The Pennsylvania State University
The Graduate School

TEMPERATURE-AWARE COMPUTING

A Thesis in
Computer Science and Engineering
by
Gregory M. Link

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2006

The thesis of Gregory M. Link was reviewed and approved* by the following:

Vijaykrishnan Narayanan
Associate Professor of Computer Science and Engineering
Thesis Advisor, Chair of Committee, Graduate Program Officer

Mary Jane Irwin
Professor of Computer Science and Engineering

Chita R. Das
Professor of Computer Science and Engineering

W. Kenneth Jenkins
Department Head, Professor of Electrical Engineering

*Signatures are on file in the Graduate School.

Abstract

In the future, the peak temperature of a chip will be a primary design constraint. Higher temperatures can accelerate various chip failure mechanisms, reducing the lifetime of the system. These high temperatures also place additional burden on cooling systems, which must prevent thermal runaways due to increased standby power consumption. Consequently, temperature must be considered in the earliest phases of the design process. Many existing thermal management techniques focus on reducing the overall power consumption of the chip by throttling performance, eventually resulting in an overall reduction in chip temperature. These techniques, while effective, often do not address location-specific temperature problems referred to as hotspots. Recent research into hotspots has shown that different functional units in general purpose processors can have significantly different temperature profiles, and that moving workloads between units can reduce the creation of hotspots on the die.

Using a newly developed thermal analysis tool, HS3d, this work explores the thermal profile of modern processor architectures, and discusses the types and characteristics of hotspots in future technologies, as process variation, multi-core design, and multi-wafer stacking techniques become prevalent. Means of mitigating these hotspots are presented, including workload migration for homogenous architectures, and means of reducing hotspots near the integer ALU. One proposed method, integer offloading to floating-point, redirects integer operations to the floating-point hardware, slightly increasing latency and power consumption, but distributing heat more evenly across the die. . Finally, a model of the impact of temperature on circuit timing is presented, and the impact of temperature gradients on multi-core processors is explored, showing that by the 45nm technology node, thermally-induced timing variations of 5% per 10° C are possible. Traditional worst-case design techniques, which assume a single high temperature for the entire device, can therefore not take full advantage of the much more common

typical-case conditions. This thesis discusses how thermal-aware design can be incorporated into the automated design flow, allowing variable frequency systems to achieve maximal performance across a wide operating range.

Table of Contents

List of Figures

List of Tables

Acknowledgments

The author gratefully gives thanks to my thesis advisor, Dr. N. Vijaykrishnan, for his attention, limitless patience, and valuable advice.

I extend my gratitude to my other committee members, Dr. Mary Jane Irwin, Dr. Chita R. Das, and Dr. Ken Jenkins for their kind guidance and discussions on my work.

I would like to dedicate this thesis to my future wife, Whitney Ortman. It was her creative thinking that allowed me to find a way to reach my academic calling, and her love and patience that allowed me to endure the hardships along the way.

Finally, I wish to express my deepest gratitude and respect to my parents and family.

Chapter 1

Introduction

1.1 Motivation

The consistent advance of CMOS technology, allowing reduced drawn widths and thinner gate oxides, has resulted in a dramatic increase in performance integration over its lifetime. To keep dynamic power consumption relatively constant over this time, the supply voltage used in designs has been reduced as well. To keep clock rates high in the presence of shrinking voltages, the power supply reduction has been mitigated by reductions in transistor threshold voltage, and currently produced designs often operate with only a 1.1V supply, and a threshold voltage in the range of 0.26V. These low threshold voltages, and thin gate oxides, have resulted in a dramatic increase in 'leakage' currents, and the associated static power consumption. The exponential relationship of static power dissipation to threshold voltages and gate thicknesses prevents the continuation of previous technology scaling trends, and leads to a difficult trade-off: decreasing the power supply and threshold voltages to limit dynamic power dissipation, or maintaining threshold voltages to help limit static power dissipation. There has been significant research into low-power design techniques attempting to keep power dissipation low, but these techniques are often "stop-gaps", allowing technology scaling to continue for a short period, but not eliminating the overall problem.

The lack of a general solution to these fundamental process-related problems has side effects, and the power densities of microprocessors continue to increase with introduction of newer and smaller technologies. The power density of cost-

performance range of microprocessors are projected to increase from around $0.65\frac{W}{mm^2}$ currently to around $0.81\frac{W}{mm^2}$ according the 2004 ITRS update. However, a more significant concern is the rapid increase in localized power densities of a processor which have been shown to be as large as 1000 $\frac{W}{mm^3}$ [1] . The resulting elevated chip temperatures cause various problems such as increased leakage (and the possibility of thermal runaway), accelerated physical failure mechanisms and timing failures. As the power dissipation on chip is non-uniform, the temperature profiles are spatially non-uniform across the chip and vary based on the power dissipation characteristics of the individual components of the chip. This leads to localized hot spots [2], regions of locally high temperature.

While attempts to reduce chip power consumption continue, these approaches are tautologically targeted at power reduction only, and are oblivious to the actual characteristics of temperature problems on chip. An example of such an optimization is the highly effective "drowsy cache" [3], which reduces power consumption in the processor cache. While effective at reducing overall processor power consumption, this has little effect on chip peak temperature, as the hotspots are traditionally located far from the caches, close to high-activity functional units, such as the integer ALU and scheduler. As such, temperature-oblivious research, while valuable for other reasons, does not and cannot address the peak temperature problems found in modern and forward-looking designs. New techniques need to be developed to first characterize, and then address, thermal problems. Such techniques will result in designs that are cheaper and easier to cool, that have a much larger effective power budget, and that can provide consistently high performance.

1.2 Background and Related Work

While the temperature of devices has been studied in the past, such work has been the domain of thermal and packaging engineers, rather than chip architects, who have only recently started focusing on this problem [4, 5, 6]. The efforts of thermal engineers focused on the design of adequate cooling solutions for already fabricated chips, and allowed for some overdesign of the cooling solution to ensure that all regions of the device operated within thermal bounds. As the power consumption (and power density) of devices has increased, however, it has become cost and

space inefficient to overdesign cooling solutions, resulting in generalized thermal responses. There are many ways to implement such generalized thermal responses, including clock gating [7], voltage and frequency scaling [8], and instruction cache gating [9]. These responses, while effective, often incur significant performance penalties as the entire processor is slowed to reduce power consumption. Skadron et al., discussed the use of control-theoretic techniques to reduce the performance penalty of such techniques, allowing a minimal amount of slowdown that still ensures the prevention of a thermal emergency [10]. These techniques, rather than emphasizing chip average temperature, focused on regions of locally high temperature, called *Hotspots*.

These regions of locally high temperature are not directly related to local power consumption, implying that while power dissipation is still the primary source of heat on a chip, blindly ignoring thermal modeling in favor of power modeling is an inadequate approach [11, 12, 13]. For example, by migrating workloads from one functional unit in a microprocessor to another, it is possible to balance the heat accumulated at each unit, and in this manner, reduce the localized heating indicative of hotspots, even though total power consumption remains equal, or perhaps increases slightly [11, 14, 15].

Many different thermal modeling tools have been developed, operating at different levels of abstraction. Chiang, Banerjee, and Saraswat have developed a SPICE-based thermal model that allows extremely high accuracy modeling of device temperatures, without the overhead of full finite-element simulation [16]. Wilkerson, Raman, and Turowski present a Python-based framework that allows thermal simulation of 3D circuits based on layout information [17]. Shang et al. use a heat spreading model of heat transfer in their work to evaluate the effectiveness of temperature-aware routing algorithms [18]. Skadron et. al provide HotSpot, a library of functions to allow fast thermal simulation of functional-unit sized blocks, providing unit-level thermal data with very low computational overhead [19].

1.3 Contributions and Future Impact

As the temperature of a given region of the chip is not directly related to the power dissipation at that region, thermal modeling tools are an important part

of the study of thermal problems. While a number of researchers have developed thermal modeling tools for use in their research, these tools are not well-suited to early-stage architectural evaluation. As such, a new thermal estimation library, *HS3d* , was developed. This tool, unlike previous tools, can model incompletely-specified floorplans in a very short time, enabling its use in automated floorplanning tools, and allowing a wide range of design spaces to be explored easily. The *HS3d* library also supports forward-looking technologies, such as multi-wafer stacking and thermal vias, providing a yet wider possible scope of investigation [20, 21]. This tool is now freely available, and is currently being used for early evaluation of 3D FPGA designs by Xilinx Corp., demonstrating the commercial need for such a tool.

This tool is first used to study the characteristics of hotspots in future technologies and architectures, where the variation in hotspot activity due to external factors such as the thermal interface material, and internal factors, such as process variation, are demonstrated. The influence of these non-design factors can have a significant impact on the thermal profile of a device, indicating a need for runtime solutions that can adapt to thermal profiles that vary not only with time, but from die to die, even with identical workloads. These results also show that many previous works exploring thermal solutions, while effective in some limited cases, can have a relatively small impact when compared to these other factors. In the future, temperature mitigation techniques will need to be evaluated in the context of the larger system, as only changes of more than 5° are truly meaningful.

This thesis then presents two different means of reducing hotspot problems, first in generally reconfigurable architectures, such as Network-on-Chip, and then a reduction targeted specifically at the integer ALU. The integer-specific optimization can reduce activity in the integer ALU by nearly 50% in the most integer-heavy applications, providing promise that even non-homogenous designs can be modified to reduce hotspot development without impacting performance. The modifications suggested can be incorporated into almost any modern processor, and will hopefully spur research into similar 'unit re-use' techniques in the future.

Finally, a model of temperature's impact on timing is presented, where preliminary results show that the performance of transistors can vary by as much as 5% per 10° of variation. These variations can be a significant roadblock to future

designs that show dynamic workload variation on a large portion of the die, such as multiprocessor systems on chip, and networks-on-chip. In such designs, it is possible for some units to operate at a 33% lower clock rate than distal units due to temperature, forcing lower frequency operation if a global signaling scheme is used. A means of minimizing these timing variations is proposed, where multiple buffer sizes and path choices ensure timing compliance in the face of varying conditions. These buffers, when combined with globally asynchronous, locally synchronous design, can allow large designs to operate within a fraction of their nominal operating speed, and providing maximum performance. This large dependence of performance on temperature indicates that the traditional design flow is unable to accurately estimate performance and timing, and thus, cannot make fully informed choices as to the placement of units or the routing of signals. This thesis discusses the inclusion of thermal modeling in the design flow, and demonstrates that current design flows can include up to a 25% inaccuracy in modeling, leading to less than optimal designs.

Chapter 2

Design of an Architectural Level Temperature Estimation Tool

Due to the large importance of temperature in many fields, such as chemical engineering and mechanical engineering, there are many existing software packages that perform temperature modeling, and a well-established body of research describing the flow of heat in a system. In addition, the computing industry has seen the use and development of such tools for use in the development and improvement of packaging and cooling systems. These tools, while suitable for macro-scale thermal evaluation, are often difficult to use, have high runtimes, and cannot be easily integrated with existing architectural exploration tools. This lack of suitable thermal estimation tools led to the simultaneous development of a number of specialized thermal estimation packages for use by the computer engineering community. These tools, while productive initial effort, did not yet meet the needs of the computer architect, and so a new tool, *HS3d* , was developed. This chapter discusses the existing thermal estimation tools, the motivation for the development of *HS3d* and its resulting capabilities, and finally the validation of the *HS3d* library.

2.1 Existing Tools

The vast majority of thermal modeling tools employ finite element modeling techniques, in which a given system is reduced into a number of adjacent cells. Through

an iterative process, the conservation laws are applied to each cell, allowing heat, charge, and mass to flow between cells. This process is iterated until the amount of flow between cells stabilizes within some error bound, at which point the time step is incremented, and the problem re-solved. This process, while highly accurate, and allowing for the simultaneous and interdependent solution of various systems, such as the self-heating of a wire, and the resulting air flow due to thermal drift. Given the complexity of the systems that such tools can be used for, including aircraft development, chemical engineering, and the design of machine rooms, the input scheme for such tools is also highly complex. The input scheme might use detailed textual input, as in CAD tools, or a graphical interface, but in general, every element in the system must be well-defined, with mechanical and thermal properties included, and every structure composed of basic parallelograms and partial spheres. Examples of such tools include ANSYS Multiphysics [22], Flotherm [23], and MSC.Marc [24]. These tools, while powerful, aren't well suited for use in computer architecture for a number of reasons. The inclusion of mass flow and electrical characteristics into the model results in high runtime, while the fully featured input model presents a steep learning curve and is difficult to automate.

As temperature variation on die has recently come under scrutiny, researchers have been developing their own, specialized tools for thermal estimation. Many different thermal modeling tools have been developed, operating at different levels of abstraction. Chiang, Banerjee, and Saraswat have developed a SPICE-based thermal model that allows extremely high accuracy modeling of device temperatures, without the overhead of full finite-element simulation [16]. Wilkerson, Raman, and Turowski present a Python-based framework that allows thermal simulation of 3D circuits based on layout information [17]. Shang et al. use a heat spreading model of heat transfer in their work to evaluate the effectiveness of temperature-aware routing algorithms [18]. Skadron et. al provide HotSpot, a library of functions to allow fast thermal simulation of functional-unit sized blocks, providing unit-level thermal data with very low computational overhead [19].

In order to thoroughly characterize a large design space, a large number of thermal profiles must be generated. While there are many existing finite-element-modeling packages that support very high accuracy thermal modeling [22, 25], there

are limitations to such tools for a large design space exploration. The user interface of such tools is designed to support the modeling of almost any imaginable object in a large number of possible environments. As such, the creation of a thermal simulation can take significant time, as each component or subsystem must be entered, often as a combination of volumes with associated thermal properties. In a traditional design flow, the time required to build individual objects is amortized by the large reuse possible. For example, once a detailed model of a rack full of servers is implemented, those racks can be replicated in a number of possible configurations, allowing datacenter designers to optimize the airflow from a cooling system. Once setup, the limited number of simulations can be performed by a large multiprocessor computer over the course of a few hours or days. This large initial entry time and simulation prohibits not only integration with automated floorplanning tools, but also is prohibits rapid evaluation of a number of designs.

Skadron, et al., distribute a library known as HotSpot, which provides a much simpler interface geared explicitly towards the temperature profiling of chips [19]. This library takes only two inputs - a device floorplan at the functional unit level, and the power dissipated by each such unit. As the library exposes a C function interface, it is easily integrated with other tools, such as the Wattch power modeling tool. Shang, et al. [18], noted that HotSpot showed a significant deviation from empirically obtained thermal data. This deviation was acknowledged, and was due to lack of thermal interface material (TIM) modeling in the early releases of the library. Subsequent versions of the HotSpot library have included TIM in the thermal model and have shown excellent thermal predictions when compared to an actual thermal test performed using an FPGA.

While this library is powerful, it has some limitations. The input format of the chip floorplan is quite restrictive, in that in all areas of the chip must be accounted for in the input floorplan - any 'unclaimed' space on the die not allocated to a functional unit will completely invalidate all results. While not a problem when modeling completed designs, this restriction limits the use of the HotSpot library in automated tools, which do not necessarily have complete information about the location or power consumption of the entire device. As the HotSpot library was originally intended to be used to determine the effectiveness of power consumption optimizations on existing floorplans, the use of the library focuses on

the creation and initialization of a floorplan object, and the subsequent application of temperature-estimation based on power consumption. This design goal resulted in a process that, while allowing for a reasonable speed of temperature estimation on a single floorplan, can result in unreasonably high performance penalties when evaluating a number of floorplans. In addition, the HotSpot library offers no support for modeling multi-layer device stacks of any sort, modeling only single-layer devices.

2.2 HS3d Features and Capabilities

To remedy these limitations, the HotSpot library was significantly extended and performance optimized, resulting in a new thermal estimation library we term "HS3d". While the basic computational model and methods are largely unchanged, they have been extended to allow for multi-layer device stacks, including optional interlayer material. The inclusion of inter-layer thermal vias can be approximated by varying the vertical thermal resistance of the materials. New library accessors allow the creation and use of incompletely-specified floorplans, and the ability to simulate floorplans at an arbitrary grid resolution has been added, ensuring accurate thermal modeling of large floorplan blocks. Many of the routines have been rewritten to optimize loop accesses, cache locality, and memory paging, while the LAPACK library was used for large matrix operations. Overall, these improvements have resulted in a significantly more static memory footprint, and runtime improvements of over 1000X when simulating a large number of floorplans.

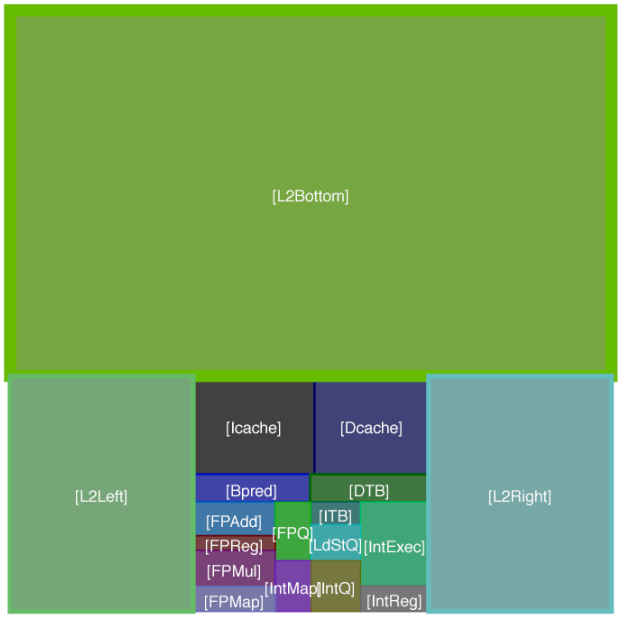
2.3 Tool Validation

To ensure accurate reporting of all results, we compare our new library to a commercial FEM software to verify accuracy. The commercial software used is Flomerics Software's "Flotherm" tool [23]. The sample device and package used for the verification process are in the following paragraph.

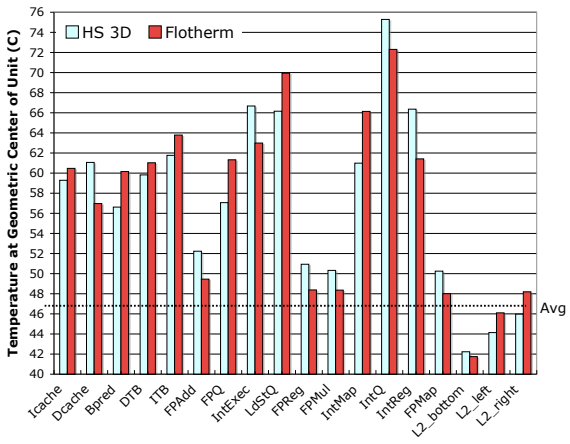
A 1.8 cm by 1.8 cm by 500 micron silicon substrate sits adjacent to a 10 micron thick silicon active layer. The substrate is connected to a 1mm thick 3cm by 3cm copper heat spreader by 75 microns of thermal interface material. The heat

spreader is connected to a 6 cm by 6cm aluminum heat sink with a 6.9mm thick base and 256 evenly distributed 2mm x 2mm pins, with an ambient temperature of 35° C. We model power dissipation based on the Wattch tool, using the default processor configuration found in [9], running the MCF benchmark from the Spec2000 integer suite [26]. While we base our discussions on a single benchmark, our conclusions are general, and do not pertain to a specific characteristic of this application. To represent the various power consumption optimizations found in modern processors, we use the "CC3" power values given by the tool. We assign the resulting power values to the ev6-like floorplan provided by the default installation of the HotSpot tool to ensure public repeatability of our results. In addition, the HS3d library will be made publicly available after the blind review process, allowing architects to perform their own thermal analyses. Figure 2.1(a) shows the floorplan used, while figure 2.1(b) shows the resulting temperatures given by each tool.

The average temperatures between the two tools differ by only 0.02 degrees Centigrade, indicating that the overall thermal path to ambient is being accurately modeled. The individual units do show some difference, however, this deviation is not so significant as to invalidate the results. We compare the results from the tools by presenting the proportion of explained variation, R^2, one possible measure of how well an approximation matches real data points [27]. In such a measure, the R^2 value is the fraction of total squared error that is explained by the model, where values closer to 1 imply closer approximations, and values closer to zero imply little predictive power of the model. HS3d approximates Flotherm with an R^2 value of 0.87, while both Flotherm and HS3d show only a weak relationship to power dissipation ($R^2 < 0.17$) and power density ($R^2 < 0.65$). In particular, the units (IntMap, FPQ, LdStQ) which differ the most from the Flotherm model are characterized as being 'downstream' of the airflow moving past the hottest part of the chip, the integer queue. As the HS3d library does not model airflow, instead using a simple thermal resistive model for its heat sink, the thermal profile is ignorant of such environment-specific details. Further testing showed that altering the direction of incident airflow continued to vary the results, but in all cases, the R^2 value remained over 0.86. Finally, the maximum error between the two models was 5.1° C, and occurred at the location of peak temperature. Average



(a) The ev6-like floorplan used



(b) Temperature Profile - Note that the average temperatures provided by the two tools are identical

Figure 2.1. Comparison of HS3d Library to Flotherm

error between the models was less than 3° C, significantly lower than the standard deviation of temperature.

The accuracy of the HS3d library in multi-layer device modeling was next modeled. Here, we model a multi-layer device fabricated using 10 micron thick silicon layers and 2 micron thick interlayer material, similar to a process described in [28].

The test case consisted of two total silicon layers, each containing a sample processor core as in the 2D verification. To more rigorously exercise the library, the two processors are given different power values for the functional units, with the processor closest to the heatsink being assigned the same distribution as in the above case, and the more distant processor being assigned a power distribution based on the SpecFP 2000 Applu benchmark [26]. Thermal profiles are again compared, with the temperature at the center of each functional unit being measured, and counted as a data point for the purposes of R^2 calculation. In the multi-layer case, the tool's correlation to Flotherm decreases slightly, to 0.87, with a corresponding R^2 value of 0.77. While this value shows that there are some differences in thermal profile, the majority of the differences are again isolated to the region of the chip 'downstream' from the hotspot at the integer remap unit. In addition, while the thermal analysis using Flotherm took almost 7 minutes using a Sun Blade 1000, the HS3d tool required less than one tenth of a second on the same machine.

To characterize the performance improvements implemented, we consider the time required to perform thermal analysis on floorplans containing a variable number of units, arranged in a grid. For a small floorplan, containing only 18 functional units, HotSpot requires a total of 0.029 user seconds, as measured by the Unix "time" tool. HS3d requires 0.004 user seconds for the same operations, a speedup of roughly 7 times. For a larger floorplan containing 81 units, as occurs if finer grained temperature information is desired, HotSpot requires 3.077s, as compared to only 0.029s for HS3d. In this case, the estimation rate is increased by over 100 times. This speedup is especially important for accurate thermal analysis of designs such as FPGAs, or tiled architectures, such as the RAW processor [29], where large portions of the chip cannot simply be clumped together as "Cache", and instead, a large number of grid cells must be accurately modeled.

Chapter 3

Characterization of Hotspot Behavior

While thermal analysis of chips is still in its infancy, early works have shown the existence of "hotspots", regions of localized high temperature [2]. Later works have built on this claim, and demonstrated various means of mitigating hotspots, and attempting to minimize thermal problems. There are no previous works, to the best of the author's knowledge, that characterize and discuss the general case of hotspot development. This lack of fundamental understanding presents a roadblock to thermal-aware design, in that architects must first become acquainted with hotspots and their properties before useful design work can begin. This chapter evaluates and characterizes the development of hotspots in a number of generalized cases, and attempts to determine generalities and guidelines for mitigating hotspots.

3.1 Background and Related Work

While the temperature of devices has been studied in the past, such work has been the domain of thermal and packaging engineers, rather than chip architects, who have only recently started focusing on this problem [4, 5, 6]. The efforts of thermal engineers focused on the design of adequate cooling solutions for already fabricated chips, and allowed for some overdesign of the cooling solution to ensure that all regions of the device operated within thermal bounds. As the power consumption

(and power density) of devices has increased, however, it has become cost and space inefficient to overdesign cooling solutions, resulting in generalized thermal responses. There are many ways to implement such generalized thermal responses, including clock gating [7], voltage and frequency scaling [8], and instruction cache gating [9]. These responses, while effective, often incur significant performance penalties as the entire processor is slowed to reduce power consumption. Skadron et al., discussed the use of control-theoretic techniques to reduce the performance penalty of such techniques, allowing a minimal amount of slowdown that still ensures the prevention of a thermal emergency [10].

Many different thermal modeling tools have been developed, operating at different levels of abstraction. Chiang, Banerjee, and Saraswat have developed a SPICE-based thermal model that allows extremely high accuracy modeling of device temperatures, without the overhead of full finite-element simulation [16]. Wilkerson, Raman, and Turowski present a Python-based framework that allows thermal simulation of 3D circuits based on layout information [17]. Shang et al. use a heat spreading model of heat transfer in their work to evaluate the effectiveness of temperature-aware routing algorithms [18]. Skadron et. al provide HotSpot, a library of functions to allow fast thermal simulation of functional-unit sized blocks, providing unit-level thermal data with very low computational overhead [19].

3.2 Package Variation

Many external factors influence the temperature distribution of a chip, including the ambient temperature, heat sink design, and heat spreader design. A general rule for calculating chip average temperature is based on the effective thermal resistance of the heat sink, a value provided by the heat sink manufacturer [30]. This rule notes that

$$\begin{aligned}\text{Chip Avg. Temperature} &= \text{Ambient Temp.} + \\ &\quad \text{Chip Power Dissipation} \times \text{Thermal Resistance to Ambient}\end{aligned}$$

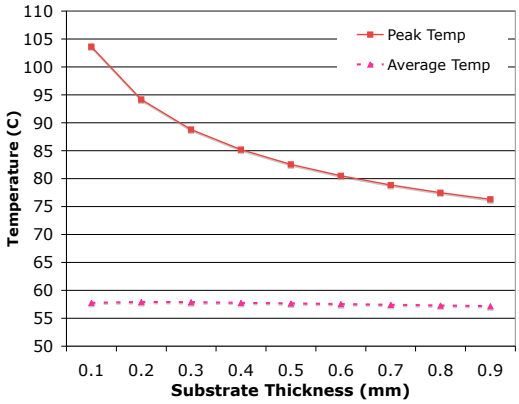
and allows a fast and simple means of calculating the average temperature of a design. Our goal, however, is not to focus on the average chip temperature,

which can be controlled by limiting total chip power dissipation and the selection of a heat sink with a low thermal resistance, but instead to focus on temperature variation across the die, regions where temperature exceeds the average, sometimes by a large amount. Varying the thermal resistance of the sink has little effect on the distance between ambient and the peak, merely increasing the average temperature. Similarly, increasing the ambient temperature has little effect on temperature distribution on the chip. To significantly impact the thermal profile, a change must somehow affect the means by which heat distributes horizontally, such as changing the horizontal thermal resistance of the silicon, or preventing heat from equalizing by moving up to the heat spreader, distributing through the low thermal resistance copper, and back down to cooler regions of the chip.

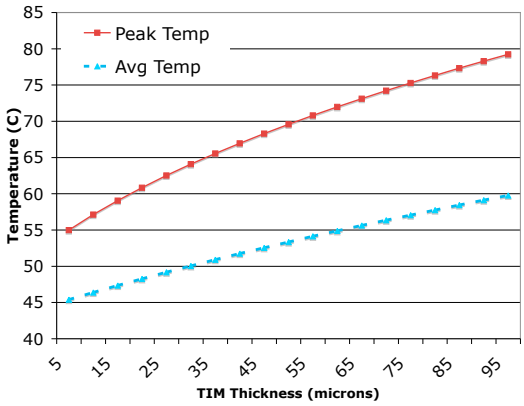
We begin our exploration of the thermal space, therefore, by considering temperature variations due to differing silicon substrate thicknesses. While many applications utilize standard thickness wafers in the end package, other application domains, such as the smart card, flash memory, and portable electronics markets, can require wafers less than 200 microns in thickness for compact form-factor reasons. Conversely, foundries using large wafer diameters tend to use thicker wafers, up to 800 microns in thickness for the most recent 300mm diameter wafers [31]. Using the same single-layer test configuration (see figure 2.1(a)) as noted in section 2.3, we vary the silicon substrate thickness from 100 microns to 1mm, and plot the peak temperature of the device in Figure 3.1(a).

While the average temperature of the chip remains nearly constant as the silicon substrate thickness varies, the peak temperature of the chip varies greatly. As the silicon is thinned from 300 microns to 100 microns, the peak temperature increases by 14.8° C, bringing the peak temperature from 88° C up to 103.6° C, possibly resulting in a large increase in standby power consumption, and a device with a less than anticipated battery life.

We next consider the impact of the thermal interface material connecting the substrate to the heat spreader. As this material is applied post-foundry, and often by end users, there is almost no way to accurately predict the eventual thickness of material applied to a chip. Publications have noted that a standard application thickness is on the range of 1 to 3 mils (25 to 75 microns) [32], but we consider a slightly larger range, as in practice, it is possible that the material might be



(a) Effect of Varied Substrate Thickness



(b) Effect of Varied Interface Material Thickness

Figure 3.1. Effects of Packaging on Temperature

applied with something as crude as a credit card in homebuilt systems [33], or simply allowed to spread due to heat sink pressure [34] . Figure 3.1(b) shows the peak temperature of the chip as the interface material thickness varies.

Unlike in the case of varied substrate thickness, varied TIM thickness does affect the average chip temperature, with the difference between a 25 micron application and a 75 micron application being roughly 8° C. More importantly, the peak temperature also increases as TIM thickness increases, resulting in a peak temperature increase of 13° C. Given the wide variations in temperature possibly resulting from variations in TIM thickness, it is important that architects remain aware of such possible differences when comparing various thermal estimates in

published works. The authors also note that while the TIM is represented as a simple thermally resistive cuboid in our model, in reality, the interation of the TIM with the silicon and heat spreader is much more complicated, due to possible air-filled gaps, as well as relative exposed surface area due to imperfections in the surfaces. As such, the temperature variations presented here can be even wider in practice. As power densities rise, such TIM thickness variation will need to be considered a form of process variation, and accurately predicted to ensure that no region of a device can reach thermal runaway or breakdown. This after-foundry variation also has implications for design-time thermal optimizations, in that there is no perfect means of estimating the device's eventual operating temperature profile. Small variations in the application of thermal interface material could result in air pockets over normally cool regions, or especially thin layers of material directly over traditional hotspots, helping reduce the peak temperature. Even a variation as small as 10 microns can result in peak temperature changes of up to 4° C, implying that any architectural or compiler technique or optimization designed with a small pre-defined temperature difference in mind can easily be rendered ineffective by process variation.

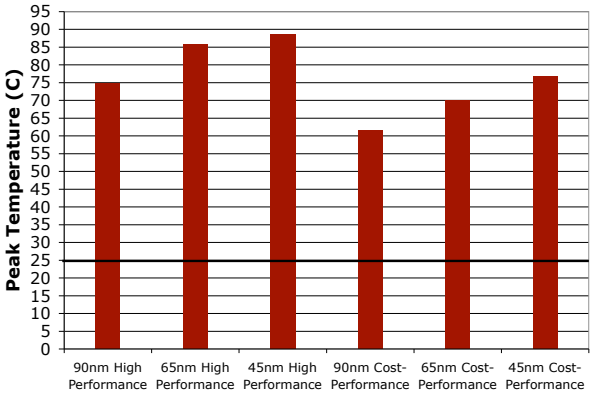
3.3 Thermal Characteristics of the ITRS Roadmap

The ITRS roadmap provides guidelines as to estimated die sizes and power densities for upcoming technology nodes [35]. Due to the relatively recent introduction of 300mm wafers into production, and distant introduction of larger wafers, the target die size for high-performance chips remains a constant 310 square millimeters until at least 2018. Given this constant die area, and the relatively slow rate of improvements in packaging and cooling technologies, the ITRS roadmap indicates that a relatively small increase in overall chip power density is expected in the future, with the $0.51\frac{W}{mm^2}$ target power density of high performance chips in 90nm technology increasing to only $0.64\frac{W}{mm^2}$ by 2008. This target power density is then fixed at $0.64\frac{W}{mm^2}$ until at least 2018. These targets indicate a peak power consumption per chip of 173W, increasing to 217W by 2008. We begin our thermal analysis of the effects of non-uniform power distribution, then, by starting with the base case of a perfectly homogenous chip design, dissipating the target power density

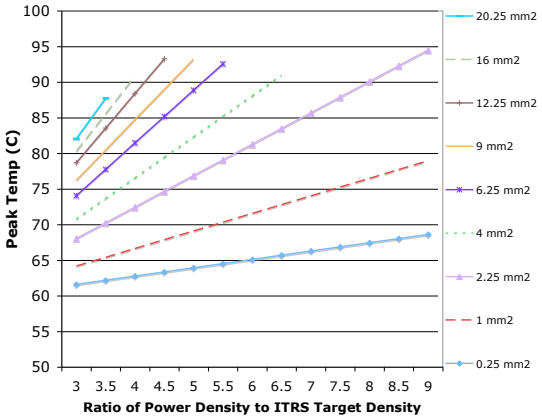
indicated by the ITRS roadmap. The resulting temperatures are nearly uniform across the chip, so only the average temperatures are shown in Figure 3.2(a). As it is reasonable to assume that future products will include significantly higher quality cooling solutions (possibly including water cooling), these (and all future) results assume a heat sink solution with an extremely low thermal resistance of only 0.1° C/W and a TIM thickness of 25 microns (1 mil). This cooling configuration, while highly effective, is also conservative, in that a less advanced cooling solution would only increase the temperatures demonstrated in our results.

Figure 3.2(a) shows that the ITRS roadmap target will be quite achievable, assuming a uniform power density, in that the average temperature of high performance chips is only expected to increase by roughly 9 degrees as technology scales from 90nm to 45nm technology. This value is unsurprising, given that simple thermal calculations show that for a heat sink solution with an efficiency of $0.1\frac{^\circ C}{W}$ and a TIM/Heat Spreader combination of similar resistance, the increase in total power by 44W results in an 8.8° C temperature difference.

In practice, however, chip power densities are often very non-uniform. We begin our analysis of such non-homogeneity by considering a very simple case of a single high-power unit in the center of an otherwise uniform power density chip. In figure 3.2(b), all configurations have an average power density of $0.51\frac{W}{mm^2}$, the ITRS specification for 90nm technology. We vary the size of the 'high power' region, as well as the factor by which the high-density region exceeds the average power density of the chip. Given this fixed average power density, the higher the power density in the 'hot' region, the less total power is being dissipated by the cold regions. Figure 3.2(b) shows that while perfectly homogenous devices yield quite reasonable temperature profiles, with peak temperatures near 60C, the peak temperature of the chip can rise quite significantly due to non-uniform power dissipation. Even a small unit, such as a 2mm x 2mm unit with a power density of $2\frac{W}{mm^2}$ (for a total power consumption of 8W) results in a peak temperature increase of 17° C, or more disturbingly, increases the amount by which chip temperature exceeds ambient by more than 50%. As technology scales, ever-shrinking processor functional units and ever-increasing cache sizes are likely to result in such small and high power density blocks becomming more common. Architects must be aware of this problem, as simply designing for a target chip-wide power consumption will no



(a) Temperatures of Homogenous Devices Following the ITRS Specification



(b) Effects of Non-Uniform Power Density on Peak Temperature - Power Density in the "High Power" Region is Some Factor Greater than the ITRS Specification

Figure 3.2. Impact of Uniformity on Temperature

longer provide suitable thermal characteristics - power density itself must become a design goal, especially when designing high-performance custom logic units that are likely to consume little area. Conversely, architectures such as FPGAs and tiled architectures, such as the RAW architecture [29], have much more uniform power density profiles. Due to their low variation in power density, excursions

from the average will be small, implying that it is possible to safely operate these devices with a higher average temperature. This higher average temperature can be exploited by allowing a higher operating frequency, and resulting total power consumption, or by allowing smaller, cheaper, cooling systems.

3.3.1 Effects of Component Redesign

As an example of a power-density optimized design process, we first define some packaging parameters for our target device. In this example, we assume a target average power density of $0.5\frac{W}{mm^2}$, continue using the same 25 micron TIM thickness and silicon thickness of 500 microns as in previous simulations, and employ a very low thermal resistance of 0.1° C/Watt. We then use the HS3d library to generate figure 3.3, which shows the temperature increase (above average) resulting from a power dissipating unit being added to the perfectly homogenous base case. The resulting lines show the minimum temperature increase resulting from the addition of such a unit, ignoring any other units. We demonstrate how these curves can give simple thermal guidance to an architect by considering the development of a new issue queue unit.

If designed using the fastest, smallest design goals, such a unit might require 2.25 sq. mm, and consume 3W of power, for a total power density of $1.5\frac{W}{mm^2}$. Figure 3.3 shows that such a unit, which has a power density 3 times the chip average, would result in a temperature spike of at least 8.9° C, as shown at point "A" on figure 3.3. A power-oriented re-design of the unit such that it provided the same performance, but required two-thirds of the power, as shown as point "B" on figure 3.3, would reduce the peak temperature to a 4.5° C peak, as the power density dropped to $1\frac{W}{mm^2}$, twice the chip average. A power-density oriented design, on the other hand, might notice that the issue queue is planned to sit adjacent to the register file. The register file might require 4 sq. mm, and consume only 2W as well, for an overall power density of $0.5\frac{W}{mm^2}$, and hence no temperature rise above ambient. Integrating the issue queue with the register file, such that the elements of each are interleaved with each other, and hence uniformly distributed over an area, would therefore require 6.25 sq. mm and consume 5W. Such a unit has a power density of $0.8\frac{W}{mm^2}$, significantly lower than that of the issue

queue unit alone. The resulting unit has a power density only 1.6 times higher than the chip average, and results in an estimated temperature rise of only 4° C, as seen in point "C". Integrating the high power density unit (the issue queue, in this example) with the lower density unit (the register file, in this example), results in a peak temperature decrease of almost 5° C, better than the decrease obtained by reducing power consumption of the problematic unit by 33%. Using the HS3d library, such figures can be easily generated, and allow architects to quickly determine which components will be most problematic without performing a complete thermal simulation, as well as determine possible options to reduce hotspot activity.

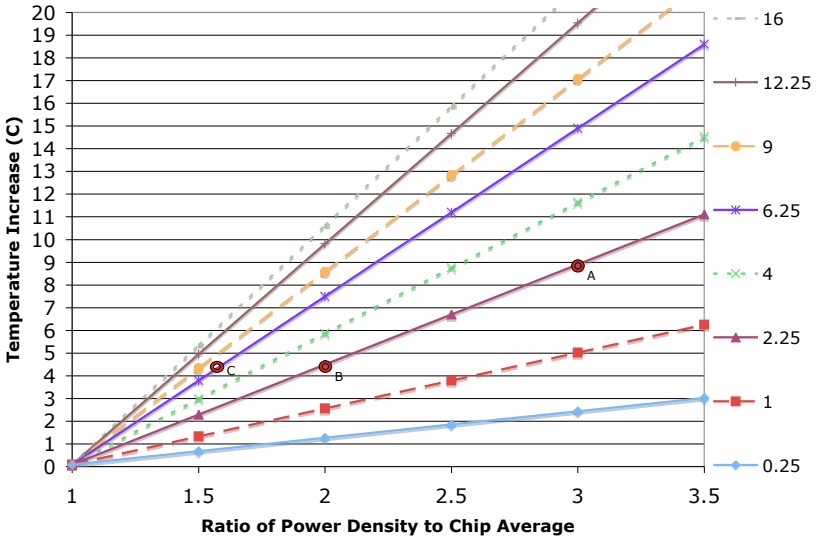


Figure 3.3. Uniform 0.25W/sq. mm chip with fixed-power region superimposed. The area of the region is varied, and is shown on the right side of the figure.

3.4 Thermal Characteristics of a Sample 90nm Processor

As the previous discussion was based only on the ITRS roadmap, it included several simplifications, such as the presence of only a single high-power unit, located at the center of the unit. In practice, the power density of functional units can vary widely, as can their location. Therefore, we now consider the characteristics of a sample 90nm processor. This processor is based loosely on the AMD Athlon 64 "Winchester" core processor [36], which has an aspect ratio of roughly 1:2, in which half of the die is consumed by the 512KB L2 cache. Overall, the entire die is 84 sq. mm. Table 3.1 shows the power dissipation of the sample processor divided into dynamic and leakage components, while figure 3.4 shows the device floorplan. The processor power consumption was determined by finding the ratio of power consumption of the various units using the *WATTCH* tool, then scaling them such that the total power consumption of the device matches the thermal design power of the Winchester core. Note that our target processor's power profile, thus calculated, matches well with other reported results, such as [37].

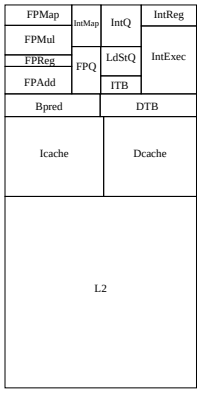


Figure 3.4. Base 90nm Processor Floorplan

The power density for the processor, thus obtained, averages $0.797\frac{W}{mm^2}$, which, while over the ITRS "high-performance" target density, is much closer to the ITRS "cost-performance" specification, which targets a die area of 140 sq. mm. and power density of $0.65\frac{W}{mm^2}$. We also note that the simple obtained standard deviation in power density is over $1\frac{W}{mm^2}$, with the power density of the various units varying by a factor of 18, even when ignoring the unused floating point units.

Table 3.1. Key Details of 90nm Sample Core

Unit	Dynamic (W)	Leakage (W)	Total (W)	Area (mm^2)	Power Density ($\frac{W}{mm^2}$)	Temperature (C)
Icache	10.39	0.95	11.34	9.04	1.25	65.9
Dcache	10.80	0.60	11.40	8.47	1.34	66.9
Bpred	2.16	1.68	3.84	2.45	1.56	67.4
DTB	0.44	0.15	0.59	2.48	0.23	65.9
ITB	0.22	0.05	0.27	0.84	0.32	68.9
FPAdd	0	2.18	2.18	2.10	1.03	63.8
FPQ	0	0.34	0.34	1.56	0.22	63.6
IntExec	7.65	2.46	10.12	4.31	2.34	85.9
LdStQ	1.66	0.23	1.90	1.35	1.40	77.6
FPReg	0	0.30	0.30	0.90	0.33	60.7
FPMul	0	2.33	2.33	2.24	1.03	60.5
IntMap	0.55	0.30	0.85	1.39	0.61	69.7
IntQ	7.32	0.42	7.75	1.90	4.06	95.7
IntReg	1.35	0.44	1.79	1.32	1.35	83.8
FPMap	0	0.34	0.34	1.56	0.22	56.7
L2	5.40	6.16	11.57	41.99	0.27	43.2
Overall	48	19.00	67.00	83.99	0.79	

Max Power Density	4.06 W/mm^2
Min Power Density	0.22 W/mm^2
Avg Power Density	0.797 W/mm^2
Max / Avg	5.1

The region of highest power density is the integer instruction queue, with a power density 5 times higher than the chip average. Table 3.1 provides the temperature distribution on this sample core, demonstrating the wide range of temperatures possible in practice.

3.5 The Impact of Technology Scaling

Technology scaling impacts both the size of units as well as the power dissipation. In previous technology generations, dynamic power was the largest component of power dissipation, and as capacitance decreases were met with frequency increases, the dynamic power consumption tended to reduce as the square of the voltage. These voltage reductions, however, were coupled with threshold voltage reductions

Table 3.2. Key Details of Sample Core Scaled to 65nm and 45nm

Unit	Dynamic (W)		Leakage (W)		Total (W)		Area(sq.mm)		$\frac{W}{mm^2}$	
	65nm	*45nm*	*65nm*	*45nm*	*65nm*	*45nm*	*65nm*	*45nm*	*65nm*	*45nm*
Icache	8.73	7.22	2.39	5.97	11.12	13.19	4.69	2.26	2.37	5.83
Dcache	9.08	7.5	1.5	3.76	10.58	11.27	4.39	2.12	2.41	5.32
Bpred	1.82	1.51	4.2	10.5	6.02	12.01	1.27	0.61	4.73	19.54
DTB	0.37	0.31	0.37	0.94	0.75	1.25	1.29	0.62	0.58	2
ITB	0.19	0.15	0.13	0.32	0.31	0.47	0.44	0.21	0.72	2.24
FPAdd	0	0	5.46	13.64	5.46	13.64	1.09	0.53	5.01	25.97
FPQ	0	0	0.87	2.18	0.87	2.18	0.81	0.39	1.07	5.56
IntExec	6.43	5.32	6.16	15.41	12.6	20.73	2.24	1.08	5.63	19.2
LdStQ	1.4	1.16	0.6	1.49	2	2.65	0.7	0.34	2.84	7.82
FPReg	0	0	0.76	1.89	0.76	1.89	0.47	0.23	1.61	8.36
FPMul	0	0	5.84	14.6	5.84	14.6	1.17	0.56	5.01	25.97
IntMap	0.46	0.38	0.77	1.93	1.24	2.31	0.72	0.35	1.71	6.66
IntQ	6.16	5.09	1.06	2.65	7.22	7.74	0.99	0.48	7.31	16.24
IntReg	1.14	0.94	1.11	2.78	2.25	3.72	0.69	0.33	3.26	11.19
FPMap	0	0	0.87	2.18	0.87	2.18	0.81	0.39	1.07	5.56
L2	4.54	3.75	15.42	38.54	19.96	42.29	21.77	10.5	0.92	4.03
Overall	40.33	33.33	47.51	118.77	87.84	152.1	43.54	21	2.02	7.24

and gate oxide thinning, resulting in increased standby power consumption. This changing relationship of power dissipation can result in a changing power profile as technology scales, and hence a changing thermal profile. As such, we now observe the effects of various types of technology scaling on thermal profiles.

We begin this analysis by scaling the sample processor design shown in figure 3.4, whose power profile is shown in table 3.1 . We scale the processor to both 65nm and 45nm technology nodes, assuming simple process scaling, with no architectural changes. To account for the increasing leakage currents as technology scales, we increase standby power by 5X per generation, as given by Borkar [38]. The resulting power dissipations are provided in table 3.2. As expected, standby power consumption is the dominant source of heat in the later technologies, however, problematically, the average power density increases to $2\frac{W}{mm^2}$ in 65nm technology, and to $7.2\frac{W}{mm^2}$ at 45nm.

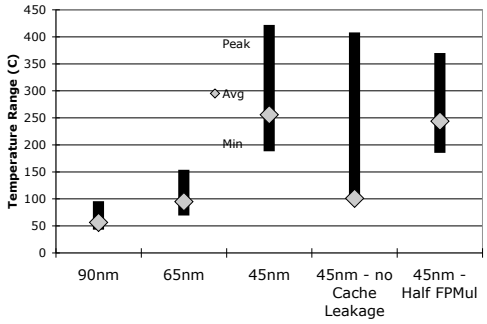
The first three columns of figure 3.5(a) show the minimum, average, and peak temperatures of the scaled processors. We see that even simply scaling the existing architectures without any additional complexity will result in problematically high temperatures at the 65nm node, and by the 45nm technology node, the power density will be too high to effectively cool.

3.6 The Impact of Gate Thickness Variation

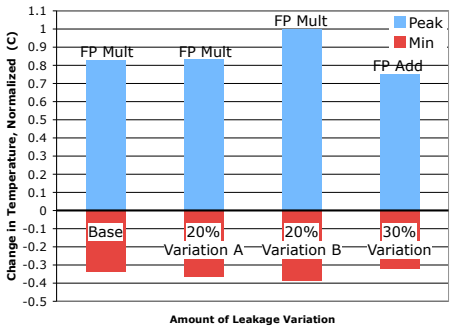
as a large percentage of power consumption at the 45nm node is due to leakage currents, which can vary greatly within a die due to process variations in threshold voltage, oxide thickness, and gate length [39], temperature variations on the die can be difficult to model accurately. We note that these variations in leakage consumption can be large, as for example, [40] notes that leakage can vary up to 30% due to gate length variations alone. Figure 3.5(b) compares the temperature variations resulting from four different leakage variation patterns, all of which have identical leakage power consumptions. The first column shows the base case, with no variation from the predicted scaling, while the second and third column are two different variation patterns where leakage variation is uniformly distributed in the range of ± 20% across the chip. The final column shows variation of ± 30%, where the traditional location of hotspot formation has its leakage reduced by the maximal value, to 0.7 times the normal value. These leakage variations on die result in hotspot magnitude changing by up to 25%, and the location of the hotspot shifting up to two functional units away from the location in the base case. These results indicate that even in the presence of highly accurate predictive modeling, on-die temperature sensors and adaptive hotspot avoidance mechanisms will be necessary in future technologies, as process variation will prevent accurate localized modeling of power distribution, even in situations where die are binned for identical power consumptions. Given this dependence of hotspot formation on leakage consumption, and hence process variation, it will be possible for two architecturally identical chips to exhibit significantly different thermal profiles, requiring intelligently responsive hotspot reduction mechanisms.

3.7 Thermal Characteristics of Multi-Layer Wafer Stacks

One of the technology enhancements currently being proposed [21] is the use of multi-layer device stacks, offering a reduction in overall wire length and inter-connect power consumption, and the possibility of integrating different process technologies into a single unit. As multi-layer device stacking has not yet reached



(a) Effects of Technology Scaling and Power Optimizations on Temperature



(b) Effects of Leakage Variation - The hottest unit is denoted at the top of each column.

Figure 3.5. Impact of Techology Scaling on Temperature

mass commercial production, there are a number of possible implementations still undergoing evaluation. While the specifics of these techniques vary, from wafer stacking to die stacking, each using different ablative techniques, only the thicknesses and composition of the final stack are important for thermal modeling. As one of the primary goals of multi-layer integration is the reduction of wire lengths, a common characteristic of multi-layer stacks is the use of thinned silicon wafers, with a very thin interlayer material between silicon layers. These thinned layers can also help keep the total thermal resistance of the system within reasonable bounds, as opposed to the use of full-thickness wafers, which would increase the

Table 3.3. Details of the Three Simulated Multi-Layer Technologies

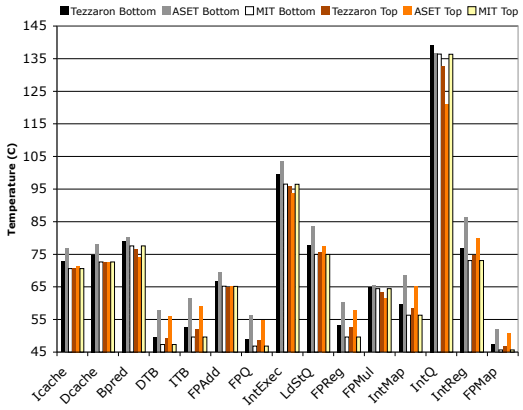
Technology Provider	Silicon Layer Thickness	Interlayer Material Thickness
Tezzaron	10 microns	2 microns
ASET	50 microns	10 microns
MIT	0.9 microns	0.3 microns

thermal resistance path to ambient significantly.

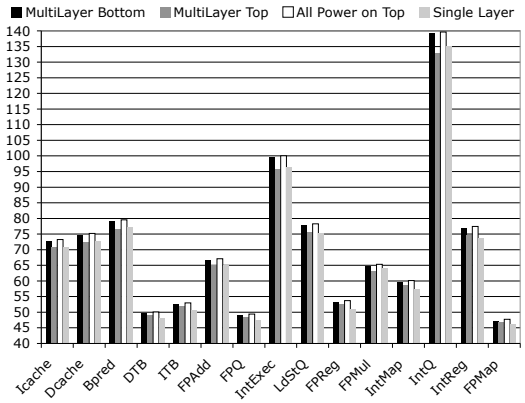
In figure 3.6(a), a single two-layer design is modeled, with only the thicknesses of the silicon layers and interlayer glue material varying. We again base our model on the 90nm sample processor shown in figure 3.4, although in this case, the design is 'folded' along the X-axis, placing the entire L2 cache farther from the heat sink, and the core itself on the layer closest to the heat sink. Table 3.3 lists the relevant details of the three technologies examined [28, 41, 42]. We note that while there are many other possible multi-layer stacking technologies that exist, the three we have chosen show a wide variation in parameters, and as a result, cover nearly all of the reasonable design space.

Figure 3.6(a) shows that while there are certainly small variations in temperature due to the choice of technology, in general, the average temperature of a region of the chip is nearly identical, regardless of the relative thicknesses involved. For example, this can be clearly seen in thermal profile of the Branch Predictor (BPred, the third column), where the temperature of a given layer in the Tezzaron stack differs from that of the ASET stack, and that of the MIT stack by only a few degrees. The difference in temperature between layers is also relatively small, with the largest differences being seen in the ASET technology, which is characterized by its thick silicon layers. Furthermore, the above simulations were performed without including the additional thermal conductivity provided by proposed "thermal dummy vias" [20], which decrease temperature variations between layers further, without significantly impacting horizontal hotspot distribution. Given the small differences in temperature variation seen between technologies, for the remainder of the multi-layer discussion, only the Tezzaron-based technology is shown.

In figure 3.6(a), we saw that the temperature difference between layers was very small, even when the interlayer material was 10 microns thick. In figure 3.6(b) we examine this further by comparing the temperature profiles of both a multi-layer chip and a single-layer chip with identical areal power densities. The first and



(a) Temperature Variations Between Different Stacking Techniques



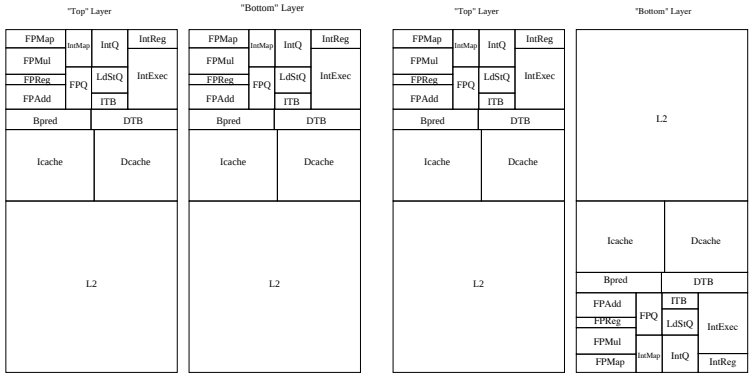
(b) Comparison Between Multi-Layer and Single Layer Designs with Equal Power Density

Figure 3.6. Effects of Moving to a Multi-Layer Process

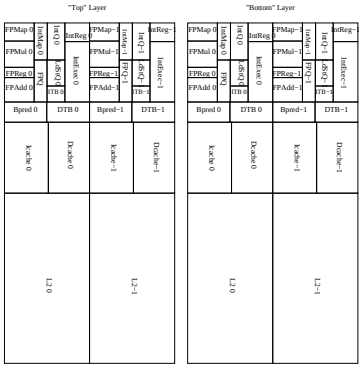
second columns shows the same small temperature deviations seen in figure 3.6(a), the third column shows the temperature of a two-layer design where all power is consumed on the top layer, and the last column shows the effects of the same power density distribution in a single layer design. Overall, the temperature profile remains relatively consistent regardless of the number of layers used in the design. This shows that even in multi-layer technologies, the temperature profile is primarily dependent on the areal power density, and the packaging's ability to dissipate the generated heat. Given the small temperature difference between layers, floorplan optimizations relying on the placement of high-power blocks closer to the

heatsink seem to provide little benefit, especially when compared to the benefits realizable by reducing areas of high power density. As technology advancement will only serve to increase the power density of components, ensuring that high-density components are not placed in the same vertical region of a stack will be a key design goal.

There are many ways in which multi-layer designs can be implemented. High power density units can be placed vertically adjacent to other high power units, or vertically adjacent to low power units. Alternately, units can be 'folded' onto multiple layers, reducing internal wire lengths, as in [43] We the temperature profiles of these designs by simulating the processor layouts shown in figure 3.7.



(a) Core-over-Core Design (b) Core-over-Cache Design



(c) Vertically Integrated Core Design

Figure 3.7. Three Different Multi-Layer Processor Design Styles

Figure 3.8 shows that due to the increase in areal power density resulting from stacking units on top of each other or folding high-power density units onto themselves can result in prohibitively high temperatures, while designs combining

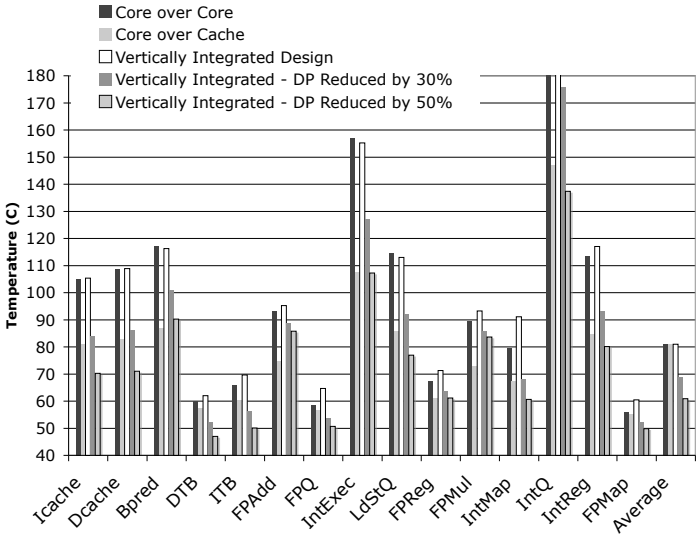


Figure 3.8. Temperature Comparison of Different Multi-Layer Processor Designs

high density units with low density units in the same vertical space have much more even temperature profiles. Given this, designs folding units onto multiple layers to reduce delays in high power consumption units can have peak temperatures far higher than those that avoid such locations of high power density. At the same time, such folded designs can expect to have an overall smaller wire capacitance due to the multi-layer design, reducing dynamic power consumption. For example, previous work notes that due to the reduced interconnect lengths, it is possible to reduce global power consumption by 15%, and replace high-power dynamic circuits by lower power implementations due to reduced timing pressures [44]. Figure 3.8 shows that in such situations, even if dynamic power consumption is reduced by 30%, the peak temperatures are significantly higher than a design that avoids placing high power units in the same region. Only when dynamic power is reduced by 45% does the core-over-core design again have the same peak temperature as the more homogenous core-over-cache design, even though the average temperature is lower by nearly 20° C. Given this high dependency on uniform power distribution, unless dramatic dynamic power reductions are possible when moving to multi-layer designs, such architectures will need to emphasize homogeneity over total power dissipation to achieve maximum performance.

Chapter 4

Techniques for Mitigating Temperature Problems

4.1 Introduction

4.2 Dynamic Workload Migration in Reconfigurable Architectures

Thermal solutions employed in current commercial processors such as dynamic clock disabling and dynamic frequency scaling shut down the entire processor for brief periods of time when the junction temperature exceeds a certain threshold [45, 46]. This results in significant performance loss during the period the chip cools down below the threshold. Additional thermal management approaches have been investigated in [12, 2]. Instead of shutting down or slowing down the entire chip, recent proposals have focused on the migration of the workload from a hot component to a cooler spare until the temperature reduces [47]. These techniques have been investigated using dual symmetric and asymmetric cores [14, 15], spare functional units [11, 48] and in high-end clusters in internet data centers [49]. However, the use of spare units adds to hardware redundancy consuming additional area. Also, changing execution between the spares incurs performance overhead. Further, as the spares should be ideally further apart for thermal consideration, the communication delay variations when using the original or the spare can in-

duce performance variations [48].In contrast to using spares, our work explores the use of dynamic reconfiguration as a mechanism for addressing localized hotspots. Our solution can be employed in programmable embedded architectures such as Network-on-Chip (NoC) designs [50] and Field Programmable devices that are being increasingly used in various applications. In these devices, the target functionality is mapped on to the underlying hardware by downloading a configuration stream. In order to avoid hot spots, we periodically modify the configuration stream to migrate the functionality performed by the programmable fabric across the chip in order to balance the temperature profile. In essence, we implement a spatial remap of the different functionalities at runtime. In order to provide a detailed insight on the different issues, we demonstrate the effectiveness of our technique using variants of a NoC design that implements a low-density parity check decoder [51, 52, 53, 54], which is used widely for error-correction in wireless communication and disk drives. Starting with a thermally-aware static mapping for this design, we explore issues such as the frequency of functional migration, choice of remap function, techniques for modifying of the configuration stream, performance impact of the reconfigured mappings, and heating patterns that can(not) be effectively handled by some migration schemes. Our experimental investigation reveals that dynamic reconfiguration approaches are successful in balancing the thermal profiles and in reducing the peak temperature by up to 8 C when starting with a thermally optimized static mapping.

4.2.1 Application Background

We use a NoC implementation of a Low-Density Parity Check (LDPC) code decoder as the target design to demonstrate the proposed hotspot avoidance techniques. This section provides an overview of this implementation to provide the reader with a basic idea of the design under experimentation. The architectural design choices explored in arriving at the design presented is beyond the purview of this work. In LDPC error coding, a message is initially encoded using a G-matrix, then decoded using a series of computations based on a large, sparse matrix known as the H-matrix. The size and contents of the G and H matrices vary based on the amount of data redundancy desired and the size of the message to be encoded.

The decoding computations are an iterative computation process based on a bipartite graph, as shown in Figure 4.1. The graph is composed of two types of node, the bit node and the check node, and the edges between the nodes are defined by the H-matrix. Every *1* in the H-matrix defines a single edge between a bit node and a check node, resulting in a bipartite graph dependent on the H-matrix, and thus the desired error protection and message size. The correlation can be seen in Figure 4.1.

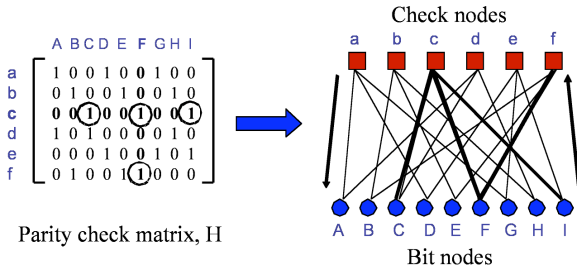


Figure 4.1. Sample H-Matrix and Corresponding Bipartite Graph

The decoding operation is initialized by seeding the bit nodes with the likelihood that a bit in the incoming message block is a *1*, based on the quality of the received analog signal. The iterative step consists of each bit node sending a value to each one of its associated check nodes, which, upon receiving values from all of their associated bit nodes, performs some computation. Each check node then sends a value to each of its associated bit nodes, which also wait to receive a value from all of their associated check nodes. Once these values have been received, computation is performed in the bit node, and another iteration is begun. Eventually, the values calculated in the bit node converge to indicate whether a given encoded bit is most likely a zero or a one. At this point, the values are sent from the bit node to the output of the chip, and another message is brought in for decoding.The communication intensive nature of the LDPC decoding has traditionally been a performance bottleneck [51]. The bottleneck is due to both the irregularity of the H-matrix, resulting in an irregular pattern of communication between nodes, as well as the dependency of the H-matrix on the amount of error protection and message block size. The dependency of the communication pattern on the particular encoding situation implies that any chip designed to decode a wide range of messages must support a flexible communication infrastructure, allowing

proper communication between nodes for each encoding supported. Consequently, a NoC communication infrastructure, where the communication between nodes is handled by a packet-based network [55] is very appropriate for implementing this application. The use of the NoC design paradigm allows for a high-performance interconnect capable of supporting arbitrary communication patterns, alleviating the traditional bottleneck to efficient decoding. To reduce the number of processing elements (PEs) required in the network, instead of representing each node in the bipartite graph as a separate PE, each PE is capable of acting as a number of nodes in the bipartite graph. This increases the storage requirement of each PE slightly, but allows the computational hardware to be shared between multiple nodes in the bipartite graph (hereafter referred to as virtual nodes). Figure 4.2 is an example of how a bipartite graph could be mapped into a NoC design.

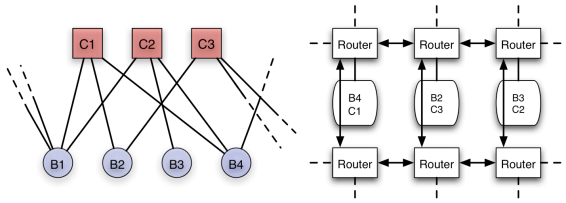


Figure 4.2. Sample Bipartite Graph and Placement Into NoC

In the traditional description of the computational work done by the bit and check nodes, the operations seem quite separate, implying that separate hardware is required to implement each node. In a traditional number representation, LDPC decoding consists of a significant number of multiplications and divisions, resulting in a slow, energy expensive process. Therefore, it is common to perform operations in the logarithmic domain, where the operation performed by the bit node is a simple summation, with each outgoing packet i to a virtual check node, being defined by the following relationship:

$$
\begin{aligned}
Bit\_Output_1 = & \\
& \left(\sum_{\substack{Associated \\ Checknodes}} \text{Check Output}_x \right) \\
& - \text{Check Output}_i \\
& + \text{Bit Output}_{i-1}
\end{aligned}
$$

Thus, the value sent by bit node x to each associated check node is dependent on the values sent by all the other check nodes associated with x. The check node operation is more complicated, requiring the computation of the bilinear transform (also in the log domain), as shown in the following equation, where B represents the bilinear transform:

$$\begin{aligned} &Check\_Output_i = \\ &\quad \text{The Bilinear Transform } B \text{ of} \\ &\quad \left(\sum_{\substack{Associated \\ Bitnodes}} B\left(\text{Bit Output}_x\right) \right) \\ &\quad -B\left(\text{Bit Output}_i\right) \end{aligned}$$

Again, the output to node i, is based on the values sent from all associated nodes other than node i. It is this bilinear transform performed by the check node that usually separates the nodes into distinct types when implemented in hardware. However, by using the distributive property of mathematics, we bring the outermost bilinear transformation from the check node operation into the bit node, resulting in the following:

$$\begin{aligned} &\text{Revised Bit Output}_i = \\ &\quad \sum_{\substack{Associated \\ Nodes}} B\left(\text{Revised Check Output}_x\right) \\ &\quad -B\left(\text{Revised Check Output}_i\right) \\ &\quad -\text{Bit Output}_{i-1} \end{aligned}$$

$$\begin{aligned} &\text{Revised Check Output}_i = \\ &\quad \sum_{\substack{Associated \\ Nodes}} B\left(\text{Revised Bit Output}_x\right) \\ &\quad -B\left(\text{Revised Bit Output}_i\right) \\ &\quad -0 \end{aligned}$$

Note that the two operations are now of the same form, and as such, the hardware required to support their computation is now functionally similar, allowing a single hardware computation unit to act as either type of node with little additional overhead.In our implementation of LDPC decoding on NoC, each PE is capable of performing the duties of 64 virtual nodes of any type. Because the communication pattern between nodes is determined by the H-matrix, each PE has a configuration memory that contains information about which of its virtual nodes communicate with what other virtual nodes, and the network addresses associated with each. This configuration memory can be written to by incoming network packets, allowing the configuration of the device and the mapping of virtual nodes onto physical PEs to be changed at runtime. In all, the configuration memory of each PE contains a total of 162 entries of information when fully loaded. The network uses a high-performance and low-latency wormhole routing, where each packet is able to be transferred one hop in each cycle. This is achieved through the use of both wide bit-width and low-voltage signaled links, as well as non-pipelined routers. Backpressure and congestion information is carried on dedicated wires between routers, and ensure that packets cannot be dropped from the network.Each packet is composed of two sections: a header, and a payload. The header information includes one bit of type information, indicating whether a packet contains configuration information or data for the computation. Addressing is done using 6 bits of physical network address, 6 bits of virtual destination address, and 6 bits of virtual source address. This allows for 64 physical addresses, each with a maximum of 64 virtual node identifiers, uniquely addressing bipartite graphs with up to 4096 nodes. The payload section is 32-bit wide and can carry two types of information: configuration data and data values. The configuration data is loaded directly into the configuration memory of the PE when received, using the 6-bit virtual address in the header and two additional bits in the payload that are unused for configuration data as indexes into the 162-entry configuration memory. As multiple on-chip routers are used on each chip, each router is designed to be as area-efficient as possible. As such, complicated adaptive routing algorithms, virtual channels, and large buffers are avoided in favor of deterministic X-Y routing and a single virtual channel [56, 57]. Buffers are limited to only 2 packets of storage due to space constraints. While in traditional networks such a design could lead to congestion and

head-of-line blocking problems, the small packet size allows each packet to require only one cycle of transfer time, reducing the congestion that can occur in wormhole networks with packets comprised of multiple flits. When the X-Y routing is used in a torus network, deadlock is avoided by only allowing edge nodes to route using the 'wraparound' links. For the small network sizes considered in our work, this results in only minor variation in network distances traveled as opposed to a fully toroidal routing, and ensures that virtual channels are not required.

4.2.2 Preventative Workload Migration

While previous research has shown that power is a poor indicator of local chip temperature, power dissipation is still the primary source of heat on a chip [11, 12, 13]. Numerous researchers have shown that by migrating workloads from one functional unit in a microprocessor to another, it is possible to balance the heat accumulated at each unit, and in this manner, reduce the localized heating indicative of hotspots [11, 14, 15]. Figure 4.3 shows an example of how this heat balancing works in practice.

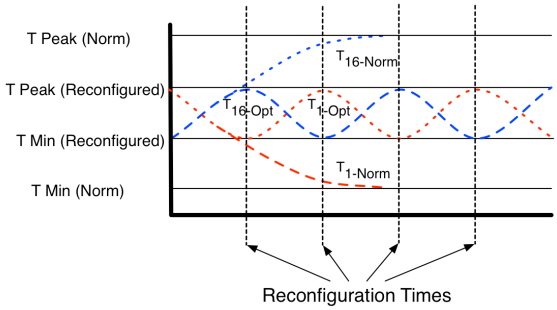


Figure 4.3. Thermal Characteristics of Systems with and without Migration

In Figure 4.3, the temperature of two functional units is shown as time progresses. In a system without hotspot avoidance (Tx-Norm), the temperature of the two points is vastly different, with functional unit 1 having a much higher temperature than functional unit 2. In a system with activity migration (T_x-Opt), the workload of the functional units is swapped periodically, resulting in balanced heat dissipation and a more even temperature profile. While functional unit 2 is, as a result, warmer, functional unit 1 has reduced in temperature significantly, reducing the possibility of operational failure due to thermal problems.

We extend this concept into the realm of a programmable NoC architecture, focusing on workload migration through dynamic reconfiguration. Figure 4.4 shows a sample temperature profile of a theoretical NoC chip, where PE 1 has a higher power dissipation than PE 16.

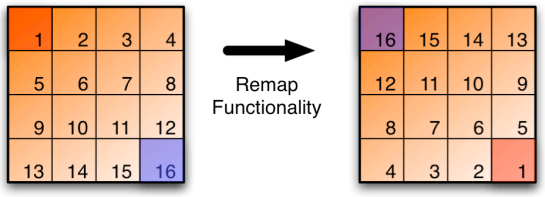


Figure 4.4. Sample NoC with Uneven Heat Distribution

As a result, significant heat has accumulated at the top left of the chip, resulting in high temperatures. By swapping the workloads of PEs 1 and 16, the heat generation in the chip will be more even, resulting in more even temperatures. However, random swapping is difficult to implement as there are various implementation issues that need to be considered. In particular, it would require knowledge of the relative power dissipation of all PEs in order to determine which are suitable for swapping, as well as fine-grained knowledge of the current temperature of the chip to determine when swapping is warranted. In addition, the movement of workload from one physical location (and thus network address) to another can result in misrouted packets if the other PEs in the network are unaware of the move. Even if they are made aware by an information broadcast, there is still the possibility of packets already in the network at the time of reconfiguration being misrouted. As such, any migration system in an NoC domain must provide a means to migrate without loss of information, and should do so with minimal performance and area cost. In addition, there are other factors that must be considered when discussing such workload relocation as discussed in the rest of this section.

4.2.3 Time between Migrations

In particular, the determination of when to migrate is important, as reconfiguring too infrequently can result in large temperature swings, as show in Figure 4.5. If the time between migration is too long, the temperatures on the chip will eventually reach those found in non-migrating systems, totally eliminating the benefits

of migration. As migrating can require some number of cycles, migrating too frequently can result in unnecessary performance penalties, however.

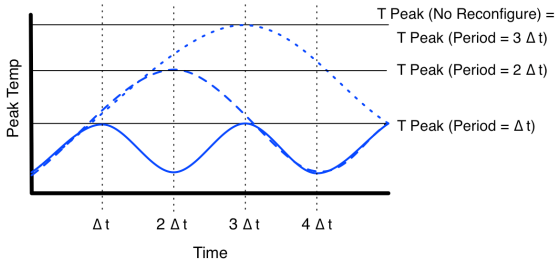


Figure 4.5. Effects of Migration on Peak Temperature

In practice, the time between migrations can be determined in one of two ways: proactively and reactively [11]. Proactive solutions migrate workloads periodically, preventing hotspots from ever forming. Our technique falls under this category. Reactive solutions use temperature or power information to migrate only when necessary, reducing the possible performance penalty due to unnecessary migration. The desired frequency of reconfiguration needs to balance the rate of temperature increase on the chip and the performance penalty for migration. The rate of temperature increase is influenced by a large number of factors, such as heat sink size, power density of the chip, ambient temperature, and thermal interface material. The performance penalty for migration is determined by the remapping technique used and the amount of data associated with the configuration.

When migration is performed, it is rarely possible to simply direct inputs from one location to another, as the original functional block will most often have some internal state information that is required for the computation. In our particular target application, computation is performed on each message block, after which all intermediate values become irrelevant, and only the configuration information in the PEs needs to be maintained. As each decoded message block leaves the network, a counter is incremented, and when the counter overflows, a migration is initiated. This allows the period between migrations to be varied easily, and requires only a small amount of hardware overhead. While our application needs no special support for state transfer during migration, other applications may require internal state information such as the internal values stored in registers to be transferred. Our architecture is general enough to support such a requirement and

can utilize the packet-based communication technique explained in the next subsection for moving such state information, albeit with appropriate modifications to the packet format, based on the amount of state data that needs to be transferred.

4.2.4 Techniques for Implementing Remapping

The first method of reconfiguration we discuss is that of maintaining multiple preloaded configuration memories on the chip. When migration is called for, reconfiguration is accomplished by multiplexing from one memory to another. This method of reconfiguration has very low performance overhead, requiring only one cycle to switch between configurations. This fast reconfiguration does come at a significant area penalty, however. Including a single additional configuration results in the area of each PE increasing by 0.65 sq. mm., for a total area increase of 10.4 sq. mm. in a 4x4 device.

As the additional chip area required to store additional copies of the configuration information is significant, it would seem more cost-effective to store the additional configuration information off-chip, and simply re-load the configuration tables from the external source, as is done at boot-time. While this requires no additional logic on-chip, it carries a significant performance penalty due to the limited I/O bandwidth available into the chip.

To minimize the reconfiguration delay penalty, it is possible to use the significant on-chip bandwidth to distribute configuration quickly, but to do so, the remapped configuration data must be available on the chip already. It must be observed that remapping not only involves moving the configuration from one PE to another but also involves updating the connectivity information contained in this configuration stream to reflect on the new remapped locations (refer to section 2 for configuration explanation). The new connectivity information can be obtained on-chip from existing configuration information in certain situations where the migration to be performed can be algebraically determined from the current configuration information. In such a situation, it is possible to introduce a limited amount of hardware to each PE, allowing it to examine its own configuration information and process it with the desired migration pattern, with the result being the new configuration information. This information can then be sent to the

proper recipient PE using the on-chip network. There are some constraints on this procedure that must be met to ensure its feasibility, the most notable being that of possible information loss due to overwriting. This situation can occur when PE X determines that it has configuration information to send to location Y in PE Z. If this configuration information reaches PE Z and is loaded into Z's configuration memory before it has parsed the previous configuration information stored there, it results in an irrecoverable loss of information about the previous state. Such a loss of state can prevent the new configuration from being calculated correctly, resulting in a failure. Such a situation can be avoided through a number of means, only two of which are discussed here. A trivial solution is to provide input buffering for incoming configuration packets, allowing them to write into the configuration memory only when the information in that location has already been analyzed by the migration algorithm, and as such, is no longer necessary. Such a solution requires buffer space similar to that discussed in previous reconfiguration methods, and as such, is undesirable.

The second and more desirable procedure that we employ allows PEs to transform and forward their own configuration information (depicted as a table in Figure 4.6) as long as the operation is performed in a well-ordered fashion. The transformed information is routed to the new workload destination determined by the remapping choice. The new destination serves as the header and the transformed configuration is the payload in the packet-based communication. Consequently, this approach utilizes the underlying packet-based communication for reconfiguration.

As long as the configuration information in each PE is processed in the same order, it is possible to begin the migration algorithm at all PE's by transforming each PE's configuration information and sending it to the proper destination. When the destination PE begins receiving new configuration information, it has already transformed its own configuration information and sent it out into the network, ensuring that overwriting cannot occur. In all situations, if location X is to be written into, that location must already have been parsed, and the resulting information also fed into the network. This particular method of reconfiguration transforms the difficulty of reconfiguration from one of hardware to the domain of the mapping algorithm, as if all PEs begin this process simultaneously, only 162

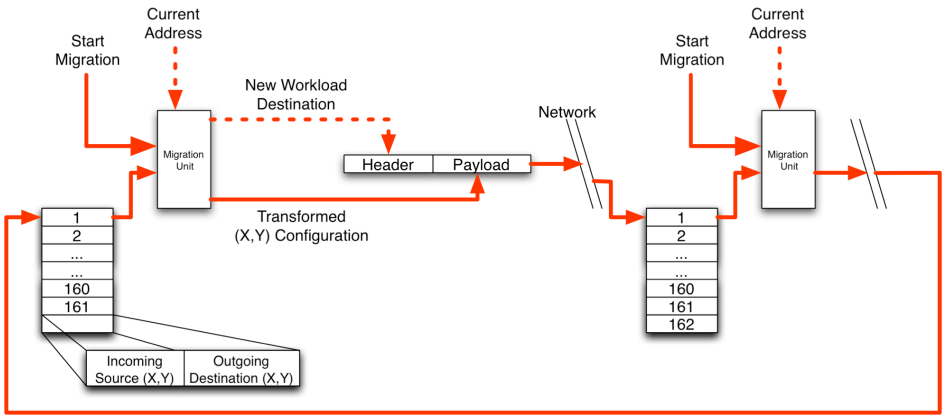


Figure 4.6. Migration Operation and Hardware

cycles (note that there are 162 entries in the configuration memory of a PE) are required to send all information into the network. This method does not require that all PEs begin the migration process simultaneously, only that all PEs involved in a particular chain of migration participate, preventing overwriting cases. The actual reconfiguration time can be higher in practice due to the possibility of network congestion caused by such a large influx of packets simultaneously. This congestion is dependent on the interaction of the network routing algorithm and the migration algorithm, and as such, exact cycle times are discussed later in the context of the particular migration algorithm.

4.2.5 Migration Schemes

A remapping scheme that itself consumes a significant amount of power is self-limiting. Hence, we seek to keep energy consumption low by limiting implementation complexity of the migration algorithms considered. Our avenue of investigation is therefore based on a logical model that ensures all new position of the workloads can be algebraically determined from the current position information and that the workloads will retain the same relative position to each other. Intuitively, we can see that there are only a limited number of ways of placing the logical functionality onto the chip, even with a perfectly homogenous chip. If we abstract this relative positioning requirement into a theoretical plane in which all workloads are statically placed, we can see that all possible migrations must oper-

ate on the plane as a whole, rather than on the workloads themselves. In practice, there are a total of three ways of adjusting a plane that, when combined, define all possible operations that we are interested in. These three operations are rotation, mirroring, and translational shifting. Therefore, we only consider each of these three operations as migration functions separately.

4.2.5.1 Rotation

The operation of rotation can, in theory, be performed in any magnitude, from 0 degrees to 360 degrees. In practice, we must map the logical functionality of a device back onto the device, restricting the rotation to certain magnitudes. In our chip, the minimum allowable magnitude of rotation is 90 degrees, due both to the topology of our network, as well as the square form factor of the chip. In a chip with a rectangular collection of PEs, rotation is limited to only 180-degree increments. The rotation operation is effective at distributing heat sources near the edge of the network, but is less effective with heat sources near the center of the chip. This is especially problematic for networks of odd-dimension (e.g., 5x5), as the center of the chip remains perfectly stationary during all configurations, and as such, can cause a stagnant hotspot. Figure 4.7 shows the four configurations possible with a rotational mapping on a 4x4 mesh.

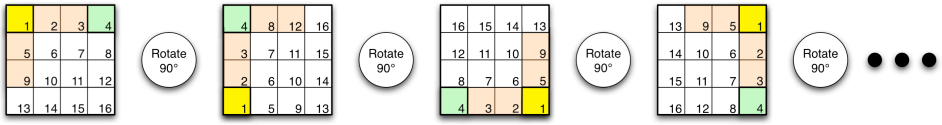


Figure 4.7. Configurations Resulting from Rotational Migration

The network congestion during configuration migration is non-trivial, as the X-Y routing algorithm maps multiple source-destination pairs onto the same link, resulting in collisions. To prevent this network congestion, the migration is performed in phases, with PEs sending out configuration information based on their location in the network. If the groupings are properly determined, no network links are utilized by more than one source-destination pair, resulting in congestion-free operation, and the time to perform a reconfiguration is deterministic. For the rotational migration function, congestion-free operation is ensured, as the groups are composed of chains of PEs as in Figure 4.8.

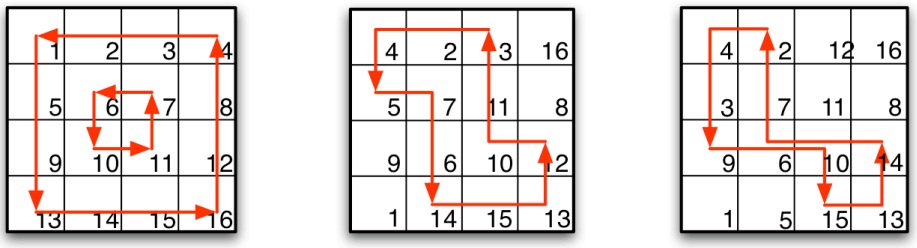


Figure 4.8. Congestion-Free Communication Pattern for Rotational Migration

In the 4x4 test chip, therefore, reconfiguration requires 501 cycles, including both the time to place the data on the network, and the network travel time. This increases to 668 cycles in the 5x5 network case, as the number of phases required increases with the size of the network. It must be observed that this transformation ensures that the relative distances between the different nodes remain the same after remapping and consequently distances between communicating nodes using the XY routing approach do not change. Hence, the performance variation across the different remapped schemes is not an issue in this case.

4.2.5.2 Mirroring

Mirroring can be performed in both the X and Y-axes of the chip, and can result in a total of four configurations, just as the rotational function discussed earlier. Figures 4.9 and 4.10 show differing ways of applying the mirroring function and the resulting configurations in a 4x4 mesh.

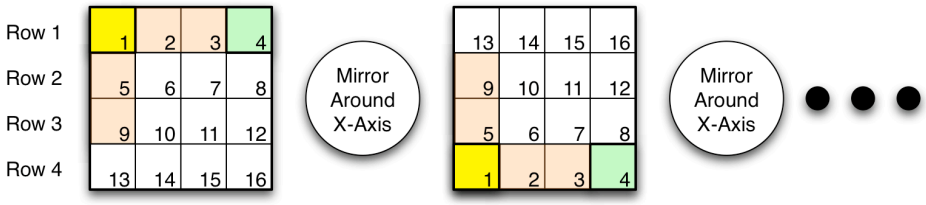


Figure 4.9. Configurations Possible with X-Axis Mirroring

While mirroring around the Y-axis alone is possible, it is very similar to X-axis mirroring in terms of performance, and as such, is omitted from this work. Unlike the rotational mapping, all configuration packets resulting from an X-mirroring

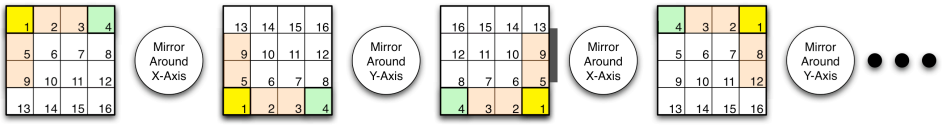


Figure 4.10. Configurations Possible by Mirroring in Both X and Y Axes

operation travel only in the Y dimension, limiting congestion to that caused by the packets from various rows interfering with each other. For example, row 1 (see Figure 4.9) wishes to relocate to the bottom of the mesh using the link between rows 2 and 3. Row 2 also wishes to use the same links, resulting in congestion. Congestion-free operation can be ensured by transmitting reconfiguration in groups, as in the rotational mapping. In this case, however, the groups are composed of pairs of rows (1 and 4 simultaneously, then 2 and 3 simultaneously). This requires only two sessions of transmission for both a 4x4 and a 5x5 network, as in both cases, only two pairs of rows must swap information. As such, reconfiguration requires only 332 cycles in the 4x4 chip, and 334 cycles in the 5x5 chip. As distances between communicating nodes employing the X-Y routing algorithm remain the same under a mirroring operation, the network performance is completely identical in any of the possible configurations, making this migration function ideal for applications with uniform timing requirements. Like the rotational function, this migration function is effective at distributing heat sources near the edge of the chip, but does little for heat sources near the center of the chip.

4.2.5.3 Translational Migration

While it would initially seem that shifting logical functionality horizontally or vertically would result in workloads being mapped off-chip, it is possible to consider translational migration in a toroidal manner, where any functionality mapped beyond the edge of the chip is simply shifted to the other edge in a circular manner. This form of migration is a superset of migrating into unused PEs at the edge of a device, as such a situation can be considered as subset of our problem by defining an initial mapping with unused units at the edges of the network. Unlike the previous migration functions, translational shifting is not limited to four possible configurations due to the square nature of our sample chip. Instead, the total number of possible configurations is defined by the X and Y dimensions of

the chip, with the total number of reconfigurations equal to product of the two dimensions. Initial results show that many of these reconfigurations do little to move heat-causing workloads far from their original location, and as such, are not effective at reducing hotspots. Based on this preliminary analysis, we limit our analysis to only shifts of 25% and 50% of the total dimension. Figures 4.11 and 4.12 show the translational migration algorithms considered.

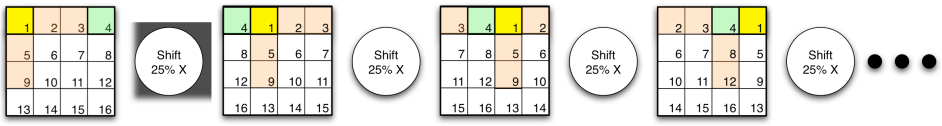


Figure 4.11. Translational Shifting by 25% in the X Direction

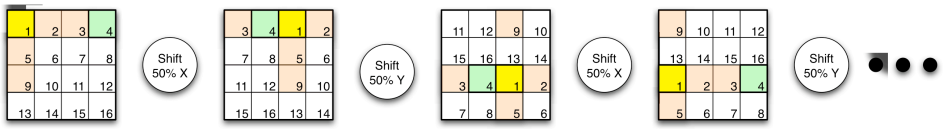


Figure 4.12. Translational Shifting by 50% in Both the X and Y Directions

Due to the similarities of Y-shifting and X-shifting, we omit the discussion of Y-shifting from the results. Like the mirroring operation, it is possible to perform reconfiguration in a congestion-free manner by performing the configuration in a column-wise or row-wise fashion in stages. As such, the total reconfiguration time at 332 cycles for the 4x4 case and 334 cycles for the 5x5 case. Unlike the mirroring configuration, however, the distance between communicating nodes using X-Y routing is not the same after a toroidal shift on a 2-dimensional mesh, and as such, the network traffic pattern can vary between the different remapped configurations, allowing for performance disparities during operation. Note that this difference occurs as this transformation maintains our relative distance constraint for the migration schemes only if we consider that there is an edge wrapping around the corners. In our test cases, these performance variations are small, with the throughput being lowered by less than one-tenth of one percent due to the traffic variation. However, this variation can be larger based on the communication pattern of the application and the size of the chip. This performance variation (and resulting penalty) can be minimized in a 2-dimensional torus network (in

Table 4.1. Algebraic Representation of the Proposed Migration Functions

Migration Pattern	New X Coordinate	New Y Coordinate
Rotation	N-1-Y	X
X Mirroring	N-1-X	Y
X Translation	(X+Offset) mod N	Y

contrast to our 2D mesh structure), as the network distance between source and destination remains more consistent due to the additional wrap-around channels.

4.2.5.4 Implementation of the Migration Functions

All of the proposed migration functions are mathematically quite simple, and require little hardware to properly implement. Each migration function takes as input the current X, Y location of the workload, and provides as output the new X, Y destination of the workload, and as such, in our architecture, operates on only 3-bit operands. Note that the on-chip routing employs an addressing scheme that uses two dimensional coordinates and should not be confused with the linear labeling used for depicting different workloads in Figures 7 through 12. Table 4.1 shows the functions implementing the different migrations, where N is the length of the array on a side.

The selection of these simple functions allows the migration unit to remain small, fast, and low power. More importantly, the simplicity and predictability of the migration functions presented allows for a simplified I/O interface to the outside of the chip, by transforming the destination address assigned to all incoming packets and transforming the source address of all packets leaving the chip. By including a migration unit at the I/O interface, the migration operation is totally transparent to the outside world.

Finally, we note that the same migration unit can perform all migration functions presented with only minor changes to the mathematical operations. As such, it is possible to dynamically alter which migration function is being used at runtime, allowing specific migrations to be performed for specific workload characteristics. We leave the investigation of such dynamic altering of the migration function for future work.

4.2.6 Experimental Platform

Our experimental platform is based on the HotSpot thermal library available from the University of Virginia [13]. The HotSpot library can be used to calculate the temperature distribution of a silicon chip and its packaging by using RC models of heat transfer. The RC model is developed from an input floorplan file, which details where each functional block is located on the chip, as well as the size of each block. Once the RC model has been developed, the power consumption of each functional block can be used to determine both steady state and transient temperature distributions. The HotSpot tool was left with all settings at the default values in order to facilitate easier comparison of our work with others. As such, the heatsink on the chip is 60mm on a side, and 6.9mm thick, and the ambient temperature is 40 degrees centigrade.

Our floorplans were taken directly from the layout of our sample chips, with each functional unit consisting of a single PE and its associated router. Two test chips were synthesized and placed and routed using a commercial 160nm standard cell library. The first uses a 4x4 architecture with 16 PEs, while the second utilizes a 5x5 architecture with 25 PEs. The chip can be clocked at up to 500MHz, and for a code with 1 check bit per 3 encoded bits and an encoded message size of 1024 bits, can decode 1.2 Gb of encoded data per second. During operation, the total chip power consumption varies between 33W and 38W at max load, dependent on configuration. In the 160nm technology our test chip was designed in, leakage power is small, and as such, we do not include temperature-dependent effects on leakage currents in our power model. The inclusion of temperature-dependent effects in technologies where leakage power is more substantial is likely to accentuate temperature differences. Consequently, we anticipate our technique to be more beneficial in this case. Overall, each functional unit has an area of 4.36 sq. mm, and the larger of the two test chips (with 25 functional units) has an area of 1.09 sq. cm. Note that our model does not include I/O pads, only the functional units themselves. The size of the functional units used in the tool compares favorably with the functional units used in the original HotSpot work, where units ranged in size from roughly 1 sq. mm to 10 sq. mm.

Power consumption of each unit is determined through the use of Synopsys' Power Compiler. A modified cycle-accurate simulator [58] is then run with an

encoded message to obtain switching rates for the components in the chip during the decoding of a message. These switching rates are used as input to the Power Compiler, ensuring that the power consumption values obtained are accurate. This is used to obtain the relationship between PE power consumption and the type and number of virtual nodes mapped into a given PE. Router power consumption is also determined using Power Compiler, but in this case, it is determined for a number of operating states, such as 'forwarding one packet', 'idle', and 'forwarding two packets'. These values are fed back into the cycle-accurate simulator, which then calculates the average power consumption of each router during the decoding of a message. Note that our simulations also include the energy consumed during the migration operation to more accurately evaluate the utility of our proposed method. The power consumed in the interconnect is lumped in the routers in our evaluation as the heating due to interconnect is not modeled more precisely in hotspot. However, due to the regular structure of the interconnect used in our design, we do not anticipate this to influence our results.

This method of calculating power consumption of units provides a high resolution method of estimating power consumption, giving power consumption values at a sampling rate of 0.854 microseconds/sample. This compares favorably with the sampling resolution used in the original HotSpot work, where temperature distributions were calculated every 3.33 microseconds.

To more fully explore the design space for reconfiguration, we evaluate multiple configurations of our test chips. In particular, the 4x4 chip is evaluated with two different H matrices (referred to as A and B), while the 5x5 chip is evaluated with three different H matrices (C, D, E). The five different matrices are based on different irregular LDPC codes, and decode message blocks with 384 bits of error protection and 576 bits of decoded data. An irregular LDPC code is one in which not all bit nodes communicate with the same number of check nodes, and not all check nodes communicate with the same number of bit nodes. We choose irregular codes because they have been shown to perform better in the presence of additive white Gaussian noise [59]. As our device decodes two message blocks in parallel, this allows the decoding of up to 1920 bits at a time.

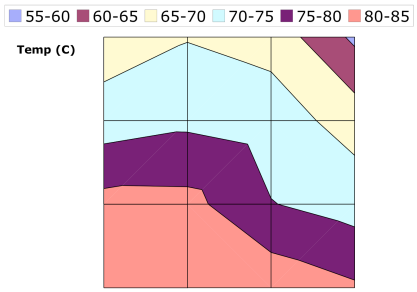
The irregularity of the codes and the variations in number of nodes mapped to a single PE are main reasons for the spatial variations in the power consumption.

Consequently, it is important to start with an optimized mapping that attempts to minimize these imbalances. Optimizations of mapping of the bit and check nodes into physical nodes have been directed primarily on minimizing the communication distances between communicating nodes [60]. We have modified this approach in our prior work to produce a thermally-aware mapping using a genetic algorithm, resulting in a more even heat distribution throughout the chip and lower peak temperatures. This thermal-aware mapping provides around 10-15 C reduction in peak temperatures for our test cases as compared to communication-oriented mapping. For example, for one of our test cases (case D), the communication-oriented mapping has a peak temperature of 84 C. Moving groups of virtual nodes as a whole starting with a communication oriented mapping in a temperature-aware fashion lowers the peak temperature to 74 C. In this case, the functionality mapped on one PE is just moved to another. Moving virtual nodes independently of each other reduces peak temperature to 72.8 C. In this case, partial portions of functionalities mapped on one PE can be moved to another to increase flexibility. To more clearly show the benefits of dynamic reconfiguration, all tests performed in this work start with an initial mapping that has been statically optimized for reducing peak temperature. [61]

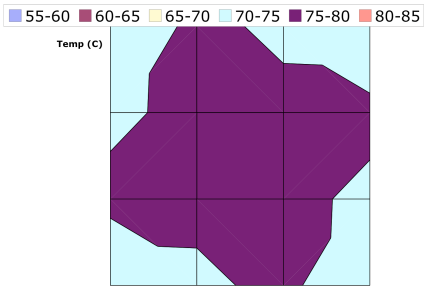
4.2.7 Effectiveness of Workload Migration

We begin our analysis by comparing the relative effectiveness of the different migrations at reducing the peak temperature of the test chips. Figure 4.13(a) shows the temperature distribution of configuration B in the initial, thermal-aware configuration, while Figures 4.13(b) through 4.13(f) show the temperature distribution of the same chip with different migration techniques used. All distributions in Figure 4.13 use the same color scheme to represent the same temperatures for easy comparison.

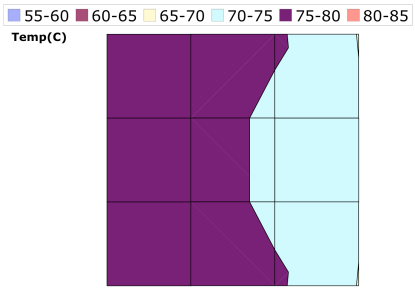
More important than an even temperature distribution however, is the peak temperature reduction. Figure 4.14 shows the reduction in peak temperature in the various circuit configurations with different migration techniques. For circuit configurations A and B, the rotational and X-Y mirroring migrations reduce the peak temperature the most, while for the larger configurations, shifting reduces



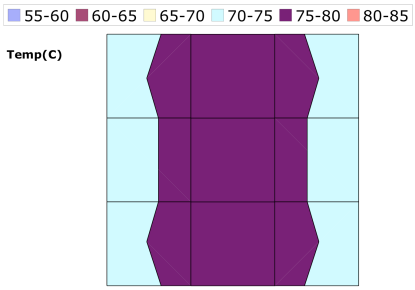
(a) Base Configuration



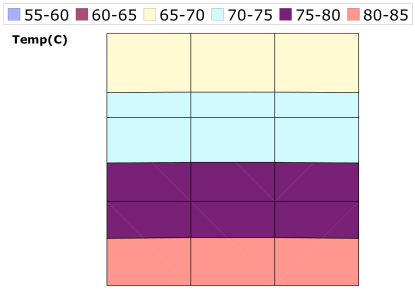
(b) Rotational Migration



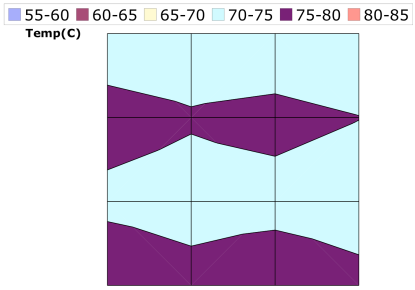
(c) X-Mirroring



(d) X and Y Mirroring



(e) X Translation



(f) X and Y Translation

Figure 4.13. Temperature Distributions Using Various Migrations on Configuration B. The grids shown in the figures do not correspond to PE granularities.

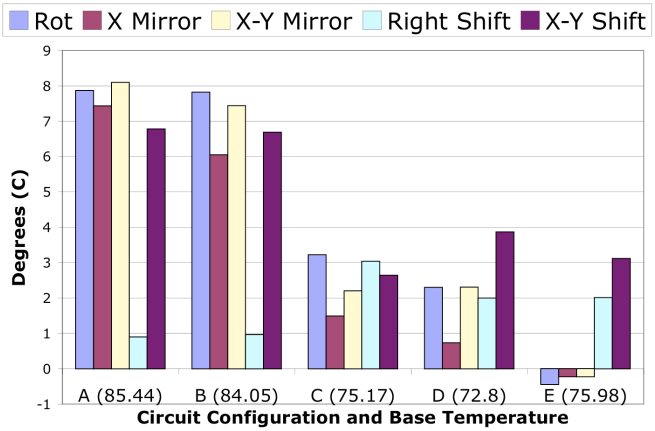


Figure 4.14. Reduction in Peak Temperature Due to Workload Migration

hotspot temperatures the most. This difference in efficacy is due, in large part, to the even dimensionality (4x4 array) of test cases A and B, as opposed to the odd dimensionality (5x5 array) of test cases C, D, and E. In the odd-dimensioned test cases, both the rotational and mirroring migration functions ignore the central PE, and as such, they are unable to balance the heat generated at the center of the device. The poor behavior of the right shift is due to the relative power output of the rows in the various test cases. In all test cases, one of the rows had a significantly higher power output than the remaining rows, generating a 'warm band' that right shifting alone is unable to distribute. While such a 'warm band' might seem to skew our results, remember that a thermally-aware placement algorithm was used to generate our initial test cases, and as such, it is reasonable to assume that such characteristics would be even more common in non-thermally-aware placements.

Among the configurations tested, X-Y shifting has the highest average temperature reduction, 4.62 C. Rotational migration has the second highest average temperature reduction, 4.15 C, but actually results in higher peak temperatures for configuration E for two reasons. The first reason, common to both the rotational and mirroring migrations, is that the hotspots in configuration E are near the center of the chip, where those algorithms are least efficient at migrating workload

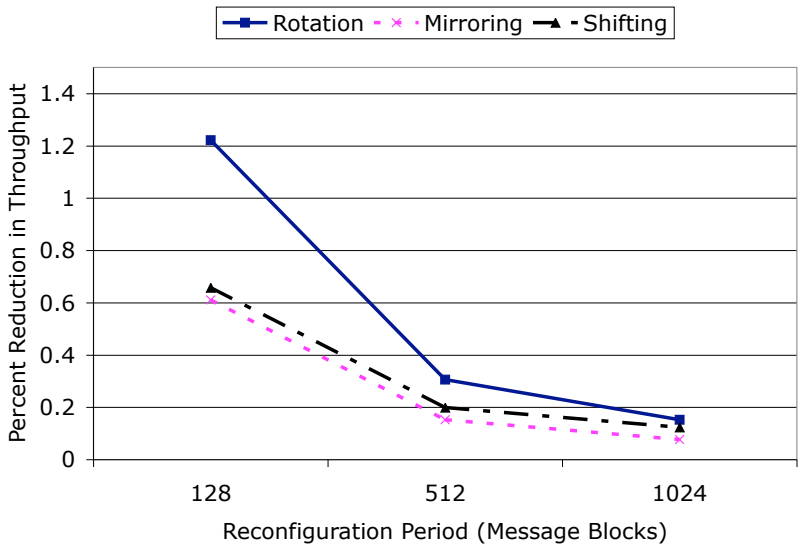


Figure 4.15. Performance Loss Due to Reconfiguration Overhead

away from the hotspot. Second, the rotational migration has the largest energy penalty for performing reconfiguration, resulting an increase in average chip temperature of 0.3 C. All of the above simulations were performed with a migration period of 128 message blocks, or every 109 microseconds (each message block requires 0.854 microseconds). Higher frequencies of migration result in a more even temperature distribution, but do so at the cost of performance. Figure 4.15 shows that the performance loss when using the various migration algorithms is minimal for migration periods equal to, or greater than 109 microseconds. As using higher periods of migration is obviously preferable, Figure 4.16 shows the increase in peak temperature when using higher periods of migration.

For a reconfiguration period of 512 message blocks (437.2 microseconds), the performance penalty drops to less than 0.4%, and the peak temperatures rise less than a tenth of a degree in the additional time between migrations. Consequently we can increase the period between reconfigurations to 1024 message blocks and reduce the throughput penalty to less than 0.2% without significant impact on peak temperature.

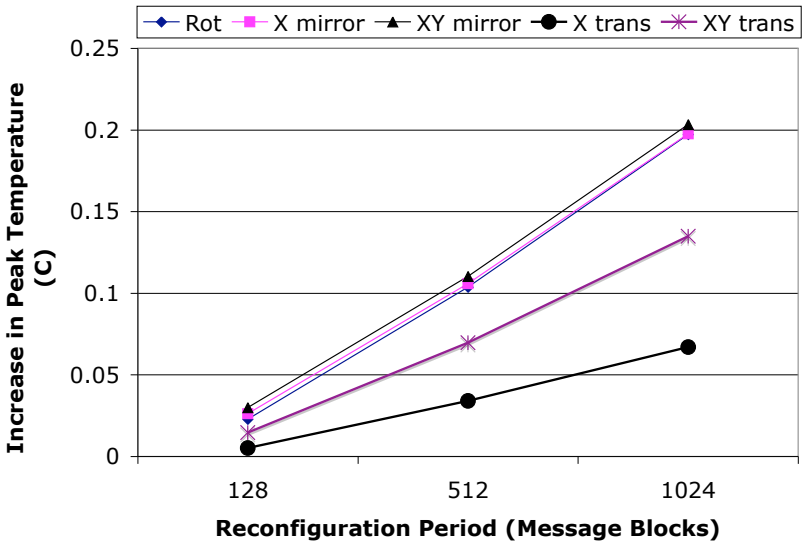


Figure 4.16. Increase in Peak Temperature Due to Longer Migration Periods

As technology scales, the power density of chips is expected to increase, resulting in higher operating temperatures and more significant hotspot problems. To model this behavior, we alter our initial floorplans to reduce the chip area by linear factors of 0.7 to represent technology scaling. Power dissipation is left at the original value from the 160nm technology node, resulting in an increased power density of roughly a factor of 2 per generation. Figure 17 shows the effectiveness of workload migration on configuration A at two such scaled technology nodes.

While the peak temperatures shown in this figure are quite large due to the static heatsink configuration, we notice that as power density (and thus hotspot temperature) increases, the benefit gained from workload migration increases. For example, while rotation reduces peak temperature by 7.87 C with current power densities, as power densities double, the peak temperature reduction increases to 13.92 C.

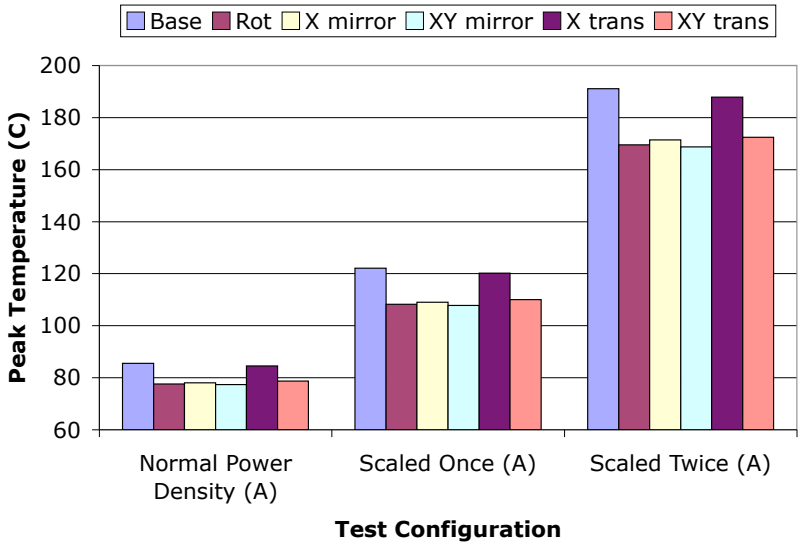


Figure 4.17. Effects of Workload Migration as Power Density Increases

4.3 Integer Computation Offloading to Floating Point Units

While workload migration is easy to conceptualize for homogenous designs, such as the network on chip discussed in Section 4.2.1, it is more difficult to envision workload migration as an effective solution for hardware-specific operations on heterogeneous architectures, such as a the functional units in uniprocessor system.

In such a uniprocessor system, the core is composed of a number of functional units, each often tailored to serve a specific purpose, such as storing the success of previous branches, ordering and issuing instructions, or performing the addition of two operands. Previous approaches to workload migration in such systems have relied on the availability of identical, sometimes redundant hardware to provide alternative locations for computation to occur. This 'alternate' hardware can be as large as an entire copy of all the integer unit, or as small as a single integer ALU. Both approaches are limiting, however, in that in the case of large hardware units,

a large amount of additional die area is used to increase the area in which power is dissipated, rather than providing additional functionality. This results in an increase in die cost, without a compensating increase in performance. Alternatively, the use of small amounts of redundant hardware reduces the area in which power is dissipated, and has little effect on the overall thermal profile of the device. In order for workload migration to be effective without a significant allocation of die area for redundant hardware, architects must seek to exploit resources already on die for the purposes of migration, or find means of exploit redundant hardware for increased performance.

As an example of such intelligent unit re-use, the author notes that it is possible, in some cases, for integer operations to be carried out on floating-point hardware, reducing the power consumed by the integer unit, and distributing power consumption across both units. There are, of course, a number of details that must be accounted for, such as the possibility of additional latency being added to the system as the operands travel to and from the floating-point unit (*FPU*), and the control overhead required to schedule integer operations (*iops*) on the FPU without conflicting with traditional floating-point operations. We discuss these details, and our proposed solutions, in the next subsection.

4.3.1 Architectural Changes

As modern processors are expected to execute floating-point operations as well as integer operations, most commercial processors have a number of floating-point ALUs available on chip that remain idle during purely integer sections of code. The floating-point format that these units traditionally support is the IEEE 754-1985 standard [62] for binary floating-point arithmetic, which specifies both 32 and 64-bit floating point formats. These formats are very similar, differing in the length of the exponent and mantissa fields only. While these formats each have special cases, such as NaN, Infinity, and Denormalized numbers, a general floating-point addition is performed by first 'aligning' the mantissas, based on the value of the exponents, performing mantissa addition, then finishing by realigning the mantissa and setting the resulting exponent.

We assume a unified instruction scheduler that issues instructions to both inte-

ger and floating-point functional units. In a value-captured scheduler, the results along with tags are broadcasted back to the instruction window at writeback stage. This unified scheduler architecture, by not differentiating between iops and fpops, prevents complications from arising due to differences in the broadcast network.

Since modern microprocessors already have their floating-point units supporting multimedia operations in a SIMD fashion [63], the hardware modifications required to support FUS are minimal. We propose that it is possible to modify the mantissa adder to support full-size integer ALU operations and bypass all unnecessary blocks such as mantissa alignment, normalization, and exponent adjustment, during integer operations. This requires very little additional hardware, with only one multiplexor placed before the mantissa adder unit, and additional load placed on the mantissa adder's output drivers. Figure 4.18 shows a block diagram of a generic floating-point unit, where the shaded blocks are bypassed during integer operation. Once an iop is issued to a floating-point unit, it receives its operands

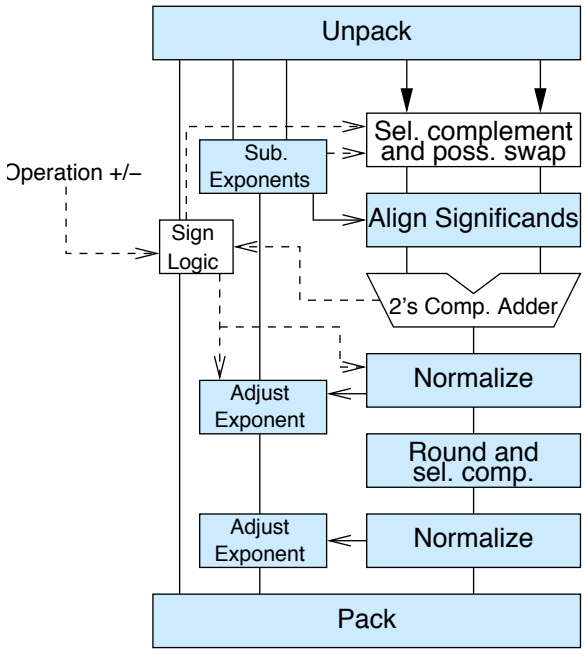


Figure 4.18. Generic floating-point unit, where shaded blocks are able to be bypassed during integer operation

from one of two sources. It either has its source operands ready (stored in its RUU entry) while waiting in the issue queue or obtains operands from the bypassing network. In the first case, the opcode and source operand values are read out from

RUU entry and sent to the functional unit simultaneously. In the second case, the instruction was issued before the operands were totally available, and as such, the instruction is expected to receive its operands through the bypass network. This requires adding bypass logic from the integer units to the floating point unit. There is no additional hardware required to broadcast the result of the shunted operation, however, as with the unified scheduler discussed earlier, all floating-point and integer operations already have their results broadcast across the same network.

As the goal of integer offloading to the FPU is to distribute heat, it is anticipated that the on-chip distance to the FPU will be non-negligible. This additional distance will result in increased latency when using the FPU as opposed to the IntALU. As the critical path in modern processors is rarely the execution unit itself, but instead the scheduling and issue logic, there is a high likelihood that there is already additional timing slack in the design to allow for some of this additional distance. In general, however, one cannot assume that such slack exists, and so our preliminary results assume an additional one cycle latency. In the future, we hope to expand our results to include a number of latencies. While this latency might normally be considered a significant performance problem, as multi-cycle ALUs normally prevent back-to-back issuing of dependent instructions, we note that the latency in this situation is not computational latency, and instead communication latency. As such, it is possible to execute dependent instructions back-to-back, as long as both are issued to the FPU, and are thus physically close enough to receive data without incurring transmission latency. The actual development of scheduling hardware that is capable of this differentiation is outside the scope of this dissertation, however. Therefore, our simulation results do note take this possible intelligent scheduling into account at this point, and we simply note that the performance penalties shown here could, in practice, be much smaller.

4.3.2 Effectiveness of Integer Offloading

In this section, we discuss the experimental platform being developed for the simulation and some preliminary results obtained through related work. Our simulator is derived from SimpleScalar V3.0 [9], an execution-driven simulator for out-of-order superscalar processors. In our implementation, the physical register file is

Parameters	Value
Processor Core	
RUU size	128 entries
Load/Store Queue (LSQ) size	64 entries
Physical Register File	128 registers
Fetch/Decode/Issue/Commit Width	8 instructions per cycle
Function Units	4 IALU, 2 IMULT/IDIV, 2 FALU, 1 FMULT/FDIV/FSQRT, 2 Mem Read/Write ports
Branch Predictor	
Branch Predictor	combined predictor with 4K meta-table, a bimodal predictor with 4K table, and a 2-level gshare predictor with 12-bit history
BTB	2048 entries, 4-way
RAS	32-entry
Memory Hierarchy	
L1 ICache	64KB, 2 ways, 32B blocks, 2 cycle latency
L1 DCache	64KB, 2 ways, 32B blocks, 2 cycle latency
L2 UCache	1MB, 4 ways, 64B blocks, 12 cycle latency
Memory	200 cycles first chunk, 12 cycles rest
TLB	4 way, ITLB 64 entry, DTLB 128 entry, 30 cycle miss penalty

Table 4.2. Parameters for the simulated baseline superscalar processor

separated from the register update unit (RUU) structure. The baseline superscalar processor is configured with parameters given in Table 4.2, which model a wide-issue high-performance superscalar processor.

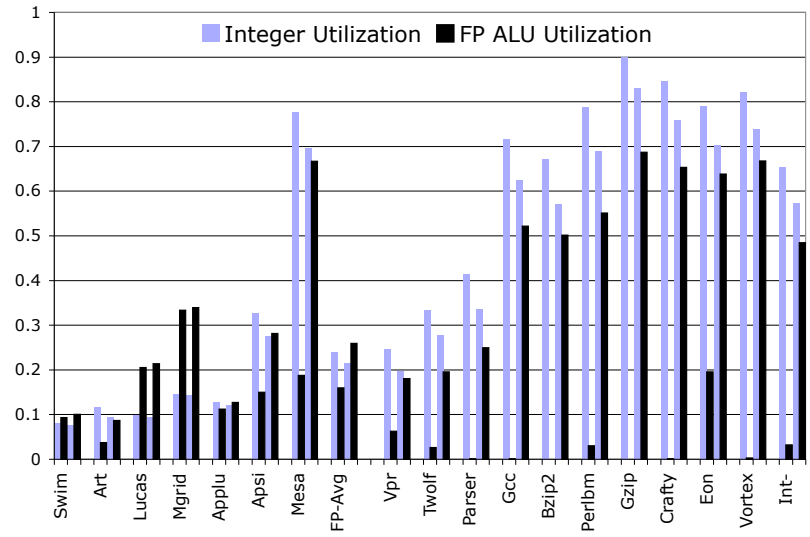
This processor configuration, while very high-performance, is significantly wider than modern processors, and as such, tends to have lower utilization of the functional units as compared to more narrow designs. As such, while the preliminary results discussed in this work are obtained with the processor configuration shown in Table 4.2, in future experiments, a more narrow processor architecture is expected to be used.

For experimental evaluation, a representative subset of SPEC2000 suite was used, including 10 integer benchmarks and 7 floating-point benchmarks. The SPEC2000 benchmarks were compiled for Alpha instruction set architecture with "peak" tuning. We use the reference input sets for this study. Each benchmark is first fast-forwarded to its early single simulation point specified by SimPoint [64]. The last 100 million instructions during the fast-forwarding phase are used to warm-up the caches if the number of skipped instructions is more than 100 million.

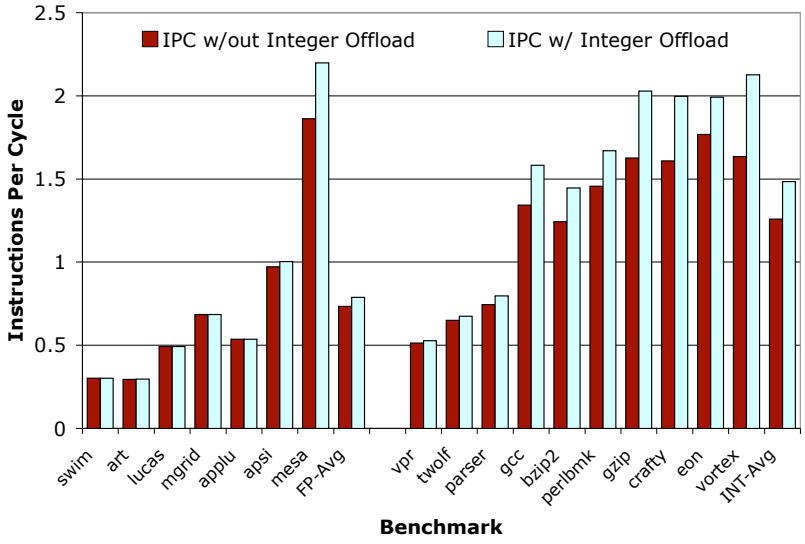
Then, the next 100 million instructions are simulated in detail, for the purposes of data collection.

In the preliminary analysis, the suitability and possible performance penalty due to integer offloading is first evaluated. For integer offloading to be possible without significantly affecting performance, two conditions must be met. First, the FPU must be sufficiently underutilized such that there are ample free cycles to execute the integer instructions without impacting the normal execution of the floating point instructions. Second, the additional latency imposed by the signal transmission time must not significantly impact the critical path of the program. The first of these conditions is demonstrated in Figure 4.19(a). Here, the utilization of each group of functional units is shown, with the darker colored segments representing unused potential, and the white segments representing utilized functional units. Figure 4.19(a) shows that in general, even in floating-point benchmarks, the floating-point ALU utilization is below 50%, indicating that on average, one iop can be sent to the FPU per cycle without impacting performance significantly. Furthermore, Figure 4.19(b) also shows that the average IPC of the system doesn't decrease when exposing the additional iALU bandwidth, rather, the additional number of effective iALUs results in performance increases in certain situations.

Given that integer offloading to the FPU doesn't seem to negatively affect performance, and that in some cases, it can result in increased performance, the amount of workload migrated was next characterized. Figure 4.19(a) shows that the average number of idle iALU units per cycle increases by *6.3* percent, on average, implying that the iALU region of the chip would consume, on average, *6.3* less power. At the same time, the usage of utilized fALU units increases by a similar amount. This, unlike the integer case, does not result in a *YYY* percent increase in power consumption for this region of the chip, as the additional operations being performed in the FPU require much less energy than the traditional floating-point operation. More usefully, in the case of integer-only benchmarks, which would be most likely to result in hotspots near the integer unit, the workload decreases 42.7%, hopefully resulting in a significant reduction in hotspot activity in the most problematic applications. Further analysis remains, however, as the thermal impact of this workload migration has yet to be estimated.



(a) Utilization of functional units. The left two columns show the utilization in the unmodified case, and the right two columns show the utilization with integer offloading.



(b) IPC of Spec Benchmarks

Figure 4.19. Preliminary results using integer offloading

Chapter 5

Effects of Temperature on Timing Closure

While the effects of temperature on semiconductor devices have been studied heavily over the years, much of this research has focused on the long-term effects and causes of failure, such as electromigration. This research has led to a number of detailed models of the effects of temperature on circuit performance, with means of modeling the changes in threshold voltage, carrier mobility, and capacitance, as temperature varies. These models currently see heavy use by those involved in process development and more specifically, in the definition of "process corners", which characterize the performance of a process under various corner cases. These corner cases are delineated by their timing characteristics, resulting in a 'slow' case and a 'fast' case. Circuit designers perform timing analysis at both corners, as well as in a 'nominal' case, in an attempt to verify that the hardware will satisfy timing constraints throughout the range of operation. These efforts, however, often assume a static temperature distribution across the chip. Even when designing units that span large regions of the device, such as the clock network, the traditional method of analysis is to develop static corner cases that predict the temperature distribution on the chip.

In the past, such static corner cases were simple to develop, with the coldest case involving a chip, just having turned on, starting from a cold temperature. This results in a nearly uniform and cool distribution. The hottest case involves the entire chip operating at or near the thermal ceiling, again, with a nearly ho-

mogenous distribution. The nominal case required estimating the temperature profile of the device when in standard operation, resulting in a non-homogenous situation, but due to the relatively low variation in power distribution on most designs, only a single case was required.

In the future, however, processor architecture is shifting towards a multiprocessor-on-chip design style for a number of reasons. Such an architecture, while homogenizing the thermal distribution of the chip slightly by doubling the number of possible hotspot locations, also presents new thermal problems. Unlike single-core designs, which have a relatively predictable thermal profile, with the core hotter than the cache, multi-core designs present a situation where the temperature gradient on the die can shift dramatically as one core receives workload, then the other. This alternating temperature gradient can present new problems with static timing closure, as the signal characteristics on the chip are no longer consistent in any one direction. In such a situation, traditional corner-case design would require that the device operate at a reduced frequency due to the thermal-induced signal skew. This chapter discusses the timing impact of temperature, and how it can impact static timing analysis. A method of dynamically adapting signal drivers to minimize the impact of timing gradients is proposed. In the future, this chapter will characterize the timing gradients, using both an analytical and a numerical model, and determine the effectiveness of dynamically adaptive drivers as a means of minimizing the timing gradients.

5.1 How Temperature Affects Timing

While many previous works have discussed the impact of temperature on one aspect of a design, or another, to the best of this author's knowledge, no work has yet detailed a comprehensive model of the fine-grained impact of temperature on the propagation delay of a circuit. There have been numerous works discussing one aspect or another, such as the impact of temperature on threshold voltage, and hence performance, or on the transmission properties of wires. This lack of a comprehensive analysis presents a limitation to temperature-aware computing in general, as many of the temperature effects involved in the analysis are complementary, indicating that even of the direction of the overall effect can vary.

Therefore, this work presents a complete model, including transistor and wiring effects, in the following subsections. As with much of the existing body of work on circuit analysis, we consider the case of the CMOS inverter.

5.1.1 Transistor Delay

The average propagation time of an inverter is composed of both the rise and fall times. As the CMOS inverter is composed of both a p-type transistor and an n-type transistor, we first discuss the n-type transistor and the fall time, and later extend the analysis to the p-type. Equation 5.1 shows the fall time is composed of two periods: that in which the output voltage drops from $0.9 * V_{dd}$ to $(V_{dd} - V_{tn})$, and that in which the output voltage drops from $(V_{dd} - V_{tn})$ to $0.1 * V_{dd}$ [65].

$$t_f = \begin{cases} t_{f_1} & (V_{dd} - V_{t_n}) \leq V_{out} \leq 0.9V_{dd} \\ t_{f_2} & 0.1V_{dd} \leq V_{out} \leq (V_{dd} - V_{t_n}) \end{cases} \quad (5.1)$$

$$t_{f_1} = 2\frac{C_L}{\beta_n (V_{dd} - V_{t_n})} \int_{V_{dd}-V_{t_n}}^{0.9V_{dd}} dV_0 = \frac{2C_L (V_{t_n} - 0.1V_{dd})}{\beta_n (V_{dd} - V_{t_n})} \quad (5.2)$$

$$t_{f_2} = \frac{C_L}{\beta_n (V_{dd} - V_{t_n})} \int_{0.1V_{dd}}^{V_{dd}-V_{t_n}} \frac{dV_0}{\frac{V_0^2}{2(V_{dd}-V_{t_n})} - V_0} = \frac{C_L}{\beta_n (V_{dd} - V_{t_n})} \ln\left(\frac{19V_{dd} - 20V_{t_n}}{V_{dd}}\right) \quad (5.3)$$

$$t_f = 2\frac{C_L}{\beta_n (V_{dd} - V_{t_n})} \times \left[\frac{V_{t_n} - 0.1V_{dd}}{V_{dd} - V_{t_n}} + \frac{1}{2} \ln\left(\frac{19V_{dd} - 20V_{t_n}}{V_{dd}}\right)\right] \quad (5.4)$$

Equation 5.4, the overall time for the output voltage to fall from $0.9V_{dd}$ to $0.1V_{dd}$, is dependent on both the mobility of the carrier, as well as the threshold voltage and the load capacitance. The mobility and threshold voltage dependence are dependent on temperature, as shown in equations 5.5 through 5.12, obtained from the BSIM 4.5 model [66].

$$\beta_n := \mu_n * \frac{\epsilon}{T_{OXE}} * \frac{W_n}{L_n} \quad (5.5)$$

$$\mu_n := \frac{U0_n}{1 + (UA + UC * V_{bseff}) \left(\frac{V_{gsteff} + C0_n * (VTH0_n - VFB - \phi_s)}{T_{OXE}}\right)^{EU_n}} \quad (5.6)$$

$$U0_n := U0Tnom_n * \left(\frac{T}{Tnom}\right)^{UTE} \quad (5.7)$$

$$U\{A, B, C, D\} := U\{A, B, C, D\}Tnom * (1 + U\{A, B, C, D\}1 * (T - Tnom)) \quad (5.8)$$

$$\phi_s := 0.4K_b\frac{T}{q}\ln\left(\frac{NDEP}{ni}\right) + PHIN \quad (5.9)$$

$$V_{th} := V_{th}@T_{nom} + \left(KT1 + \frac{KT1L}{L_{eff}} + KT2\cdot V_{bseff}\right)\left(\frac{T}{Tnom} - 1\right) \quad (5.10)$$

$$\begin{aligned} V_{th}@T_{nom} := & \\ & + \left(\frac{K1T_{OXE}\sqrt{\phi_s - V_{bseff}}}{TOXM} - K1\sqrt{\phi_s}\right)\sqrt{1 + \frac{LPEB}{L_{eff}}} \\ & -K2_{ox}V_{bseff} \\ & +K1_{ox}\left(\sqrt{1 + \frac{LPE0}{L_{eff}}} - 1\right)\phi_s \\ & +\frac{(K3 + K3B\cdot V_{bseff})\,T_{OXE}\;\phi_s}{W'_{eff} + W0} \\ & -\frac{1}{2}\left(\frac{DVT0W}{\cosh\left(\frac{DVT1W\;\;L_{eff}\;\;W'_{eff}}{ltw}\right) - 1} + \frac{DVT0}{\cosh\left(\frac{DVT1\;\;L_{eff}}{lt}\right) - 1}\right)(V_{bi} - \phi_s) \\ & -\frac{1}{2}\frac{(ETA0 + ETAB\cdot V_{bseff})\cdot V_{ds}}{\cosh\left(\frac{DSUB\cdot L_{eff}}{lt0}\right) - 1} \\ & -nvt\ln\left(\frac{L_{eff}}{L_{eff} + DVTP0\,(1 + e^{-DVTP1\;\;V_{ds}})}\right) \end{aligned} \quad (5.11)$$

where

$$V_{bi} := \frac{kb\;T\;\;\ln\left(\frac{NDEP\cdot NSD}{ni^2}\right)}{q} \quad (5.12)$$

and all other referenced terms and variables do not vary with temperature

This sequence of equations shows a complicated dependence on temperature, as higher temperatures lead to lower threshold voltages, and thus higher performance. At the same time, these increased temperatures decrease mobility, and increase wire resistance. Depending on the technology and operating voltage, then, the mobility change or the threshold voltage change can dominate - or cancel each other out. The case in which the two effects are perfectly complementary results in a *Zero-Temperature Coefficient (ZTC)*, in which temperature variation has no impact on performance [67]. One previous researcher has noted that "ZTC Voltage for MOSFETs is about 1V, and therefore, circuits with power supply voltages below

1V have a positive temperature dependence on drain current. [67]", however, the issue is not necessarily clear, as another researcher notes that "..the optimum supply voltage which results in temperature insensitive operation is proportional to the threshold voltage. [68]" To clarify this discrepancy, we perform a timing analysis using SPICE to quantify the effect of temperature on device timing.

5.2 Performance-Temperature Coefficient

The performance/temperature coefficient, dt/dT, is a single scalar value that indicates the change in circuit timing as temperature varies. To determine this coefficient through simulation, we characterize the delay of a sample circuit as temperature varies, using simple interpolation to determine a first-order value of dt/dT. To ensure generality of this evaluation, we use the a chain of minmal-sized inverters, each driving a fan-out of four identical inverters. The delay through four of these inverters is measured as temperature varies. Our SPICE simulations of these sample circuits use the latest release (Feb 22, 2006) of the widely-used Predictive Technology Model from Arizona State University [69, 70]. As temperature, technology, and supply voltage are independent variables, resulting in a large data space, we simplify the analysis by normalizing the delay to a base case of 25C at each supply voltage, as shown in Figure 5.1.

It can be seen in Figure 5.1 that across a wide range of technologies and operating voltages, the variation in delay with respect to temperature is nearly constant, with the largest variation occurring when operating voltages are lowest. This indicates that for the vast majority of situations, dt/dT can be assumed to be roughly constant, with a value of 0.655% variation in delay for every 1° variation in temperature. Thus, over a 10° temperature variation, delay would vary by 6.5%.

5.3 Temperature and Timing Gradients in Practice

Given that we now have a metric for how much temperature variation impacts timing, it becomes necessary to evaluate the range of temperature variation possible

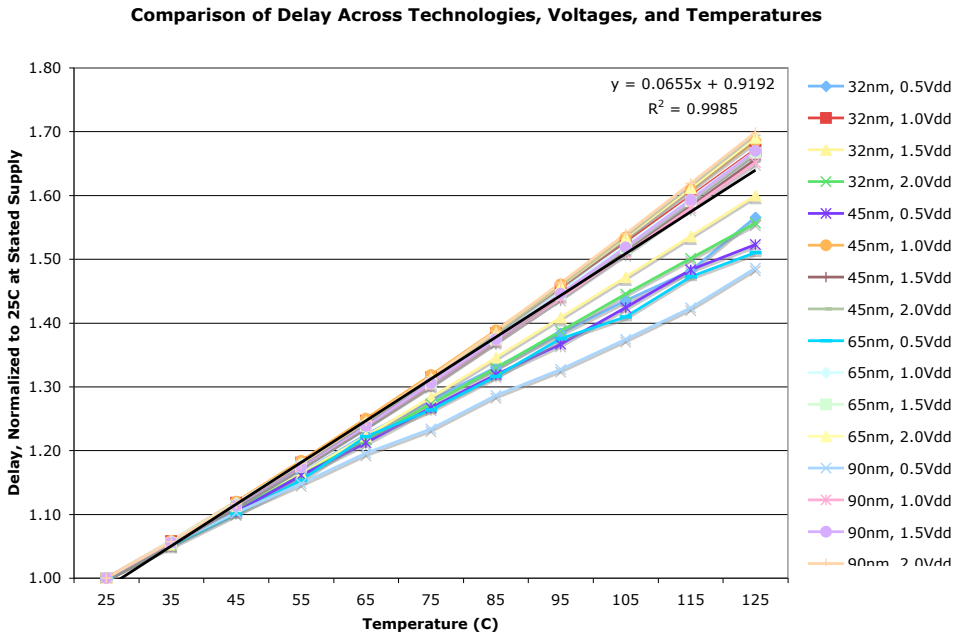


Figure 5.1. Impact of Technology, Operating Voltage, and Temperature on FO4 Delay

on die. This work explores two different means of defining this range. The first is the thermal variation occurring during device power up, as the device warms from room temperature to operating temperature. This variation, while it can affect timing, has always been present in the design process, and can be mitigated by operating with a reduced clock during warmup, or any number of other design-time solutions. The second type of significant thermal variation occurs at runtime, as heat generation is not uniform across the die. As an example, we consider a sample 4-core processor, as shown in Figure 5.2. While this processor fully meets all aspects of the ITRS 65nm high-performance roadmap, the power consumption is skewed towards the first of four cores, and away from the last. This results in a heavily skewed thermal profile, with a 100° variation between core 0 and core 3, as shown in Figure 5.3.

While on-die temperature gradients have also always been present to some extent, it is only the advent of multi-core design that allows these thermal hotspots to develop at one location on the die, or another, and thus presents the possibility for one side of the die to be 100° warmer than the other, and then, only 2 seconds later, for the opposite to be true. This 200° variation in the thermal gradient is both significant, and unpredictable.

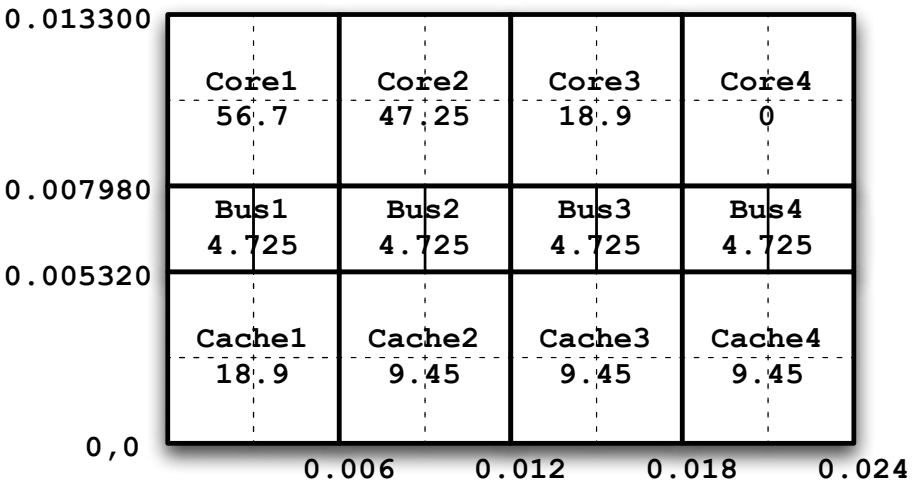


Figure 5.2. Floorplan for an arbitrary four-core processor meeting the ITRS roadmap. The power consumption of each unit is shown.

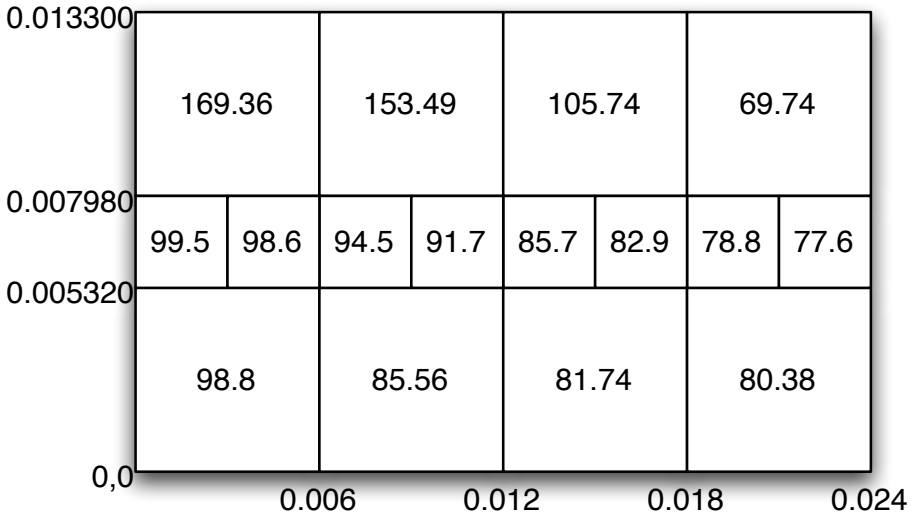


Figure 5.3. Steady-state thermal variation on sample four-core processor.

Continuing with the example four-core processor, we annotate the floorplan to include to two chip-wide wires, each running the length of the bus, used for cache coherency purposes. We denote the propagation delays on these wires at a nominal and uniform temperature of 27° as $t_{right_{27°}}$ and $t_{left_{27°}}$. Considering the case where the left side of the die is at 170° , and the right side is at 70° , and

applying the analytical transistor timing model developed in Section 5.1.1, we find that the $t_{right_{70^\circ}} = 1.18 \cdot t_{right_{27^\circ}}$ and that, even worse, $t_{left_{170^\circ}} = 1.62 \cdot t_{left_{27^\circ}}$. Thus, not only has temperature caused a 62% slowdown in inverter timing, but there is now a 50% disparity in the speed at which signals move across the chip. In the static design case, this can be accounted for, by noting that the two signals will not meet at the center of the design, but instead, 29% of the way from the left edge of the design. This can be incorporated as part of the static timing analysis model, or alternately, sizing techniques can be used to ensure that the signals meet at the center of the design. The situation is worse, however, when the workload shifts from core 0 to core 3, resulting in the thermal profile flipping accordingly. The two signals will now meet at a point 71% of the way from the left edge of the design, a variation of 42% of the total signal path. If both of these signals were required for some decision-making structure, located in the center of the chip, there is a 21% variation in timing, dependent on the thermal profile of the design. This is a purely run-time variation, and, if to be accounted for in static timing and the design phase, results in up to a 21% timing overhead, and lost performance.

Note that the above discussion only includes the transistor model, as the analytic wiring model has not yet been fully evaluated. It is noted, however, that metal wires become more resistive at higher temperatures, with the resistance of copper increasing by 39% as temperature increases from 70° to 170° , resulting in a yet wider performance variation. This variation is even more problematic on long wires, as the timing characteristics of long wires are heavily dependent on the resistance early in the sequence. As such, signals traveling in the direction of increasing temperature have a low-resistance path to source at their beginning, while signals traveling away from regions of high temperature are limited by a larger resistance early in the signal chain.

5.4 Mitigating Timing Gradients

In the past, static design techniques could account for the temperature variation of designs, with a well-characterized thermal profile of a uniprocessor accounting for the vast majority of thermal variation expected. New multicore designs, however, can have dramatic and shifting thermal gradients, and resulting timing gradients.

The dynamically changing nature of the thermal profile results in dynamically changing performance characteristics of circuits, especially those traversing large regions of the chip and potentially operating at a number of different temperatures along these regions, such as clock buffers. Traditional timing analysis, sizing, and insertion techniques do not include these dynamic characteristics, and result in designs that must be operated at a worst-case frequency that is much lower than the typical case, resulting in lower performance.

In this thesis, we characterize the impact that temperature has on the timing characteristics of long signal wires with multiple repeaters, across a range of deep sub-micron technologies. We see that designs focused on worst-case performance inhibit performance at typical temperatures. We demonstrate that the optimal buffer sizing and insertion varies with temperature in both the 45nm and 32nm technology nodes. We then propose a design technique where both typical-case and worst-case signal paths are included in the design, and the signal is sent on whichever path provides optimal timing characteristics at a given temperature. This results in additional timing slack at the cooler edge of the operating temperature range, allowing either higher frequency operation or a lower operating voltage than would be possible without adaptation. The decreased operating voltage results in power savings, and thus helps ensure that temperatures remain low during operation.

5.4.1 Background and Related Work

The problems of optimal buffer insertion and optimal wire sizing have been heavily studied. One of the most commonly cited works on the topic is Bakoglu's work, *Optimal interconnection circuits for VLSI* [71]. In this work, it is shown that the optimal buffer size for a long interconnect can be calculated as a function of the resistances and capacitances of the minimum-sized inverter and wire, as shown in (5.13).

$$\frac{S_{Optimal}}{S_{Minimum}} = \sqrt{\frac{R_D C_{int}}{R_{int} C_{in}}} \tag{5.13}$$

Bakoglu's analysis considers only RC components, ignoring inductive effects. Cong and Pan investigated the importance of RLC interconnects, showing that the opti-

mal interconnect width could be determined analytically [72]. In [73], the problem of simultaneous wire and buffer sizing is analyzed using a transmission line model, extending their analysis to include a general routing tree.

The effect of temperature on circuit performance has also been studied heavily. Of particular interest is Ajami's work, [74]. In this work, the impact of a static temperature gradient on buffer sizing is discussed. It is shown that static temperature gradients impact the performance characteristics of drivers and wires, and as such, the optimal buffer and wire size is dependent on temperature. Ajami's work, while similar, only considers 100nm and larger technologies, where thermal variations are smaller, and the dependence of driver resistance on temperature is not as severe. In addition, in [75], Ajami studies the impact of substrate temperature variation on clock skew by characterizing a symmetrical H-tree structure. In this work, a method of minimizing the resulting clock skew through the use of an asymmetrical H-tree is proposed. These works, while fundamentally similar to this work, only consider a static thermal gradient, therefore underestimating the performance variations possible in practice, and proposing solutions that do not compensate for dynamic thermal variation.

On-die thermal variations have come under close scrutiny recently, most notably in [19], where a distributed RC model of temperature is used to estimate the temperature variations on a processor, demonstrating the presence of localized regions of high temperature, termed *hotspots*. A number of means of reducing these hotspots are evaluated, including migrating workload from one copy of the integer register file to another, dynamic voltage scaling, and global clock gating. An overview of hotspot behavior is given in [76]. In this work, the steady-state temperature profiles of sample designs are characterized across a wide range of situations, including technology scaling and multi-layer wafer integration. This work notes that due to the power density variation seen among functional units, the steady-state temperature on die can vary by 50° in a 90nm technology, and that this variation increases as technology scales, resulting in steady-state thermal gradients of nearly 100° in 65nm technology.

A thermally-aware routing mechanism is proposed in [18], where packets are adaptively routed around hot regions of an on-chip network, allowing those regions to cool and distributing further heat generation across the chip. The link design

for such an on-chip network is considered in [77], where long-distance inter-router connections are designed for typical case, rather than a worst-case. In this proposed system, error detection systems are integrated into the link design, and a fail-safe transmission mode is supported, allowing a much higher operating frequency in the typical case.

5.4.2 Dynamically Adaptive Driver Selection

As the characteristics of transistors vary with temperature at a different rate than wire characteristics, the particular combination of driver sizing and wire lengths for an optimal repeater varies with temperature. We propose that these variations can be accounted for through the use of dynamically adaptive driver selection, where larger drivers can be used to account for reduced current flow and increased wire resistance. In situations where the larger drivers aren't needed, smaller, lower capacitance, and more highly spaced drivers can drive signals with less energy and performance overhead.

5.4.3 Experimental Platform and Preliminary Exploration

We evaluate this dependence of optimal buffer sizing on temperature using SPICE simulation. Inverter characteristics are obtained from the Predictive Technology Model [78], and wire characteristics are based on those found in [79] and the ITRS roadmap [35]. These wire dimensions are then used as input to the Predictive Technology Model's interconnect estimator to obtain appropriate resistance and capacitance values per unit length. Each wire is then modeled as 100 separate RC elements of appropriate length. Inductance effects are not included in these simulations, as according to [80], the impact of inductance on global interconnects on delay diminishes with technology scaling, and as such, will be of decreasing importance. In each simulation, a square wave input is fed into a chain of inverters, as in Figure 5.4, and the time for the signal to propagate from point "A" to point "B" is measured. The physical distance between points "A" and "B" is calculated by multiplying the length of the wire by the number of wires traversed, and the propagation speed of the interconnect calculated as $(distancetraveled)/(delay)$. The buffer size and wire length are varied across a wide range, and the propagation

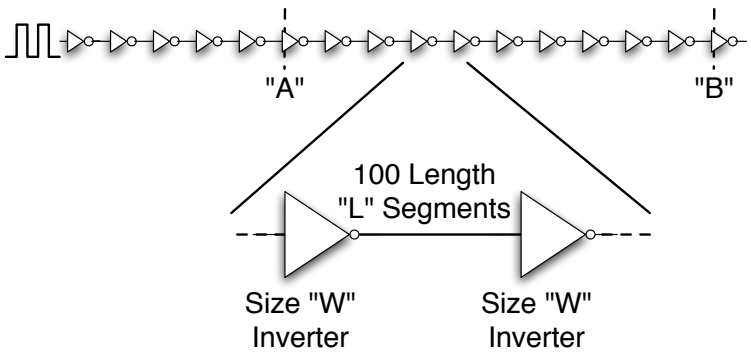


Figure 5.4. Circuit for Timing Analysis

speed of the interconnect is shown in Figure 5.5. Paths perpendicular to the con-

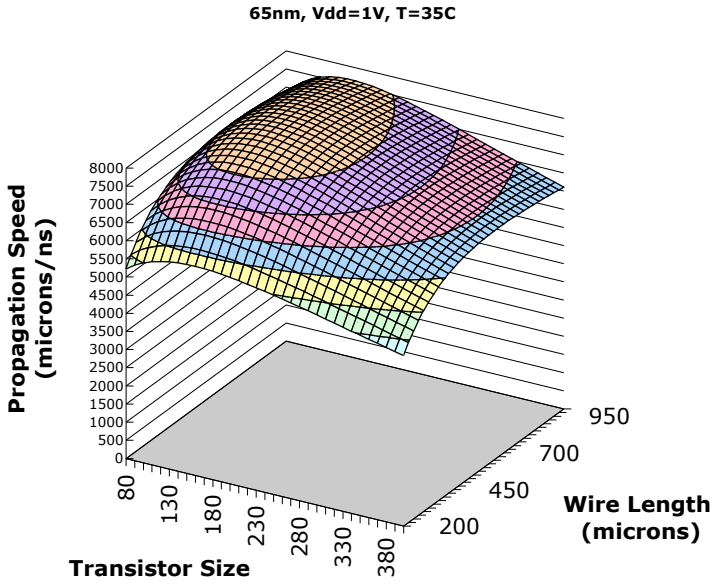


Figure 5.5. Buffer and Wire Sizing for Optimal Propagation Speed in 65nm Technology with V_{dd}=1V and T=35° C. Each contour represents the performance available at a given configuration, with the fastest possible design being located at the center of the contours, with a transistor size 160 times larger than minimum, and wires 650 microns long. The highest-performance configuration will propagate a signal at 8000 μ/ns when operating at 35° C.

tours indicate the steepest rate of performance decrease, and thus, increasing wire length beyond 650 microns has little impact on performance, but decreasing wire size and transistor size even slightly results in a sharp reduction in performance.

Figure 5.6 shows the same range of circuit designs, this time operating at

125° C. In Figure 5.6, the highest propagation speed occurs with a configuration

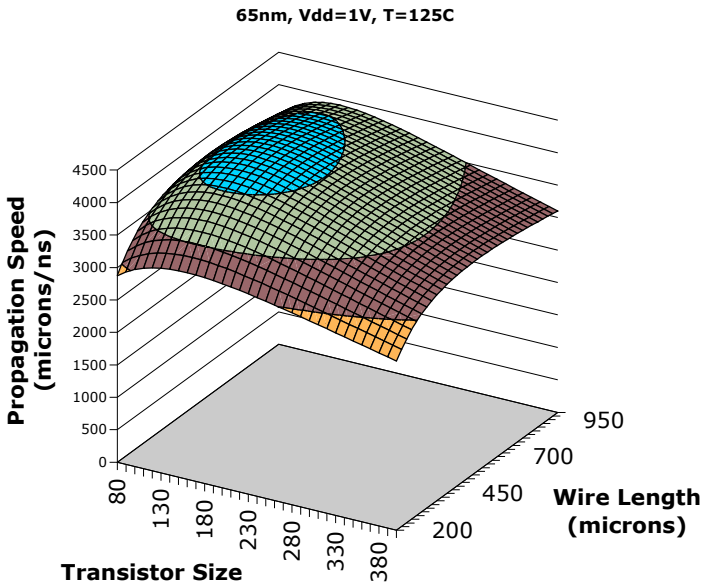


Figure 5.6. Buffer and Wire Sizing for Optimal Propagation Speed in 65nm Technology with V_{dd}=1V and T=125° C. The optimal configuration has drivers 155 times the minimum width and wires 600 microns long. At 125° C, this configuration will propagate a signal at 4170 μ/ns.

almost identical to the previous optimal case, with drivers 3% smaller and wires 8% smaller than the optimal configuration at 35° C. The propagation speed is greatly reduced however, as signals only move at 53% the rate at which they move with optimal sizing at 35° C. We then compare the performance of the two optimal driver designs (for 35° C and 125° C) across a wide range of operating temperatures to see how well they would perform in the general case.

The two designs have nearly identical performance characteristics across all temperatures studied, varying by only 105 microns per nanosecond at most. At first glance, this would indicate that optimal buffer sizing is nearly independent of temperature. As shown in (5.13), the optimal buffer size is dependent on the ratio of driver resistance to wire resistance. The lack of change in optimal buffer sizing therefore indicates that at the 65nm technology node, wire resistance varies with temperature at the same rate as driver resistance, resulting in the near independence of buffer sizing on temperature. However, as noted in [74], as technology

scales, driver resistance depends more heavily on temperature. As the dependence of wire resistance on temperature doesn't scale with technology, this would indicate that optimal buffer sizing in future technology nodes should be temperature dependent.

We therefore evaluate a 45nm technology, to see if the increased dependence of driver resistance on temperature results in a change in optimal buffer sizing and spacing. Figure 5.7 shows the performance contours and optimal buffer design in the 45nm technology, when operating at 35° C, with the ITRS specified 1V supply voltage. Figure 5.8 shows the performance contours (and thus optimal buffer configuration) at 125° C. It is seen from the shape of the contours that the optimal

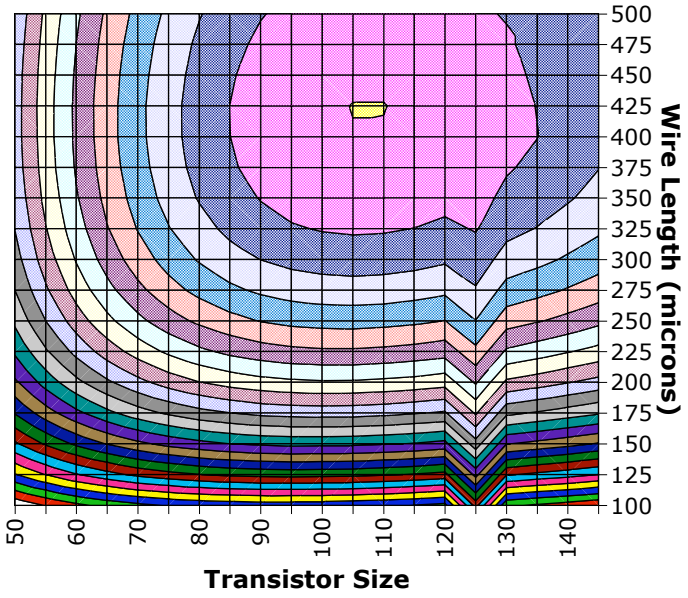


Figure 5.7. Propagation Speed of 45nm Technology at 35° C and V_{dd}=1V. The optimal configuration has a transistor width 105 times the minimum, and wires each 425 microns long. It has a propagation speed of 7250 μ/ns at 35° C.

design point has moved significantly, with the optimal driver size having reduced to only 75 times minimum width, and the wires have shrunk as well, to only 275 microns in length. This is a 29% reduction in driver size, and a 36% reduction in wire length. We again compare the effectiveness of these optimal designs across the possible temperature range to determine their more general effectiveness. Figure 5.9 shows that unlike the 65nm technology, there is a significant difference

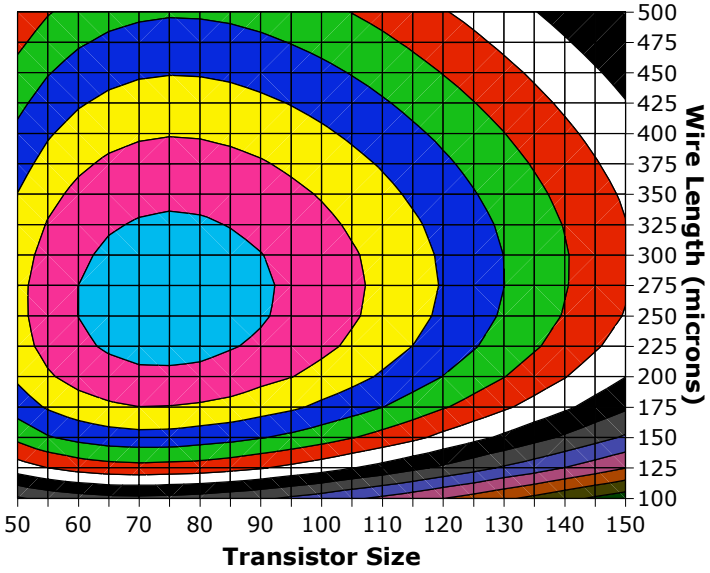


Figure 5.8. Propagation Speed of 45nm Technology at 125° C. The optimal configuration has a transistor width 75 times the minimum, and wires each 275 microns long. Ithas a propagation speed of 4943 μ/ns at 125° C.

in performance between the two designs, as the design optimized for higher temperature achieves higher performance above approximately 70° C, while the other design is faster at lower temperatures.

Figure 5.9 also compares the propagation speeds of the two optimal buffer and wires sizes for a 0.7V technology. Here, we see that the optimal design at high temperatures provides a relatively flat performance characteristic, resulting in an unnecessary 15% timing overhead at 35° C . This is especially important in designs with a significant dynamic thermal variation, such as multiprocessors-on-chip. As the worst-case temperature in such designs is often much higher than the typical temperature, interconnects designed with the worst-case temperature in mind will limit performance unnecessarily, resulting in slow chip-to-chip signals, such as inter-processor communication.

We finally consider the very forward looking 32nm technology. Figure 5.10 shows the propagation speed provided by the 35° C and 125° C optimized design. The 32nm technology has very similar characteristics to the 45nm technology, with the peak performance and transition temperature nearly identical between the two technologies. The largest difference is that in the 32nm technology, the worst-

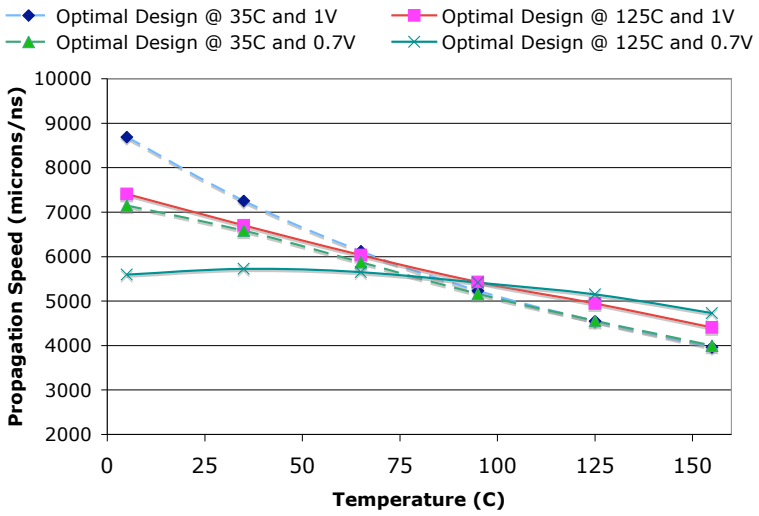


Figure 5.9. Comparison of Optimal Designs at 35° C to the Optimal Designs at 125° C. The dashed lines indicate designs that are optimal at 35° C.

case design does not suffer as severe reduction in performance as in the 45nm process technology. This indicates that a single worst-case design will be able to effectively cover a wide range of high temperatures, allowing a single typical-case buffer system to optimize heavily for colder temperatures.

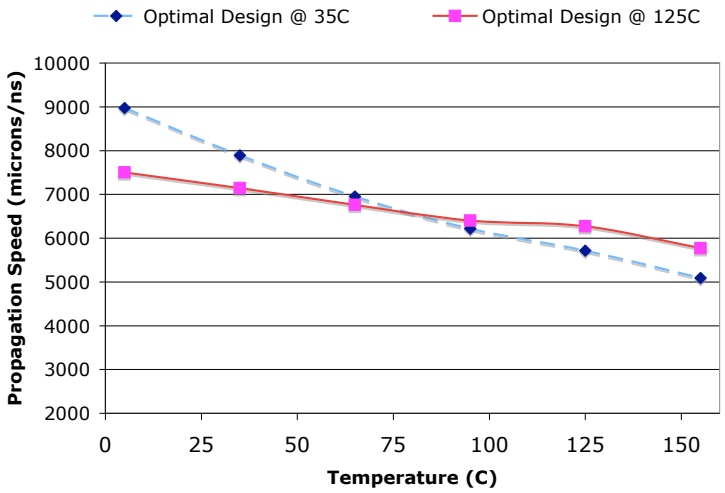


Figure 5.10. Comparison of Optimal Design at 35° C to the Optimal Design at 125° C in 32nm Technology.

5.4.4 Adaptive Interconnect

Due to the increasing importance of temperature on optimal design sizing, it will become difficult to design a single static signal path that will achieve optimal performance. Designs assuming a typical-case temperature will slow considerably as temperatures increase, leading to timing failures, while more conservative designs emphasizing high-temperature performance will be restricted to lower performance at all times, limiting overall device performance and increasing cycle times. We suggest that rather than including a single signal path for long-distance interconnects, two separate paths be designed. One path will have optimal characteristics for the high temperature case, ensuring that the design will meet some minimum timing, even at high temperatures. The other path will be designed with the typical case temperature in mind, allowing fast signal propagation across the chip in the typical case, and creating additional timing slack. This additional timing slack can be used for a number of purposes, including reducing the operating voltage to conserve energy, or dynamically adjusting timing to take advantage of the reduced latency [81, 82]. In high frequency designs, the additional propagation speed may even allow signals to arrive at their destination a number of cycles before they might arrive if using a slower interconnect, reducing latency across the design.

The proposed interconnect system is shown in Figure 5.11. In Figure 5.11, a

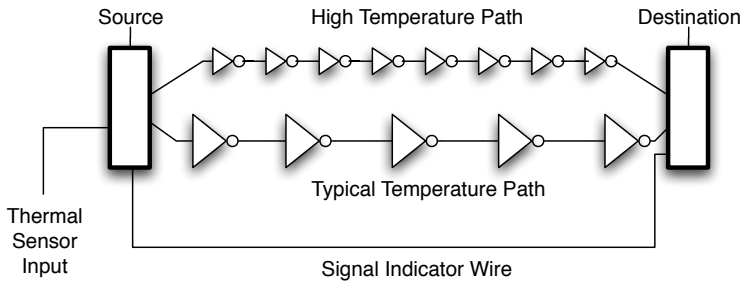


Figure 5.11. Block Diagram of Temperature-Adaptive Signal Path

given signal can be sent on either the high temperature path or the typical temperature path. The choice of which path to use is based on input from one of the device thermal sensors in the region. This one-bit input line indicates whether the current temperature of the region is above or below the transition temperature at which the high temperature path becomes necessary, and therefore, whether the signal

should be sent along the high temperature path. In the event a transition from one path to the other becomes necessary, the source unit activates both sending paths, allowing the signal to travel on both wires for a brief period. When unused, each signal path is gated to reduce leakage currents. This transition behavior is shown in Figure 5.12. During this time, the signal indicator wire would be toggled,

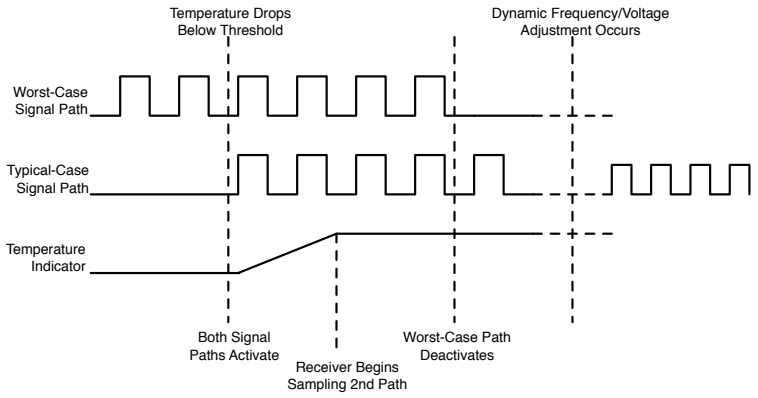


Figure 5.12. Timing Diagram Demonstration Dynamically Adaptive Buffer

indicating to the receiver that it should begin listening to the appropriate buffer chain. As the signal indicator is not significantly buffered (to reduce leakage power and area requirements), it can take a number of cycles for the destination control block to receive the indicator signal. As the transition occurs near the performance crossover point, however, both signal paths have nearly identical timing characteristics, and therefore, the destination block can listen to either buffer chain without experiencing a timing violation. After some fixed number of cycles (determined in simulation), the destination block is sure to have received the slow-moving signal indicator, and thus, the sender may stop toggling the now unneeded buffer chain to reduce power consumption. By including hysteresis at the thermal sensor, it is possible to prevent excessive transitioning, only changing signal paths whenever temperature varies by some non-trivial amount, such as 10° C. As temperature fluctuations occur at very large time scales compared to clock cycles, temperature variations also do not result in thrashing. Figure 5.13 shows the resulting performance characteristics of such a dual-mode buffer system as temperature varies. As changing environments and workloads can result in temperature changes of up to 100° C, the adaptive buffer system can therefore provide up to a 17% boost in

performance.

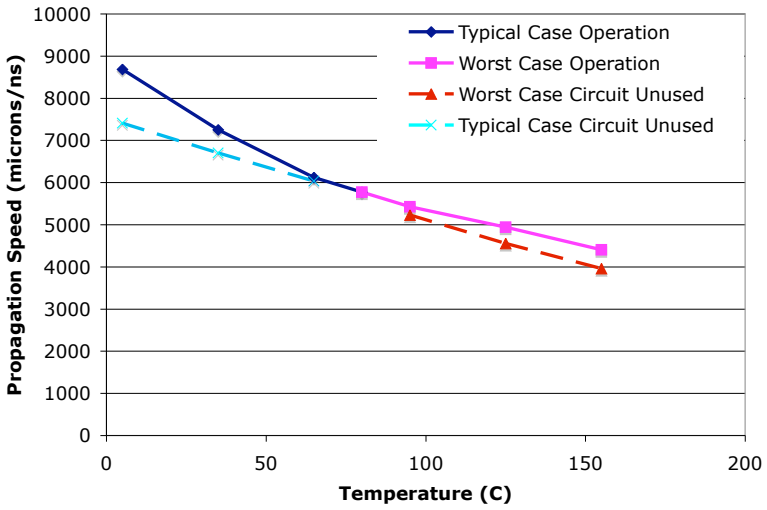


Figure 5.13. Performance Characteristics of Adaptive Buffer in 45nm Technology

In a system that can dynamically alter its frequency and voltage, such as Intel's Montecito core [82] or the Razor processor [81], the impact of a dynamic buffer system can be substantial. The typical case path can be used for the vast majority of computation, but in the event the temperature exceeds some boundary, the secondary path can be exploited, maintaining timing closure (and providing the ability to operate) at higher temperatures.

Of particular note is that it would be possible to include these dynamically adaptive buffers as part of an automated design flow by annotating the existing large-scale driver cells. In such a situation, the timing of the 'fast' process corner could be reduced to that of the typical case path, while the timing of the 'slow' process corner could assume the much more stable performance characteristics of the high temperature path. The routing tool would include a double copy of any signal path using these modified cells, reducing the need for designer intervention. Even in designs where dynamic frequency scaling is prohibitively complex, the typical case path can be statically designed to operate at a lower voltage, providing the same operating characteristics as the worst-case path, but with substantial energy savings. As an example, we consider the 45nm technology characterization shown in Figure 5.9. Here, rather than continue to send the signal along the worst-case delay path as temperatures drop, the signal would instead use a buffer

chain optimized for 35° C, and operating at 0.7V. This lower-voltage chain provides performance nearly identical to the worst-case design, but does so while consuming only half of the power. This is an especially promising approach, as the reduction in power consumption would lead to lower temperatures, and thus more opportunities to employ the typical-case path.

There is obviously overhead associated with such an adaptive buffer system, the most obvious of which is the additional drivers and wiring channels. As the drivers are particularly large, we base our area estimate the area required for these buffers. In the 45nm high performance technology, the two designs compared have buffers of width 105 and 75, with wires of length 425 microns and 275 microns, respectively. Thus, the buffer area per unit length can be calculated. There is a variation of less than 10% between the two configurations indicating that each path is roughly area-equivalent, and that therefore the total area required for this scheme is twice that of the base case.

5.4.5 Discussion

As significant on-die temperature variation becomes more commonplace, the performance characteristics of devices will vary as well. The dependence of temperature on workload characteristics and environmental factors implies that the resulting performance variations will also be unpredictable. In this work, we demonstrate that it is realistic for future designs to experience temperature variations of over 100° C in daily use. The impact of this temperature variation on long repeater chains is characterized in three future technologies, demonstrating that while the variation impacts only overall circuit performance in the 65nm technology node, in 45nm and 32nm technology nodes, temperature variation also affects optimal circuit design choices. This implies that while a given buffer sizing and wire length might be optimal at one temperature, it is not optimal for all temperatures. If designers continue to apply traditional worst-case design techniques in this situation, it will result in designs that, while functional at higher temperatures, do not achieve peak performance in the typical case. Designing for the typical case allows a significantly increased signal propagation speed at low temperatures, but is also highly susceptible to timing failures at high temperatures. This work therefore pro-

poses a dynamically adaptive interconnect structure that transmits signals using a temperature-appropriate signal path, ensuring functionality at high temperatures, while providing significant timing slack at more typical operating temperatures. This slack can be exploited for both increased performance, by increasing the signaling rate by nearly 20% at the edges of the temperature range, but also for lower energy consumption, as the typical case path can be operated at a lower voltage to reduce power consumption by 50%, while still maintaining timing closure.

5.5 Including Temperature in the Automated Design Process

The traditional design flow is a multi-step process, with high-level floorplanning or synthesis occurring at an early stage, with additional refinement steps including global routing, local routing, and power and clock grid design. At each step, the various design options are automatically evaluated according to some 'goodness' metric, such as total area, cumulative wirelength, and optimal timing. For performance reasons, it is infeasible to precisely and completely evaluate each design, and as such, numerous approximations and assumptions are made. Of particular importance to this thesis are the assumptions made regarding device timing, and the performance of the underlying circuits on which it depends. In the floorplanning process, two metrics are usually optimized - chip area, and cumulative wire length. The optimization of chip area is for yield (and hence cost) purposes, while the cumulative wire length is an indication of the delay incurred in traversing the various nets on die. Intuitively, longer wires lead to longer delays, and as such, the manhattan distance between the start and end points of a wire is used as a simple-to-calculate metric for overall wire delay. Notably, the *cumulative* wire length is used rather than the worst-case wire length, as the floorplanner has no information on the relative criticality of the various nets, and as such, cannot make informed decisions about which wires correlate to the worst-case signal path.

While historically this methodology has worked well, shrinking process technologies lead to higher peak power densities, larger power density variations on die, and thus larger temperature disparities across the design. As noted in section

5.2, these temperature variations lead to timing variations, and as such, signals traveling through hot regions have a much higher delay than those going through colder regions, even though both might have identical manhattan distances. The reduced correlation between pin locations and effective delay means that design tools are making less-than-optimal decisions early in the design process, and as such, designs will not achieve maximal performance, or might require expensive iterations through the design flow. As such, incorporating more accurate estimates and metrics in the design flow is increasing in importance.

This thesis focuses on the temperature-related aspects of variation and optimization in the design flow, and expands upon previous work in this area on thermal-aware floorplanning [83]. In this work, a traditional simulated-annealing floorplanner was originally designed to include the peak temperature of the chip as a tertiary optimization metric, allowing the annealing engine to prefer designs that have lower peak temperatures, assuming they have roughly equivalent area and wire length. Over the course of the entire annealing process, this allows designs with lower peak temperatures to be preferred, resulting in a floorplan with roughly equivalent area and wire length, but lower peak temperatures.

Unfortunately, while *HS3d* was developed to be a very high performance thermal estimation library (partially with this floorplanner in mind) even at only 8ms per estimation, the millions of floorplans generated during the annealing process resulted in intractably long runtimes, on the order of days for even 'small' floorplans with 30-50 blocks. As such, the tool was altered such that temperature was no longer evaluated on each floorplan, and instead peak power density used in it's place. Only after the annealing process had completed was a full thermal analysis run to estimate the resulting design's peak temperature. Comparisons between traditional floorplanners and this "power density aware" floorplanner showed a roughly 8° improvement in peak temperature, with less than a 1% impact on chip area or wire length.

Previous works, however, have shown that power density correlates poorly to temperature [19]. In addition, due to lack of spatial information as regarding the power density, it is impossible to use the power density information to more accurately estimate the interconnect delays. As such, there is still a compelling need for a full thermal map of the design. This thesis proposes a method by which

such thermal maps can be created much faster by using a pre-computation method.

5.5.1 Pre-Computation of Thermal Resistances

While there are many different means of estimating temperature, the most widely-used method to date relies on the similarity between the electrical concepts of resistance, capacitance, current, and voltage and the thermal concepts of resistance, thermal mass, power, and temperature, respectively. In this method of temperature calculation, the chip design is used to create a parallel thermal resistor network, and then the power consumption is injected into the network by means of perfect current sources. The solution to the network can be computed by any number of means, including SPICE simulation or direct matrix-vector multiplication between the resistance matrix and vector of power values. The tool *HS3d* was developed to address the performance problem associated with computing large resistance matrices, as discussed in Chapter 2.2. One of the most significant performance improvements was the use of optimized matrix routines such as those found in the LAPACK and BLAS libraries [84, 85], which significantly improve the performance of basic matrix operations such as matrix-vector multiply. Even with these routines, however, the calculation of a resistance matrix is a time-consuming process, requiring between 8-20ms for each individual block permutation.

In existing tools, the resistance matrix is unique for each permutation of blocks because the resistor network is configured in such a way that there is a node at the geometric center of each block, ensuring that each block has a uniquely computed temperature, as shown in Figure 5.14.

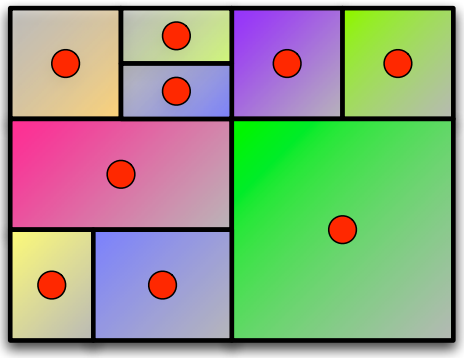


Figure 5.14. Temperature Estimation Points in Traditional Tools

As the goal of our tool is not to compute each individual block temperature, but instead the temperatures across the design through which wires must pass, this one-to-one correspondence provides no significant benefit. Instead, we estimate the temperature in a regular grid pattern on the chip, as shown in Figure ref-fig:PreComputedEstimationPoints.

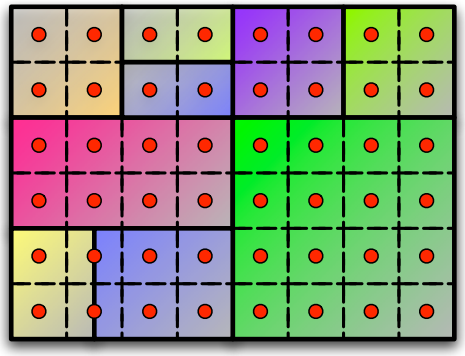


Figure 5.15. Temperature Estimation Points in Pre-Computed Matrix

Using this regular grid structure as a block-level floorplan, a corresponding thermal resistance matrix is calculated and stored in memory. To estimate the steady-state temperature, then, only one last step is required - the matrix-vector multiply between this pre-computed resistance matrix and the power dissipated in each cell. If, at any point during the annealing process, a plan is created that is roughly the same dimensions as this pre-computed resistance matrix, we can estimate the temperatures of this plan by translating the block location and power consumption into the respective locations in the power vector for the pre-computed plan, then performing the multiply. Unlike the calculation of the thermal resistance matrix, an operation employing $O(NumBlocks^3)$ divisions, the block power consumptions can be assigned to the appropriate slot into the grid power vector in only $O(NumBlocks)$ divisions. In practice, the thermal resistance matrix creation takes over 1,000,000 times longer than the power vector assignment for 33-block floorplans, implying that pre-solving thermal matrices provides yet further opportunity for a dramatic speedup of thermal estimation.

5.5.1.1 Accuracy of Estimation

While providing a huge potential for speedup, the pre-computation of thermal resistance matrices also introduces the possibility for error. As the actual floorplans created during the annealing process do not necessarily match the dimension of the pre-computed floorplans, a naive mapping of floorplan to a small number of pre-computed matrices might result in the mapping of a $2cmx2cm$ floorplan onto a $4cmx4cm$ pre-computed matrix, or worse, a $1cmx1cm$ matrix.

In the case of mapping a design to a larger pre-computed matrix, the temperature at edges of the design would be incorrectly estimated, due to the additional thermal pathways. In the case of a design being mapped to a smaller pre-computed matrix, the mapping can be done in one of two ways - overlaying or scaling. When overlaying the total power of the design can be severely underestimated, as in the case of the $2cmx2cm$ design being mapped onto a $1cmx1cm$ matrix, the map would only include 25% of the actual power dissipating area. In the case of scaling, the power density can be much higher due to scaling the power consumption into a smaller area. In the example used above, the power density would be incorrect by a factor of four, resulting in drastic inaccuracies in thermal estimation.

Given the relatively minor problems imposed by mapping a design onto a larger matrix as compared to mapping it to a smaller matrix, this thesis explores the thermal inaccuracy incurred for mapping onto larger matrices, and thus provides guidance as to the number of pre-computed floorplans required for a range of possible floorplan sizes.

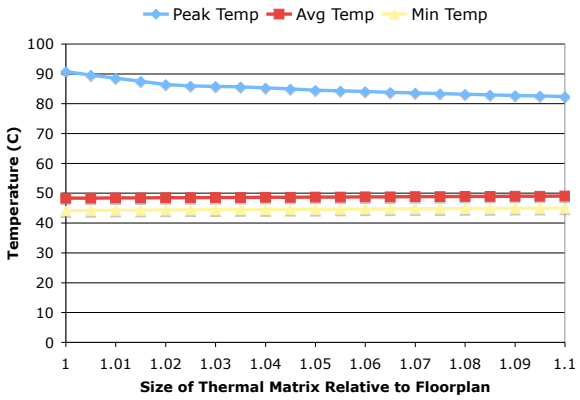


Figure 5.16. Impact of Size Mis-match Between Actual Floorplan and Thermal Matrix. Starting size of the floorplan is 4 sq. cm.

Figure 5.16 shows the range of temperatures obtained for the floorplan shown in 2.1(a), used earlier for the validation of the *HS3d* tool itself. Figure 5.16 shows the temperature ranges resulting from temperature estimation using a pre-computed thermal resistance matrix with a $20x20$ mesh resolution. Examining Figure 5.16, we note that at a relative size of 1, where the estimation matrix is identical to the floorplan, the peak temperature is 90.7° C. As the size of the estimation matrix increases, the peak temperature decreases slightly, resulting in a 2° mis-estimation for every 1% size mis-match between the floorplan and the thermal matrix. Alternately, it can be viewed as a 2° mis-estimation for every $0.2mm$ size mis-match. To clarify this situation, the floorplan from Figure 2.1(a) was scaled to 25% of its original area and power, and the resulting temperature variations are shown in Figure 5.17. Here, we see that the inaccuracy in peak temperature decreases at a much smaller rate, if measured as $\frac{PeakTemp^\circ\ Inaccuracy}{\%SizeMismatch}$. Instead, the amount of inaccuracy is a constant factor in both cases if measured as $\frac{PeakTemp^\circ\ Inaccuracy}{SizeMismatchmm}$. Therefore, we note that for a size mismatch of $0.1mm$, the peak temperature is reduced by approximately 1° . For size mismatches of $0.25mm$, the inaccuracy increases to 2.5° , and for a size mismatch of $0.75mm$, the inaccuracy is only 5° . As thermal variations less than 5° are possible for a large number of reasons, such as variations in the thickness of the thermal interface material, variations in leakage currents, and variations in operating mode, for the remainder of the work, our simulation results include floorplans such that the maximum size discrepancy is $0.75mm$. Therefore, the average size mismatch is only $0.375mm$, and an average of 3.5° of error in the peak temperature estimation.

5.5.2 Implementation and Evaluation

Having found that using pre-computed resistance matrices allows the thermal qualities of a floorplan to be evaluated without a significant drop in estimation accuracy, the floorplanning tool was enhanced to create a library of pre-computed matrices at run-time, generating a set number of matrices, with a fixed step size between them. These values are manually set to 'reasonable' values, ensuring that any approximation will find a pre-computed matrix of an appropriate size, and that the library contains a large enough number of such plans as to allow thermal estima-

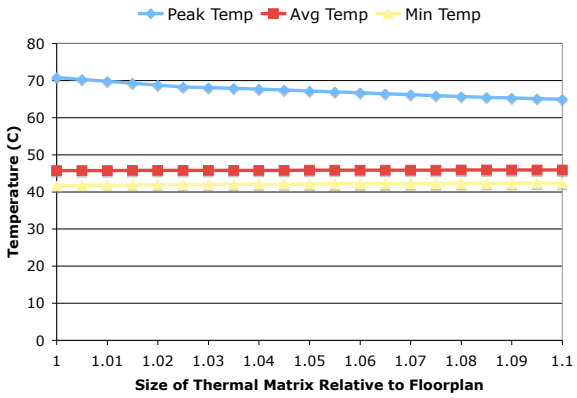


Figure 5.17. Impact of Size Mis-match Between Actual Floorplan and Thermal Matrix. Starting size of the floorplan is 1 sq. cm.

tion on all reasonable floorplans generated during the annealing process. While this technique does require some foreknowledge of the floorplanning process (gained by running the floorplanner in a non-thermally-aware mode), in a more commercial setting, such a library could be generated dynamically at runtime, as each floorplan checks the existing library for a thermal resistance matrix of appropriate size, generating one if none is available. After some initial cold-start time, the library would contain a sufficient index of pre-computed matrices, keeping runtime low, and requiring no manual input from the operator, or foreknowledge of floorplan sizes.

In order to evaluate the impact of a true thermal-aware optimization on floorplanning, we compare the performance of a peak-temperature-aware placement to both a power-density-aware placement, as well as a purely area and wire driven placement. Figure 5.18 shows the area, wire length, and peak temperatures obtained using all three methods with the benchmark *ami49* [86] with power consumption as noted in [83] . Note that for the thermally-aware results presented in Figre 5.18, a mesh resolution of 20x20 was used, with a library of floorplans generated with $0.75mm$ size increments.

In Figure 5.18, we see that the thermally-aware floorplanning process results in peak temperatures that are, on average, 9° C lower than those obtained with traditional floorplanning techniques. Comparing to the previously published power-density aware methodology, the thermal-aware floorplan further reduces peak temperature by an additional 7° , on average. Ignoring the rather poor results the

power-density aware tool generated on the ami33 benchmark, the thermal-aware tool results in an average of 2° reduction in peak temperature. Most importantly, the power-density aware tool does not necessarily reduce peak temperature overall, as in two of the four test-cases, the power-density aware tool actually results in a higher peak temperature than the traditional placement technique.

Regarding the more common metrics of chip size and wire length, we observe that the thermally-aware technique does result in a slightly increased average deadspace, of 9.9%, as compared to 8.67% for the traditional technique, and 8.7% for the power-density aware technique. This slight increase in area is to be expected from a thermally-aware tool, as increased die area provides positive feedback in the form of reduced temperature.

While the thermal-aware floorplanning technique is effective at reducing peak temperatures when compared to both the both the traditional and power-density aware techniques, the runtime (shown in Figure 5.18(e)) is much larger, requiring 10 to 50 times as long to run to perform the annealing process. This isn't merely a drawback as regards to time required to run the tool, but also in regards to the quality of results, as there is some question that simply running the traditional algorithm 20 times and selecting the one with the lowest peak temperature might be a better selection process. Profiling the floorplanner as it runs, we find that the vast majority of the run-time (96%) is spent doing the matrix-vector multiplication of the thermal resistance matrix and the power vector. While this multiplication is still much faster than the creation of the thermal resistance matrix, even with optimized libraries, it still requires some time to run. As matrix multiplication on 'small' matrices is best performed as an $O(matrixsize^3)$ process, reducing the thermal estimation mesh resolution from 20x20 to 16x16 matrix should reduce run-time by almost 50%. Further reducing resolution to only a $10x10$ mesh will bring run-time from a 10-50X penalty to only a 1.25X-6X% penalty, making it much more comparable to the other methods. In addition, the reduced memory footprint will allow for better usage of the memory hierarchy, and thus hopefully reduce run-time yet further. Figure 5.19 shows the effectiveness and run-time of the thermal-aware floorplanner when used with a lower resolution pre-computed thermal resistance matrixes.

Figure 5.19 shows that while the 10x10 matrices provide a greatly reduced run-

time, only 2-3 times larger than the power-density aware algorithm, the thermal optimization suffers greatly. Due to the lower mesh resolution, the floorplanner is unable to discern a useful difference between many configurations, and as such, there is little forward progress towards a lower peak temperature. Moving to a 16x16 resolution alleviates much of this problem, as the thermal-aware floorplanner is better able to compare nearly identical floorplan configurations. The resulting peak temperatures are comparable to those found with the more accurate 20x20 mesh, but the run-time is reduced by a large factor, making the thermally-aware floorplanning more viable in a large-scale annealing process.

5.6 Thermal-Aware Signal Delay Estimation

As the estimation of peak temperature on the die requires the calculation of a complete thermal map, the calculation of a 'speed map' for the die, indicating which regions have the lowest delay per logic stage or the fastest signal propagation, is quite simple, requiring only a nonlinear mathematical operation on each temperature in the thermal map. Armed with this information, it is possible to improve the accuracy of wire delays provided by and used in the floorplanning process, hopefully not only providing more accurate estimates of wire delay to downstream tools, but also providing better placements based on these more accurate delay estimates.

Using the correlation of temperature to delay obtained shown in Figure 5.8, a 'speed map' of the die is generated, containing the relative propagation speed of each cell in the thermal map. As the uneven thermal characteristics lead to uneven performance characteristics, the shortest path between two cells in the mesh is no longer necessarily a straight line, and can vary as the temperature profile of the device changes. Figures 5.20 and 5.21 demonstrate this effect on a sample dual-core processor floorplan similar to that of the AMD Athlon X2 [87].

In Figure 5.21, the latency for both the manhattan paths and the shortest path between two points are shown. Notably, the delay of the manhattan paths is not constant, varying by roughly 1%. The shortest path between the two points is over 3% faster than the manhattan paths, indicating that any timing estimation based on manhattan paths will be incorrect by 3%. While this might not seem

like a dramatic variation, the author notes that the temperature variation in the region between the two points has only a small temperature variation, between 86° to 69° , a range of only 17° . As future designs are expected to have increased thermal variation, the variation in signal delays can be expected to increase as well.

Integrating this delay estimation with the floorplanning tool, the normal manhattan distance estimation of wire length is replaced with one that accounts for temperature-induced delay variation. The runtime for this tool is increased significantly over the basic thermal-aware case, as an all-pairs shortest paths algorithm must be run on each evaluated floorplan. With a mesh resolution of 16x16, the run-time required for the placement of the *xerox* benchmark increased from 1221 seconds to 4743 seconds, an increase of 3.8 times. While this runtime is quite large, the additional delay is algorithmically linear with respect to the resolution of the thermal map, rather than with respect to the number of blocks in the floorplan, implying that for large floorplans, the complexity doesn't grow with the number of blocks that need to be placed (or connections between them), instead, growing only linearly with the number of floorplans to be evaluated. Still, the resulting run-time increase is unlikely to be widely accepted, and as such, the resolution of the tool was reduced to only a 10x10 mesh. While this does reduce the accuracy of the thermal estimation in the regions of peak temperature, due to the lack of resolution near particularly small components, the overall thermal estimate is still accurate. In addition, as the peak temperature within a cell, if higher than that obtained with a low-resolution mesh, will be located in only a fraction of the low-resolution cell, it is reasonable to assume that a thermal-aware routing tool would use this lower-temperature portion of the low-resolution cell to ensure high signal propagation speed. Hence, while the actual peak temperature estimation suffers from reduced accuracy, the wire-delay variations are much more tolerant of lower resolution meshes. Figures 5.22 and 5.23 shows the run-time and wire-delay variations observed by the thermal-aware floorplanner during the optimization process.

While Figure 5.21 showed only a small variation in wire delay between the two arbitrary points, the addition of thermal-aware wire delays in the floorplanning process had a much more noticeable effect. Figure 5.23 shows that during

the floorplanning process, effective wire lengths deviated up to 55% from their non-thermal-aware values, with an average deviation of approximately 20%. Even assuming that the average chip temperature was available before the floorplanning process (perhaps by an initial run of a thermal-aware floorplanner), assuming a single static temperature profile during floorplanning leads to inaccurate wire length estimations, with the final floorplan having, on average, a 3% deviation from the average delay during the floorplanning process. As such, including temperature during the entire optimization process yields more accurate wire-length estimations.

Figure 5.24, then, shows the impact that having these more accurate wire delays can have on the floorplanning process.

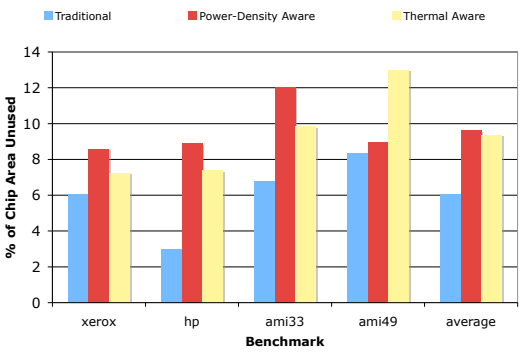
While the inclusion of thermally-aware wire delays in the floorplanning process is effective at generating floorplans with lower effective wire lengths, the runtime penalty is quite high, and as such, is unlikely to be introduced into the industrial tool flow. Instead, a more likely option is the inclusion of thermal-aware characteristics into the global routing process. While this thesis does not address the inclusion of thermal-aware characteristics in the global routing process, preliminary work has been done to characterize the efficacy of thermal-aware global routing over simple manhattan routing. Given a sample die, such as the ev6 core shown in Figure 2.1(a), traditional global routing tools ignore temperature variations, and focus on the utilization of wiring tracks, congestion, and crosstalk problems. In operation, however, the thermal variation on die will cause performance variations depending on which particular locations a given signal traverses. Figure 5.25 compares the delay of temperature-ignorant routing, which assumes a static default temperature of 85° C, and thermal-aware delay. As the average temperature of the die is below 85° C, the thermal-aware routing estimates much lower delays for all signals. At a finer detail level, however (not easily visible on the figure), the timing differences between blind manhattan routing and shortest-path routing are shown. Due to variations in temperature along the die, blind manhattan routing can traverse hot (and thus slower) regions, resulting in lower performance than shortest-path routing. On this die, the difference isn't large, with the worst-case signal delay for blind manhattan routing only 1.8% longer than that obtained through shortest-path routing. Figure 5.26 shows the floorplan for a sample dual-

core processor, similar to the AMD Athlon X2 core, and the path delay variation across the die.

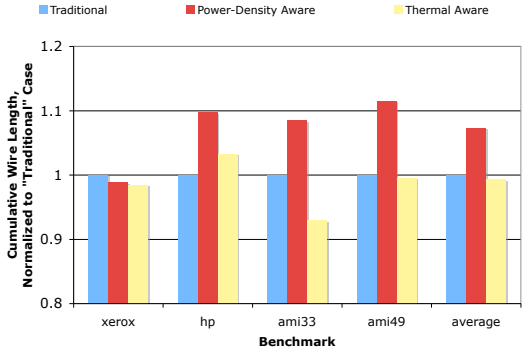
Due to the additional temperature disparity on die, as well as the increased number of hotspots, the worst-case path delay is over 8% longer if using blind manhattan routing, rather than thermal-aware shortest path routing. Therefore, as technology moves forward and wire delay and thermal variation both increase in importance, the impact of thermal variation on delay must be included in routing tools to ensure optimal design.

5.6.1 Discussion

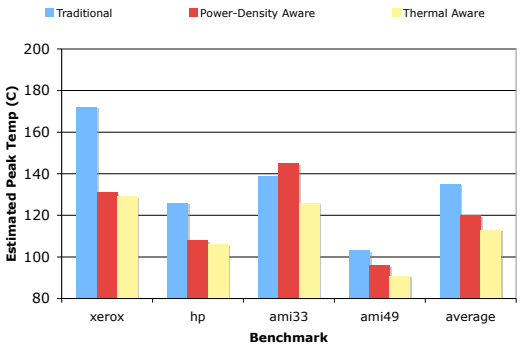
In this chapter, the relationship between temperature and timing has been explored, and the performance of FO4 logic has been shown to vary roughly 1% for every 1.5° C variation in temperature. The impact of temperature on long signal wires was also characterized, and shown to follow a much less linear trend, due to the conflicting behavior of wires and transistors as temperature varies. One means of minimizing long signal wire delay variation was presented, through the use of temperature-sensitive signal paths that dynamically select the better of two repeater paths to provide guaranteed timing across a wide range of temperatures with minimal energy cost. The integration of temperature-aware design in the design tool flow was explored, and a new method of fast thermal estimation for use in floorplanning algorithms was presented. This method allows true thermal-aware floorplanning with a much smaller performance penalty than previous methods, and results in designs that, on average, have peak temperatures 22° lower than thermally-blind methods, and 12° lower than power-density aware methods. The impact of temperature on wire estimation was then included into the floorplanning process, allowing more accurate wire length estimation during the floorplanning process.



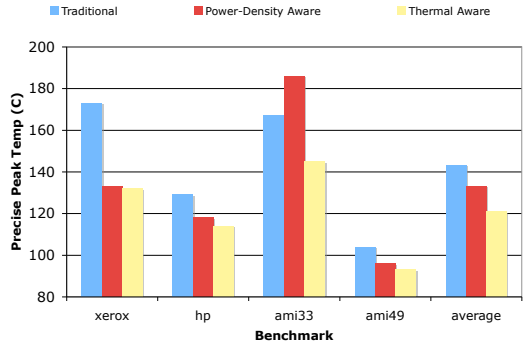
(a) Percent of Deadspace in Final Floorplan. Lower values indicate smaller overall chip area.



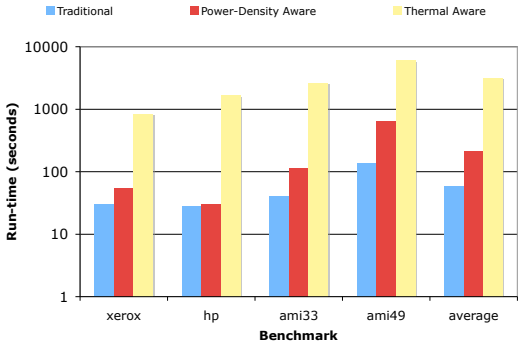
(b) Cumulative Wirelength, Normalized to that obtained by the "Traditional" floorplanning tool.



(c) Peak Temperature of the Design, obtained using a pre-computed thermal matrix of the appropriate size.

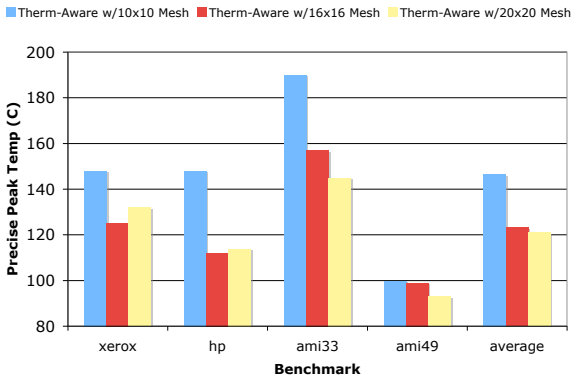


(d) Peak Temperature of the Design, obtained using a precisely-sized thermal matrix.

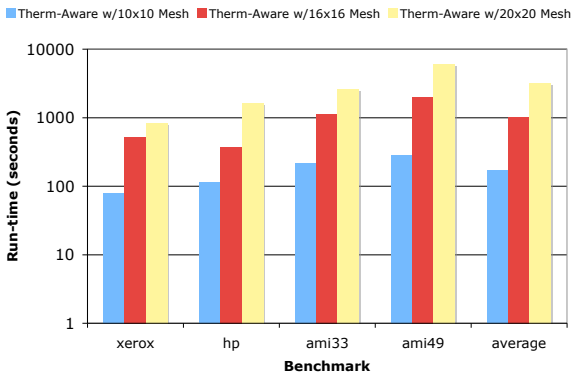


(e) Runtime comparison between the various methods of floorplanning. A mesh resolution of 20x20 was used for thermal estimation.

Figure 5.18. Comparison of traditional floorplanning techniques to power-density aware, thermally-aware, and thermally-aware with wire-delay variation. A 20x20 mesh resolution was used for thermal estimation.



(a) Peak Temperature of the Design, obtained using a precisely-sized thermal matrix.



(b) Runtime comparison between the various mesh sizes.

Figure 5.19. Evaluation of smaller mesh sizes for temperature and runtime tradeoffs.

Figure 5.20. Test Device Floorplan. Total die area is 6.7 sq. cm. and total power consumption is 134W.

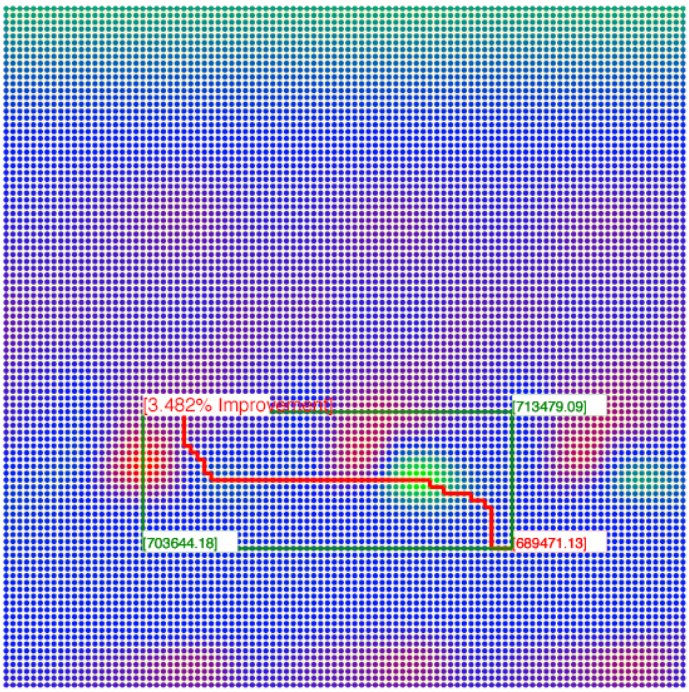


Figure 5.21. Shortest Path Between Two Arbitrary Points on Design Shown in Figure 5.20. Latency is normalized such that 10,000 units represent the time taken to travel 1 micron at 85° C. Temperatures range from 86° to 58° .

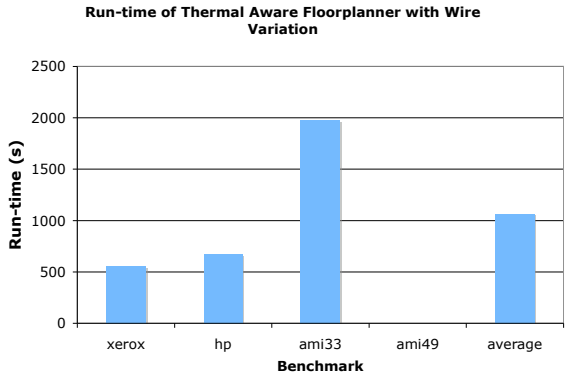


Figure 5.22. Run-time required for thermal-aware floorplanning with wire-delay variations included. A 10x10 mesh resolution was used.

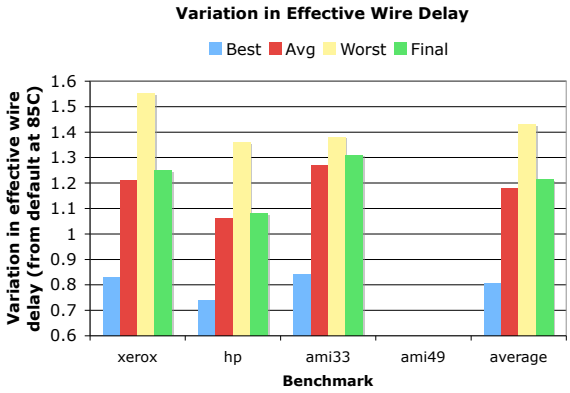


Figure 5.23. Variation in Wire Delays During the Floorplanning Process

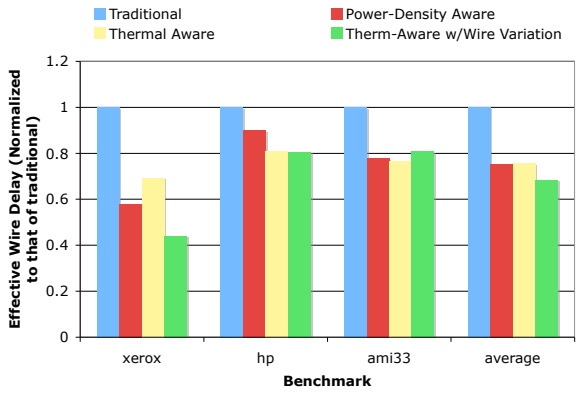


Figure 5.24. Comparison of effective wire delays between floorplanning algorithms

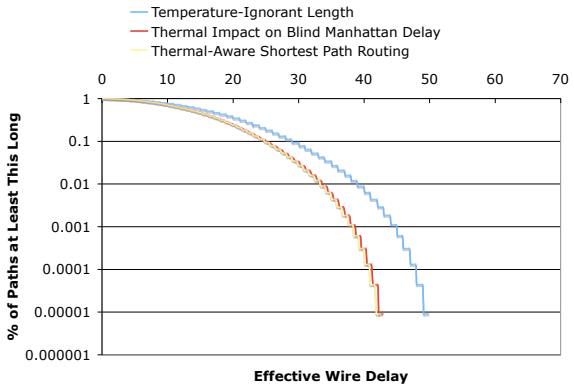
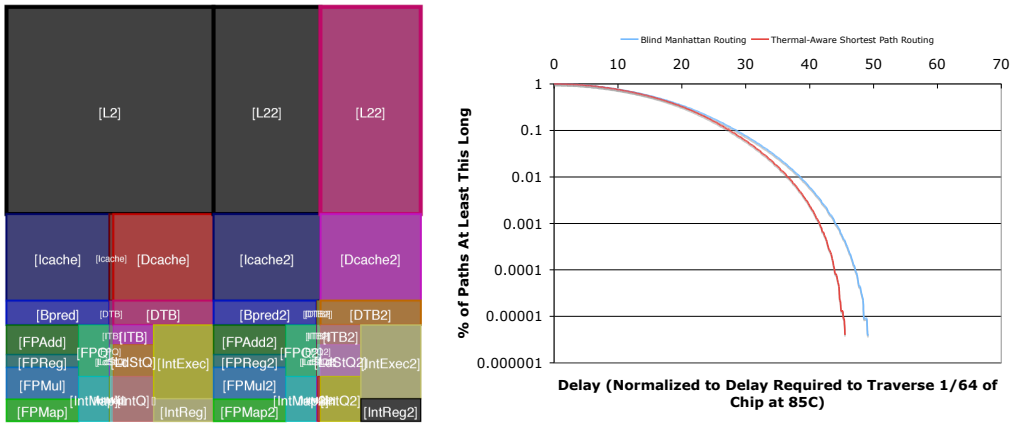


Figure 5.25. Variations in delay among paths on the sample floorplan. Temperature varies only 46° on this die.



(a) Sample dual-core processor floorplan

(b) Variations in delay due to routing choices

Figure 5.26. Variations in delay on a sample dual-core floorplan. Temperature varies by 110° on this die.

Chapter 6

Conclusion and Future Research

As technology scales, the power density of functional units is expected to increase. This increase in power density, when coupled with the relatively stagnant ability of the package to distribute and remove heat, results in ever-increasing chip temperatures, both locally and globally. These high temperatures lead to reduced device lifetime, reduced reliability, increased standby power consumption, and increased timing variation. While low-power design techniques address the global heating problem, such techniques have little impact on local thermal problems. To ensure the viability of future technologies, means of characterizing, analyzing, and mitigating these thermal problems must be developed. This thesis discusses previous research modeling and mitigating these problems, and presents a new modeling tool, *HS3d* , capable of high-performance modeling of a large number of designs, and the first tool to support multi-level floor plans. This tool has been used for an evaluation of floorplanning algorithms in [88] and is currently being evaluated by Xilinx for use in the development of multi-layer FPGA designs. The type and magnitude of thermal problems likely to be encountered in future designs is demonstrated, and the impact of different design attributes on these problems is discussed.

Two means of mitigating these problems are discussed, including runtime planar remapping [89] and integer offloading to floating-point units [90]. An numerical model of the importance of temperature on circuit timing is extracted from simulation, showing the impact of temperature on circuit performance. Across a wide range of technologies and operating voltages, the delay through a fan-out four

inverter increases roughly 5% for every 10° increase in temperature. The buffer-insertion problem is also shown to be temperature dependent, with optimal inverter sizings and wire lengths varying as the operating temperature varies, indicating that a single design cannot be optimal across a wide range of temperatures. A possible means of mitigating this sub-optimality is presented, allowing near-optimal signal propagation across a wide range of temperatures.

The integration of thermal estimation into the design flow is explored, integrating *HS3d* into a simulated-annealing floorplanner, and showing that reduction in peak temperatures of over 20° , or, alternately expressed, 33%, with only a 3% increase in chip size. The use of a library of pre-computed thermal resistance matrices is proposed, allowing for a significant performance advantage when compared to traditional techniques. Using such grid matrices with a moderate resolution, such as 12x12, allows true thermal estimation to be incorporated into the floor-planning process without a significant increase in run-time. The impact of temperature on effective wire delays is also considered, and while the inclusion of these more accurate metrics has little impact on the floorplanning process, it does provide a more accurate means of estimating delays for use in the global routing process. These more accurate delays could be provided to the global routing tool, providing preference for longer wires to prefer cooler paths, giving higher performance than traditional 'blind' manhattan routing.

6.1 Future Work

While this document discusses a number of different parts of the complete thermal 'picture', it is by no means complete. Much of this thesis employs average power consumption values to determine operating temperature ranges, and as low-power techniques continue to advance, these time-varying power consumption will deviate further and further from the average. This time-varying temperature profile will make many design-time temperature-aware solutions much more complex, and may result in a need for more advanced run-time solutions.

The proposed adaptive buffer path selection is one such solution, providing a means of ensuring high signal propagation speeds across a range of operating temperatures. Still, the solution, as currently described, requires a near doubling

of resources, and more important, requires the creation of parallel signal wires, causing coupling capacitance, and therefore increasing delay. One possible improvement is to use adaptively-sized buffers, where the size of the inverter used to drive the signal can be changed dynamically, requiring only one signal path. This may allow superior performance over traditional designs, without the overhead of paired signal paths.

While the variation in wire propagation delay due to temperature have been studied, and the use of shortest-path routing has been shown to be superior to thermally-blind manhattan routing, this information has yet to be included into a global routing tool. This is a non-trival task, as the primary task of a global router (to evenly distribute wires so as to reduce congestion) is directly at odds with thermally-aware design, which prefers clustering signals into cold regions, allowing increased performance. As such, a detailed analysis of the trade-offs involved will have to be performed.

More generally, as technology scales, unless significant and dramatic changes are made to current architectural and design techniques, power density variation, both on die and temporally, will continue to increase, leading to larger and larger variations in delay, reliability, and leakage. These variations can be considered in the same light as process variation, power rail variation, and other types of variation. Looking forward, then, it is not unreasonable to assume that a generally 'variation tolerant' field of research will emerge in computing, with circuits and architectures able to handle variation at different levels of abstraction. These variation tolerant systems will employ variable clocking schemes (or fully asynchronous), as well as some means to intelligently schedule workload among processing units whose capabilities and performance can vary significantly, both between die, as well as intra-die.

Bibliography

[1] P. Li and L. Pileggi and M. Ashegi and R. Chandra, "Efficient Full-Chip Thermal Modeling and Analysis," in *IEEE/ACM Intl. Conf. on CAD*, November 2004.

[2] K. Skadron, T. Abdelzaher, and M. Stan, "Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management," in *High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on*, 2002, pp. 17–28.

[3] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy caches: simple techniques for reducing leakage power," in *Computer Architecture, 2002. Proceedings. 29th Annual International Symposium on*, Anchorage, AK, 2002, pp. 148–157.

[4] J. Srinivasan and S. V. Adve, "The Importance of Heat-Sink Modeling for DTM and a Correction to "Predictive DTM for Multimedia Applications"," in *Proceedings of the Fourth Annual Workshop on Duplicating, Deconstructing, and Debunking (WDDD) at ISCA-05*, June 2005.

[5] M. D. Powell, M. Gomaa, and T. Vijaykumar, "Heat-and-Run: Leveraging SMT and CMP to Manage Power Density Through the Operating System," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, October 2004.

[6] S. Heo, K. Barr, and K. Asanovic, "Reducing Power Density through Activity Migration," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, August 2003.

[7] H. Li, S. Bhunia, Y. Chen, T. N. Vijaykumar, and K. Roy, "Deterministic Clock Gating to Reduce Microprocessor Power," in *Proceedings of the Ninth International Symposium on High-Performance Computer Architecture (HPCA)*, 2003, pp. 113–122.

[8] D. Brooks and M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors," in *Proceedings of the 7th International Symposium on High-Performance Computer Architecture (HPCA)*, 2001.

[9] D. Brooks, V. Tiwari, and M. Martonosi, "WATTCH: A Framework for Architectural-level Power Analysis and Optimizations," in *ISCA*, 2000, pp. 83–94.

[10] 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 International Symposium on High-Performance Computer Architecture (HPCA)*, 2002. [Online]. Available: http://lava.cs.virginia.edu/HotSpot/

[11] S. Heo, K. Barr, and K. Asanovic, "Reducing power density through activity migration," in *Low Power Electronics and Design, 2003. ISLPED '03. Proceedings of the 2003 International Symposium on*, 2003, pp. 217–222.

[12] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on*, 2001, pp. 171–182.

[13] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, S. K., and D. Tarjan, "Hotspot: Techniques for modeling thermal effects at the processor-architecture level," in *ThermalInvestigation of ICs and Systems, 2002. THERMINIC. The International Workshop on*, October 2002.

[14] S. Ghiasi and D. Grunwald, "Thermal management with asymmetric dual core designs," in *University of Colorado Technical Report CU-CS-965-03*, 2003.

[15] C. H. Lim, W. Daasch, and G. Cai, "A thermal-aware superscalar microprocessor," in *Quality Electronic Design, 2002. Proceedings. International Symposium on*, 2002, pp. 517–522.

[16] K. B. T-Y Chiang and K. C. Saraswat, "A New Analytical Thermal Model for Multilevel ULSI Interconnects Incorporating Via Effects," in *IEEE International Interconnect Technology Conference (IITC)*. San Fransisco, CA, USA: IEEE, June 2001, pp. 92–94.

[17] P. Wilkerson, A. Raman, and M. Turowski, "Fast, Automated Thermal Simulation of Three-Dimensional Integrated Circuits," in *Proceedings of the Inter Society Conference on Thermal Phenomena*, 2004.

[18] L. Shang, L.-S. Peh, A. Kumar, and N. Jha, "Thermal Modeling, Characterization and Management of On-Chip Networks," in *IEEE Micro*, 2004.

[19] Skadron,Kevin and Stan,Mircea and others, "Temperature-Aware Microarchitecture," in *Proceedings of the 30th International Symposium on Computer Architecture (ISCA)*, 2003.

[20] B. Goplen and S. Sapatnekar, "Thermal via placement in 3D ICs," in *ISPD '05: Proceedings of the 2005 international symposium on physical design*. New York, NY, USA: ACM Press, 2005, pp. 167–174.

[21] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat, "3-D ICs: A Novel Chip Design for Improving Deep Submicrometer Interconnect Performance and Systems-On-Chip Integration," in *Proceedings of the IEEE*, vol. 89, no. 4, May 2001, pp. 602–633.

[22] Ansys Inc., "Ansys Multiphysics." [Online]. Available: http://www.ansys.com/products/multiphysics.asp

[23] Flomerics Corp., "Flotherm Modeling Software," Flomerics and Flotherm are Trademarks of Flomerics Corp. [Online]. Available: http://flotherm.com

[24] MSCSoft Inc., "MSC.Marc Nonlinear Analysis, Part of SimOffice." [Online]. Available: http://www.mscsoftware.com/products/products_detail.cfm?PI=1

[25] ThermoAnalytics Inc., "WinTherm Heat Transfer Simulation Software." [Online]. Available: http://www.thermoanalytics.com/products/wintherm/index.html

[26] J. L. Henning, "SPEC CPU2000: Measuring CPU Performance in the New Millennium," *Computer*, vol. 33, no. 7, pp. 28–35, 2000.

[27] E. Weisstien, "Correlation Coefficient," From MathWorld – A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/CorrelationCoefficient.html

[28] S. Gupta, M. Hilbert, S. Hong, and R. Patti, "Techniques for Producing 3D ICs with High-Density Interconnect," 2005, available from Tezzaron Semiconductor. [Online]. Available: http://www.tezzaron.com/about/papers/ieee%20vmic%202004%20finalsecure.pdf

[29] M. Taylor, J. Kim, J. Miller, *et al.*, "The Raw Microprocessor: A Computational Fabric for Software Circuits and General Purpose Programs," in *IEEE Micro*, Mar/April 2002.

[30] Electus Distribution, "Heatsink Basics," July 2005. [Online]. Available: http://www1.jaycar.com.au/images_uploaded/heatsink.pdf

[31] WaferNet Inc., "Availability for 300mm wafers and 200mm wafers." [Online]. Available: http://www.wafernet.com/products.html#300

[32] Intel Corp., "Single Edge Contact Connector 2 Thermal Interface Material Functional Requirements," November 1998, Accessed July 2005. [Online]. Available: http://www.intel.com/design/pentiumii/packtech/24445801.pdf

[33] PC Tech Talk.com, "Applying Arctic Silver Thermal Grease," July 2005. [Online]. Available: http://www.pctechtalk.com/?m=show&id=138

[34] Arctic Silver Inc., "Application Instructions for Premium Silver Thermal Compound on a Large Contact Area," 2005. [Online]. Available: http://www.arcticsilver.com/arctic_silver_instructions_big2.htm

[35] "International Technology Roadmap for Semiconductors," 2004. [Online]. Available: http://public.itrs.net

[36] AMD Corp., "AMD Athlon 64 Processor Power and Thermal Data Sheet," October 2004, accessed July 2005. [Online]. Available: http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/30430.pdf

[37] Y.-F. Tsai, A. H. Ankadi, N. Vijaykrishnan, M. J. Irwin, and T. Theocharides, "ChipPower: An Architecture-Level Leakage Power Simulator," in *Proceedings of the IEEE International SOC Conference*, September 2004.

[38] S. Borkar, "Design Challenges of Technology Scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23–29, July-August 1999.

[39] J. P. Gyvez and H. P. Tuinhout, "Threshold Voltage Mismatch and Intra-Die Leakage Current in Digital CMOS Circuits," *IEEE Transactions on Solid-State Circuits*, vol. 39, no. 1, pp. 157–168, January 2004.

[40] R. Rao, A. Srivastava, D. Blaauw, and D. Sylvester, "Statistical Esimation of Leakage Current Considering Inter- and Intra-Die Process Variation," in *ISLPED*, 2003, pp. 84–89.

[41] Y. Yamaji, T. Ando, *et al.*, "Thermal Characterization of Bare-Die Stacked Modules with Cu through-vias," in *Electronic Components and Technology Conference*, vol. 51, 2001, pp. 730–732.

[42] C. S. Tan and R. Reif, "Multi-Layer Silicon Layer Stacking Based on Copper Wafer Bonding," *Electrochemical and Solid-State Letters*, vol. 8, no. 6, pp. G147–G149, 2005.

[43] J. Mayega, O. Erdogan, P. M. Belemjian, K. Zhou, J. F. McDonald, and R. P. Kraft, "3D Direct Vertical Interconnect Microprocessors Test Vehicle," in *ACM Great Lakes Symposium on VLSI*, 2003, pp. 141–146.

[44] B. Black, D. W. Nelson, C. Webb, and N. Samra, "3D Processing Technology and its Impact on IA32 Microprocessors," in *Proceedings of the International Conference on Circuit Design (ICCD)*, October 2004, pp. 316–318.

[45] Intel Corp., "Intel Pentium III Processor for the PGA370 Socket at 500Mhz to 1.13Ghz," 2001, Available from *www.intel.com/design/pentiumiii/datashts/245264.htm*.

[46] Transmeta Corp., "The Crusoe Processor," information available at `http://transmeta.com/crusoe`.

[47] M. Kanellos, "At Intel - The Chip With Two Brains," Available on C-Net, ZDNet, or http://tinyurl.com/ak5xb, August 2002.

[48] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*, 2003, pp. 2–13.

[49] R. K Sharma et al., "Balance of Power: Dynamic Thermal Management for Internet Data Centers," HP Technical Report HPL-2003-5, 2003.

[50] A. Jantsch and H. Tenhunen, Eds., *Networks on chip*. Hingham, MA, USA: Kluwer Academic Publishers, 2003.

[51] E. Yeo, B. Nikolic, and V. Anantharam, "Architectures and Implementations of Low-Density Parity Check Decoding Algorithms," *IEEE International Midwest Symposium on Circuits and Systems*, August 2002.

[52] M. Mansour and N. Shanbhag, "Low-power vlsi decoder architectures for ldpc codes," in *Low Power Electronics and Design, 2002. ISLPED '02. Proceedings of the 2002 International Symposium on*, 2002, pp. 284–289.

[53] B. Levine, R. Reed Taylor, and H. Schmit, "Implementation of near shannon limit error-correcting codes using reconfigurable hardware," in *Field-Programmable Custom Computing Machines, 2000 IEEE Symposium on*, Napa Valley, CA, 2000, pp. 217–226.

[54] R. Gallager, "Low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[55] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Design Automation Conference, 2001. Proceedings*, 2001, pp. 684–689.

[56] W. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, 1992.

[57] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1997.

[58] D. Whelihan, "The NOCSim Simulator," 2004, available from `http://www.ece.cmu.edu/ djw2/NOCsim`.

[59] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 585–598, 2001.

[60] G. Al-Rawi, J. Cioffi, and M. Horowitz, "Optimizing the mapping of low-density parity check codes on parallel decoding architectures," in *Information Technology: Coding and Computing, 2001. Proceedings. International Conference on*, Las Vegas, NV, 2001, pp. 578–586.

[61] W.-L. Hung, Y. Xie, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Thermal-aware ip virtualization and placement for networks-on-chips architecture," in *In the Proceedings of the International Conference on Computer Design (ICCD)*, 2004.

[62] I. C. Society, "Ieee standard for binary floating-point arithmetic," in *IEEE Std 754-1985*, 1985.

[63] G. Hinton, D. Sager, M. Upton, D. Boggs, A. Kyker, and P. Roussel, "The microarchitecture of the pentium 4 processor," in *Intel Technical Journal, Q1 Issue*, 2001.

[64] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically characterizing large scale program behavior," in *In the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, October 2002.

[65] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective*. Addison Wesley, 1994.

[66] B. Sheu, D. Scharfetter, P.-K. Ko, and M.-C. Jeng, "Bsim: Berkeley short-channel igfet model for mos transistors," *IEEE Journal of Solid-State*, vol. 22, no. 4, pp. 558–566, 1987.

[67] K. Shakeri and J. D. Meindl, "Temperature Variable Supply Voltage for Power Reduction," in *IEEE Computer Society Annual Symposium on VLSI*, 2002.

[68] A. Bellaouar, A. Fridi, M. Elmasry, and K. Itoh, "Supply voltage scaling for temperature insensitive cmos circuit operation," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 45, no. 3, 1998.

[69] W. Zhao and Y. Cao, "New generation of Predictive Technology Model for sub-45nm design exploration," in *ISQED*, 2006.

[70] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," in *CICC*, 2000.

[71] H. Bakoglu and J. Meindl, "Optimal interconnection circuits for vlsi," *IEEE Transactions on Electron Devices*, vol. 32, no. 5, 1985.

[72] J. Cong and D. Pan, "Interconnect estimation and planning for deep submicron designs," in *Proceedings of the Design Automation Conference*, 1999.

[73] T.-C. Chen, S.-R. Pan, and Y.-W. Chang, "Performance optimization by wire and buffer sizing under the transmission line model," in *International Conference on Computer Design (ICCD)*, 2001.

[74] A. Ajami, K. Banerjee, and M. Pedram, "Analysis of substrate thermal gradient effects on optimal buffer insertion," in *International Conference on Computer Aided Design (ICCAD)*, 2001.

[75] A. Ajami, M. Pedram, and K. Banerjee, "Effects of non-uniform substrate temperature on the clock signal integrity in high performance designs," in *IEEE Conference on Custom Integrated Circuits*, 2001.

[76] ..., "Blind reference," in ..., 2005.

[77] R. Tamhankar, S. Murali, and G. De Micheli, "Performance driven reliable link design for networks on chips," in *Proceedings of the Asia and South Pacific Design Automation Conference.*, vol. 2, 2005.

[78] Y. Cao, T. Sato, M. Orshansky, D. Sylvester, and C. Hu, "New paradigm of predictive mosfet and interconnect modeling for early circuit simulation," in *IEEE Custom Integrated Circuits Conference*, 2000, pp. 201–204. [Online]. Available: http://www.eas.asu.edu/~ptm

[79] S. Im, N. Srivastava, K. Banerjee, and K. Goodson, "Scaling analysis of multilevel interconnect temperatures for high-performance ics," *IEEE Transactions on Electron Devices*, 2005.

[80] K. Banerjee and A. Mehrotra, "Analysis of on-chip inductance effects for distributed rlc interconnects," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 8, 2002.

[81] T. Austin, D. Blaauw, T. Mudge, and K. Flautner, "Making typical silicon matter with razor," *Computer*, vol. 37, no. 3, 2004.

[82] C. Poirier, R. McGowen, C. Bostak, and S. Naffziger, "Power and temperature control on a 90nm itanium-family processor," in *Solid-State Circuits Conference (ISSCC)*, 2005.

[83] W.-L. Hung, G. Link, Y. Xie, N. Vijaykrishnan, and M. Irwin, "Interconnect and thermal-aware floorplanning for 3d microprocessors," in *Quality Electronic Design, 2006. ISQED '06. 7th International Symposium on*, 2006, pp. 98–104.

[84] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.

[85] J. Dongarra, "Basic Linear Algebra Subprograms Technical Forum Standard," in *International Journal of High Performance Applications and Supercomputing*, 2002, pp. 1–111 and 115–199.

[86] The Gigascale Systems Research Center, "VLSI CAD Bookshelf 2," 2006. [Online]. Available: http://www.gigascale.org

[87] AMD Corp., "AMD Athlon Multi-Core Processors White Paper," May 2006, accessed May 2006. [Online]. Available: http://multicore.amd.com/WhitePapers/Multi-Core_Processors_WhitePaper.pdf

[88] W.-L. Hung, Y. Xie, G. M. Link, and J. Conner, "Temperature-aware voltage islands architecting in system-on-chip design," in *In the Proceedings of the International Conference on Computer Design (ICCD)*, October 2005.

[89] G. M. Link and N. Vijaykrishnan, "Hotspot prevention through runtime reconfiguration in network-on-chip," in *In the Proceedings of Design, Automation, and Test in Europe (DATE)*, March 2005.

[90] J. Hu, G. M. Link, J. K. John, S. Wang, and S. G. Ziavras, "Resource-driven optimizations for transient-fault detecting superscalar microarchitectures," in *In the Proceedings of the Asia-Pacific Computer Systems Architecture Conference (ACSAC)*, October 2005.

Vita

Gregory M. Link

Greg Link was born in Altoona Pennsylvania, on September 23rd, 1979. He began his undergraduate career at Juniata College, where he participated in the 3-2 engineering program, in participation with the Pennsylvania State University. In 2002, he was awarded a degree in general physics with honors from Juniata College, and a degree in electrical engineering with highest distinction from the Pennsylvania State University. In the fall of 2002, he enrolled in the doctoral program in the department of computer science and engineering at the Pennsylvania State University.

From 2002-2003, he was supported by a University Fellowship, and began working under the advisement of Dr. N. Vijaykrishnan. During 2003-2004, he worked as a research assistant for Dr. Vijaykrishnan in the Microsystems Design Laboratory. For the remainder of his graduate career, he was supported by the University's prestigious Academic Computing Fellowship, receiving one of the two awards given annually. His research interests include microprocessor architecture, on-chip interconnect (specifically network-on-chip), and adaptive computer systems.

He will begin a position as an assistant professor of electrical and computer engineering at the York College of Pennsylvania in the fall of 2006.