Transactions on Architecture and Code Optimization*, 1(1):94–125, Mar. 2004.
- [102] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware computer systems: Opportunities and challenges. *IEEE Micro*, 23(6):52–61, Nov-Dec. 2003.
- [103] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Proceedings of the 30th Annual ACM/IEEE International Symposium on Computer Architecture*, pages 2–13, Apr. 2003.
- [104] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture: Extended discussion and results. Technical Report CS-2003-08, University of Virginia Department of Computer Science, Apr. 2003.
- [105] sp3 diamond technologies, diatherm heat spreaders. [http://www.sp3diamondtech.com/prod\\_diatherm.asp](http://www.sp3diamondtech.com/prod_diatherm.asp).
- [106] Standard Performance Evaluation Corporation. SPEC CPU2000 Benchmarks. <http://www.specbench.org/osg/cpu2000>.
- [107] L. Stockmeyer. Optimal orientations of cells in slicing floorplan designs. *Information and Control*, 57(2-3):91–101, 1983.
- [108] V. Szekely, C. Marta, M. Rencz, G. Vegh, Z. Benedek, and S. Torok. A thermal benchmark chip: design and applications. *IEEE Transactions on Components, Packaging, and Manufacturing Technology*, 21(3):399–405, September 1998.

- [109] V. Székely, A. Poppe, A. Páhi, A. Csendes, and G. Hajas. Electro-thermal and logi-thermal simulation of VLSI designs. *IEEE Transactions on VLSI Systems*, 5(3):258–69, Sept. 1997.
- [110] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi. Cacti 5.1. Technical Report HPL-2008-20, HP Laboratories, Palo Alto, April 2008.
- [111] K. Torki and F. Ciontu. IC thermal map from digital and thermal simulations. In *Proceedings of the 2002 International Workshop on THERMal Investigations of ICs and Systems (THERMINIC)*, pages 303–08, Oct. 2002.
- [112] S. Velusamy, W. Huang, J. Lach, M. R. Stan, and K. Skadron. Monitoring temperature in FPGA based SoCs. In *Proceedings of the 2005 IEEE International Conference on Computer Design*, Oct. 2005.
- [113] S. Velusamy, K. Sankaranarayanan, D. Parikh, T. Abdelzaher, and K. Skadron. Adaptive cache decay using formal feedback control. In *Proceedings of the 2002 Workshop on Memory Performance Issues*, May 2002.
- [114] R. Viswanath, W. Vijay, A. Watwe, and V. Lebonheur. Thermal performance challenges from silicon to systems. *Intel Technology Journal*, Q3 2000.
- [115] D. F. Wong and D. L. Liu. A new algorithm for floorplan design. In *Proceedings of the ACM/IEEE 23rd Design Automation Conference*, pages 101–107, June 1986.
- [116] Y-W. Wu, C-L. Yang, P-H. Yuh, and Y-W. Chang. Joint exploration of architectural and physical design spaces with thermal consideration. In *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*, pages 123–126, August 2005.
- [117] Y. Yang, Z. Gu, C. Zhu, R. P. Dick, and L. Shang. ISAC: Integrated Space-and-Time-Adaptive Chip-Package Thermal Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(1):86–99, 2007.

- [118] Y. Zhan and SS Sapatnekar. A high efficiency full-chip thermal simulation algorithm. In *IEEE/ACM International Conference on Computer-Aided Design, 2005. ICCAD-2005*, pages 635–638, 2005.
  
- [119] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan. Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects. Technical Report CS-2003-05, University of Virginia Department of Computer Science, Mar. 2003.