TU Delft

Thermal-Aware Design and Runtime Management of 3D Stacked Multiprocessors

SUMEET KUMAR

Thermal-Aware Design and Runtime Management of 3D Stacked Multiprocessors

Thermal-Aware Design and Runtime Management of 3D Stacked Multiprocessors

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen
op dinsdag 15 september 2015 om 12:30 uur
door

Sumeet Susheel KUMAR

Master of Science in Microelectronics, Technische Universiteit Delft

geboren te Kuwait City, Kuwait

This dissertation has been approved by:

Promotor:

Prof. dr. ir. A.-J. van der Veen

Copromotor:

dr. ir. T.G.R.M. van Leuken

Composition of the doctoral committee:

Rector Magnificus,	voorzitter
Prof. dr. ir. A.-J. van der Veen	Technische Universiteit Delft, promotor
dr. ir. T.G.R.M. van Leuken	Technische Universiteit Delft, copromotor
dr. A. Zjajo	Technische Universiteit Delft

Independent members:

Prof. dr. J. Pineda de Gyvez	Technische Universiteit Eindhoven
Prof. dr. F. Pêcheux	Laboratoire d'Informatique de Paris-6
Prof. dr. D. Stroobandt	Universiteit Ghent
Prof. dr. K.L.M. Bertels	Technische Universitieit Delft
Prof. dr. ir. G.J.T. Leus	Technische Universitieit Delft, reservelid

The research described in this dissertation was performed in the Circuits and Systems Group, Faculty of Electrical Engineering, Computer Science and Mathematics, of Delft University of Technology, Delft, The Netherlands. This work was partially supported by the CATRENE programme under the Computing Fabric for High Performance Applications (COBRA) project CA104.

ISBN 978-94-6186-513-7

Copyright © 2015 by Sumeet Susheel Kumar [*sumeetskumar@ieee.org*]

Printed by: Gildeprint Drukkerijen - The Netherlands

Summary

The sustained increase in computational performance demanded by next-generation applications drives the increasing core counts of modern multiprocessor systems. However, in the dark silicon era, the performance levels and integration density of such systems is limited by thermal constraints of their physical package. These constraints are more severe in the case of *three-dimensional (3D)* integrated systems, as a consequence of the complex thermal characteristics exhibited by stacked silicon dies. This dissertation investigates the development of efficient, thermal-aware multiprocessor architectures, and presents methodologies to enable the simultaneous exploration of their thermal and functional behaviour.

Chapter 2 examines the efficiency of multiprocessor architectures from the perspective of the the memory hierarchy, and presents techniques that focus on the effective management and transfer of on-chip data in order to minimize the time spent waiting on memory accesses. In the case of shared-memory multiprocessors, this is achieved through the proposed *Persistence Selective Caching (PSC)* and *CacheBalancer* schemes that influence what data is stored in on-chip caches, where it is stored, and for how long. This enables the memory hierarchy to adapt to changing execution behaviour, balance resource utilization, and most importantly, reduce the average latency and energy per memory access. Further to this, Chapter 2 presents the *Pronto* system, which enables efficient data transfers in message-passing multiprocessors by minimizing the role of the processing element in the management of transfers. Pronto effectively decreases the overheads incurred in setting up and managing data transfers, thereby yielding shorter communication latencies. In addition, it also simplifies the semantics of data movement by abstracting implementation details of communications from the programmer, thus enabling transfers to be specified entirely at the task level.

The issue of thermal-aware design for 3D *Integrated Circuits (IC)* using *Nagata's* equation – a mathematical representation of the dark silicon problem – is investigated in Chapter 3. Significantly, the chapter explores the thermal design space of 3D ICs in terms of this equation, and proposes a high-level flow to characterize the specific

influence of individual design parameters on thermal behaviour of die stacks. The results of this exploration advance the state-of-the-art by providing new insights into the critical role of power density, thermal conductivity and stack construction in the formation of hotspots in 3D ICs. Building on these insights, the *Ctherm* framework is proposed for the thermal-aware design of *multiprocessor systems-on-chip (MPSoC)*. Ctherm enables the concurrent evaluation of thermal and functional performance of MPSoCs using automatically generated fine-grained area, latency and energy models for system components, and facilitates the exploration of thermal behaviour early in the system design flow. The efficacy of the framework is demonstrated using a number of practical design cases ranging from floorplanning and temperature sensor placement to application tuning. Together, the characterization and the Ctherm framework further our understanding of the thermal behaviour of die stacks, and provide a practical template for the realization of thermal-aware electronic design automation tooling for 3D ICs.

The management of thermal issues that arise in 3D MPSoCs at runtime is examined in Chapter 4. Temperature control is typically exercised by means of *Dynamic Thermal Management (DTM)* which continuously adapt the activity and power dissipation of system components. A significant disadvantage of state-of-the-art DTMs lies in their inability to account for the non-uniform thermal behaviour of die stacks, leading to the ineffective management of temperatures and in degraded system performance. In Chapter 4, a novel 3D *Dynamic Voltage Frequency Scaling (DVFS)* scheme is proposed that takes these non-uniformities into account within its power management algorithm, effectively maintains operating temperatures within a safe range, and maximizes system performance within the available thermal margins at individual processing elements. Furthermore, the chapter also presents an adaptive routing strategy to decrease the magnitude of thermal gradients in network-on-chip based 3D architectures, by directing traffic along paths of low temperature. The proposed *Immediate Neighbourhood Temperature (INT)* adaptive routing scheme actively steers interconnect traffic away from regions with thermal hotspots based only on temperature information available in the immediate neighbourhood, relying on the heat transfer characteristics of 3D ICs to avoid the need for a global temperature monitoring network. The consequent spreading of interconnect activity over multiple paths results in balanced thermal profiles, and decreased operating temperatures across the system.

Over the course of these chapters, this dissertation explores the critical issues impeding the realization of thermal-aware 3D stacked multiprocessors, and details a multifaceted approach towards addressing the challenges of dark silicon.

For my mother and father,

You have no idea how hard I've looked for a gift to bring you. Nothing seemed right. What's the point of bringing gold to the gold mine, or water to the ocean. Everything I came up with was like taking spices to the Orient. It's no good giving my heart and my soul because you already have these. So I've brought you a mirror. Look at yourself and remember me.

Rumi

Contents

1 Introduction **1**

1.1 Motivation 4
  1.1.1 Architectural Efficiency 4
  1.1.2 Thermal Constraints 5
  1.1.3 Temperature Management 6
1.2 Research Questions 6
1.3 Dissertation Outline 7
1.4 Publication List 9

2 Architectural Techniques for Efficient On-Chip Data Management **11**

2.1 Naga Architecture Overview 13
2.2 Low-overhead Message Passing with Pronto 15
  2.2.1 Related Work and Motivation 16
  2.2.2 The Pronto Message Passing System 17
    2.2.2.1 Pronto API 17
    2.2.2.2 Hardware Architecture 19
  2.2.3 Experimental Evaluation 22
    2.2.3.1 End-to-end Message Transfer Latency 24
    2.2.3.2 Communication Overheads 25
    2.2.3.3 Application performance with Pronto 26
    2.2.3.4 Impact of input dataset size 28
    2.2.3.5 Impact of extraneous interconnect traffic on output jitter 28
  2.2.4 Conclusions 31
2.3 Improving Data Cache Performance using Persistence Selective Caching 32
  2.3.1 Related Work 32

2.3.2 Persistence Selective Caching 33
2.3.2.1 Selective Caching Criteria 34
2.3.2.2 Significance of Persistence Threshold 36
2.3.2.3 Limitations 37
2.3.3 Evaluation 38
2.3.3.1 AMAT and Energy 38
2.3.3.2 Overheads and Implementation Cost 41
2.3.4 Conclusions 41
2.4 Runtime Management of Shared Caches using CacheBalancer . . . 42
2.4.1 CacheBalancer 43
2.4.1.1 Access Rate based Memory Allocation 43
2.4.1.2 Pain-driven Task Mapping 45
2.4.2 Evaluation 47
2.4.3 Conclusions 51

3 Exploring the Thermal Design Space in 3D Integrated Circuits 55
3.1 Significance of parameters 56
3.2 Thermal Characterization of Die Stacks 57
3.2.1 Experimental Setup and Validation 60
3.2.2 Characterization 61
3.2.2.1 Thermal Conductivity (κ_{eff}) 61
3.2.2.2 Die Thickness and Stack Depth ($l_{x,y,z}$) 64
3.2.2.3 Power Density (Q/A) 65
3.2.3 Conclusions 66
3.3 Vertical Interconnect 66
3.3.1 Electrical Performance 67
3.3.2 Area 69
3.3.3 Thermal Performance 70
3.3.4 Conclusions 72
3.4 Thermal-Aware Design Space Exploration 72
3.4.1 Ctherm Framework 75
3.4.1.1 Physical Model Generation 75
3.4.1.2 Thermal-Functional Co-simulation Platform . . . 79
3.4.2 Evaluation 81
3.4.2.1 Validation, Accuracy and Simulation Speed . . . 81
3.4.2.2 Design Cases 84
3.4.3 Additional Media 89
3.4.4 Conclusions 90

4 Runtime Temperature and Power Management for 3D Multiprocessors 93

4.1 Temperature-Aware DVFS for Stacked Die Architectures 95
  4.1.1 3D DVFS 96
    4.1.1.1 Initial Updates 98
    4.1.1.2 Thermal Runout 98
    4.1.1.3 Convergence Check 99
    4.1.1.4 Pull Up/Pull Down 100
    4.1.1.5 Write-Back and Reset: 100
  4.1.2 Implementation Considerations 100
  4.1.3 Evaluation 101
    4.1.3.1 Per-core Granularity 102
    4.1.3.2 Island Granularity 104
  4.1.4 Conclusions 107
4.2 Temperature-Aware Adaptive Routing for Dynamically-Throttled 3D Networks-on-Chip 108
  4.2.1 Background 109
  4.2.2 Immediate Neighbourhood Temperature (INT) Adaptive Routing 110
    4.2.2.1 Temperature Monitoring 111
    4.2.2.2 Temperature Channel Considerations 111
    4.2.2.3 Thermal-Aware Dynamic Throttling 112
    4.2.2.4 Temperature-Aware Adaptive Routing Algorithm 113
  4.2.3 Evaluation 114
    4.2.3.1 Characterization of Throttling 114
    4.2.3.2 INT Evaluation 115
  4.2.4 Conclusions 119

5 Conclusions **123**
Additional Contributions 128
Future work 129

Acronyms **133**

Notation **137**

Bibliography **141**

Samenvatting **155**

Acknowledgements **159**

Curriculum Vitae **165**

Propositions **167**

Stellingen **171**

1

Introduction

The increasing computational requirements of next-generation applications is an important driver for the development of high-performance microprocessors. Desktop processors from the early 2000s supported performance in the range of 100 *billion operations per second (BOPS)*, and traditionally utilized increasingly higher clock frequencies to scale performance. Consequently, these devices were rated with a *thermal design power (TDP)* of over $100W$ [1], necessitating the use of extravagant heatsinks and exotic methods for cooling [2]. Small form factor computing devices such as mobile phones and ultrabooks on the other hand impose extremely restrictive TDPs. For instance, in a modern smartphone, the digital workload consisting of control, data and signal processing aggregates to over 100 BOPS, however, with a power budget of only $1W$ [1]. Furthermore, even though the performance requirements of this workload increase by two orders of magnitude every five years, the power budget grows only minimally.

Chip multiprocessors (CMP)[1] are an effective means of realizing such high computational performance. CMPs integrate a large number of simple *processing elements (PE)* that dissipate a relatively small amount of power, into a single *integrated circuit (IC)* package [3]. Workloads are divided into tasks that execute concurrently on PEs, yielding performance improvements that surpass conventional frequency upscaling. CMPs are thus based on the rationale that high performance can be realized better through *computing in strength* rather than *computing in speed*, with superior power efficiency. The viability of the concept is evident from the number of production ready CMPs in the market. Offerings from Ambric [4–7], PicoChip [8, 9], Tilera [10–12], Intellasys [13, 14] and NEC [15] integrate anywhere between 40 and 336 processing

---

[1] Also referred to simply as multiprocessors

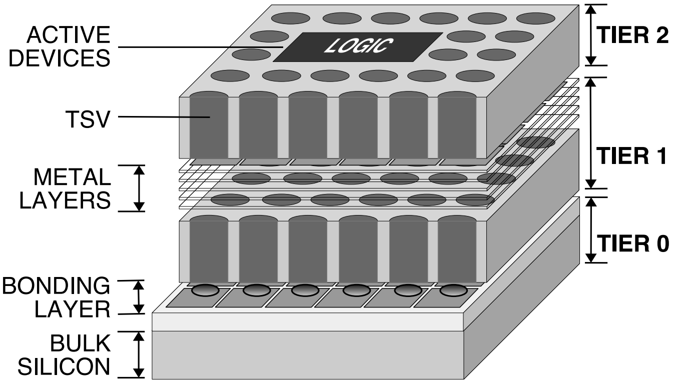
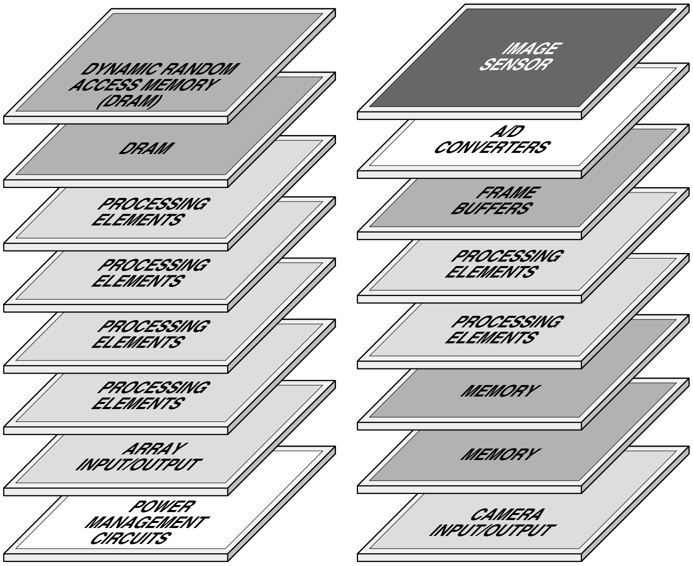


Figure 1.1. Cut away of a die stack illustrating *Through Silicon Via (TSV)* based vertical interconnections

elements within a single chip. The application domains for these processors are increasingly in the computing, multimedia and signal processing areas, involving large data sets with high throughput requirements. For instance, the 248-core PicoChip PC203 is primarily intended as a baseband processor for wireless networks. Similarly, the 128-core NEC IMAPCAR serves as an image recognition processor at the heart of automotive collision avoidance systems [16]. The physical constraints imposed by their operating environments, in both cases, limit TDP to under $3W$.

Although the addition of PEs to multiprocessor arrays improves system performance, technology-related challenges limit the extent to which such arrays can be scaled up. The first challenge stems from the increase in die size that accompanies the integration of additional PEs. As yield decreases with increasing die size [17, 18], performance gains are obtained at the cost of manufacturability. The second, yet equally important challenge arises due to the limited *input/output (I/O)* bandwidth of pin-constrained multiprocessor and memory packages [19], which results in the performance improvements from the additional PEs being diminished due to memory and I/O contention [20].

Three-dimensional (3D) integration is a promising solution to these limitations, facilitating the realization of large multiprocessors in the form of a stack of silicon dies [21–23]. The stacked dies are interconnected by means of vertical metal wires known as *Through Silicon Vias (TSV)*, as illustrated in Figure 1.1. 3D integration essentially reduces the area footprint of multiprocessors by converting planar area into stack height, effectively reducing the size of individual dies, and thus improving manufacturing cost. It further facilitates the integration of dies varying in function-



(a) (b)

Figure 1.2. Illustration of 3D integrated systems. (a) Multiprocessor with stacked DRAM, I/Os and power management circuits (b) Integrated computer vision system with stacked image sensor, data conversion circuitry and processing elements.

ality and process technology node into a single IC package [20, 24]. Consequently, components such as the *Random Access Memory (RAM)* can be integrated within the stack [25–28], and made accessible to PEs through a high-bandwidth wide I/O interface [19], allowing system performance to be scaled with PE counts. Potentially, 3D could also be used to enable fully integrated systems incorporating sensors, data converters and PEs, as illustrated in Figure 1.2.

Current application trends indicate that future workloads will require computational performance in the range of 1 *trillion operations per second (TOPS)* [1], necessitating the use of *many-core* CMPs. At such large scales, however, architectural inefficiencies have a significant impact on both performance, as well as dependability. In addition, despite the benefits of 3D integration, die stacks exhibit complex thermal behaviour that can be detrimental to system performance.

1.1 Motivation

The challenges accompanying the efficient design and dependable operation of large scale CMPs can be grouped into three categories - architectural efficiency, thermal constraints, and runtime temperature management.

1.1.1 Architectural Efficiency

Multiprocessor architectures can be broadly classified based on their communication model into two types - shared-memory and message-passing architectures. Shared-memory multiprocessors use a global memory space that is shared amongst all PEs. Data transfer between tasks executing on PEs is implicit, and is managed in hardware by the underlying memory hierarchy consisting of multiple on-chip caches. However, synchronization and data sharing must be explicitly managed within shared-memory architectures, and this increases complexity in applications with significant inter-task communication. In message-passing based dataflow architectures [29] on the other hand, applications are described as a set of communicating tasks, with well defined input and output dependencies. Communicating tasks run asynchronously on separate PEs and exchange data between their local memories. Tasks fire once their inputs become valid, and are thus implicitly synchronized. In comparison with shared-memory, message-passing incurs a higher overhead since data transfers must be explicitly managed due to the absence of a global address space.

In both message-passing as well as shared-memory architectures, execution time refers to the amount of time required to complete execution of all constituent tasks within a given application program. For a task executing on a PE, the execution time (t_{task}) is given as:

$$t_{task} = t_{instructions} + t_{memory} \tag{1.1}$$

where $t_{instructions}$ represents the fraction of the task's execution time spent in arithmetic, logic and control instructions, and t_{memory} the time spent performing memory load-stores. $t_{instructions}$ is largely a function of the code's complexity and size, as well as the PE's microarchitecture - specifically, factors such as instruction latency, branch predictor accuracy and issue-width [30]. On the other hand, t_{memory} is influenced by the management and transfer of on-chip data, and includes the time spent waiting for data to be fetched from remote memories, or lower levels of the memory hierarchy. The access latency of the memories, layout of data across memory banks, and the overheads incurred in managing transfers between them together contribute to t_{memory}, and influence the effective performance of the system. The efficient management and transfer of on-chip data is therefore essential to realizing high-performance many core CMPs.

1.1.2 Thermal Constraints

Power dissipation by components in ICs results in the generation of heat, which causes operating temperatures to rise. The generated heat is therefore evacuated from the system by means of a heatsink so as to maintain temperatures within the safe operating range, and prevent device failure [31]. Nagata [32] determined that the maximum allowable power dissipation in an IC is constrained by its physical construction, and the thermal efficiency of its cooling interfaces. Their relationship is given as:

$$\frac{\alpha N_G E}{t_{pd}} \leq g \cdot \Delta T \tag{1.2}$$

where ΔT is the maximum permissible difference between on-chip and ambient temperatures, and represents the available temperature margin at zero power dissipation. The relation thus indicates that for a planar IC, the activity rate (α), energy dissipation (E), clock period (t_{pd}) and number (N_G) of gates that can be integrated into a single chip is limited by the thermal conductance (g) of its interface with the ambience, as well as the ambient temperature. This limitation best describes the phenomenon termed by the semiconductor industry as *Dark Silicon*, a reference to the large sections of modern ICs powered down due to thermal considerations. Essentially, Nagata's equation dictates that to improve integration densities, components must either be utilized less, or must dissipate a smaller amount of power (E/t_{pd}). Alternatively, either conductance to ambient must be improved, for instance using a heatsink with a larger surface area, or ambient temperature must be decreased. The latter serves to increase the magnitude of available temperature margin ΔT.

The issue of dark silicon is further complicated in the case of die stacks on account of their distinct thermal behaviour as compared to planar ICs. In 3D ICs, thermal conductance g is a function of the physical construction of the die stack, and the TSV-based vertical interconnect. The value of g drops as distance from the heatsink increases and as a result, the thermal constraints imposed by (1.2) vary throughout the die stack. System design approaches that ignore this effect run the risk of yielding thermally inefficient designs that inadequately utilize available temperature margins. In order to maximize the performance of a stacked die architecture, it is essential that the unique thermal characteristics of 3D ICs be taken into account during early stages of system design. A significant obstacle to achieving this lies in our relatively shallow understanding of the thermal behaviour of die stacks, and the influence of the design parameters established by Nagata's equation on operating temperatures. The realization of a thermal-aware design flow for 3D architectures is consequently predicated on the characterization of the thermal design space for die stacks.

1.1.3 Temperature Management

The operating temperature at any point in the IC is dependent on the amount of heat generated within the system, and the rate at which it is conducted away towards heat sinking surfaces. The power dissipation of components is determined by α and E, which vary depending on the nature of the workload being executed on the multiprocessor, and t_{pd}. When α is balanced, all PEs dissipate a similar amount of power, and produce a uniform power density that results in heat generation spread across the complete area of the IC. However, imbalances in α can lead to a spike in spatial power density and cause the formation of thermal hotspots. *Dynamic Thermal Management (DTM)* strategies are typically invoked in such circumstances to arrest rising temperatures and maintain them within safe margins. Such an action, on the other hand, imposes a performance penalty. The unique thermal characteristics of 3D ICs further complicate this behaviour by producing non-uniform temperature margins at different tiers of the stack. In order to uniformly extract the full performance of PEs in a multiprocessor, the DTM strategy must take into account the thermal characteristics of 3D ICs and the non-uniformities in temperature margins.

Thermal gradients are another undesirable consequence of unbalanced activity in multiprocessors, and result in the accelerated degradation of devices [31][33]. Reducing the magnitude of these gradients requires temperature awareness within the architecture, and mechanisms to dynamically steer system activity away from regions of high temperature.

1.2 Research Questions

The presented motivations can be condensed into the four research questions that are addressed by this dissertation.

1. How can the performance and efficiency of on-chip memory operations in multiprocessors be improved?
2. How do the physical design parameters in Nagata's equation affect the thermal behavior of 3D Integrated Circuits?

3. How can the knowledge of thermal behaviour be effectively leveraged in the design of 3D stacked multiprocessors?
4. How can the architecture and operating parameters be efficiently adapted at runtime to mitigate the severity of thermal issues, and improve execution performance?

Together, these questions encapsulate the key challenges and issues impeding the design and efficient operation of 3D stacked multiprocessor systems.

1.3 Dissertation Outline

This dissertation presents architectural techniques to enable the realization of efficient, high-performance multiprocessors, and facilitate runtime temperature management to ensure their dependable operation. Most importantly, it provides new insights into the complex thermal behaviour of 3D ICs, and illustrates how the design space of stacked die architectures can be effectively explored in order to maximize performance in the dark silicon era. This dissertation consists of two main themes, architecture and temperature, examined in light of the research questions outlined in the previous section.

Chapter 2 presents techniques that aim at minimizing the amount of time spent by PEs in waiting on memory accesses (t_{memory}), as well as decreasing the energy dissipated within the memory hierarchy, through the efficient management of on-chip data. These include:

- *Pronto* - a message-passing system that decreases the overheads for data transfers between communicating tasks in dataflow multiprocessors, yielding shorter transfer latencies than competing schemes.
- *Persistence Selective Caching (PSC)* - a selective caching scheme for first level data caches. PSC minimizes t_{memory} by decreasing the average latency for memory accesses, and improves efficiency by reducing the energy dissipated per access.
- *CacheBalancer* - a runtime resource management scheme that balances the utilization of shared caches in multiprocessors, reducing the latency of accesses to dynamically allocated memory, as well as the system's energy density.

Chapter 3 investigates the complex thermal characteristics of die stacks in terms of the design space described in (1.2). A high-level exploration flow is presented to examine the influence of stack composition, physical construction, power density and design of the vertical interconnect on the thermal behaviour of 3D ICs. This exploration provides new insights into the formation of hotspots in die stacks, and the role of individual design parameters in their mitigation.

In order to apply these insights, Chapter 3 presents the *Ctherm* framework for the thermal-aware design of *multiprocessor systems-on-chip (MPSoC)*. Ctherm facilitates the concurrent evaluation of the thermal and functional performance of MPSoCs, enabling the holistic exploration of candidate design options. The framework automates the generation of fine-grained area, latency and energy models for components in order to accurately model power density, and hence thermal behaviour. Chapter 3 highlights the impact of modelling component internals on the accuracy of thermal estimates, and illustrates the potential of thermal-aware approaches across the design flow.

Chapter 4 proposes DTM strategies for runtime temperature management in 3D MPSoCs. These include:

- *Temperature-Aware Dynamic Voltage Frequency Scaling (DVFS)* - a runtime power manager that takes into account the non-uniformities in temperature margins within die stacks when scheduling voltage frequency levels for processing elements in the 3D MPSoC. The proposed strategy maximizes execution performance within the non-uniform thermal constraints prevalent in die stacks, and outperforms the conventional DVFS approach.

- *Immediate Neighbourhood Temperature (INT)* Adaptive Routing - an adaptive routing strategy that balances thermal gradients and decreases hotspot magnitudes in 3D *networks on chip (NoC)* by actively steering interconnect activity away from regions of high temperature. The high degree of thermal coupling between stacked dies eliminates the need for system-wide propagation of temperature information, and enables adaptive routing decisions to be driven by a simple temperature monitoring network.

Together, the two strategies alleviate the severity of thermal issues in 3D MPSoCs, and maximize the performance of stacked-die multiprocessors within available temperature margins at runtime.

1.4 Publication List

The contributions of this dissertation have been disseminated through a number of refereed conference, journal and book publications, in addition to poster presentations and live demonstrations. Individual chapters include at the end a listing of their relevant publications. A comprehensive list of all the publications arising out of this dissertation is provided here as a general overview.

Book Chapters

1. S.S. Kumar, A. Zjajo, R. van Leuken, ”Exploration of the Thermal Design Space in 3D Integrated Circuits”, *Physical Design for 3D Integrated Circuits*, CRC Press, December 2015, *Invited Book Chapter*

Journal Papers

1. S.S. Kumar, A. Zjajo, R. van Leuken, ”Immediate Neighbourhood Temperature Adaptive Routing for Dynamically-Throttled 3D Networks-on-Chip” *IEEE Transactions on Circuits and Systems II (TCAS-II), in press*

2. S.S. Kumar, M.T.A. Djie, R. van Leuken, ”Pronto: A Low Overhead Message Passing System for High Performance Many-Core Processors.” *International Journal of Networking and Computing - Special Issue*, vol. 4, no. 2, pp. 307-320, July 2014

3. S.S. Kumar, A. Aggarwal, R. Jagtap, A. Zjajo, R. van Leuken, ”System Level Methodology for Interconnect Aware and Temperature Constrained Power Management of 3-D MP-SOCs” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 7, pp. 1606-1619, July 2014

Conference Papers

1. S.S. Kumar, A. Zjajo, R. van Leuken, ”Physical Characterization of Steady-State Temperature Profiles in Three-Dimensional Integrated Circuits” *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015

2. S.S. Kumar, A. Zjajo, R. van Leuken, ”Ctherm: An Integrated Framework for Thermal-Functional Co-simulation of Systems-on-Chip ” *Proceedings of the IEEE/Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pp.674-681, 2015

3. J. de Klerk, S.S. Kumar, R. van Leuken, ”CacheBalancer: Access Rate and Pain Based Resource Management for Chip Multiprocessors”, *Proceedings of the International Symposium on Computing and Networking (CANDAR)*, pp. 453-456, 2014
4. S.S. Kumar, R. van Leuken, ”Improving data cache performance using Persistence Selective Caching,” *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp.1945-1948, 2014
5. S.S. Kumar, M.T.A. Djie, R. van Leuken, ”Low Overhead Message Passing for High Performance Many-Core Processors,” *Proceedings of the International Symposium on Computing and Networking (CANDAR)*, pp. 345-351, 2013
6. R. Jagtap, S.S. Kumar, R. van Leuken, ”A Methodology for Early Exploration of TSV Placement Topologies in 3D Stacked ICs” *Proceedings of the IEEE/Euromicro Conference on Digital System Design (DSD)*, pp.382-388, 2012
7. A. Aggarwal, S.S. Kumar, A. Zjajo, R. van Leuken, “Temperature constrained power management scheme for 3D MPSoC,” *Proceedings of the IEEE Workshop on Signal and Power Integrity (SPI)*, pp. 7-10, 2012

Posters and Demonstrators

1. S.S. Kumar, A. Aggarwal, R. Jagtap, A. Zjajo, R. van Leuken, “Interconnect and Thermal Aware 3D Design Space Exploration”, Invited Presentation and Poster, *ICT.OPEN*, Eindhoven, The Netherlands, 2013
2. S.S. Kumar, R. van Leuken, A. Michos, A. Chahar, J. de Klerk, “Naga High-Performance Array Processor”, Poster and Demonstrator, *University Booth – Design Automation and Test in Europe (DATE)*, Grenoble, France, March 2013

2

Architectural Techniques for Efficient On-Chip Data Management

The evolution of microprocessors from single-core towards the present day many-core is driven by the performance requirements of next-generation applications. The effective translation of their large PE counts into actual execution performance, however, remains hinged upon the efficiency of their underlying hardware and software architectures.

Reiterating from Chapter 1, the execution time (t_{task}) for a task executing on a PE is given as:

$$t_{task} = t_{instructions} + t_{memory} \tag{2.1}$$

where $t_{instructions}$ represents the fraction of the task's execution time spent in arithmetic, logic and control instructions, and t_{memory} the time spent performing memory load-stores. While $t_{instructions}$ is primarily a function of the PE's microarchitecture, t_{memory} is influenced by the efficiency of on-chip data management. Specifically, in the case of message-passing architectures, t_{memory} depends on the efficiency of data transfer between distributed memories, and the efficiency of the memory hierarchy in the case of shared-memory architectures.

In message-passing dataflow architectures, tasks execute concurrently on separate PEs, and communicate in a producer-consumer fashion according to their task graph [29]. Tasks begin execution once their inputs become ready, and upon completion, pass their output data to the next waiting task. Since communication in conventional message-passing architectures [34] is managed explicitly, the latencies for these operations is reflected in the execution time of tasks.

$$t_{task} = (t_{computation} + t_{mp}) + (t_{transfer} + t_{fc}) \tag{2.2}$$

Thus, in addition to the time $t_{computation}$ spent performing computations, t_{mp} is spent in executing message-passing library functions to manage transfers. Furthermore, although the actual transfer incurs an aggregate latency of $t_{transfer}$, an additional overhead t_{fc} is incurred in synchronizing and implementing flow control between the communicating tasks.The magnitude of t_{mp} and t_{fc} together indicate the efficiency of the message-passing implementation, and influence the overheads incurred during data transfers. Reducing these overheads is essential in improving execution performance.

In cache-based shared-memory architectures, t_{memory} can be represented in terms of *average memory access time (AMAT or t_{AMA})*, a metric that indicates the efficiency of the memory hierarchy.

$$t_{task} = t_{instructions} + (M_A \cdot t_{AMA}) \tag{2.3}$$

$$where\ t_{AMA} = \mu_{hit} \cdot t_{hit} + \mu_{miss} \cdot t_{miss} \tag{2.4}$$

M_A refers to the total number of memory accesses made by the application, μ_{hit} and μ_{miss} are the hit and miss rate of the data cache, with latency t_{hit} and t_{miss} respectively. Although this equation holds for both instruction as well as data caches, we focus on the data memory hierarchy alone. *Hit rate* refers to the fraction of memory references for which the requested data is found within the PE's private *Level-1 data cache (L1D)*, while the remaining references constitute misses that necessitate a fetch operation from lower levels of the memory hierarchy. The latency for a reference resulting in a hit (t_{hit}) depends on the configuration of the data cache, and is typically in the range of $1-3$ clock cycles [35]. The penalty for a miss (t_{miss}), on the other hand, is much larger, and depends on the cache line size, bandwidth of the interconnect, miss rate of the lower level caches and their miss penalty. In *network-on-chip (NoC)* based multiprocessors, t_{miss} is also influenced by the communication distance (hop count) between PEs and the lower level caches, as well as by contention within the interconnect and shared caches. Improving execution performance for cache-based multiprocessors necessitates efficient data management within the system so as to increase hit rate, as well as minimize t_{hit} and t_{miss}.

In this chapter, we present three architectural techniques for efficient on-chip data management for both message-passing as well as shared-memory multiprocessors. These include:

1. *Pronto*: A low overhead message-passing system which simplifies the semantics of data movement between communicating tasks by performing buffer management, message synchronization and address translation directly in hardware. This results in transfer latencies upto 30% shorter than state-of-the-art proposals. In terms of (2.2), Pronto reduces t_{mp} by minimizing the role of the PE in

managing transfers, and t_{fc} through its use of reservation-based message flow control.

2. *Persistence Selective Caching (PSC)*: A selective caching scheme that identifies reusable data at runtime, and services references to them from a low-latency assist cache. This reduces the hit latency t_{hit} for a majority of references, and yields t_{AMA} (AMAT) upto 59% lower than conventional data caches. In addition, the small size of the assist, coupled with its high hit rate result in a 75% reduction in average energy per access.

3. *CacheBalancer*: A runtime resource management scheme that balances the utilization of shared caches, and minimizes the cost of fetching data from lower levels of the memory hierarchy. CacheBalancer reduces shared cache contention by up to 60% and improves execution performance by 22%.

2.1 Naga Architecture Overview

The base platform for this dissertation is a generic many-core processor architecture - *Naga* - targeted towards the acceleration of applications in the multimedia and computing domains. The distinct characteristics of each of these domains makes them well suited for either shared-memory, or message-passing architectures. While this makes the choice of communication model trivial for applications from either of the two domains, the general purpose nature of Naga necessitates the inclusion of both models in the architecture. The *Alewife* machine [36] was based on a similar line of reasoning, where Kubiatowicz argued that the advantages of a dual model stemmed from practical design considerations including communication cost, memory access latencies and programmability [37]. While the end goal for such integration is to enable flexibility in the selection of a communication model at the task level, this leads to a number of complications pertaining to the management of the memory address space, and interference between communications of the two models [37]. For the purpose of this dissertation, we avoid the complexities of such fine-grained mixed models, and instead utilize two separate sub-arrays, each implementing either shared-memory or message-passing. Naga thus incorporates two sub-arrays – the message-passing *NagaM*, and shared-memory *NagaS*. An illustration of these sub-arrays is shown in Figure 2.1. Among the architectural techniques presented in this chapter, Pronto uses the NagaM as its base system, while PSC and CacheBalancer use the NagaS.

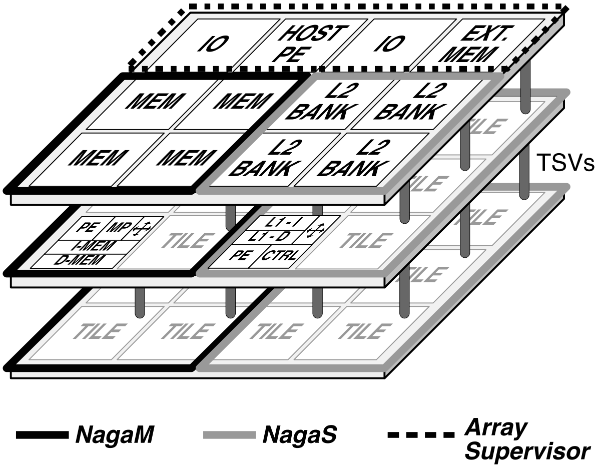


Figure 2.1. Illustration of the Naga many-core with the shared-memory based NagaS, and message-passing NagaM sub-arrays

The *NagaM* sub-array is a dataflow accelerator primarily for multimedia and signal processing workloads. It uses ρ-VEX *Very Long Instruction Word (VLIW)* PEs [38] which provide a performance benefit by exploiting inherent *Instruction Level Parallelism (ILP)* within executing tasks. PEs are placed within tiles containing private data and instruction memories, and a message passing communication interface. At runtime, tasks are spawned and pinned onto PEs according to the task graph, with communicating nodes mapped as close to one another as possible, by a runtime mapper in the host processor. Each task executes asynchronously on a ρ-VEX PE upon its input data becoming ready, and produces data that similarly triggers the next task in the graph. Fast dual-ported memories (MEM) serve as data I/O for the accelerator, and store the input data to, and output data from the head and tail of the task graph respectively. Although multiple are shown in Figure 2.1, we assume only a single dual-ported memory in the array for simplicity. A conceptual overview of the NagaM accelerator is provided in Figure 2.2.

NagaS on the other hand, is a conventional shared-memory architecture with a two-level cache memory hierarchy, intended for general purpose compute applications. Tiles incorporate a simple *Reduced Instruction Set Computing (RISC)* PE, private instruction and data caches, and a cache controller. In order to maximize cache memory bandwidth, L2 cache banks implement slicing [39] to split the shared address

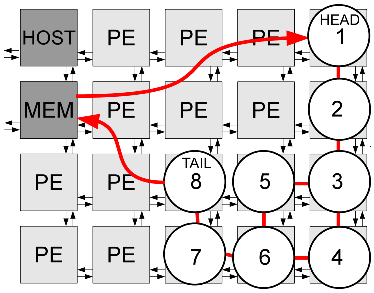


Figure 2.2. NagaM accelerator with host processor (HOST), and dual ported memory buffer (MEM). Only the Head and Tail tasks of the mapped task graph read from and write to the memory buffer respectively.

space into *slices*, with each mapping to a single bank [40]. NagaS uses the threaded model of execution, allowing applications executing on the host to spawn threads on the array's PEs.

Both NagaM and NagaS use the *R3* NoC interconnect [41]. R3 is a wormhole routed, best-effort, packet-switched NoC that enables stacked-die architectures by facilitating the creation of 3D meshes with *Through Silicon Via (TSV)* based vertical links. The network uses a dimension-ordered *Z-X-Y* routing algorithm that routes interconnect traffic based only on source and destination network addresses. The simplistic architecture of the R3 NoC allows us to evaluate the actual impact of the architectural techniques presented in this chapter without any performance boosts due to interconnect optimizations. Interconnect performance is treated separately in Chapter 4.

2.2 Low-overhead Message Passing with Pronto

Existing message-passing implementations rely largely on feature-rich software libraries to manage the transfer of messages between PEs. Thus, in addition to specifying what data must be moved between executing tasks, the programmer must also manage the actual transfer and the corresponding resource reservations. This is detrimental for two reasons. Firstly, it results in communication operations being managed through the PE, thus increasing execution time as well as communication latency. Secondly, it requires the implementation aspects of the underlying message passing communications architecture to be exposed to the programmer, thereby increasing complexity.

In this section we present *Pronto*, a low overhead message passing system for many-core processors. Data transfers with Pronto are initiated using a compact set of simple yet highly effective functions that provide a layer of abstraction separating the programmer's view of inter-task communication, and its actual implementation in the underlying hardware architecture. Operations such as address translation, synchronization of transfers and resource management are handled entirely in hardware, simplifying the programming model and minimizing the time spent by PEs in executing non-task related operations.

2.2.1 Related Work and Motivation

A number of many-core processors, both in academia as well as the industry, implement message passing for inter-task communication. For instance, the 430-core *picoArray* uses basic message passing *put* and *get* functions to transfer data between concurrently executing tasks [42]. During compilation, tasks are mapped onto PEs and their communication flows converted into interconnect schedules. Since interconnect arbitration and resource reservations are performed at compile-time, communications do not incur any additional latency penalties related to these operations are runtime. The dataflow based *Ambric Massively Parallel Processor Array* [5] implements a similar methodology although with a hierarchical interconnect structure. The *Intel SCC* [43] on the other hand performs all required reservations at runtime rather than statically. Message passing is implemented through a global shared address space accessible through each PE's *Message Passing Buffer (MPB)* [44]. Tasks executing on PEs share data through virtual connections established by dynamically allocating common memory objects within this space, using functions from the *RCCE* library [45]. Synchronization, ordering of messages and shared accesses must be managed through a programmer-enforced protocol in software.

Apart from these implementations, there also exist individual message passing proposals based on the *MPI* standard [46] often with specific objectives. For instance, *QoS-ocMPI* adds *Quality of Service (QoS)* support into a subset of MPI functions, specifically for NoC based multiprocessors [34], thus allowing critical transfers to occur through a reserved channel, i.e. with *throughput guarantees*. Another proposal, *TMD-MPI* [47], adapts MPI towards supporting message passing between processors across multiple *Field Programmable Gate Arrays (FPGA)*. It essentially abstracts the complexity of inter-chip communication, instead providing the programmer with a homogeneous view of the system. Despite their merits, these proposals are largely based on the original MPI standard, which itself is intended for large distributed memory systems [46]. This objective of the standard reflects in the overheads incurred due to its use in resource constrained many-core processors. Psota and Agarwal noted this in their proposal *rMPI*, indicating the need for a simple message passing *Application Programming Interface (API)* with a small memory footprint to replace MPI in chip multiprocessors [48].

The drawbacks of heavy software libraries reflect primarily in the latency of data transfers. Proposals without static scheduling and resource reservations often require the MPB and synchronization of data transfers to be explicitly managed by the programmer. These operations are performed through functions of the software library executed on the PE, and result in a non-zero t_{mp} that is dependent on the operation's latency. Consequently, the latency incurred to setup and manage transfers is higher than if the same were managed in hardware. Therefore, by removing the need for explicit management of the MPB and synchronization of data transfers through function calls, the latency of transfers could be greatly reduced. This would also serve to abstract the implementation of the message passing system from the programmer, and simplify the semantics of inter-task communication.

2.2.2 The Pronto Message Passing System

The performance gains of many-cores over sequential implementations are quickly lost as communication overheads approach task execution times [48]. In order to maximize throughput of the many-core array, it is important that message transfer latencies be kept low. By implementing transfer management functions in hardware, PEs are released from having to explicitly oversee data transfers, thereby allowing them to perform useful work instead. Pronto uses a *Direct Memory Access (DMA)* engine based message passing system for data transfers. Data blocks are moved between tile-local memories using hardware managed *Message Passing Buffers (MPB)* over the R3 NoC interconnect. Figure 2.3 illustrates the architecture of a NagaM PE tile with the Pronto message-passing interface.

2.2.2.1 Pronto API

Executing tasks communicate through calls to four simple message passing functions of the Pronto API, as listed in Table 2.1. These functions are essentially shells that set Pronto's hardware registers with the parameters of the message transfer. In contrast to the heavy *send* and *receive* primitives of existing message passing libraries, our API's functions are extremely light-weight, consisting only of a few writes to memory mapped control registers.

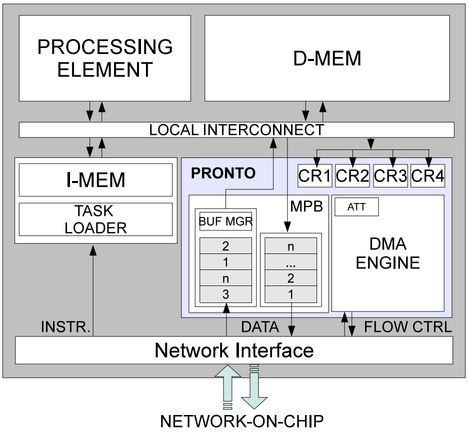


Figure 2.3. NagaM tile containing a ρ-VEX processing element, local memories, Pronto message passing interface and a network interface

Table 2.1. Pronto Message Passing API for NagaM

Function	Arguments
MP_send()	destination task id, length, local memory address of data
MP_receive()	source task id, length, local memory address for data
MP_mread()	local memory address for data, length, MEM address
MP_mwrite()	local memory address of data, length, MEM address

The *MP_send* and *MP_receive* functions are always called in pairs between communicating tasks, with the calls specifying only the size of the message, its location in the tile's local memory, and the sender/recipient's task ID. This provides a high level of abstraction, hiding details such as the actual physical PE onto which tasks are mapped. Each argument of the function calls maps to a particular control register of the Pronto interface, as listed in Table 2.2. The Pronto architecture allows programmers to extend the software API by defining multiple message types through the *CR4* control register. During message transfers, the contents of this register are encoded into the message header (also known as *message envelope*), enabling control signaling between executing tasks.

2.2.2.2 Hardware Architecture

The control registers together with the software API act as an interface between the executing application code and the Pronto hardware. Rather than actually performing

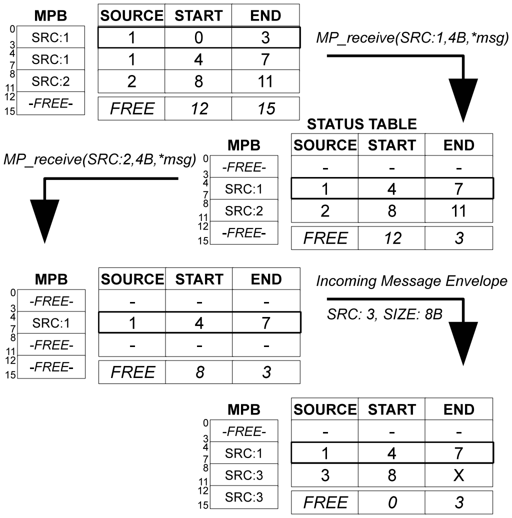
Table 2.2. Control Register Mappings

	CR1	**CR2**	**CR3**	**CR4**
MP_send()	Local memory address	Length	Dest. Task ID	Type (DAT)
MP_mwrite()	Local memory address	Length	MEM address	Type (MWR)
MP_receive()	Source Task ID	Length	Local memory address	Type (DAT)
MP_mread()	MEM address	Length	Local memory address	Type (MRD)

the transfer through software, the message passing functions of our API only configure Pronto's control registers to initiate transfers between communicating tasks. The actual transfer is performed and managed by the hardware architecture itself. The following subsections examine Pronto's management of the MPB, flow control and synchronization of messages, and its abstraction of physical addressing from the programmer.

(i) Address Translation As previously mentioned, the *MP_send* and *MP_receive* functions specify message transfers using task IDs of the recipient and source respectively, instead of their physical PE addresses. This is enabled by a per-tile *Address Translation Table (ATT)* programmed during task mapping, which translates programmer specified task IDs into the physical network addresses of the corresponding PEs. Consequently, the communication semantics for Pronto completely abstract details such as the physical address of PEs, and allow all inter-task communications to be specified at the task level itself. In addition to reducing the complexity of programming using message passing, this abstraction also permits task mappings to be adapted at runtime without requiring the software to be recompiled since physical addresses of PEs are not specified anywhere in the code.

(ii) Buffer Management Before any data can actually be transmitted, it is essential for the sending node's message passing interface to determine whether sufficient free space exists in the downstream MPB. This is achieved through the use of a *message envelope* containing the source node's physical address, the amount of MPB space requested and the type of the message. Envelopes are handled at the downstream node on a first-come-first-served basis, with accepted envelopes resulting in the MPB reserving the requested chunk of memory for the impending message. The *buffer manager* actively tracks the utilization of the MPB through a status table, as shown in Figure 2.4. Upon arrival of each message, the buffer manager translates the source node address into its corresponding task ID, and places this information together with the MPB memory address at which the message is located into a free tuple of the status



SOURCE	START	END
1	0	3
1	4	7
2	8	11
FREE	12	15

SOURCE	START	END
-	-	-
1	4	7
2	8	11
FREE	12	3

SOURCE	START	END
-	-	-
1	4	7
-	-	-
FREE	8	3

SOURCE	START	END
-	-	-
1	4	7
3	8	X
FREE	0	3

Figure 2.4. Illustration of buffer management and message ordering in the *Message Passing Buffer (MPB)*

table in a circular *FIFO*-like manner. A pointer indicates the oldest waiting message entry in the table, illustrated as an emboldened tuple in Figure 2.4. A successful reservation results in an acknowledgement to the upstream node indicating that the transfer may commence. In the event of insufficient MPB space, the corresponding envelope is buffered until the requested space becomes available. Therefore, no negative acknowledgements are returned, preventing repeated envelope transmissions from the stalled sender. Since only a single envelope is required per message regardless of its size, the overhead it poses remains fixed, and is quickly amortized during burst transfers.

Envelopes are generated automatically once an *MP_send* call moves a complete block of data into the MPB. Therefore, destination MPB reservations are handled automatically by the DMA engine rather than explicitly by the programmer. The motivation for using a message envelope is two fold:

1. The R3 NoC used in the NagaM enforces a protocol allowing for a maximum payload of 64B (16 words) per packet. Larger payloads are split into multiple packets, each of which is arbitrated separately by the R3 router's round robin arbiter. Multiple tasks communicating concurrently with a downstream task would result in the latter's MPB being inundated with only parts of messages, necessitating a buffer of a larger capacity. On the other hand, the use of message

envelopes and the reservation based message flow control system ensures that received messages can always be stored as a whole, and that transfers commence only upon reservation of sufficient storage in the MPB. Furthermore, the mechanism simplifies buffer management by allocating memory on a per-message basis rather than per-source.

2. The message envelope and reservation based message flow control further ensure that packets belonging to messages in flight do not end up blocked in router FIFOs due to a full downstream MPB. Given the NoC's best effort nature, this would lead to blocked links, and give rise to the possibility of network deadlocks due to the absence of time-outs and packet dropping in the R3 architecture. Our mechanism therefore separates flow-control and buffering for the message passing system from that of the NoC.

Multiple requests from different upstream nodes to a single MPB are handled sequentially, although once accepted, transfers may proceed concurrently. This is possible since the buffer manager allocates disjoint blocks of memory to each transfer, allowing received words to be placed in their appropriate MPB memory locations simply based on their source. The *MP_send* function does not specify the destination memory address for any transfer. Where this data is placed in the receiving node's local memory is determined by the arguments of the *MP_receive* call at the destination, essentially simplifying the semantics of data movement in the system. Needless to say, each node may only hold one request (both active and pending) to any particular downstream node at any given point in time. Furthermore, words constituting a message must form a contiguous block in memory, i.e. they must be located at sequential memory addresses.

(iii) Ordering of Messages at Destination The buffer manager preserves the entry order of incoming data blocks using the status table, ensuring that the oldest received block is popped from the buffer when requested by the executing task. In the case of concurrent tasks with uneven loads where the upstream task generates multiple data blocks during a single run of the downstream task, this mechanism guarantees that blocks are consumed in the same order as they are generated. Received blocks are moved into the local data memory of the PE once the *MP_receive* function with the corresponding source task ID is called.

In case a task produces more than one type of output data, a programmer defined protocol must be enforced to order the *MP_send* and corresponding *MP_receive* calls. This is because the functions do not include any details of the destination memory address for the remote task, thus making it difficult to determine which data block the message contains. Given the nature of dataflow based programs and their definite input-output dependencies, this ordering is trivial to enforce. Therefore, if a task generates two outputs and sends them in one order, the downstream task must call *MP_receive* in this exact same order. This is illustrated in Figure 2.4 which shows the MPB of a destination node receiving messages from a number of nodes, even before older messages already waiting in the buffer are consumed. When *MP_receive* is called, the waiting messages from the requested source task ID are returned to the PE in the order in which they arrived.

The *MP_receive* function is blocking, and hence stalls the PE until data from the specified source is received by the MPB. The *MP_send* function, on the other hand, is non-blocking except for when the local MPB's output buffer is full in addition to the downstream MPB's input buffer. In this case, execution is stalled by clock-gating the local PE. Proper load-balancing of tasks to ensure that they incur similar execution times minimizes the occurrence of such buffer full/empty stalls. We illustrate this in the following subsection with the *JPEG* decoder.

2.2.3 Experimental Evaluation

We evaluate Pronto using a cycle-accurate HDL based simulation model of NagaM. The model uses 18 ρ-VEX processing elements connected over a 4x5 mesh topology network, with a single data memory buffer from which task graphs fetch their input data, and write their output to. Although a practical hardware implementation would place limitations on the size of this buffer, for the purpose of our simulations, we impose no such constraints. This does not affect the validity of the presented results since the evaluation focuses primarily on the message passing system within the array, and its consequent impact on application performance. The MPB is sized at 512B (128-words) for the input, and 256B (64-words) for the output.

Three dataflow workloads are used to analyze the performance impact and scalability of Pronto: *JPEG* decoder, *Moving Average FIR* filter and a custom test workload. The *JPEG* decoder from the *MiBench* benchmark suite [49] implements the decoding of JPEG images into the Bitmap format. The conversion process involves three stages, namely *Huffman decoding*, *Inverse Discrete Cosine Transform (IDCT)* and *colour conversion*. The original sequential implementation of the JPEG decoder from the benchmark suite was parallelized manually by converting each of its three stages into concurrently executable tasks, with the Pronto API functions used for data transfer. After initial experiments, a more effective four-stage JPEG decoder was developed to overcome inefficiencies noted in our three-stage implementation. The two versions are

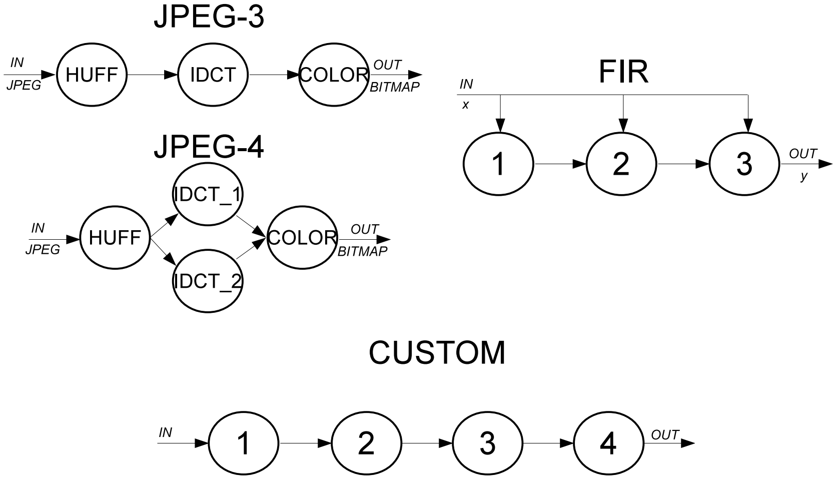


Figure 2.5. Task graphs for the *JPEG-3*, *JPEG-4*, *FIR* and *CUSTOM* workloads.

identified as JPEG-3 and JPEG-4, with the suffix signifying the number of concurrent stages in their task graphs. The input data set for these workload consists of a 512x512 pixel JPEG encoded image.

The *Moving Average FIR filter* workload is used in signal processing applications to remove unwanted noise in signals. The filter essentially implements the equation listed in (2.5), where *x* and *y* represent the input and output signals respectively, with *N* samples.

$$y[i] = \frac{1}{N}x[i] + \frac{1}{N}x[i-1] + \frac{1}{N}x[i-2] + ... + \frac{1}{N}x[i-N-1] \tag{2.5}$$

The nature of this algorithm allows it to be partitioned into multiple concurrent tasks, each with a similar computational load. However, partitioning may only be beneficial upto a certain point, after which communication latencies become comparable to the execution time of tasks themselves, thus limiting further performance gains. The *Custom* application represents an ideal dataflow workload with identical concurrent tasks. Such partitioning can be expected to minimize execution stalls. Figure 2.5 illustrates the task graphs for the JPEG-3, JPEG-4, Moving Average FIR filter and Custom workloads.

The evaluation of Pronto consists of five separate experiments that determine:

1. End-to-end message transfer latency

Table 2.3. Comparison of average transfer latency per word

	LATENCY PER WORD (CYCLES)	BURST SIZE (WORDS)
OCMPI [34]	32.9	256
[50]-SHARED QUEUE	20	64
[50]-SCRATCH QUEUE DMA	9	64
PRONTO	**6.48**	64

2. Communication overheads
3. Application performance with Pronto
4. Impact of input dataset size
5. Impact of extraneous interconnect traffic on output jitter

The following subsections describe each of these experiments, and provide an overview of the obtained results.

2.2.3.1 End-to-end Message Transfer Latency

The performance of Pronto is first evaluated in terms of its message transfer latency per hop, i.e. the latency incurred in transferring a message between two adjacent nodes. For this, two tasks are pinned onto neighbouring PEs in the NagaM array. The first task generates a burst of 64 data words and transfers these using an *MP_send* call to the second task which then receives the burst using *MP_receive*. In order to accurately estimate the transfer latency, these measurements are performed without any extraneous interconnect traffic (zero network load). The obtained latencies are listed in Table 2.3. The same table also includes the transfer latency for similar sized bursts from literature - Francesco's Shared Queue and Scratch Queue DMA [50], and the MPI derivative for multiprocessors with on-chip interconnects - ocMPI [34]. Pronto is observed to have a transfer latency 30% lower than the closest distributed memory based proposal, Scratch Queue DMA [50]. Note that the use of a larger burst size of 256 words works in favour of ocMPI since the overheads of transfer setup are better amortized by large bursts. Despite this, the overall per word transfer latency of ocMPI is observed to be significantly larger than that of Pronto, indicating the higher transfer overheads of MPI-based systems.

The transfer of the message envelope and the downstream node's acknowledgement of buffer reservation impose a one-time latency overhead for each message. While message envelopes indicate the source node and quantum of MPB space

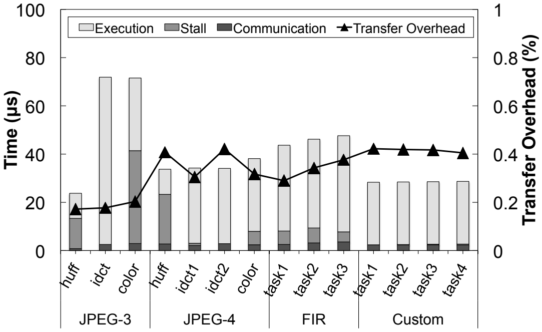


Figure 2.6. Breakdown of task execution as a fraction of its total execution time. The *transfer overhead* reflects the overhead imposed by message envelopes as a percentage of total execution time.

required at the destination, the former is already included into the packet header according to the R3's protocol. Therefore, the message envelope in NagaM is a 2-flit packet consisting of the header and a single flit containing an integer value of the required MPB space. The envelope length remains the same regardless of message size. A 64 word message on the R3 NoC is sent in 4 packets, or 68 flits in total. A single message envelope and the corresponding downstream MPB acknowledgement result in 3 additional flits (2 for the envelope and 1 for the acknowledgement) being exchanged between the nodes. This constitutes an overhead of under 5% for a 64 word message.

2.2.3.2 Communication Overheads

In order to determine the transfer overhead for messages in terms of total execution time, we mapped single instances of workload task graphs onto the array with zero network load. Figure 2.6 illustrates the fraction of total execution time spent in execution, stalls due to a full/empty MPB and in communication across different workloads. The same figure also indicates the overhead imposed by message envelopes as a fraction of total execution time. As previously mentioned, only one envelope and its corresponding acknowledgement are generated for each message transfer. Consequently, the number of envelopes and acknowledgements exchanged over the interconnect depends only on the number of messages transferred, and not their size. In general, the transfer overhead of the message envelopes constitutes less than 0.5% of the total execution time across all workloads. The time spent stalled due to a full/empty buffer is primarily caused by imbalances between the tasks, and this can be reduced by precise partitioning and load-balancing. Frameworks such as Daedalus [51] enable such analysis and help in precise partitioning of workloads for high performance and scalability.

The consequences of inefficient partitioning in the case of the JPEG-3 workload are also illustrated in Figure 2.6. Initial runs of the workload on the NagaM array revealed an imbalance in the runtime of its three constituent stages. The IDCT stage in particular was observed to run close to six times as long as the Huffman decoding stage, resulting in repeated execution stalls for the latter. The IDCT stage was subsequently partitioned further into two concurrent tasks to address the imbalance in task loads, as shown earlier in Figure 2.5. The resulting implementation reduced buffer-related execution stalls by 38% and reduced execution time by 45% as compared to the three-stage version. Although the number of message transfers in both implementations is identical, Pronto's transfer overhead appears higher in the case of JPEG-4 due to the reduced execution time.

2.2.3.3 Application performance with Pronto

Execution performance can be improved in two ways - by increasing the number of concurrently executing tasks through fine grained partitioning, and by increasing the number of instances of the task graph executing in parallel. We observe from Figure 2.7 that the former does not always yield significant returns. In the case of the FIR filter for instance, the speedup obtained through fine-grained partitioning tends to flatten out beyond 6 tasks for a 6400 sample input size as a consequence of the reduction in computational load per task to a level where communication latencies become significant.

Instantiation of multiple instances of the task graph on the other hand allows for exploitation of data-level parallelism, thus achieving greater speedup and higher throughput. Figure 2.8(a) reports the execution speedup for the workloads with a varying number of parallel instances of the task graph, over sequential execution on the host PE. A linear improvement in speedup is observed as the number of parallel instances executing on the NagaM array is increased. The corresponding throughputs for these workloads considering a 200MHz clock frequency are reported in Figure 2.8(b). Note that since the FIR workload generates output data blocks of size 376B as against 256B for the Custom workload, the two yield very similar throughputs despite their significantly different speedups. In comparison, the JPEG decoder generates larger output data blocks of size 768B, thus explaining its prominently higher throughput.

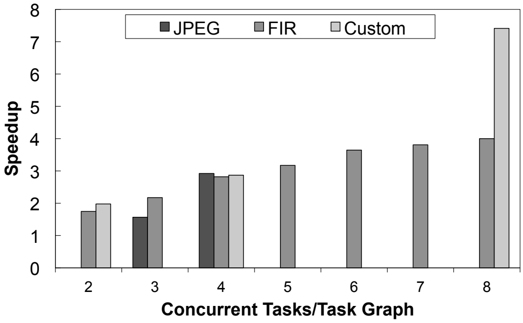


Figure 2.7. Performance improvements obtained with fine-grained partitioning

2.2.3.4 Impact of input dataset size

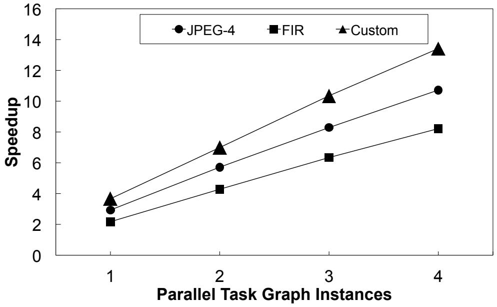
The runtime of all workloads is influenced by the size of their input datasets. When the number of concurrent tasks per task graph as well as the number of parallel instances of the task graph are fixed, the runtime can be expected to increase as the input dataset size is increased. A longer runtime can however be beneficial as it tends to amortize the impact of communication and configuration overheads. Most significantly, a longer runtime softens the impact of ATT configuration that occurs when the task graph is spawned on the NagaM array. To estimate the performance impact of such overheads, we varied the input dataset size for workloads, effectively changing their runtime. Figure 2.9 reports the speedup obtained over sequential execution across different dataset sizes.

The figure indicates small improvements in speedup with increasing dataset size. Note that with the 64x64/800 samples dataset, the overall execution time for most workloads is low enough for the the ATT configuration operation to constitute a moderate overhead. With larger dataset sizes on the other hand, speedup improvements are less pronounced, since the configuration operation no longer forms an appreciable fraction of total execution time. These results suggest that overheads of Pronto within the NagaM architecture are sufficiently low so as to yield similar speedups across a range of input dataset sizes.

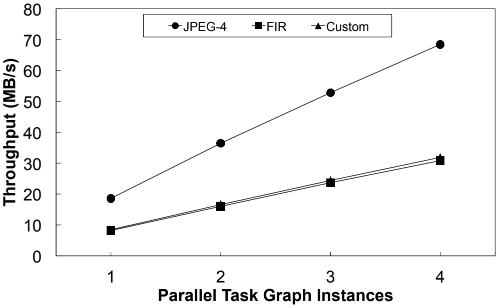
2.2.3.5 Impact of extraneous interconnect traffic on output jitter

Given the best-effort nature of NagaM's NoC, it is prudent to evaluate the impact of extraneous interconnect traffic on the variation in arrival rate of data blocks.

The eventual integration of NagaM and NagaS into a single multiprocessor array



(a)



(b)

Figure 2.8. (a) Execution speedup relative to sequential execution on the host PE (sequential exection times - *JPEG-4*: 105.3μs, *FIR*: 115.8μs and *Custom*: 106.1μs) (b) Throughput at 200MHz

will cause the message-passing and shared-memory architectures to share a common system interconnect. For this reason, it is prudent to evaluate the impact of extraneous interconnect traffic due to the shared-memory NagaS, on the variation in arrival times for data blocks in the message-passing NagaM. This is done by emulating cache related traffic in the interconnect through a set of *Traffic Injectors (TI)* placed at the North and South edges of the array. Injectors at the northern edge emulate cache miss and write-back requests directed towards those on the southern edge. These requests vary in size from 4B (cache miss) to 64B (cache line write-back) at various injection rates, emulating extremely pessimistic miss rates. The injectors on the southern edge respond with appropriately sized packets to the requesting injector, as illustrated in Figure 2.10. Multiple parallel instances of a task graph are mapped onto PEs of the array, with task data blocks moving in a direction orthogonal to the injected synthetic

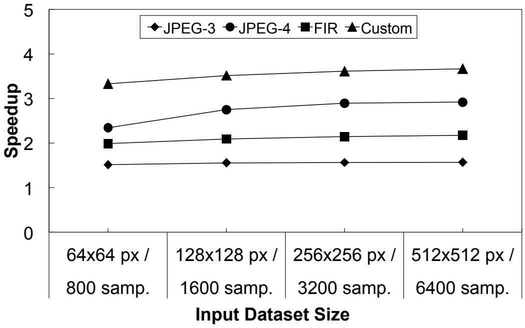
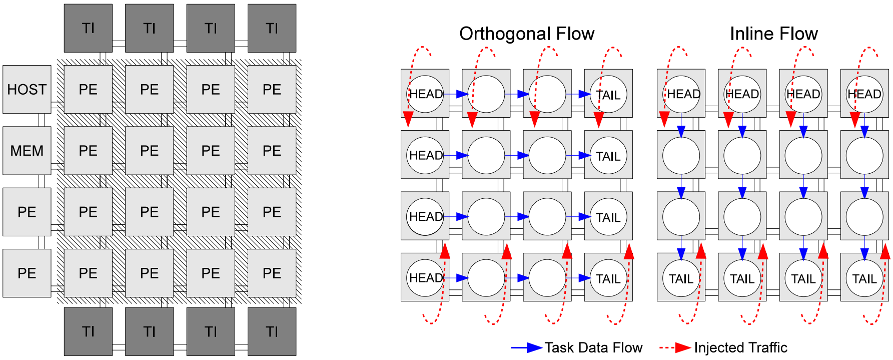


Figure 2.9. Speedup with varied input dataset sizes

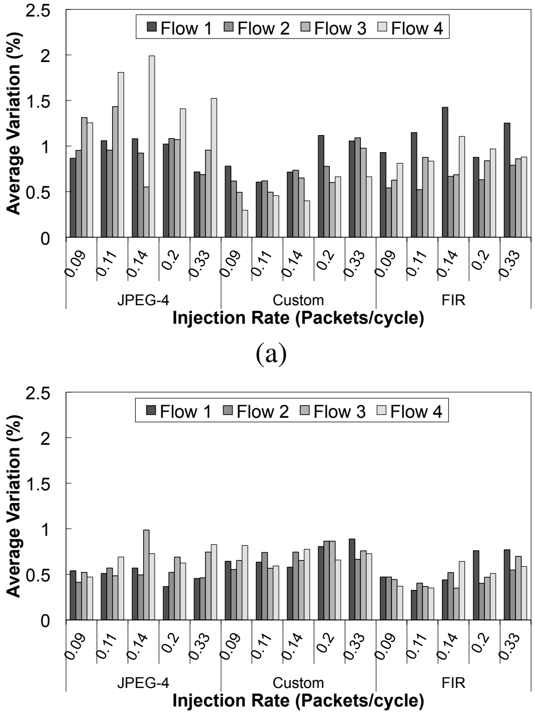


(a) (b)

Figure 2.10. (a) NagaM array with traffic injectors simulating cache traffic. Tasks are mapped to PEs within the highlighted region of the array. (b) Illustration of relative directions of task data and injected traffic - orthogonal and inline.

traffic. The output jitter is measured as the variation from expected arrival time for data blocks at the memory buffer *(MEM)* averaged over the entire execution of the workload for a given input dataset.

The measured output jitter for workloads at different injection rates for the case when task data blocks and synthetic traffic flow in orthogonal directions is reported in Figure 2.11(a). In order to provide a comparison, we adapted the traffic injectors and task mapping such that the injected traffic and task data blocks flow inline with one another as shown in Figure 2.10(b). The average variation across workloads and injection rates is observed to drop from the earlier peak of 2% to under 1% with this



(a)

(b)

Figure 2.11. Average arrival time variation for: (a) orthogonal flows and (b) inline flows

new mapping. This can be seen in Figure 2.11(b). Rather than the injection rate, it is the relative direction of interconnect traffic that significantly influences arrival time variations for data blocks.

In the first case, the XY routing algorithm of the network results in increased contention in the North-South network links on account of their utilization by both injected traffic as well as data blocks moving to and from the memory buffer. As a consequence, the head stages of all graph instances remain stalled until their requested data blocks arrive, resulting in accumulation of the delay at all subsequent stages. In the second case, due to the location of the head task for each task graph instance, input data blocks are routed in a direction orthogonal to the injected traffic. Consequently, input data blocks encounter little contention in their path, and therefore do not delay task execution. Output data blocks from the tail moving towards the memory buffer similarly incur minimal delays. Contention at the memory buffer itself, on the other hand, contributes to the variations observed for workloads in Figure 2.11(b).

Although interconnect traffic affects the transfer latencies for intermediate data blocks moving between stages of a task graph, the actual impact is minimized due to the relatively smaller hop count for such transfers as compared to those directed at the memory buffer. This is a consequence of the mapping algorithm's placement of communicating tasks on neighbouring cores, often resulting in single hop transfers. Even in the case of JPEG-4 where the *IDCT_1* and *IDCT_2* tasks communicate with the colour conversion and Huffman stages over a multi-hop path, interconnect contention is seen to have a smaller impact on output jitter. Interestingly, it is also observed that due to imbalances in execution times of workload tasks, arrival time variations are evened out by buffer full stalls, resulting in decreased jitter.

These results indicate that contention is better tolerated by data blocks moving between intermediate stages of the task graph, than by those moving between the head/tail tasks and the memory buffer. Contention in the path of data blocks moving to and from the memory buffer are observed to cause high arrival time variations. Mapping strategies that result in such critical data flows following orthogonal paths to extraneous interconnect traffic can significantly reduce output jitter, as observed in Figure 2.11(b).

2.2.4 Conclusions

This section presented the Pronto low overhead message passing system for many-core processors. By implementing functions to manage message transfer, synchronization, address translation and buffer management directly in hardware, Pronto reduces message transfer latencies by upto 30% in comparison to state-of-art DMA-based proposals. The reservation based message flow control system implemented through the use of message envelopes imposes an overhead of under 5% for 64 word burst transfers, constituting less than 0.5% of the total execution time of workloads such as the JPEG decoder and FIR filter. In addition to the low latency, the presented system simplifies the semantics of data movement by abstracting the implementation details of the communications architecture from the programmer, enabling data transfers to be specified at the task level. The speedup over sequential execution obtained with Pronto in an 18-core NagaM array is found to scale linearly with the number of parallel task graph instances, with similar performance across a range of input dataset sizes. An analysis of the impact of the relative direction of task data flows and extraneous interconnect traffic on the arrival time for output data indicated that blocks moving between the memory buffer and the head/tail of task graphs result in the highest arrival time variations when delayed due to contention. These effects are mitigated by adapting the task mapping to ensure that performance critical data move in a direction orthogonal to extraneous interconnect traffic. This finding is critical towards enabling the future integration of NagaM and NagaS.

2.3 Improving Data Cache Performance using Persistence Selective Caching

The limited size of L1 caches in comparison to the data set of modern applications leads to the emergence of expensive misses, necessitating latency and energy intensive accesses to lower levels of the memory hierarchy (t_{miss}). Although large set-associative caches appreciably reduce miss rates, their size causes them to have a higher hit latency (t_{hit}), and consume more energy per access than smaller direct-mapped caches.

This section presents the *Persistence Selective Caching (PSC)* scheme which reduces *average memory access time* (*t_{AMA} or also AMAT*) through the selective caching of reusable lines in a small, fully-associative assist cache. The reuse potential of a line is estimated at runtime based on its *access persistence*, i.e. the number of accesses to the line within a certain window of data references by the processor. Lines with sufficient access persistence are moved from the *L1 data cache (L1D)* into the assist cache from where subsequent references to them are serviced. Due to the assist's small size, these references incur a shorter hit latency, and consume considerably lesser energy than an L1D access. PSC's selectivity ensures that only the most reusable lines are moved to the assist, leading to a significant reduction in the number of cache line movements (*swaps*), and thus lower energy per access than competing schemes.

2.3.1 Related Work

A number of studies have, in the past, used small memory buffers to augment the capacity of the main L1D, and thus improve performance and energy consumption. In this dissertation, such memory structures are referred to as assist caches. The *Filter cache* [52] for instance is an assist that reduces energy consumption for cache memory accesses by using a very small memory buffer in between the processor and L1D. However, these energy savings are obtained at the cost of increased access latencies, and thus higher AMAT. The *Victim Cache (VC)* [53] on the other hand aims to decrease AMAT by reducing the cost of conflict misses. The VC stores victims of L1D evictions such that in the event of future references to them, the lines can be returned to the L1D in a single cycle rather than through a long latency, energy consuming cache miss. On a VC hit, the requested line is moved to the L1D, and the corresponding entry from the L1D evicted to the VC. This swap operation constitutes an energy overhead, and is a significant disadvantage of the victim cache. Stiliadis et al. overcame this disadvantage with their proposal, *Selective Victim Caching (SVC)* [54]. In SVC, the swap operation is prevented from occurring if the incumbent L1D cache line is found to be more reusable than the requested VC line. SVC considerably reduces the number of swaps as compared to a conventional victim cache with the same miss rate and latency improvements.

However, these proposals consider the L1D as the primary target for data references by the processor, and the assist as an auxiliary cache. A majority of references are thus serviced by the larger L1D cache, and consequently, the relatively shorter latency and energy per access of the assist cache remain unexploited. Duong's *L0* scheme (*I1P101* in particular) [55] and the *HitME* cache [56] are examples of cache organizations that service a majority of data references through an assist placed alongside the L1D. In these proposals, any cache line that is referenced atleast once in the L1D is immediately moved to the assist. Subsequent accesses to these lines, for as long as they remain in the assist, incur a relatively shorter access latency and energy. However, L1D cache lines are moved to the assist without determining how much potential they have for reuse. If the assist-cached lines are not sufficiently reused so as to amortize the energy consumed in moving them from the L1D, any latency benefits obtained will be at the cost of increased energy consumption. In addition, moving lines following just a single access results in the small cache quickly becoming inundated, further resulting in a large number of evictions back to the L1D. These cache line movements constitute an energy overhead, and thus it is imperative that the assist caching strategy implement some selectivity in moving lines to the assist cache.

2.3.2 Persistence Selective Caching

PSC is a selective caching scheme that aims to reduce AMAT and energy per access by servicing a majority of data references through a small, fully-associative assist cache, while minimizing the number of cache lines moved to this cache from the L1D. It achieves this by limiting use of the assist cache to only those cache lines that are most frequently accessed. PSC is based on the premise that frequently accessed cache lines have potential for reuse, and are therefore likely to be referenced again. The reuse potential of a line is determined based on its *access persistence*, i.e. the number of accesses to the cache line within a small window of references. Lines that exhibit

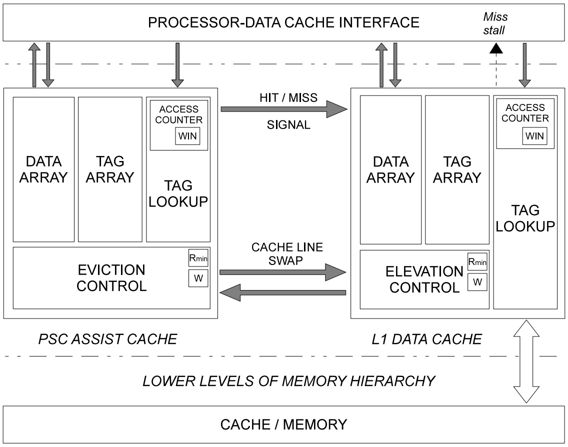


Figure 2.12. Architecture of the PSC assist cache and its interface for cache line swaps with the L1D

persistence equal to or greater than a certain threshold level are regarded as reusable and subsequently *elevated* to the assist cache. Due to its small size, references to data stored in the assist incur a relatively shorter access latency, and consume a smaller amount of energy than the L1D.

The PSC assist cache is illustrated in Figure 2.12. Both the assist and L1D are probed in parallel on references by the processor. Since the two are mutually exclusive in their content, cache lines may reside in either cache but not both simultaneously. Consequently, a reference to a line may hit in either, or miss in both, with the latter case serviced through the L1D alone. To enable the swapping of cache lines between the two caches, the architecture includes an interface for the exchange of data, tags and hit/miss information as illustrated in Figure 2.12. PSC does not influence the caching of lines by the L1D from lower level caches and is thus non-filtering, unlike the Filter cache [52] and SVC [54]. This ensures that the full capacity of the L1D is always available irrespective of the amount of data reuse present in the application.

2.3.2.1 Selective Caching Criteria

The threshold level of access persistence that lines require in order to be elevated to the assist cache is defined in terms of two parameters:

- R_{min}**:** The number of references required to a line for it to be regarded as frequently used and hence, as reusable.

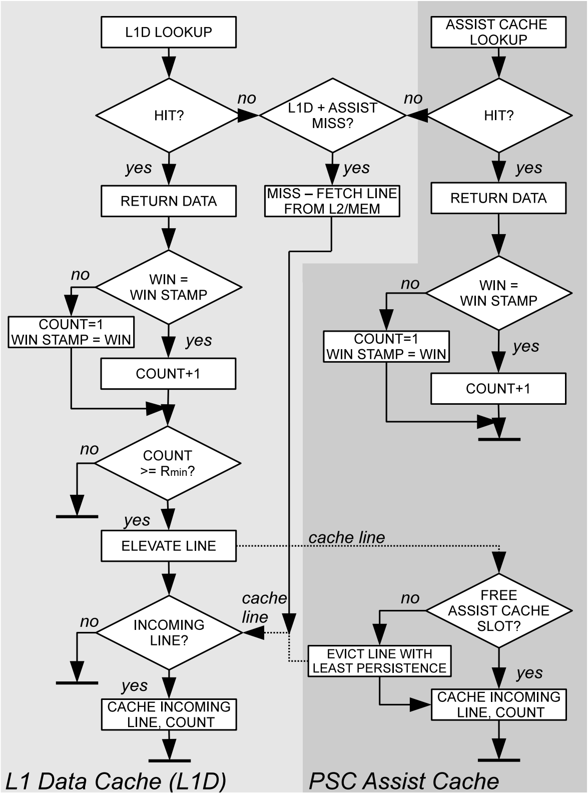


Figure 2.13. Algorithms for the L1D and PSC assist cache

- W**:** The number of references within which the R_{min} condition must be met. Each sequence of W references by the processor is considered as a window.

These two parameters essentially define PSC's selectivity in elevating L1D cache lines to the assist cache. Since access persistence is strongly dependent on memory access patterns, these parameters can be expected to have different optimal values for different applications.

The working of a data cache with a PSC assist is illustrated in Figure 2.13. Elevation decisions are made following L1D hits, after the requested data is returned to the processor. The decision to elevate a cache line to the assist is based on its access count. However, in order to account for changing memory access patterns of the workload and thus temporal variations in cache line usage, this count is considered valid only within the window of W references during which it is updated. The active window at any point in time is indicated by the *WIN* register, the value of which is incremented after every W references. Cache lines are stamped with this value to indicate the window active at the time of the reference. If during a subsequent access to the line, this stamp is found to differ from the value in the WIN register, the line's access count is considered invalid and is reset, and its WIN stamp updated.

On the other hand, if the values are found to be matching, the access count is considered valid and is compared with R_{min} to evaluate whether the line can be elevated to the assist. Therefore, if a line is accessed atleast R_{min} times within the current reference window, it is moved across the swap interface and placed within an available slot in the fully-associative assist cache. In the event that no free slot exists, the assist cache line with the least persistence is evicted and moved back to the L1D, while the newly elevated line is stored in the resulting free slot. The line evicted from the assist may however map to a different set inside the L1D and subsequently displace other useful data in that set. This is referred to as an *induced replacement* and can ironically increase contention in the L1D. In general, the occurrence of induced replacements decreases as PSC's selectivity is increased.

2.3.2.2 Significance of Persistence Threshold

The R_{min} parameter of the threshold has a significant influence on AMAT and energy consumption since it determines the selectivity of PSC, and controls the number of elevations to the assist. Effectively, it determines the number of references for which the lower access latency of the assist is incurred. The use of an R_{min} value lower than the application's optimal results in a large number of cache lines being elevated to the assist, and consequently, an aggravated induced replacement rate and energy overhead. On the other hand, a value higher than the optimal limits elevation to only those lines with exceptional amounts of access persistence. Both of these strategies can be employed to achieve specific performance or energy targets.

The W parameter determines the period in which reuse is measured. It enables the access count to not only reflect how often cache lines are referenced, but also when. This allows the PSC assist to selectively cache and accelerate references to the currently active data set. The parameter also helps limit the width of access counters used to track persistence since the maximum number of accesses possible to a cache line within any window is always W, and thus the counter width is $log_2(W)$.

Table 2.4. System Configuration

PROCESSOR	
Architecture	32b SimpleScalar/ARM
Issue Width	1 instruction/cycle
MEMORY HIERARCHY	
L1 Data Cache	1KB, 8KB, 32KB, 64KB / 4-way
L1 Hit Latency	1, 2, 3, 3 cycles resp.
Miss Penalty	64 cycles
Assist Cache (PSC/SVC/L0)	512B / Fully Associative
Assist Hit Latency	1 cycle
L2 Data Cache	1MB / 8-way
Line Size/Write Policy	64B / Write-back

2.3.2.3 Limitations

For an application, the overall optimal persistence threshold is one that results in a significant portion of data references being serviced through the assist cache, while minimizing the number of cache line swaps performed. Such a threshold ensures a shorter latency and lower energy per access, and its value can be expected to vary for different applications. To evaluate the complete envelope of operation of PSC, we determine the optimal persistence thresholds used in this dissertation through an exploratory simulation. Although time consuming, the simulation provides valuable insights into the characteristics of the PSC assist cache, and how the persistence threshold affects its performance. However, given that systems incorporating the PSC assist may execute a range of applications with distinct characteristics, it may not be realistic to expect such an exhaustive characterization of each. One promising approach to determining the persistence threshold is *compile-time analysis* [57–59]. Gonzales et al. [57] incorporate reuse, volume and interference analyses into their locality analysis compiler, providing information on the type of reuse in cache lines, the reuse distance, and potential conflicts that could occur between these lines in the data cache. Using such analysis methodologies, the value of the parameters R_{min} and W can be determined based on characteristics such as *reuse type*, *reuse distance* and *reuse volume*.

We note that even optimal thresholds represent only an approximate best case. Through the course of execution, memory access patterns and application behaviour often change significantly, and factors such as the reuse distance and stride length [60] of accesses change as well. This indicates the need for an adaptive strategy that tunes both R_{min} and W components of the persistence threshold accurately based on the variations in execution behaviour.

Table 2.5. Access Persistence Thresholds

	blowfish	*dijkstra*	*ghostscript*	*gsm*	*ispell*	*mad*	*sha*
R_{min}	70	30	70	70	100	90	130
W	2048	512	1024	2048	2048	1024	1024

2.3.3 Evaluation

We evaluate *Persistence Selective Caching (PSC)* using trace driven simulations. Data memory traces are generated using *SimpleScalar/ARM* [61] for a single issue processor, and subsequently passed into a trace-driven cache simulator. The simulator models a conventional data cache together with the PSC assist, *Selective Victim Cache (SVC)* and write-back *L0 cache* according to the configurations listed in Table 2.4. For each simulated cache, the corresponding latency, area and power data are gathered using *CACTI* [35] for the $32nm$ technology node. The access latencies of the caches are also listed in Table 2.4. Seven workloads from the *MiBench* embedded benchmark suite [49] are used in the evaluation. The persistence threshold for each is determined through an exploratory simulation as previously mentioned, and the corresponding R_{min} and W values obtained are listed in Table 2.5. These reflect the access persistence of the most frequently used data set within the workloads.

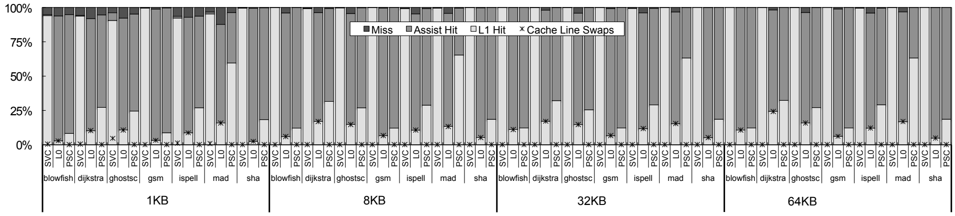
2.3.3.1 AMAT and Energy

Figure 2.14(a) illustrates the percentage of memory references that hit in the L1D, hit in the assist cache, and miss in the L1D. In a conventional cache, all data references either result in an L1D hit, or in a miss, causing the line to be fetched from lower level caches. In the case of the L0, on each L1D hit, the requested line is moved to the assist cache from where subsequent references to it are serviced. Though this yields a benefit in terms of access latency, since cache lines are moved into the L0 following just a single hit, it results in a large number swaps between the two caches. The swap count for the L0 is accordingly observed to be higher than for other schemes across workloads. Since PSC applies access persistence as the condition for elevations, lines are not moved into the assist until they demonstrate sufficient potential for reuse. This drastically reduces the number of swaps as compared to the L0. For instance, in the case of the $dijkstra$ workload using an L0 assisted 32KB data cache, over $20E6$ swaps occur, while with PSC, this is reduced to $8E3$ with better AMAT and energy reductions. Similarly, for $blowfish$ with a PSC assisted 8KB data cache, a 45% improvement in AMAT is obtained from only 18 cache line elevations to the assist. Effectively, 80% of all data references by *blowfish* hit this small set of cache lines in the assist as seen from Figure 2.14(a). This observation highlights the reusability of cache lines that exhibit access persistence, and illustrates the significant performance gains that can be obtained from the selective caching of reusable lines in low-latency assists.

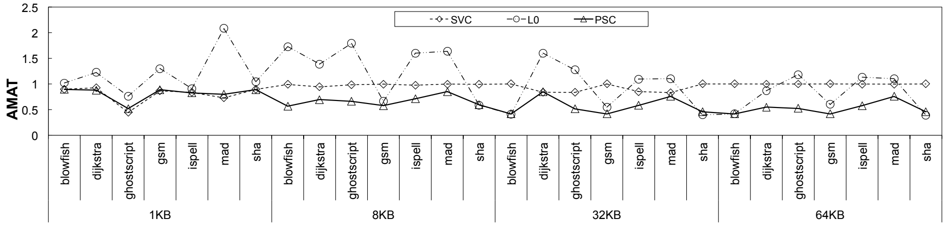
Figure 2.14(b)-(d) illustrate how the servicing of references from the assist, selectivity in elevations and reduction in cache contention translate into AMAT improvements and energy savings. PSC is observed to yield consistently lower AMATs than conventional data caches across all workloads and cache configurations. When compared to SVC and L0, PSC offers superior improvements in terms of both access latency as well as energy. These improvements are more pronounced for the larger caches whose higher access latency and energy consumption considerably magnify the negative repercussions of the SVC and L0. The SVC offers little benefit for L1D caches larger than 8KB with the workloads used. Conversely, in the case of the small 1KB cache, the benefits of PSC are not as pronounced since both the assist and the L1D have the same access latency. The AMAT improvements obtained for small caches stem from the miss rate reductions achieved through the dynamic associativity [53] added to contentious sets by the assist. PSC's performance in such caches matches that of SVC in terms of AMAT as well as energy. The most significant improvements are observed when PSC is used with larger set-associative caches with access latencies greater than that of the assist.

Although unselectively moving cache lines to the assist and servicing references to them with a short 1 cycle latency can sometimes yield AMAT benefits, the frequent occurrence of swaps always results in higher energy consumption. An example of this is the $blowfish$ workload with the 32KB data cache where the PSC and L0 are observed to offer similar AMAT improvements, but while the L0 incurs a large number of swaps in doing so, the PSC yields the same improvements with only 29 swaps. Using access persistence as the selectivity criterion therefore results in a large reduction in the number of cache line movements as compared to the L0 assist. Unselective movement of data in the case of the L0 also results in aggravated miss rates due to induced replacements, and higher energy consumption.

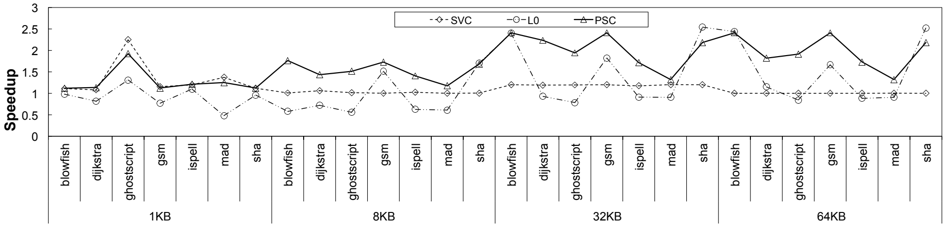
In the case of the SVC, despite its miss rate reduction, since a negligible fraction of references are serviced through the small assist, the AMAT and energy improvements obtained are small. For most cases in Figure 2.14(b) and (c), PSC results in AMAT upto 59% shorter, and energy consumption upto 75% lower than conventional data caches, as well as competing schemes. Note that in computing the energy per access in the case of PSC, we took into account the overheads of selective caching and maintaining of access counts and window timestamps as well.



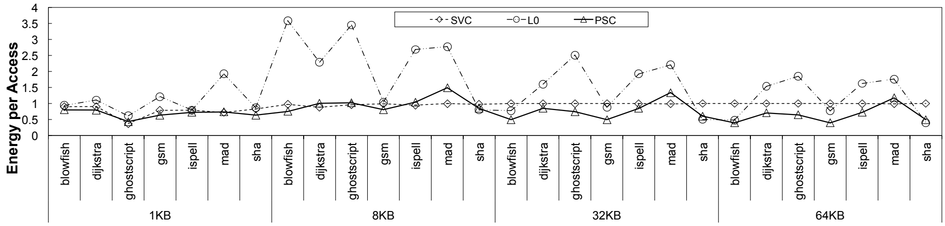
(a) Distribution of Memory References



(b) Normalized Average Memory Access Time (AMAT)



(c) Normalized AMAT Speedup



(d) Normalized Average Energy per Access

Figure 2.14. (a) Distribution indicating the fraction of memory references that result in an L1D hit, miss and assist cache hit for each workload with the *Selective Victim Cache (SVC)*, *L0* and *Persistence Selective Caching (PSC)* assist. The number of cache line swaps normalized to the total number of references is also indicated. (b) *Average Memory Access Time (AMAT)* (c) AMAT speedup and (d) Average energy per access normalized to a conventional 4-way L1D of size 1KB, 8KB, 32KB and 64KB.

Table 2.6. Area overhead relative to a conventional L1D

	SVC	L0	PSC
1KB	46%	46%	52%
8KB	1.2%	1.2%	7.3%
32KB	1%	1%	7%
64KB	0.8%	0.8%	7%

2.3.3.2 Overheads and Implementation Cost

The swap operation for the PSC is identical to that of the SVC except for the fact that lines evicted from the PSC assist may map to a different set in the L1D than from which the newly elevated line originated. This necessitates an additional L1D lookup and in case the set is fully occupied, an extra cycle for the corresponding eviction (an induced replacement). However, the extremely low number of swaps that occur with PSC offsets any disadvantages that such overheads may pose. Adopting a very pessimistic estimate for time per PSC swap as twice that of SVC swap time (6 cycles and 3 cycles [54] respectively), the total time spent in cache line swaps is still upto 90% lower for PSC as compared to SVC, and over 99% lower than the unselective L0 assist.

Table 2.6 lists the area overhead of the PSC scheme as compared to a conventional data cache. For the PSC, a $11b$ access count (corresponding to a window size of 2048 references), and a $21b$ window stamp were considered, translating to $4B$ of extra tags per line in both the L1D and the assist cache. The implementation overheads for the selective caching logic itself are minimal, comprising primarily of a small number of comparators and adders. Also listed in the same table are the overheads of the SVC and L0 (all assists in the table are of size $512B$).

2.3.4 Conclusions

Persistence Selective Caching (PSC) improves AMAT and energy by caching reusable lines in a low-latency, low-energy assist cache. PSC identifies reusable cache lines at runtime based on their access persistence, and by reducing the hit latency for these lines, improves AMAT by upto 59% as compared to conventional data caches. Furthermore, allowing only reusable lines into the assist, PSC reduces the number of cache line swaps between the L1D and assist cache, and thus decreases average energy per access by upto 75% over competing assist caches. These results illustrate the benefits of selective caching using low-latency assists, and highlight the significant amount of data reuse present in some applications.

2.4 Runtime Management of Shared Caches using CacheBalancer

The dominance of miss penalty (t_{miss}) in the AMAT of caches in large multiprocessors is indicative of the growing importance of on-chip data management. With increasing system size, it is imperative that close attention be paid to where data used by tasks is stored in order to minimize access costs.

Memory allocators are used by application programs to dynamically manage their memory at runtime, through the allocation of additional blocks for storage, or the release of unused blocks back to the system. The allocated memory forms part of the global *heap*. In conventional allocators [62], memory is allocated at the first available address where a contiguous free block of the requested size exists. Consequently, the allocated memory can reside anywhere within the bounds of the heap. Considering the use of cache slicing in *non-uniform cache access (NUCA)* based multiprocessors such as NagaS, this raises the possibility of the allocated memory block residing in a distant cache bank. Accesses latencies are aggravated as a result of the longer interconnect path between PEs and the cache bank. Since the lower level cache banks are accessed primarily on L1D misses, the increased access latency translates to a higher miss penalty (t_{miss}).

There exists a significant body of work that addresses the issue of memory allocation to improve access latencies. Distance awareness [63] is a scheme that allocates memory pages in cache banks close to the requesting PE, thereby reducing the communication latency for accesses. However, in doing so, the scheme forces all data allocated by a PE to be localized in a single cache bank, resulting in expensive capacity misses. This drawback is avoided by tracking the number of active pages in every cache bank in the system. The authors in [63] consider a threshold beyond which pages spill into neighbouring caches. [64] proposes a similar principle using software counters to track the difference in the number of allocated memory pages per cache bank. Both proposals aim to localize memory allocations around the requesting processor, and relax this rule progressively based on the size of the dataset already allocated in target banks. Although these prevent caches from overflowing due to excessive allocations in any single bank, such approaches ignore the actual utilization of data cached in the banks. For instance, a cache may have only a few allocated pages but be very heavily utilized by PEs, resulting in long access latencies. Allocating more data in the same bank would only exacerbate this problem [65]. It is therefore essential to consider the actual utilization of caches alongside conventional communication distance when performing memory allocations.

In multiprocessor environments, task mapping also plays a crucial role in determining the spread of data across cache banks, and thus memory access latencies. For instance, tasks sharing data must be mapped onto PEs located close to the bank caching this data in order for their access latency to be low [66]. However, each new task mapped onto the system increases utilization of cache banks to a different extent based on its memory access patterns and execution characteristics. It is therefore prudent to guide task mapping based on the characteristics of the application's tasks in order to uniformly utilize available cache memory resources, and thus minimize memory access latencies.

In this section, we propose a runtime resource management scheme named *CacheBalancer* that improves the utilization of on-chip shared caches and reduces access latencies in chip multiprocessor systems. CacheBalancer incorporates a utilization and communication distance-aware memory allocator that avoids heavily accessed cache banks, easing contention, and consequently reducing the access latency for allocated data. Furthermore, CacheBalancer uses information on the execution characteristics of tasks gathered from profiling and static analysis to determine their influence on other tasks in the system, through a metric called *Pain*. This metric is used at runtime to determine a task map that balances cache utilization, and minimizes inter-task interference due to shared system resources such as interconnect links and caches.

2.4.1 CacheBalancer

The CacheBalancer consists of two parts: An *Access Rate* based memory allocator, and a *Pain*-driven task mapper. The following subsections examine each part in detail.

2.4.1.1 Access Rate based Memory Allocation

While existing schemes within the state-of-the-art implement distance-aware [63] and cache capacity aware [64] memory allocation to overcome the limitations of conventional allocators, they result in the non-uniform utilization of on-chip cache resources. CacheBalancer uses a metric called *Access Rate* to determine the relative utilization of cache banks. Access rate is defined as the ratio of the number of accesses made to a particular cache bank versus the total number of shared cache accesses in the system. A high access rate indicates that a cache bank is frequently accessed, i.e. a *hot cache*. Conversely, a low access rate indicates a relatively less accessed cache bank, or a *cold cache*. CacheBalancer uses two access rate thresholds to determine if cache banks are *hot* or *cold*. These are determined as:

$$\theta_{hot}(p_i, m_j) = \phi \cdot \beta \tag{2.6}$$

$$\theta_{cold}(p_i, m_j) = \frac{\beta}{\lambda \cdot HC(p_i, m_j)} \tag{2.7}$$

where θ_{hot} and θ_{cold} represent the access rate thresholds for a cache bank to be considered as hot and cold respectively, β is the mid-range value of the access rate across N cache banks in the system, and $HC(p_i, m_j)$ is the hop-count along the interconnect path between PE p_i and cache bank m_j. The ϕ and λ parameters are integer constants that control the distribution of the allocated memory. ϕ for instance influences the memory allocator's sensitivity to hot caches. λ on the other hand affects the spread of data by limiting candidate cache banks based on their distance from the allocating PE. A high value forces the localization of data, while a low value enables the allocation of data across a larger number of cache banks. Due to their characteristics, these parameters could be used within a runtime manager to adapt memory allocations based on system utilization, power and reliability. This is an important topic to be explored in the future.

β is derived using Ψ, the set of access rates for N cache banks, as:

$$\beta = midrange(max(\Psi), min(\Psi)) \tag{2.8}$$

$$\Psi = \{\psi_0, \psi_1, \psi_2, ..., \psi_{N-1}\} \tag{2.9}$$

where ψ_x is the access rate of cache bank x.

Access rates are measured using two $8b$ access counters per cache bank. In order to prevent temporal fluctuations in access rates from causing oscillations in memory allocation targets, access rate samples are filtered using a moving average filter. Thus, the counters are sampled every 500 cycles and averaged with a finite number of previous samples in order to eliminate temporal spikes, and determine the current access rate. The calculated value is subsequently stored in a dedicated subsection of the address space accessible through the memory hierarchy. In order to allocate memory, the memory allocator function is called from tasks executing on PEs. This function first computes the θ_{hot} and θ_{cold} thresholds using the cache bank access rates, and invokes the selection algorithm listed in Figure 2.15 to obtain a target cache bank.

The algorithm identifies a candidate cache bank with access rate below the threshold θ_{hot}, preventing over-utilization by checking the number of pages already allocated at the banks. Memory allocation is avoided at cache banks that are heavily accessed, and those that would overflow with further allocation. Page counters track allocation at each bank relative to the lowest page count amongst all banks. The

```
initial cache bank ← closest cache bank to PE
x ← initial cache bank
if Access rate[x] $> \theta_{hot}$ || Allocated page count[x] $>$ Max. pages per bank then
    x ← next candidate cache bank
    while Access rate[x] $> \theta_{cold}$ && Allocated page count[x] $>$ Max. pages per bank do
        x ← next candidate cache bank
        if x==NULL then
            x ← initial cache bank
            break
    if Allocated page count[x]==0 then
        Allocated page count[x] ← number of pages to allocate
        minimum pages ← find minimum non-zero page count
        for every cache bank $y$ do
            Allocated page count[y] ← Allocated page count[y]-minimum pages
    else
        Allocated page count[x] ← number of pages to allocate
Target cache bank ← x
Return Target cache bank
```

Figure 2.15. Access rate based cache bank selection

lowest count is re-computed every time an allocation occurs in the bank with the least allocated pages. Once a suitable candidate bank is found, memory is allocated in a corresponding subsection of the address space.

2.4.1.2 Pain-driven Task Mapping

The utilization of cache banks is also very strongly influenced by the nature of tasks executing on PEs within the system. Memory-intensive tasks are more likely to utilize cache banks heavily than their counterparts with smaller data sets. In addition, the performance of tasks can become inter-dependent when they share a cache, especially when their execution characteristics vary significantly. It is therefore essential for task characteristics to be taken into consideration during task mapping in order to uniformly utilize system resources and minimize inter-task interference.

CacheBalancer uses an adapted version of the *Pain* metric of Zhuravlev et al. [67] to guide the mapping of tasks onto PEs. The metric describes, for each executing task, the performance degradation that would be experienced if a certain new task were to be mapped onto a particular PE. Pain is measured in terms of three parameters: *Cache Intensity*, *Cache Sensitivity*, and *Communication Impact*.

(i) Cache Intensity This parameter indicates how aggressively a cache bank is used by tasks. For each cache bank i, it is given as:

$$Z(m_i) = \sum_{\pi_j \in \Pi_i} z_i(\pi_j, m_i) \tag{2.10}$$

where π_j is a task within the set of tasks Π_i that use the cache bank m_i. $z_i(\pi_j, m_i)$ is the average number of task π_j's memory accesses that reference cache bank m_i. Its value can be estimated using memory access profilers such as *plugin* [68], *cprof* [69] and *Quad* [70]. Since the actual number of accesses may only be known at runtime, the obtained estimates can be coupled with appropriate runtime meta data to obtain an accurate value. Cache intensity hence measures how heavily cache banks are used by executing tasks.

(ii) Cache Sensitivity This parameter indicates the sensitivity of tasks to contention in shared cache banks. It is measured as the likelihood of a task's memory hits turning into misses as a result of cache contention. Sensitivity is specified as the product of the stack distance profile [71] of the task's hits in the cache [67], and the probability of eviction from each stack position. The stack positions correspond to the order in which cache lines would be evicted in the event of contention in the set. Thus, $x = 1$ corresponds to the most recently used, and $x = n$ the least recently used sets. Cache sensitivity is given as:

$$S(\pi_i) = \sum_{x=1}^{n} \frac{x}{1+n} \cdot h(x) \tag{2.11}$$

where n is the set-associativity of the cache bank, and $h(x)$ is the number of hits by the task at stack position x. $h(x)$ is similar to the access persistence metric used in the context of PSC, and requires characterization of the task's memory access profile [72]. This can be done using the same tools as used in the case of *cache intensity*, and also using Zhuravlev's extension [67] for the *PIN* dynamic instrumentation framework [73].

(iii) Communication Impact This parameter encapsulates the cost of accessing shared data located in a specific cache bank, over the shared interconnect. It therefore determines the degree of performance loss induced due to interconnect contention and communication distance resulting from a certain task mapping. For each candidate mapping $map(\pi_i) \rightarrow p_q$ where p_q is a PE, the communication impact is given as:

$$CI(\pi_i \mid map(\pi_i) \rightarrow p_q) = \sum_{m_j \in M_i} 2 \cdot HC(p_q, m_j) \cdot Z(m_j) \cdot S(\pi_i) \tag{2.12}$$

$HC(p_q, m_j)$ represents the hop count between PE p_q and cache bank m_j used by the task. Note that the hop count is doubled in order to account for the fact that memory accesses consist of both requests and responses.

The pain in terms of cache intensity, cache sensitivity and communication impact is given for a task π_i assuming a candidate mapping on PE p_j as:

$$Pain(\pi_i)_{\pi_i \rightarrow p_j} = \underbrace{\left(\sum_{m_j \in M_{\pi_i}} S(\pi_i) \cdot Z(m_j) \right)}_{\text{due to shared caches}} + \underbrace{\left(CI(\pi_i \mid map(\pi_i) \rightarrow p_j) \right)}_{\text{due to interconnect}} \tag{2.13}$$

where M_{π_i} represents the set of caches used by a task π_i. At runtime, the task mapper evaluates the effective pain for each candidate mapping, and assigns tasks onto available PEs that yield the lowest pain.

Since this mapping is performed at runtime, searching through the combination space could be a time-consuming process. In order to reduce the complexity of the search operation, the mapper orders the waiting task pool based on task importance. This is determined as the product of cache sensitivity, cache intensity, and data set size. Tasks with higher importance are mapped first, with full fidelity in evaluations of mapping options. On the other hand, tasks with lower importance can be mapped with a lower search effort. As mapping progresses, the available options decrease and thus, the search space also decreases. In total, the mapping function has a complexity of $O(p)$ per task in a system with p processors. In the average case, for n tasks, the complexity is $O(pn \cdot log(n))$, and $O(pn^2)$ in the worst case.

2.4.2 Evaluation

CacheBalancer is evaluated using a cycle-accurate SystemC model of the R3 NoC [41] based NagaS array, illustrated in Figure 2.16. The simulator is configured according to the configuration in Table 2.7. CacheBalancer is implemented in C as a linkable library. Its memory allocator and task mapper can be invoked from application software by including the library at compile-time. In order to emulate worst-case conditions, CacheBalancer is tested with a synthetic workload that uses parallel tasks to repeatedly allocate and write to memory pages, thus stressing the shared caches, and exposing their resulting memory access latency.

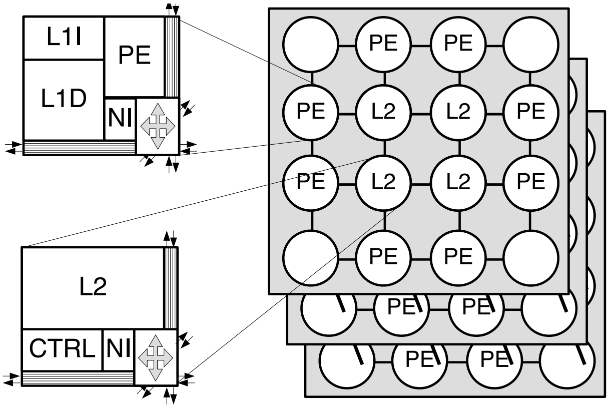


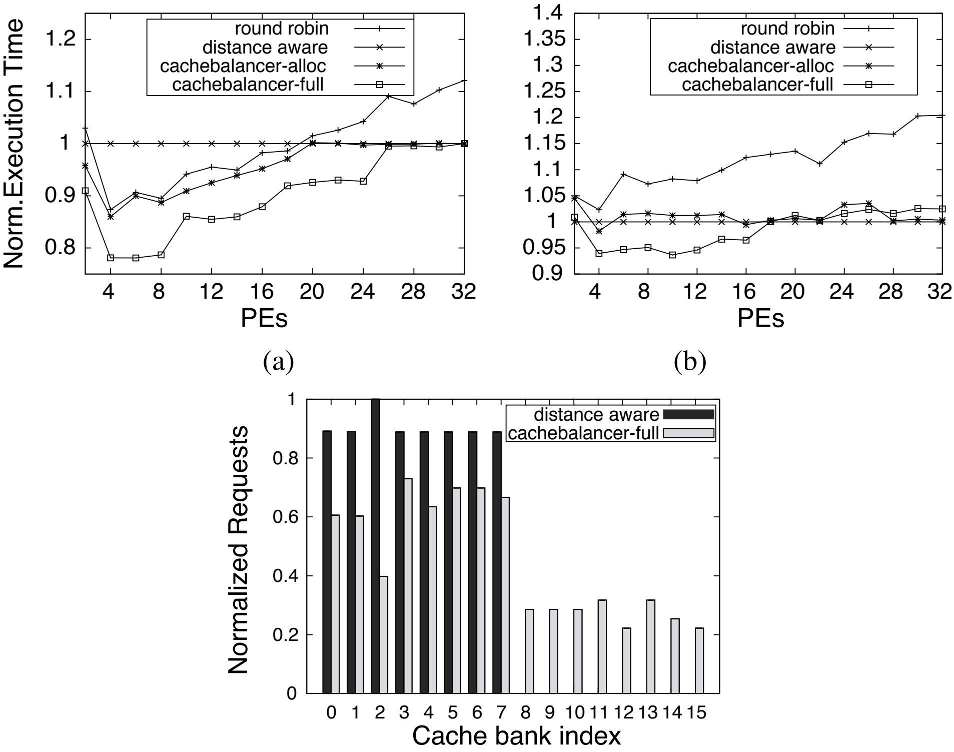
Figure 2.16. R3 network-on-chip based NagaS array

Table 2.7. System Configuration

Processing Elements	32
Instruction Set Architecture	$32b$ Microblaze
L1 Instruction Cache	4KB/2-way/64B
L1 Data Cache	64KB/4-way/64B
L2 Cache Banks	128KB/8-way/64B
Number of L2 Cache Banks	8/16
Total L2 Cache Capacity	1024KB/2048KB
Interconnect Topology	$4 \times 4 \times 4$ (3D Mesh)
Tech. Node/Clock	$90nm$/200MHz

Figure 2.17 reports the normalized execution time for the workload with varying number of PEs, using the conventional memory allocator (*round robin*) [62], distance aware allocator [63], CacheBalancer's access rate based allocator (*cachebalancer-alloc*), and the complete CacheBalancer scheme comprising of the allocator as well as the pain-driven task mapper (*cachebalancer-full*). The system is evaluated with two different ratios of PEs to shared caches - $32:8$ and $32:16$, and the execution time results corresponding to these are reported in Figure 2.17(a) and (b) respectively.

In Figure 2.17(a), the ratio of PEs to shared caches is high, thus emulating a high-contention scenario. In this case, *cachebalancer-full* is observed to provide upto 22% lower execution time compared to the *distance aware* scheme. The minimization of communication distance in the case of the distance aware allocator reduces communication latency at the cost of cache contention. CacheBalancer on the other hand prioritizes the minimization of cache contention, even at the cost of slightly

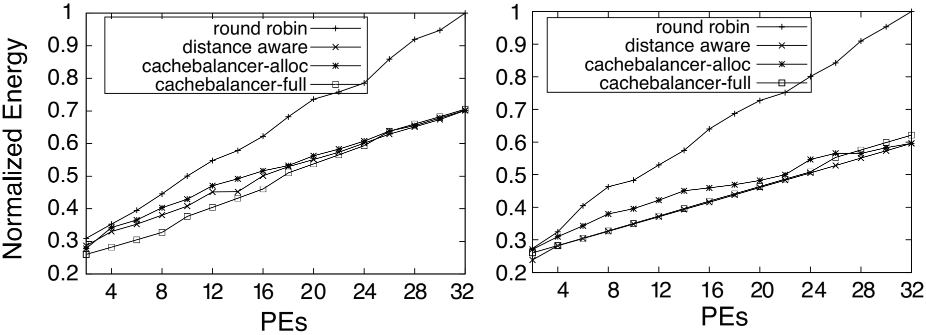


(a) (b)

(c)

Figure 2.17. Execution time using different schemes normalized to the distance aware allocator. Multiprocessor system includes 32 PEs and varying number of cache banks - (a) 8 cache banks, (b) 16 cache banks. (c) Cache bank utilization for system with 32 PEs and 16 banks

increased communication distances. The rationale behind this lies in the fact that communication latencies can be improved by means of optimizations to the interconnect architecture [74], and also because minor differences in communication distance have a relatively small impact on overall performance. For instance, even with a slightly higher (1 hop) communication distance, CacheBalancer has a positive influence on performance. This benefit is obtained mainly from the reduction in contention at the shared cache banks. However, with increasing number of PEs, the available cache memory bandwidth is quickly exhausted. At this point, CacheBalancer's performance matches that of the distance aware scheme. Figure 2.17(b) reports the execution time



(a) (b)

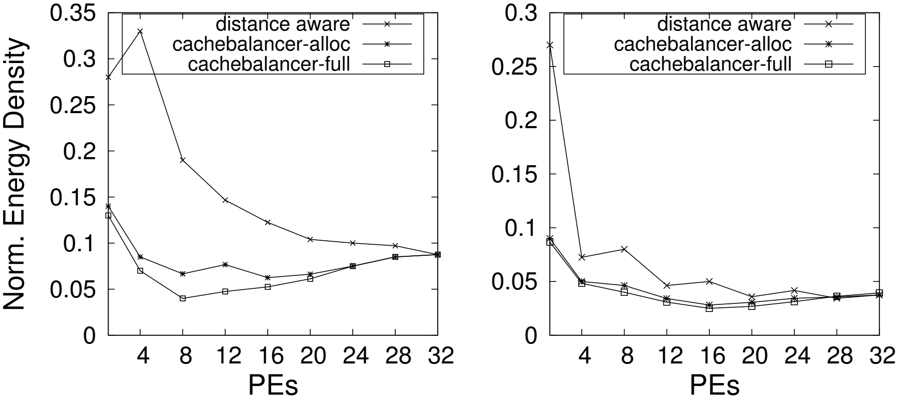
Figure 2.18. Normalized energy dissipation with (a) 8 cache banks (b) 16 cache banks

results for a larger number of shared caches. In this case, the topology of the system results in a larger number of shared cache banks in the vicinity of PEs. Consequently, the distance aware allocator experiences reduced contention at the banks as compared to the round robin scheme. Due to this, the execution time benefits of CacheBalancer appear smaller than for the previous case.

The distance-aware allocator prioritizes the use of the closest cache banks, and consequently, utilizes only 8 of the 16 banks in the system. CacheBalancer on the other hand utilizes all 16 banks, yielding a more uniform spread of access across the shared caches, and reduces contention by upto 60%. This is clearly evidenced in Figure 2.17(c). These simulations use a hot cache sensitivity (ϕ) of 1 and allocation spread (λ) of 2 hops.

In addition to decreasing execution time, decreasing contention also has a significant impact on the energy dissipation of caches. This is especially true when cache bandwidth is constrained. The energy dissipated in the two cases, normalized to round robin allocation with 32 PEs, is reported in Figure 2.18. Reducing contention at cache banks decreases the occurrence of conflict misses, and eases congestion in the interconnect. The benefits of CacheBalancer in this regard are observed in Figure 2.18(a) where the ratio of PEs to caches results in high contention at cache banks. On the other hand, the already low contention at banks in Figure 2.18(b) causes CacheBalancer to yield little benefit in terms of energy dissipation.

Nevertheless, in both cases, energy density is decreased by over 50% due to the spreading of cache utilization across all banks in the system. Thus for a given number of cache accesses, CacheBalancer spreads the corresponding energy dissipation over a larger area than competing allocators. Figure 2.19 reports the energy density obtained



(a) (b)

Figure 2.19. Energy density normalized to the round-robin allocator with (a) 8 cache banks (b) 16 cache banks

with the different allocators. Energy density is a critical factor that influences operating temperatures in ICs. Its reduction prevents the occurrence of performance degrading thermal emergencies. The effects of energy density are treated in more detail in Chapter 3.

2.4.3 Conclusions

This section presented CacheBalancer, a runtime resource management scheme for chip multiprocessors that uses the *access rate* and *pain* metrics to guide memory allocation and task mapping. The scheme achieves upto 60% lower contention by improving utilization of available cache banks, and reduces execution time by upto 22% as compared to competing proposals. Furthermore, by improving utilization of shared cache resources, it yields upto 50% lower energy density than other proposals. The presented CacheBalancer in this dissertation is only a proof of concept, and requires further exploration to determine the full significance and impact of parameters such as ϕ, λ, θ_{hot}, θ_{cold}, and their derivation. These topics require further investigation, and integration within Naga and its tooling.

Summary

The performance of a multiprocessor is strongly influenced by the efficiency of its underlying communication model, which determines t_{memory}, the fraction of execution time spent stalled on memory load-stores. In the first part of this chapter, we presented the Pronto message-passing system to improve the efficiency of data transfer between distributed memories in the NagaM multiprocessor. The proposed system minimizes PE involvement in the management of data transfers, and reduces transfer overheads through the use of message envelopes for flow control.

In the second part, we proposed Persistence Selective Caching (PSC) to improve management of data in the L1 cache so as to reduce hit time (t_{hit}) and decrease miss rate. By accelerating references to reusable data, PSC yields upto 59% reductions in average memory access time (AMAT), and upto 75% reduction in average energy per access, over competing schemes. The efficacy of PSC is indicative of the significant data reuse present in general applications.

In the third part, we proposed the CacheBalancer scheme to efficiently manage cache resources in non-uniform cache access (NUCA) multiprocessors. CacheBalancer combines a dynamic memory allocator and a runtime task mapper, in order to minimize access costs for the last level caches. In addition to improving execution performance by 22%, CacheBalancer mitigates contention by increasing the utilization of available cache resources. This yields upto 50% lower energy densities within the system.

Associated Publications

The topics presented in this chapter are covered in the following publications:

1. S.S. Kumar, M.T.A. Djie, R. van Leuken, ”Pronto: A Low Overhead Message Passing System for High Performance Many-Core Processors.” *International Journal of Networking and Computing - Special Issue*, vol. 4, no. 2, p. 307-320, July 2014

2. S.S. Kumar, M.T.A. Djie, R. van Leuken, ”Low Overhead Message Passing for High Performance Many-Core Processors,” *Proceedings of the International Symposium on Computing and Networking (CANDAR)*, pp. 345-351, 2013

3. S.S. Kumar, R. van Leuken, ”Improving data cache performance using Persistence Selective Caching,” *Proceedings of the IEEE International Symposium on*

Circuits and Systems (ISCAS), pp. 1945-1948, 2014

4. J. de Klerk, S.S. Kumar, R. van Leuken, ”CacheBalancer: Access Rate and Pain Based Resource Management for Chip Multiprocessors”, *Proceedings of the International Symposium on Computing and Networking (CANDAR)*, pp. 453-456, 2014

3

Exploring the Thermal Design Space in 3D Integrated Circuits

Three-dimensional integrated circuits (3D IC) are composed of multiple stacked dies interconnected using *Through Silicon Vias (TSV)*. These enable the high-density integration of devices without increasing the area footprint of the system. However, the number of devices that can be integrated within the system is limited by its thermal behaviour. Nagata [32] determined this limit as:

$$\frac{\alpha N_G E}{t_{pd}} \leq g \cdot \Delta T \tag{3.1}$$

where N_G is the number of gates in the system, E is their energy dissipation, and t_{pd} is the clock period. The relation dictates that the maximum number of gates that can be integrated within an IC is limited by the average thermal conductance g and temperature difference ΔT between the dissipating elements and the ambient air. In order to increase integration density, either energy dissipation of the gates, or their activity rate must be decreased. Alternatively, the thermal conductance of the system must be improved so as to efficiently dissipate the larger amount of generated heat. (3.1) essentially specifies the thermal design space for 3D ICs in terms of its most significant parameters.

In this chapter, we characterize the influence of these design parameters on the thermal behaviour of die stacks, and examine the factors influencing the formation and mitigation of hotspots. In the second half of this chapter, the insights obtained from this exploration are applied within a novel framework to enable the thermal-aware design of 3D stacked multiprocessors, providing a means for uncovering thermal issues and addressing them early in the design flow.

3.1 Significance of parameters

The heat flow between power dissipating elements and sink surfaces in 3D ICs follows the Fourier heat transfer equation [75]:

$$c_v \frac{\delta T}{\delta t} = \nabla(G \cdot \nabla T) + Q \tag{3.2}$$

$$where \quad \nabla T = \begin{bmatrix} \frac{\delta T}{\delta x} \\ \frac{\delta T}{\delta y} \\ \frac{\delta T}{\delta z} \end{bmatrix} \quad and \quad Q \propto \frac{\alpha N_G E}{t_{pd}} \tag{3.3}$$

Q is the heat flowing from a power dissipating element towards multiple sink surfaces, at a rate $\delta T/\delta t$ through a material of volumetric heat capacity c_v. The conductance matrix G defines the thermal conductance along the orthogonal x, y and z axes of the material as a function of effective thermal conductivity (κ_{eff}) of the materials encountered along those respective axes. Ignoring the direction of heat flow, equation (3.2) can be rewritten in steady-state form, for any single dimension x, y or z as:

$$Q = g_{x,y,z} \cdot \Delta T \quad where \;\; g_{x,y,z} = \kappa_{eff} \frac{A}{l_{x,y,z}} \tag{3.4}$$

where $g_{x,y,z}$ is the effective thermal conductance along the selected dimension. This equation indicates the relationship between thermal conductance, and effective thermal conductivity of the material. The conductance $g_{x,y,z}$ across a die layer is thus a function of its κ_{eff}, its area (A), and its material thickness ($l_{x,y,z}$). The κ_{eff} of a layer is not only dependent on its material, but also on structures such as TSVs which act as high conductance thermal pathways, improving the overall conductivity of the layer itself. For a material layer with TSVs, the κ_{eff} may be computed as the weighted average of the thermal conductivities of the TSV material (κ_{tsv}) and the layer material (κ_{mat}) respectively, as given in (3.5).

$$\kappa_{eff} = A_{tsv}\kappa_{tsv} + A_{mat}\kappa_{mat} \tag{3.5}$$

$$A_{tsv} = n(\pi r_{tsv}^2) \;\; and \;\; A_{mat} = (h_{die} w_{die}) - A_{tsv} \tag{3.6}$$

A_{mat} is the area of the material layer in a die with length h_{die} and width w_{die}, and A_{tsv} represents the total area of n TSVs of radius r_{tsv} on this layer.

From (3.4), the magnitude of thermal gradients observed at a given point across any layer of the stack is determined by effective thermal conductivity (κ_{eff}) of that layer, its thickness, and the power density at that point in the die stack. Lateral thermal gradients on stacked dies are predominantly influenced by the effective thermal conductivity of the bulk silicon. As κ_{eff} increases more heat is conducted away from the power dissipation site resulting in a better spread of temperatures. Low κ_{eff} on the other hand results in the stagnation of heat and thus the formation of hotspots. As the thickest die layer, the bulk silicon's κ_{eff} also determines the heat flow to other dies in the stack, thereby affecting vertical thermal gradients. Improving conductivity increases heat flow towards the sink surfaces, yielding lower peak temperatures in stacked dies.

The spread of temperatures is also dependent on die thickness ($l_{x,y,z}$). The use of die thinning to improve integration density causes a decrease in the thickness of bulk material, which hampers the lateral spreading of heat, and leads to the formation of high temperature hotspots. The spread of temperatures is however limited by lateral thermal resistance beyond a certain die thickness. Therefore, dies thinner than this threshold can be expected to experience hotspots of greater magnitude than thicker dies. In 3D ICs, $l_{x,y,z}$ can also be considered as the effective length of the heat flow path between the power dissipating elements and the sink. Consequently, the thermal conductance of the heat flow path decreases as stack depth increases.

Power density influences the concentration of generated heat (Q) in die stacks, and can increase the peak temperatures of hotspots. However, since vertical heat flow is determined by κ_{eff}, the temperature gradient of a die is predominantly affected only by local power dissipation as well as dissipation in the tiers directly above and below it. Heat from other tiers in the stack, in the steady-state condition, raises the overall operating temperature of all the dies. Consequently, increasing power density causes the temperature profiles to shift to higher ranges, while temperature gradients are affected only if the dissipated power is in a tier with sufficient κ_{eff} to the observation die.

3.2 Thermal Characterization of Die Stacks

In conventional single-die ICs, the heat dissipated by circuit elements is conducted to the heatsink through a relatively small number of intermediate layers. In 3D ICs on the other hand, each die can add up to 12 layers of varying thickness and conductivity to the heat flow path between dissipating elements and the heatsink [76]. This impedes the flow of heat away from power dissipation sites, and results in aggravated operating temperatures. A number of studies in literature examine specific thermal characteristics of die stacks. Oprins et al. [77] investigated the thermal

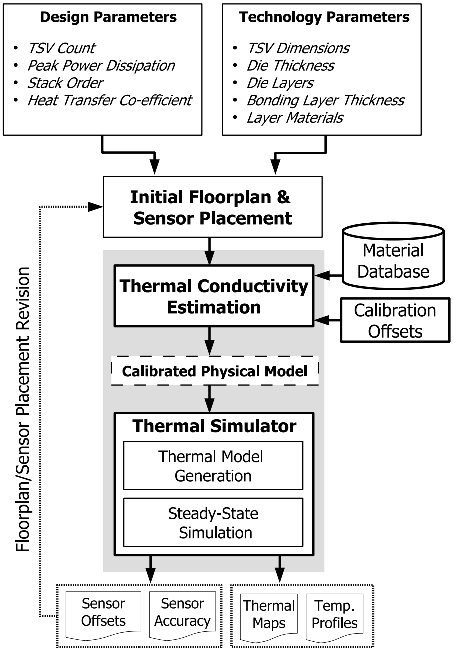


Figure 3.1. Thermal characterization flow

coupling between memory and logic dies in a two-tier stack in order to determine operating temperatures and thermal profiles in the memory. Their study used a thermal model calibrated with a $130nm$ stacked-die silicon test chip, and examined the influence of microbumps and underfill thermal conductivity on peak temperatures. Clarke et al. [78] characterized the influence of stacked memory on hotspot formation in underlying logic tiers. Their work revealed the non-trivial nature of heat flow within 3D ICs, and uncovered the benign influence of symmetric metallization in memories on hotspot temperatures. Matsumoto et al. [79] performed an experimental measurement of thermal resistance of multi-layer die stacks interconnected with C4 microbumps, highlighting the dependence of thermal resistance on microbump pitch and size. Similarly, Vaisband et al. [80, 81] investigated thermal conduction paths within a two-tier die stack in order to evaluate the magnitude of heat transfer between dies, and to characterize the temperature dependence of thermal conductivity.

The design of the *Through Silicon Via (TSV)* based vertical interconnect significantly influences thermal conductivity, and thereby impacts system performance. In addition to TSVs, the composition and depth of die stacks, die thickness, location

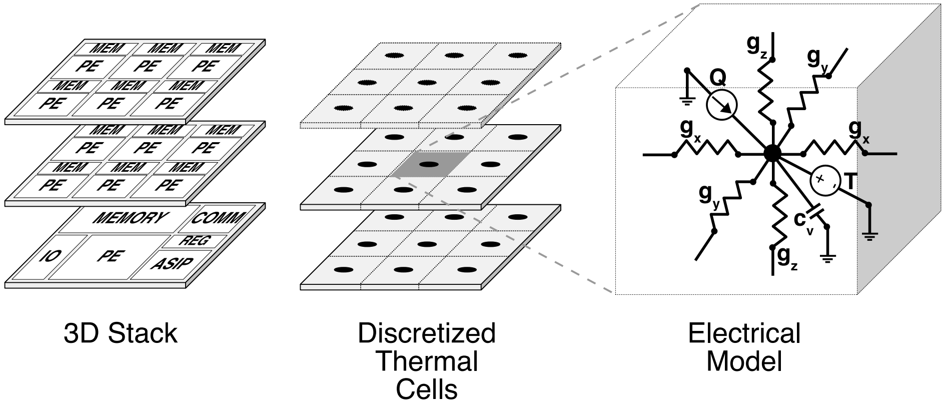


Figure 3.2. Illustration of the equivalent electrical model used to determine the temperature of thermal cells

of power dissipating elements, and stack power density also form critical design parameters that influence thermal behaviour. A high-level exploration flow is used to characterize the influence of these parameters on the thermal behaviour of die stacks. The flow comprises of two stages: thermal conductivity estimation, and thermal simulation, as shown in Figure 3.1. In the first stage, input design and technology parameters are translated into a physical model of the die stack. This describes the dies and their constituent layers, materials and dimensions, floorplans, peak power dissipation values for components, locations of temperature sensors, and the number and dimensions of TSVs used in the design. The thermal conductivity of each layer in the stack is computed using (3.5) alongwith a material database. The resulting physical model contains all the requisite data for accurate thermal modelling of the die stack. In the second stage of the flow, the thermal behaviour of this model is simulated in order to obtain fine-grained steady-state temperature maps corresponding to the peak power dissipation of components. In order to do this, a thermal simulator is used [82].

Thermal simulation begins with the discretization of the die stack into a grid of thermal cells, each representing a unit of heat generation (Q), and temperature (T), as illustrated in Figure 3.2. The thermal conductance (g) between cells essentially determines the heatflow across the die, and is derived from (3.4). The *Resistance-Capacitance (RC)* network within each cell represents the electrical equivalent of that cell's thermal relationship with its neighbours. In order to simulate thermal behaviour, power values are inserted into the appropriate thermal grid current sources corresponding to the locations of power dissipating elements. This results in a rise in local temperature, and depending on the magnitude of conductance ($g_{x,y,z}$), the temperature of neighbouring thermal cells as well. The inclusion of a capacitance representing the specific heat capacity (c_v) of the silicon allows modelling of both steady-state as well as transient thermal behaviour. The accuracy of the model, however, is largely dependent on the dimensions of the thermal grid, which in turn has a considerable influence on runtime. Since power and temperature are considered to be uniform within a thermal cell, large dimensions can result in loss of accuracy. For instance, if a small circuit element dissipates 100% of the power within a thermal cell, large cell dimensions would result in that total power being considered as having been dissipated over the entire cell's area - thereby erroneously modelling a lower power density. Furthermore, this would result in overly optimistic temperature estimates. On the other hand, very small cell dimensions result in the generation of a massive thermal model, increasing system memory usage considerably, and resulting in long runtimes. More recent thermal simulators eliminate this problem by using a non-uniform grid that uses variable sized thermal cells to better model the spatial differences in floorplan complexity and power dissipation [75][83].

3.2.1 Experimental Setup and Validation

In order to validate our characterization flow, we created a physical model of the 3D stacked test chip described in [76] and carefully calibrated the heat transfer co-efficient of the connection to ambient using available data. Thereafter, we carried out a series of thermal simulations using test floorplans containing heating elements of size and power dissipation identical to those in [76]. Our setup accurately reproduced the same temperature profiles, with a maximum temperature deviation of 6% at hotspot peaks.

The experimental setup utilizes a 3D stack consisting of a $250\mu m$ base die with multiple thinner dies bonded above using $1\mu m$ thick bonding layers, as illustrated in Figure 3.3. Each die in the stack contains five configurable power dissipating elements of $50\mu m \times 50\mu m$ to generate heat. The complete stack is connected to ambient air through an interface with the previously calibrated heat transfer co-efficient. The material composition and layer dimensions for dies is identical to the 3D test chip described in [76]. The dies used in the characterization have a size of $2000\mu m \times 2000\mu m$, and the thermal simulations use a grid width of $10\mu m$, which provides good accuracy with acceptable simulation times. Simulations are carried out using the 3D-ICE thermal simulator [82] and are run until steady state temperatures are reached. To quantify the variation in temperatures with changing parameter values,

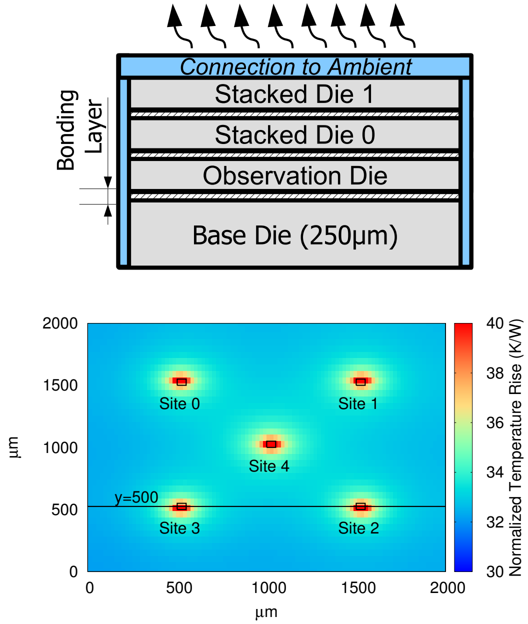


Figure 3.3. Composition of the die stack and normalized map of temperature rise across the die with all heaters dissipating 25mW

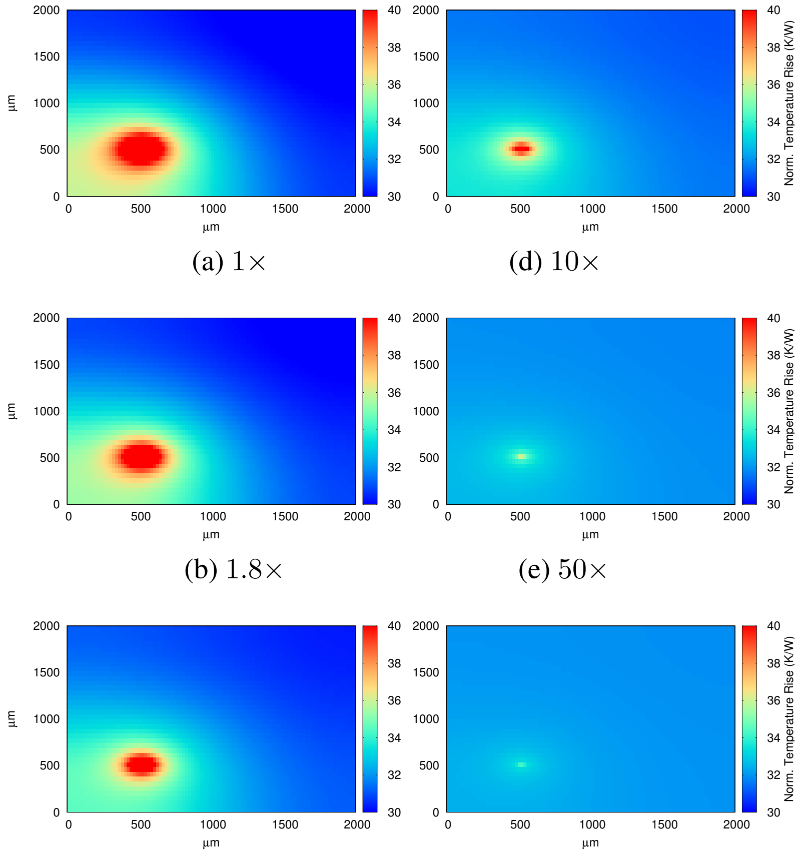
the normalized temperature profile along the horizontal line at $y = 500\mu m$ is plotted. For clarity of presentation, we only apply power at Site 3, keeping all other heaters on the die idle. This allows us to clearly observe the temperature gradients that result from power dissipation at a single site under different physical conditions.

3.2.2 Characterization

The following sections examine the influence of effective thermal conductivity, die thickness and stack depth, and stack power density on temperature profiles.

3.2.2.1 Thermal Conductivity (κ_{eff})

The formation of hotspots is critically influenced by the effective thermal conductivity of the bulk silicon, as observed in Figures 3.4*(a)*-*(f)* which depict the temperature rise across the dissipating die with increasing κ_{eff}. This die forms the middle layer of a five-die stack mounted on a $250\mu m$ base, and dissipates $120mW$ of power at the Site 3 heater. As effective conductivity increases, the heat produced due to this



(a) 1×
(d) 10×
(b) 1.8×
(e) 50×
(c) 4×
(f) 100×

Figure 3.4. Thermal map depicting normalized temperature rise across the observation die with increasing effective thermal conductivity relative to conventional silicon (1×).

localized power dissipation is conducted more efficiently to adjacent dies in the stack, and in turn to the sink surfaces. The generated heat is thus prevented from stagnating in the die, thereby resulting in a decreased hotspot temperature, and smaller thermal gradients. Figure 3.5 shows the change in temperature profiles observed with varying κ_{eff} values. Note that κ_{eff} is specified relative to the thermal conductivity of pure silicon.

The Figures 3.4 and 3.5 also illustrate the improved heat spread due to decreased lateral thermal resistance resulting from the higher κ_{eff}. The lateral spread of heat

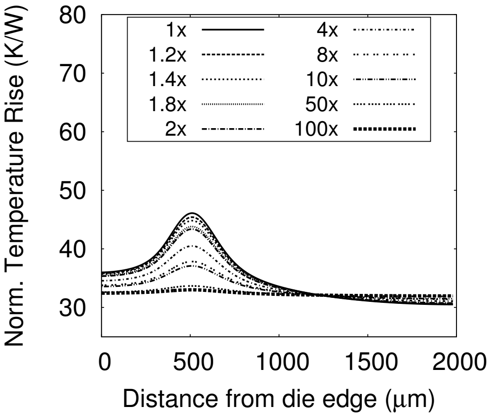
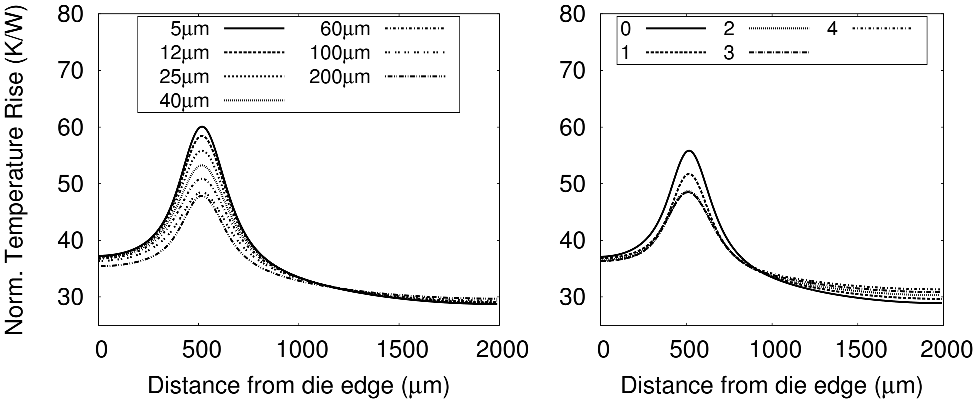


Figure 3.5. Influence of effective thermal conductivity on normalized temperature rise in the observation die

Table 3.1. Variation in the size of sensing zones for different measurement accuracies with changing thermal conductivity

	ACCURACY-LINKED SENSING ZONE RADIUS (μm)			
κ_{eff}	*0K*	*1K*	*2K*	*3K*
1×	0	80	360	1460
1.2×	0	80	480	1460
1.4×	0	100	620	1460
1.8×	0	120	1140	1460
2×	20	120	1460	1460
4×	20	280	1460	1460
8×	20	1460	1460	1460
10×	20	1460	1460	1460
100×	300	1460	1460	1460

holds a number of implications for the placement of temperature sensors. The accuracy of temperature sensors is influenced by their distance from the hotspot [75], and owing to their size, sensors cannot always be placed at close proximity to the region of interest. This necessitates the use of calibration techniques to offset induced measurement errors [75][84]. We observe that the radius of the zone within which hotspot temperatures can be tracked with 100% accuracy increases with κ_{eff}. Table 3.1 lists the maximum radius of sensing zones within which temperatures can be measured to within $0K$, $1K$, $2K$ and $3K$ accuracy of the hotspot. The radius is observed to increase with κ_{eff}, indicating that temperature sensor placement becomes less restrictive as effective thermal conductivity increases.

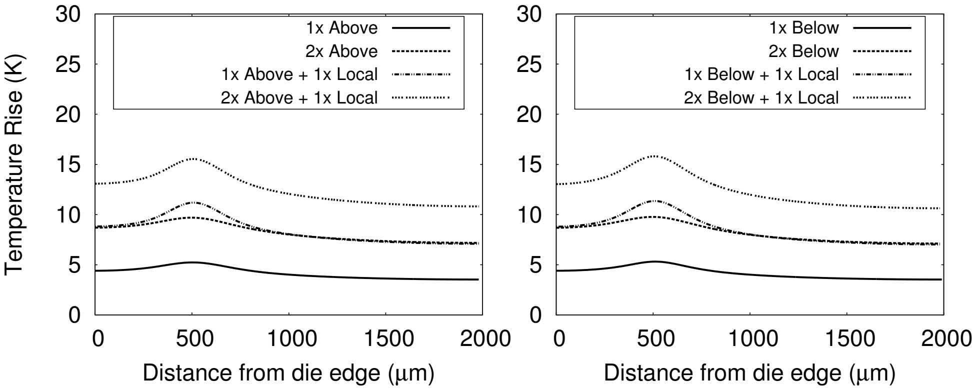


(a) (b)

Figure 3.6. (a) Influence of varying die thickness on normalized temperature rise. (Thickness of base die: $250\mu m$) (b) Influence of stack depth on normalized temperature rise on an active observation die mounted on a $250\mu m$ thick base die. (Thickness of observation/stacked dies: $25\mu m$)

3.2.2.2 Die Thickness and Stack Depth ($l_{x,y,z}$)

Figure 3.6(a) shows the temperature profile on a single die of varying thickness, stacked above a $250\mu m$ base die. The thickness of the die is observed to influence the peak temperatures of hotspots. The relatively shallow depth of bulk silicon in thin dies inhibits the spread of heat away from the hotspot, and thus causes the highest peak temperatures. While increasing the thickness improves lateral spreading, this benefit diminishes as die thickness exceeds $100\mu m$. The influence of die thickness can also be emulated by a stack of dies. Figure 3.6(b) reports the temperature profiles resulting from the stacking of multiple $25\mu m$ thick dies on top of the dissipating die used in the previous case. Peak temperatures are seen to decrease as stack depth increases. The additional dies stacked above the dissipating die in this case serve as bulk material and facilitate the lateral spreading of generated heat, as evidenced by the increased temperature rise registered at the eastern edge of the die. The heat spreading effect is observed for stacks with up to two-dies above the dissipating die, and translates to an effective thickness ceiling of $75\mu m$ (excluding the base die), which is lower than the $100\mu m$ ceiling observed in the previous case. This observation is explained by the



(b)

Figure 3.7. Influence of stack power density on the actual temperature rise in an observation die. Power sources (a) above observation die (b) below observation die. Note that the case marked +*Local* uses an active Site 3 heater on the observation die. Each heater dissipates $120mW$ of power.

fact that die stacks contain a range of materials such as *silicon dioxide*, which have a thermal conductivity much lower than that of silicon. Consequently, the thermal characteristics of a die stack with effective thickness of $100\mu m$ are certain to differ from those of a single die of the same thickness.

3.2.2.3 Power Density (Q/A)

Figure 3.7 shows the temperature profile of an observation die located in the middle of a five-die stack mounted on a $250\mu m$ base die. The graph reports the temperature rise experienced in the observation die due to power dissipated at Site 3 heaters on other tiers of the stack, i.e. changing stack power density. Since the primary sink in the stack is located at the surface of the topmost die, generated heat must flow through all intermediate dies. This is evidenced by the higher temperature rise noted when the dissipating dies are located below the observation die. However, it is interesting to note that similar behaviour is observed even when the dissipating dies are placed above the observation die. This leads us to conclude that in addition to the heat flowing towards the primary sink, significant conduction also occurs towards other sink surfaces in the stack, resulting in complex heat flow patterns. The overall steady-state temperature, on the other hand, is observed to be notably influenced by the stack power density and the heat transfer co-efficient of the sink. It is therefore important to take the maximum power density of the stack into consideration when determining the specifications of the heatsink and heatspreader.

3.2.3 Conclusions

The magnitude of hotspots and the shape of temperature profiles in dies are observed to be inherently linked to the power density, stack height and thermal conductivity of the die stack. This indicates that the available temperature margin at any point in a 3D IC is dependent on a complex set of design parameters. This observation indicates that it is necessary to evaluate the thermal behaviour of the system early in the exploration flow, in order to obtain a dependable, thermally efficient design.

3.3 Vertical Interconnect

Thermal conductivity (κ_{eff}) is observed to have a significant influence on the thermal behaviour of die stacks, and is determined by the number, size and material of the *Through Silicon Vias (TSV)* that form the vertical interconnect. The vertical interconnect is therefore a crucial component, and its design must take into account the unique electrical, mechanical and thermal characteristics of TSVs.

Although TSVs can be placed anywhere in the design, their use is disruptive due to their occupation of active silicon area. The placement of TSVs at certain locations could result in the displacement of circuit elements, and necessitate physical design re-runs. Jagtap et al. [85] determined that four generic placement topologies - *Border*, *Bundle*, *Shielded* and *Isolated*, form the corner points of the vertical interconnect design space in terms of noise voltage, area penalty and placement feasibility. The *Border* topology is based upon the rationale that if TSVs are placed along the periphery of the design like I/O pins then the resulting central rectangular area can be completely used for placement of devices. Thus, existing planar design practices and algorithms can be retained. With a closely packed *Bundle* of TSVs, an enclosure around the bundle can be blocked out for placement of devices causing minimal disruption to existing place and route tools. The *Shielded* topology is a form of the bundle topology with improved signal integrity. In the *Isolated* topology, TSVs are placed at a large distance from one another in order to minimize electrical interference and other coupling effects. We use these topologies to highlight the effects of TSV placement on performance and cost. The four topologies are illustrated in Figure 3.8.

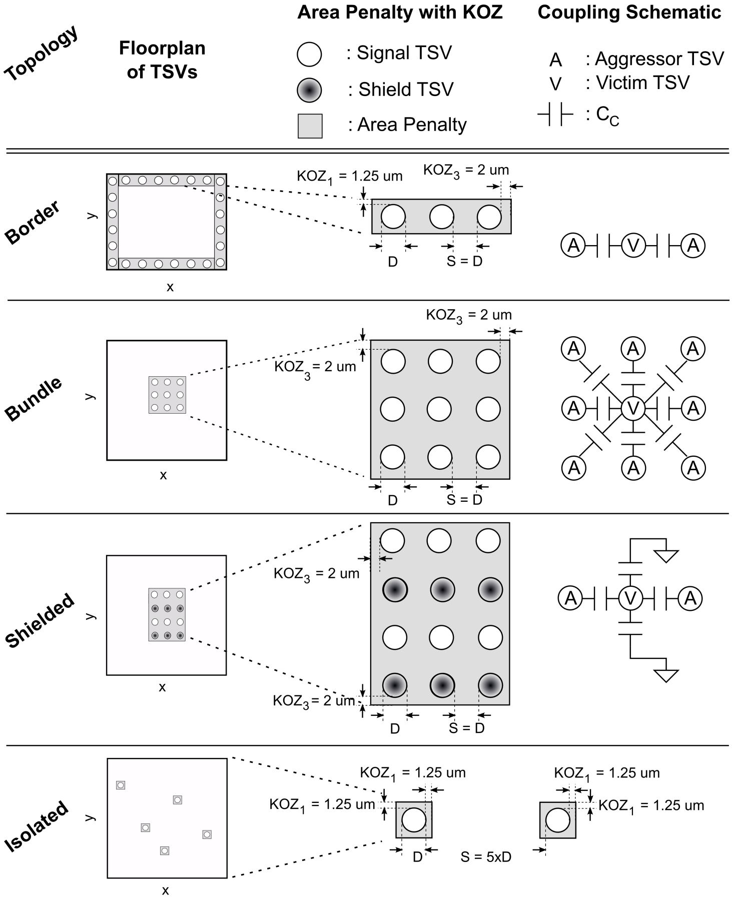


Figure 3.8. Illustration of *Through Silicon Via (TSV)* placement topologies. *Keep Out Zone (KOZ)* dimensions are indicated for each, together with the *TSV diameter (D)* and *spacing (S)*. Schematics also illustrate capacitive coupling between aggressor and victim TSVs.

3.3.1 Electrical Performance

The short vertical communication path facilitated by 3D stacking enables faster communication as compared to longer lateral wiring [86]. However, the disruptive nature of TSVs often means that they must be placed away from active devices, and interconnected using lateral wires. This results in an increased effective capacitance between the driver and load, and consequently, in a higher propagation delay and

Table 3.2. Results of TSV topology exploration for an on-chip communication block in 45nm

	Border	*Bundle*	*Shielded*	*Isolated*
Performance metric				
Min. Capacitance (fF)	26	80	80	26
Max. Capacitance (fF)	184	152	176	186
Min. Delay (ps)	7	20	19	7
Max. Delay (ps)	47	38	44	47
Max. Increase in Delay (ps)	5	23	8	1
Normalized Noise (Margin: 0.34)	0.16	0.42	0.11	0.05
Freq. (GHz)	13.9	15.8	14.5	13.9
Freq. With noise (GHz)	13.0	11.6	12.9	13.7
Percent decrease	7	27	11	1
Cost metric				
Total capacitance (fF)	8080	8276	9092	6970
Total TSV area (μm^2)	3328	5220	10422	3126
Percent TSV area	4.0	6.2	12.4	3.7

power dissipation. Since the length of lateral wiring is determined by the topology in use and the location of TSVs with respect to their driver/load, the propagation delay can vary between nets. This is seen in Table 3.2, which lists the results of a performance-cost evaluation of different TSV topologies for a $300\mu m \times 300\mu m$ macro of an on-chip communication block in the $45nm$ technology node.

Noise due to capacitive coupling between TSVs in close proximity also influences electrical performance. Firstly, simultaneous signal transitions in aggressor TSVs can induce propagation delays in critical victim nets. Secondly, coupling can induce spurious transitions in victim nets and cause errors in the transferred data. It is therefore essential to space TSVs sufficiently, or incorporate measures to mitigate signal integrity issues within the vertical interconnect. Shielding is one such method which reduces coupling between nets by separating them with a grounded line. Although this increases capacitance to ground, and thus propagation delay as well as power, it decreases noise voltage to within acceptable margins.

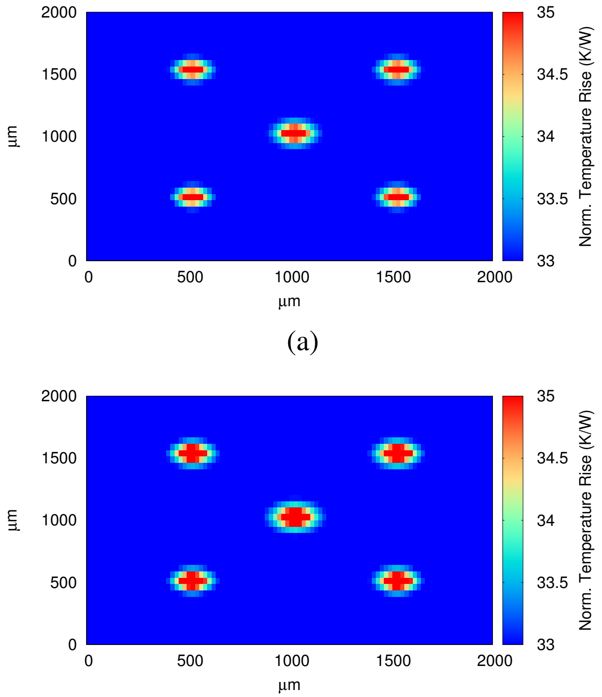
In Table 3.2, the Bundle topology shows a worst case normalized capacitive coupling noise of 0.42, which violates the noise margin of 0.34 for the 45 nm technology node [87]. The Isolated topology on the other hand, exhibits a much lower normalized noise value of 0.05, well within the margin, on account of the wide spacing between its TSVs. Similarly, increased capacitance to ground with the Shielded topology results in better noise performance than the Border topology. The resulting frequency for vertical links in each topology is calculated by adding a Setup Time of 25 ps to the path delay, and any increase in this delay due to noise limits the maximum achievable operating frequency. This is highlighted in the case of the Bundle topology where the relatively high coupling noise causes a 27% reduction in operating frequency compared to that of the same topology without noise. The wide spacing of TSVs in the Isolated topology reduces the impact of capacitive coupling, resulting in a mere 1% difference between frequencies with and without noise.

3.3.2 Area

The difference in the co-efficient of thermal expansion of TSVs and silicon means that the two expand by different amounts when heated. As a consequence, TSVs expand more than their surrounding silicon, and create regions of stress in their immediate vicinity. In order to insulate circuit elements from potential damage due to this stress, TSVs are enveloped by a *Keep Out Zone (KOZ)* within which no active devices are placed. In electrical terms, KOZ is defined as the area around a TSV where change in saturation current $\Delta Idsat$ for MOSFETs is greater than 5% [88]. Mercha et al. [88] analyzed the impact of thermo-mechanical stresses induced during TSV formation on device integrity and observed that for TSVs arranged in a row or in a bundle, the stress components add up and thus propagate larger distances into the surrounding silicon. Consequently, the dimensions of the KOZ vary with the topology. For 40 nm devices [88], given TSV diameter (D) and spacing (S), the guidelines on KOZ sizing can be summarized as:

1. KOZ is equal along both axes of a single TSV.
2. Devices cannot be placed between two TSVs when spacing $S = D$ as $\Delta Idsat$ in this region exceeds 5%.
3. KOZ decreases with increasing TSV spacing. $KOZ_{2D} = 2.5\mu m$ when S=2D, $KOZ_{3D} = 2\mu m$ when $S = 3D$, and $KOZ_{4D} = 1.25\mu m$ when $S \geq 4D$.
4. KOZ dimensions vary when TSVs are placed in a row with $S = D$. $KOZ_2 = 1.53\mu m$ for a row of 2 TSVs, $KOZ_3 = 2\mu m$ for a row of 3 TSVs, and $KOZ_4 = 2.125\mu m$ for a row of more than 4 TSVs.

For an isolated TSV, the aggregate area including the KOZ is about 2.6 times the area of the TSV itself, which is a significant overhead. Moreover, this factor varies according to whether TSVs are arranged in a row or in a matrix, implying that the KOZ must be accounted for in the estimation of total TSV area penalty. This is observed Table 3.2, where the Border, Bundle and Isolated topologies, despite being



(a)

(b)

Figure 3.9. Heatmaps corresponding to power dissipation at 5 heaters on the middle die of a five-tier stack, with the (a) Shielded and (b) Isolated topologies

equal in terms of their TSV count, pose significantly different area overheads. In area constrained designs, KOZ spacing requirements can make topologies unfeasible if all TSVs cannot fit within the available placement windows.

3.3.3 Thermal Performance

The total number of TSVs in the vertical interconnect is effectively a function of the system architecture, electrical noise performance requirements, and the area constraints for the design. Since TSV count influences the effective thermal conductance k_{eff} in (3.5), topologies also have an influence on the resulting thermal profile in die stacks. In topologies that incorporate shielding TSVs, the additional thermal conductance results in decreased peak temperatures and hotspot dimensions. Figure

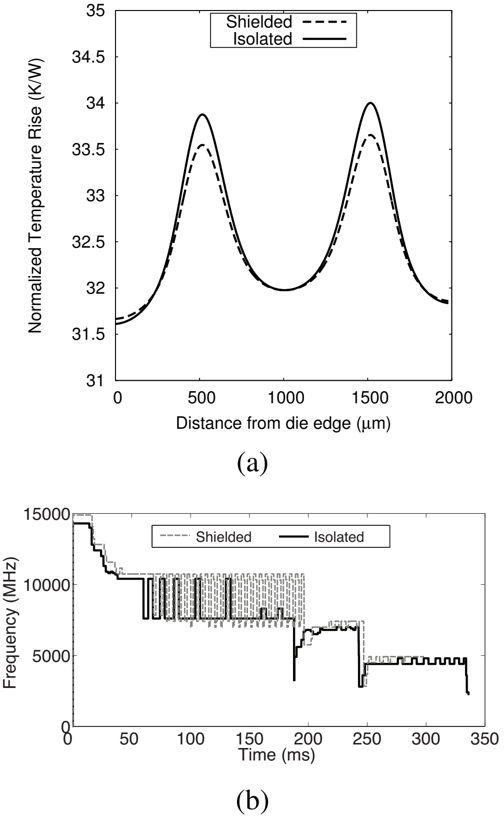


Figure 3.10. (a) Steady-state peak temperature profile corresponding to Shielded and Isolated TSV topologies. (b) Aggregated frequency profile corresponding to execution of computational workload on a three-tier multiprocessor with Shielded and Isolated topologies. Note: The trace for Shielded ends 38ms earlier than that of Isolated due to faster completion of execution.

3.9 illustrates the heatmaps corresponding to power dissipation at five heaters on the middle die of a five-tier stack, with the Shielded and Isolated TSV topologies. Due to its greater number of TSVs, the Shielded topology exhibits an effective thermal conductance 1.8× greater than that of the Isolated topology. This results not only in decreased hotspot size, but also in a lower steady-state peak temperature profile as depicted in Figure 3.10(a). This additional temperature margin, although small, can yield an appreciable improvement in system performance.

To illustrate this, the two topologies are deployed within a three-tier mulitprocessor with 12 PEs, and a temperature-aware power manager. During thermal emergencies, voltage and frequency levels are scaled down in order to limit power dissipation and prevent hotspots from forming [83]. The additional temperature margin realized with the Shielded topology enables the operation of PEs at higher frequencies, and improves overall system performance. This is observed in Figure 3.10(b) where the the execution of the application program completes $38ms$ faster with the Shielded topology, than with the Isolated topology. Essentially, the additional thermal conductance afforded by the shield TSVs yields a 11% improvement in the multiprocessor's execution performance without any modifications to the architecture.

3.3.4 Conclusions

The vertical interconnect design space is complicated by the mechanical and thermal effects of TSVs. KOZ sizing is strongly dependent on the TSV topology, and can influence placement feasibility in area constrained designs. Similarly, system performance is affected not only by the electrical performance of TSV topologies, but also by their thermal effects. These effects of TSV based vertical interconnects make it essential for topology exploration to be performed early in the system design flow, and to be integrated within the design space exploration environment.

3.4 Thermal-Aware Design Space Exploration

Design space exploration (DSE) involves the evaluation of architectural and system design options in terms of performance and associated cost. In light of increasing integration densities and the shift towards stacked-die architectures, the thermal constraints indicated by Nagata's equation (3.1) begin to assume greater significance, and form an important aspect of the design space that must also be concurrently evaluated. Early knowledge of runtime thermal behaviour of systems is invaluable in guiding architectural and system-level design decisions, and help minimize the mismatch between design-time estimates and actual runtime system performance. In addition, it can yield efficient designs that maximize performance within the available thermal constraints [83].

Over the years, a number of methodologies, frameworks and tool flows have been proposed to meet this need for *thermal-aware DSE (tDSE)*. Skadron et al., proposed one of the earliest thermal-functional co-simulation frameworks comprising of the *SimpleScalar* architectural simulator [61] and the *Hotspot* thermal model [83, 89]. Their pioneering work was followed by a number of other proposals such as [90], which integrates a multiprocessor simulator with a static thermal model to enable thermal-aware performance evaluation of a low power *multiprocessor system-on-chip (MPSoC)*. Proposals like [91] and [92] improve the coupling between functional and thermal simulators to enable modelling of *Dynamic Thermal Management (DTM)* schemes, and others such as [93] and [94] enable the thermal-exploration of system-designs using *SystemC/TLM* models. The growing popularity of 3D design for *systems-on-chip (SoC)* has led to the development of methodologies such as *Pathfinder3D* [95] and *PathfindingFlow* [96] for the thermal-aware exploration of the 3D design space, and the *MEVA-3D* [97] floorplanner for physical design and performance estimation of 3D microarchitectures.

Despite their merits, all the surveyed proposals exhibit a number of limitations that motivate this paper. Firstly, the existing proposals require power and latency models to be provided as an input along with system floorplans. Since such methodologies are most often used in tDSE scenarios, abstractions are used to avoid the time-consuming development of detailed models and component floorplans for each configuration at each design point. While this enables the fast and simple revision of system specifications, it leads to inaccuracies in thermal characterization [83]. Secondly, existing methodologies do not model the internal organization of components such as caches, resulting in abstractions that hide the internal power dissipation characteristics of individual components. The ability to accurately characterize component thermal behaviour is a prerequisite to developing thermally efficient systems. Thirdly, existing methodologies require the thermal simulation to be repeated from scratch even for minor modifications and optimizations to the system, eg. a change in the DTM's critical temperature.

We seek to address the shortcomings of state-of-the-art methodologies with our proposal - *Ctherm* - which is an integrated co-simulation framework that enables thermal-aware design of MPSoCs. It simplifies the design process by automating the generation of detailed floorplans and *area-latency-energy (ALE)* models for components from the input system specification. Abstract components in the system-level floorplans are replaced with the generated floorplans, preserving the flexibility afforded by abstraction while removing the inaccuracy associated with their use. The modeled system is instantiated as a SystemC virtual platform together with an embedded thermal engine in the cycle-accurate co-simulation stage to estimate functional as well as thermal performance.

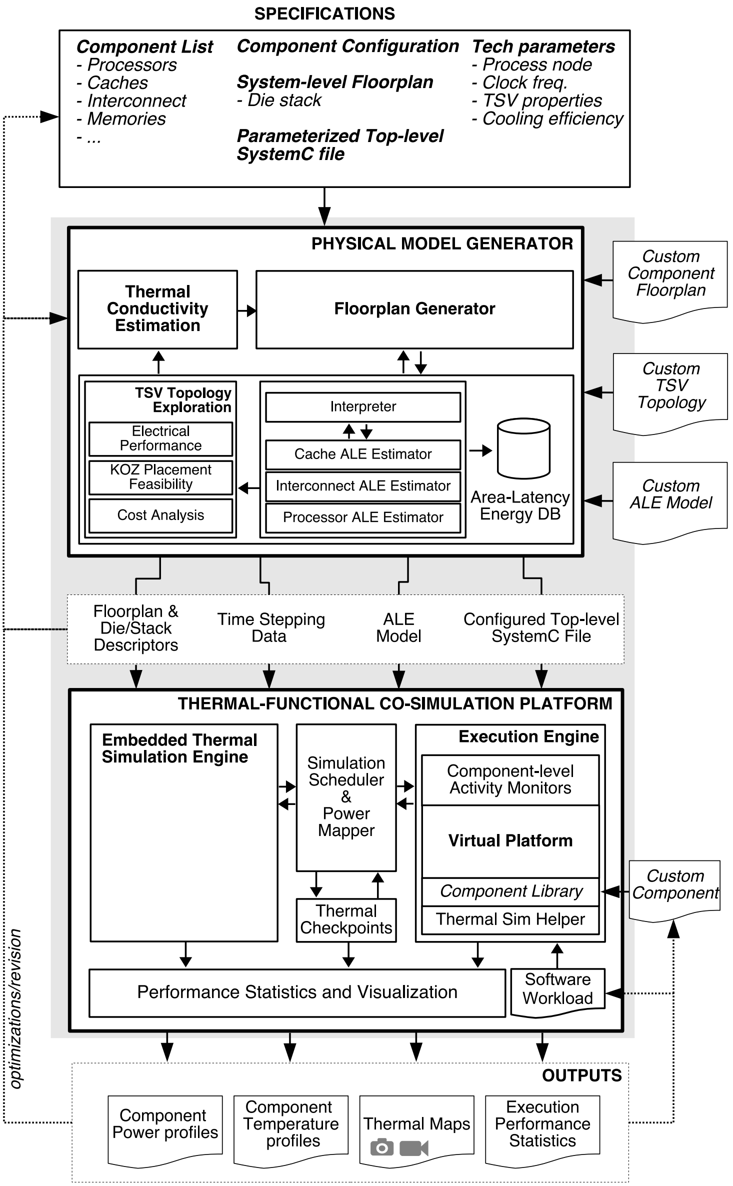


Figure 3.11. Ctherm framework for thermal-functional co-simulation

3.4.1 Ctherm Framework

The Ctherm framework consists of two stages: the physical model generator, and the thermal-functional co-simulation platform. These stages enable the translation of input specifications into a physical model of the system, and subsequently, the thermal-functional evaluation of the model to determine system performance and thermal efficiency. The Ctherm framework is illustrated in Figure 3.11.

3.4.1.1 Physical Model Generation

The physical model consists primarily of a system-level floorplan and *area-latency-energy (ALE)* models for components. We generate the physical model in two stages; first, by estimating the latency, energy and dimensions for components based on their configuration, and second, generating fine-grained floorplans for individual components based on these estimates, and inserting them into the system-level floorplan.

(i) Area-Latency-Energy Model The thermal behaviour of components is largely dependent on their internal organization, power dissipation characteristics and area. Accurate characterization requires detailed models of the energy and latency per operation of each component, together with the area of its constituent functional units. ALE data for generic components such as cache memories, interconnects and simple processor cores can be generated using existing parameterizable estimators [35][98][99]. Since SoCs are often composed of such generic components, Ctherm integrates a number of state-of-the-art estimators within its configuration generator. A Python interpreter is used to translate input system specifications into a suitable format for each estimator, and further convert the outputs of each into usable ALE data. ALE models contain estimates of dimensions for components and their constituent functional units, architectural organization (such as number of cache banks, interconnect ports), as well as energy and latency per operation, and per functional unit.

Cache ALE estimation The internal organization of on-chip cache memories often vary based on the configuration (size, associativity and cache line size) as well as the chosen design target (minimized area, latency and power dissipation). For instance, optimizations such as wordline and bitline segmentation that are used to reduce access energy also result in large *Static RAM (SRAM)* arrays being divided into multiple smaller sub-banks, each with its own decoder, sense amplifier, comparators and output drivers. This change in organization can significantly affect how dissipated power

func. unit dimensions	bitline segmentation	access latency
# decoders	wordline segmentation	cache size
# output drivers	sense amp/driver mux	associativity
# comparators	sharing factor (N_{spd})	line size
# sense amplifiers	energy per func. unit	

Figure 3.12. ALE data for cache memories

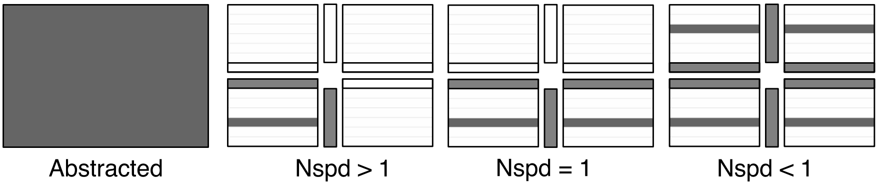


Figure 3.13. Access power distribution across sub-banks in a cache data array with varying N_{spd}

is distributed across the the entire cache area. Thus, it is essential to include such estimates into ALE models in order to obtain an accurate thermal characterization of components. A summary of data included within cache ALE models is listed in Figure 3.12.

The *wordline sharing factor (N_{spd})* parameter best illustrates the effect of changing component internal organization on power distribution. This factor specifies the number of cachelines stored per *SRAM* wordline, and is used to optimize caches to meet specific power, area or latency constraints. N_{spd} essentially also determines which SRAM banks are activated on memory accesses. For instance, a value greater than one results in the same SRAM wordline line being activated for accesses to all cache lines that share it, and conversely for N_{spd} values lesser than one, multiple SRAM wordlines being activated for accesses to a single cache line. The location of the dissipated power for accesses is observed to be strongly linked to the N_{spd} value, and in cases can result in the formation of thermal hotspots. Ignoring this parameter during thermal simulation results in the power dissipated on accesses to be distributed across the entire cache's area, as illustrated by the Abstracted case in Figure 3.13.

Interconnect ALE estimation *Network-on-Chip (NoC)* interconnect routers consist of multiple input/output ports arranged around a central crossbar switch. The area, latency and energy of routers is influenced by the width of the NoC links, their physical length (effective capacitance), and the number of ports. In 3D routers, the

# ports	# TSV
driver energy	TSV topology
crossbar energy	TSV conductivity
input port energy	func. unit dimensions
arbiter energy	

Figure 3.14. ALE data for interconnect routers

effective capacitance of the vertical interconnect is influenced by the TSV dimensions and placement topology. For this reason, interconnect ALE estimation also incorporates a methodology for TSV topology exploration [100], and the generated estimates are integrated into the router ALE model, shown in Figure 3.14.

TSV Topology Exploration Ctherm integrates Jagtap et al.'s methodology for TSV topology exploration [85]. The methodology evaluates candidate topologies in terms of their electrical performance, implementation cost, and feasibility after incorporating topology-specific KOZ. The results from this exploration allow the initial floorplan to be revised in order to incorporate the optimal TSV topology, and thus account for the thermal influence of the vertical interconnect on system performance.

Processor ALE estimation The ALE model for processors can be generated in one of two ways - per instruction, and per pipeline stage. The former lumps the power of individual pipeline stages into an aggregate value for each instruction, while the latter specifies a generalized average power value per pipeline stage regardless of the actual instruction. Processor ALE models consist primarily of functional unit dimensions, and the power dissipation at the chosen granularity.

For other components, custom ALE estimators can be added to the framework by simply extending the Python interpreter. However, for components with optimized implementations, and those for which no parametrizable estimators exist, custom ALE models must be provided as an input to the configuration generator.

(ii) Floorplan Generation To enable fine-grained thermal characterization of systems, Ctherm automates the generation of floorplans based on component dimensions extracted from ALE models, using specific Python based planning routines for each component type. These routines only require specification of the anchor position for components in the system floorplan. The pseudo-code for the planning routines used for memories, interconnect routers and processing elements is listed in Figure 3.15. For components without a rigid internal organization, such as PEs, functional units are

```
——————— Caches ————————
placement position ← initial anchor position
for each functional unit in cache do
    GET(dimensions, pitch, count) from ALE model
    GET(row count, column count) from ALE model
    for rows in [0 to row count] do
        for cols in [0 to column count] do
            PLACE UNIT(identifier,placement position,dimensions)
            placement position ← CALCULATE NEXT PLACEMENT POSITION( )
```

```
———— Interconnect Routers —————-
GET(dimensions, ports) from ALE model
GET(TSV topology, TSV count) from ALE model
placement position ← CALCULATE CENTER POINT(initial anchor position,dimensions)
for each functional unit in interconnect router do
    if functional unit == Crossbar then
        PLACE UNIT(identifier, placement position, dimensions)
    else if functional unit == I/O Port then
        for each port in interconnect router  do
            placement position ← CALCULATE NEXT PLACEMENT POSITION( )
            PLACE UNIT(identifier, placement position, dimensions)
    else
        PLACE TSVs(topology, count)
```

```
————— Processing Elements —————
GET(dimensions) from ALE model
placement position ← initial anchor position
for each functional unit in processing element do
    PLACE UNIT(identifier, placement position, dimensions)
    placement position ← CALCULATE NEXT PLACEMENT POSITION( )
```

Figure 3.15. Routines for generation of fine-grained cache memory, interconnect router and processing element floorplans.

placed at Manhattan distance from one another with a target aspect ratio determined by the system floorplan. Although basic, Ctherm's routines can be extended with techniques such as [101] for fast thermal-aware floorplanning of tiles, and [102] for minimizing the wirelength between functional units. Furthermore, support for components such as programmable accelerators can be added through the inclusion of additional planning routines.

The floorplanner computes the overall die size, and generates a die descriptor containing a physical description of the die and its material properties, in addition to the detailed system floorplan. For 3D stacked SoCs, the floorplanner is executed iteratively till floorplans for each die have been generated, and the stack descriptor describes each die of the stack together with a path to its corresponding floorplan. This descriptor also includes the inter-tier thermal conductivity corresponding to the chosen TSV count and topology.

3.4.1.2 Thermal-Functional Co-simulation Platform

The second stage of the Ctherm framework performs the thermal-functional evaluation of the SoC using the generated physical model, and the input SystemC top-level file configured with the system specifications. The co-simulator consists of a cycle-accurate simulation engine integrated with an embedded thermal simulator. The simulation engine instantiates components from the *SoCLiB* IP library [103] which consists of an extensive set of SystemC behavioural models for processor cores, interconnects, caches, memories, controllers and accelerators. Thermal simulation of modeled platforms is enabled by an adapted version of the *3D-ICE* thermal simulation engine [82], embedded within the co-simulation platform core.

(i) Power Mapper A thermal model for the system is generated based on the die descriptor and floorplan generated in the previous stage of the framework. The die is discretized into a grid of thermal cells, with cell size determining the resolution of the resulting thermal maps, and also complexity of the thermal model. The thermal model generation step is performed only once per simulation run, and usually completes in under a minute for cell sizes of 50μm. Logging of thermal maps on the other hand poses a significant overhead that is dependent on thermal cell size. However, since temperatures on die do not change at the one-cycle time scale, a logging interval of 50μs provides sufficient resolution for visualization of hotspots in most simulations, with an acceptable overhead.

SystemC components of the SoCLiB IP library are augmented with an activity tracking function that logs the operations performed by their constituent functional units on a cycle-accurate basis. All activity frames are evaluated at the start of every thermal simulation time step, and are converted into a detailed power map using the corresponding ALE model for each component. Since the model contains internal organization details of components, an exact list of units activated and their corresponding power dissipation can be computed by the power mapper.

The thermal simulation is triggered once activity data has been collected from all components, at a rate determined by the thermal time step. Time steps larger than the

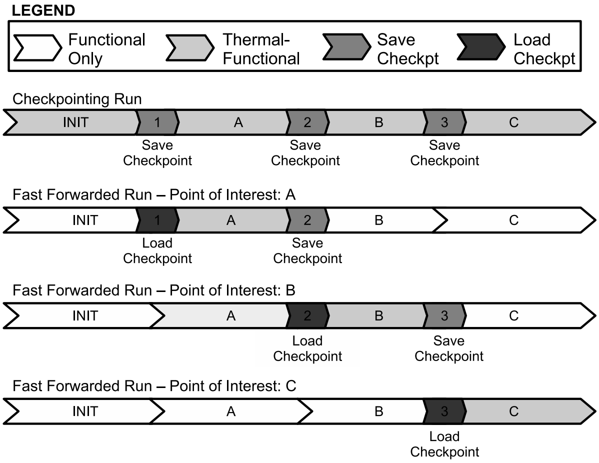


Figure 3.16. Illustration of checkpointing run generating thermal checkpoints, followed by fast forwarded runs using saved thermal checkpoints.

execution step requires aggregation or averaging of power maps until insertion, which can result in false hotspots and false blurring of hotspots respectively.

(ii) Checkpointed Thermal Simulation Thermal simulations are normally carried out for the same duration of time as the functional simulation. However, in some cases, the thermal behaviour that we want to characterize can occur much later in the simulation. In order to observe this behaviour in isolation, Ctherm supports the discrete starting and stopping of the thermal simulation engine at any time during the execution. This is achieved through the *Thermal Simhelper* component which acts as an interface between the virtual and co-simulation platforms, enabling control of the thermal simulation both from the behavioural model of components, as well as from software executing on the virtual platform. With thermal simulation disabled, functional simulation speed increases by a minimum of 30%.

A drawback of this approach is that it results in the loss of thermal continuity until the point thermal simulation is enabled. Therefore, fast-forwarding from the start of functional simulation results in the thermal simulation starting with a die at initial temperature (ambient). To overcome this, we integrated the ability to save thermal checkpoints to disk using the Thermal Simhelper. Furthermore, we adapted the 3D-ICE core to initialize the system's thermal model using user-specified checkpoints. Consequently, Ctherm allows the saving of thermal checkpoints to disk during simulation, and the initialization of the system's thermal state with any user specified checkpoint during the simulation. In order to do this, thermal simulation is first performed for the time interval that will be fast-forwarded, and a thermal checkpoint is created at the end, as illustrated in Figure 3.16. Subsequent thermal simulations can begin directly at the point of interest, after initializing the thermal state with the saved checkpoint. Once initialized, the system appears as if thermal simulation has been running since the beginning. Checkpointed simulations can drastically cut time spent in simulating specific temporal effects, and testing systems post optimization. In addition to thermal checkpointing, Simhelper also allows the temperature map of the die to be saved and read at any point during the co-simulation. These maps can be sampled at discrete locations to emulate temperature sensor readouts.

3.4.2 Evaluation

The effectiveness of Ctherm is illustrated using real design cases, each performed on an Intel PentiumD 3.0GHz machine with 4GB memory. *CACTI* [35], *WATTCH* [98] and *ORION* [99] are used as ALE estimators for caches, processors and the system interconnect respectively. All design cases consider the $90nm$ technology node, a 200MHz clock frequency and a $340K$ ($67°C$) critical temperature. The results of the evaluation are presented in two parts, starting with the accuracy of thermal characterization and simulation speed. This is followed by the four design cases.

3.4.2.1 Validation, Accuracy and Simulation Speed

Since Ctherm implements changes to the 3D-ICE core in order to enable thermal checkpointing, the effect of these modifications was validated against an unmodified version. The two versions were found to produce matching thermal profiles, and the implemented modifications were found to induce no errors or variations in the thermal simulation results. The modified version therefore has the same accuracy and correctness as the standard 3D-ICE core. To verify the power mapper, the total power inserted per operation during the simulation was validated against the values reported by the ALE estimators, and the locations of these insertions were compared against manually computed locations and verified to be correct across different system configurations.

Ctherm improves thermal simulation accuracy by using generated fine-grained internal floorplans for components. This enables the accurate distribution of dissipated power across the constituent functional units of components. To illustrate the advantage such fine-grained modelling provides, we examine the influence of varying the wordline sharing factor (N_{spd}) on the temperature profile of a 32KB data cache

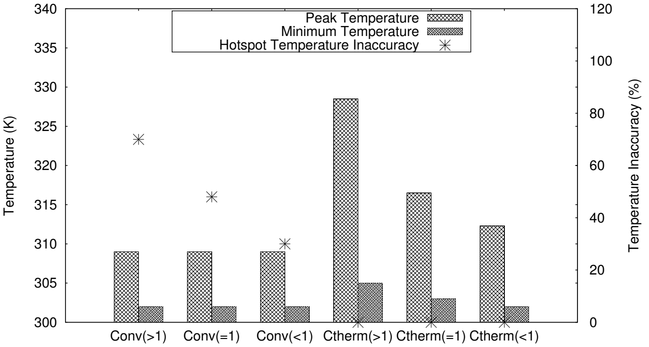
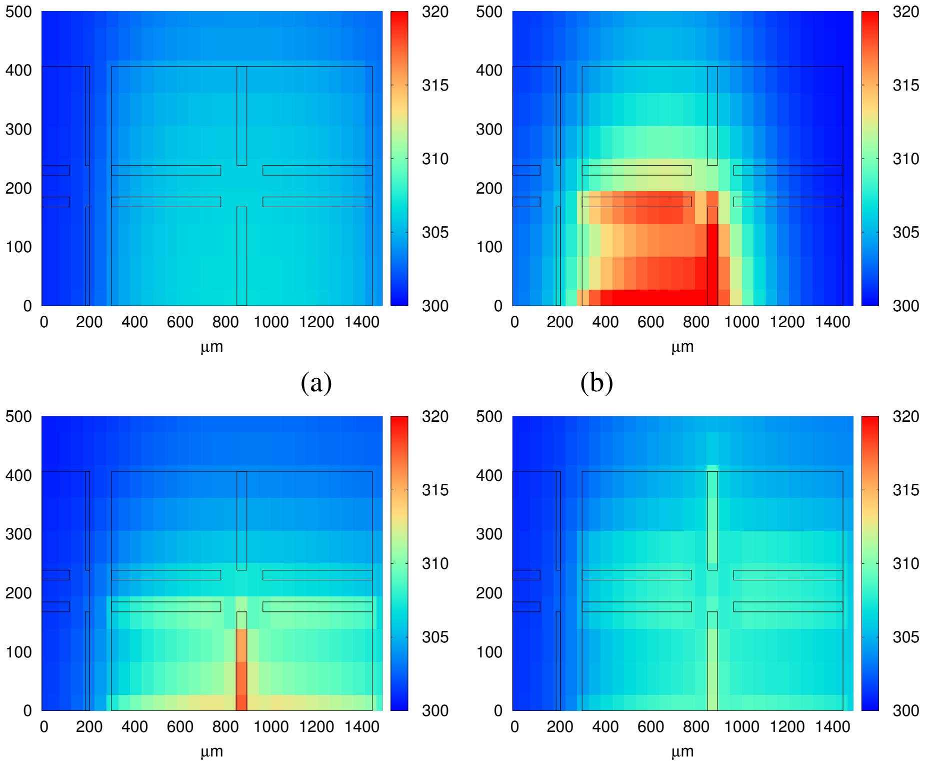


Figure 3.17. Comparison of thermal simulation with abstract floorplans (*Conv*) and thermal simulation using Ctherm's fine grained floorplans (*Ctherm*) for a cache memory with varying wordline sharing factor (N_{spd}). Three cases: $N_{spd} > 1$, $N_{spd} = 1$, $N_{spd} < 1$.

following $1E5$ sustained accesses to a single cacheline. This evaluation follows the illustration previously shown in Figure 3.13.

The three cache organizations ($N_{spd} > 1$, $N_{spd} = 1$, and $N_{spd} < 1$) are first evaluated using the conventional thermal simulation approach (*Conv*), i.e. abstracting component internals. This abstraction results in the distribution of dissipated power across the complete area of the cache, and thus the peak and minimum temperatures for the three are identical as observed in Figure 3.17. The organizations are subsequently evaluated using Ctherm's fine-grained floorplans which account for the N_{spd} parameter. The results of this evaluation reveal remarkable differences in the peak temperature for each case, indicating the presence of a hotspot especially in the case of $N_{spd} > 1$. The difference in temperature profiles as a result of varying distribution of dissipated power are clearly visible in Figure 3.18(b)-(d). The temperature map obtained with abstracted component internals in Figure 3.18(a) on the other hand incorrectly reports peak temperatures up to 70% lower than those obtained with fine-grained simulation. This highlights the importance of fine-grained floorplans during tDSE and thermal-aware system design.

The speed of the Ctherm framework largely depends on the complexity of the system being evaluated, granularity of thermal simulation and die size. In order to provide an idea of the average simulation speed in realistic design scenarios, we report the runtime Ctherm incurred in performing the four design cases listed in the following section. The simulation speeds per design case are reported in Table 3.3 alongside the corresponding die and *thermal cell (Tcell)* sizes.



(a) (b)

(c) (d)

Figure 3.18. Heatmaps for 32KB data cache using (a) conventional thermal-simulation with abstracted cache internals, and Ctherm with fine-grained cache floorplans for (b) $N_{spd} > 1$, (c) $N_{spd} = 1$, (d) $N_{spd} < 1$. Temperatures are measured in *Kelvin (K)*.

Table 3.3. Simulation speed across design cases listed in Section 3.4.2.2

Case	*Tcell Size (µm)*	*Die Size (µm × µm)*	*Sim. Speed (cycles/min)*
(i) (FP_A,FP_B)	100	1400 × 1200	200K
(i)(FP_C,FP_D)	100	700 × 1200 (×2)	200K
(ii)	100	1400 × 1200	400K
(iii)	50	1350 × 850	215K
(iv)	100	700 × 600	600K

Table 3.4. Thermal-aware performance estimates for floorplan exploration. * marks thermal runaway.

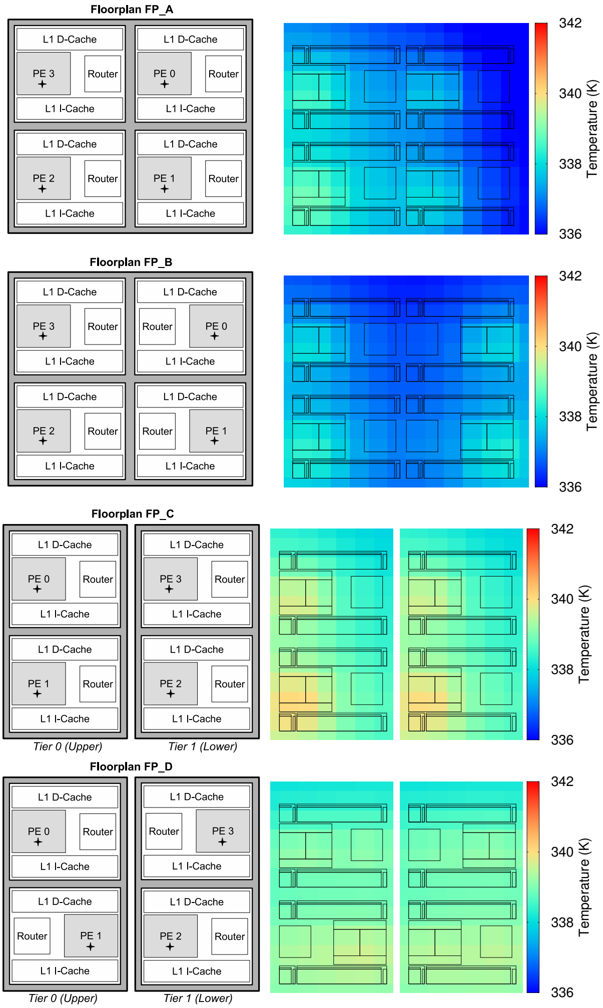
	FP_A	*FP_B*	*FP_C*	*FP_D*
Cumulative CPI	1.07	1.05	1.33	1.24
Data Refs/cycle	0.265	0.270	0.215	0.229
Avg. Off Time	55%	45%	100%*	100%*

3.4.2.2 Design Cases

In this section, we illustrate the applicability of Ctherm using four specific design cases.

(i) 2D/3D Floorplan Selection On account of severe space constraints in portable devices, embedded MPSoCs cannot afford to have extravagant heatsinks, and it is therefore important for them to be designed for operation within an extremely narrow thermal envelope. This design case involves the evaluation of floorplan options for a multiprocessor array with four *processing elements (PE)*. Candidate floorplans for the array are illustrated in Figure 3.19(a). Each floorplan depicts four tiles containing a simple RISC PE, private *4KB 2-way* instruction and data caches, a $64b$ 5-port network-on-chip router connecting to neighbouring tiles and a temperature sensor at the location marked as $+$. Floorplan FP_A represents a homogeneous array of identical tiles arranged across the die. In FP_B, tiles are mirrored across the vertical axis in order to increase the distance between PEs and thus decrease thermal interference. FP_C is a two tier die-stacked implementation of the same system. Such an arrangement however creates a region of high power density by stacking PEs one above the other. FP_D mitigates this issue by mirroring the floorplans of both tiers with a checkerboard pattern of PEs. In terms of (3.1), these floorplans vary in their power density (Q/A), and distance of power dissipating elements from the heatsink surfaces ($l_{x,y,z}$). A *Dynamic Thermal Management (DTM)* scheme is included in PE tiles to disable switching activity as soon as the critical temperature is breached, with a reactivation temperature margin of $2K$. The $dijkstra$ shortest-path benchmark from the $MiBench$ suite [49] is used as a test workload for the virtual platform. The thermal and functional performance of the system for each floorplan is reported in Table 3.4.

Average Off Time indicates the fraction of the simulation time for which PEs remained disabled due to a DTM action, thus indicating the thermal efficiency of floorplans. FP_B's spreading out of PEs is seen to result in decreased average off time, causing a decrease in the number of cycles taken to execute the workload, i.e. *cycles per instruction (CPI)*. Figure 3.19(b) illustrates the thermal maps for each



(a) (b)

Figure 3.19. (a) Floorplan options for the four-PE multiprocessor array. Temperature sensor locations are marked as +. (b) Thermal maps sampled during execution of the *MiBench-dijkstra* workload. Note: Tier 0 is located close to the heatsink/connection to ambient, temperatures are measured in *Kelvin (K)*.

floorplan. These were automatically generated by Ctherm together with the floorplan overlay. The 3D options FP_C and FP_D split the four PEs over two dies, thus halving the overall area footprint for the system, without changing the total power dissipation. Since the power dissipated by PEs is conducted to the ambience through

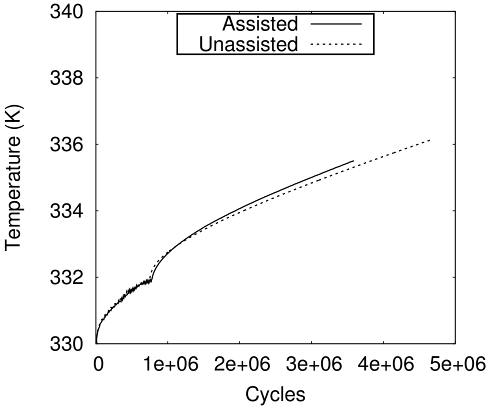
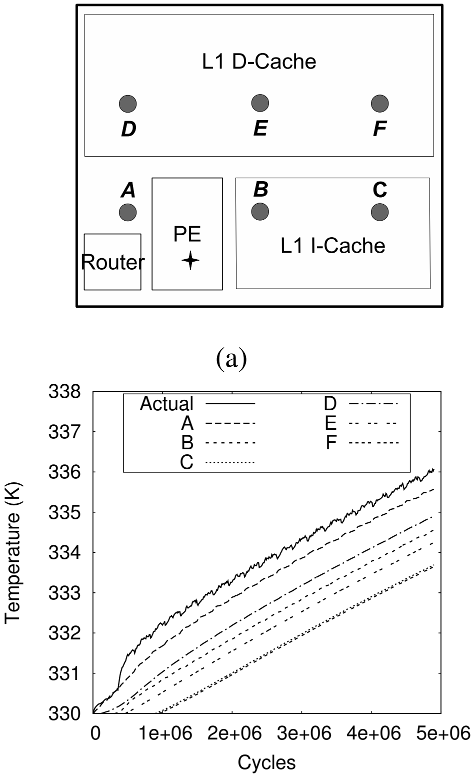


Figure 3.20. Comparison of temperature profiles obtained using a conventional data cache (*Unassisted*), and a data cache with an 8-entry assist (*Assisted*).

the surface of the die, this reduction results in decreased cooling efficiency. In fact, the cooling efficiency of the 3D configuration is constrained to such an extent that even after disabling all switching activity in the system, temperatures continue to rise due to the leakage power dissipation. Floorplans FP_C and FP_D thus encounter a thermal runaway. This result indicates that in order to use the 3D design points, either cooling efficiency must be improved, or leakage control/power gating mechanisms must be integrated into the architecture to limit leakage power dissipation especially in lower tiers of the stack.

(ii) Thermal-aware Architecture Exploration In Chapter 2.3.2, we presented the *persistence selective caching (PSC)* scheme which used a small fully associative assist cache to decrease the average latency and energy for data cache accesses. We evaluate the thermal impact of such a cache assist on the performance of uniprocessor SoC consisting of a single PE, a *64KB 4-way* data cache and an *8KB 2-way* instruction cache. A multi-dimensional array implementation of the *first sum (kernel11)* workload from the *Livermore Loop Kernels* benchmark [104] is executed on the platform, with an array size of 168×168. The assist scheme is observed to reduce the average latency of memory accesses, and consequently CPI by up to 24% compared to a conventional data cache. While this improves performance, it results in execution proceeding at a faster rate, leading to a quicker ramp up of temperatures as seen in Figure 3.20. However, since CPI is reduced, execution performance completes approximately 1 million cycles earlier, leading to a peak temperature that is $0.9K$ lower than the conventional data cache. In terms of (3.1), PSC increases activity rate (α) of PEs,



(a)

(b)

Figure 3.21. (a) Floorplan indicating candidate locations for temperature sensors. (b) Tracking from each sensor location compared to the actual temperature of the PE.

and decreases heat generation by reducing energy dissipation (E). Consequently, any architectural optimization that affects activity rate must be evaluated in light of their thermal effects in order to obtain a realistic estimate of their actual impact on performance.

(iii) Temperature Sensor Placement The ability to monitor die temperature is a prerequisite for performing runtime thermal management. Modern on-chip sensors rely on a range of techniques to measure temperature [105][106][107]. However, in order to enable accurate digital readouts of the measurement, sensors usually integrate an *analog-to-digital converter (ADC)* [105][108] or a *time-to-digital converter (TDC)* [106], which occupies a considerable amount of area. Their size therefore limits the number, as well as the locations at which such sensors can be placed. Sensors located far from hotspots exhibit delayed responses and inaccurately measure hotspot temperature, as we illustrate in this design case. It is therefore prudent to evaluate placement options for sensors by using thermal-functional co-simulation of the system. In the event that a sensor cannot be placed close to a hotspot, results of the co-simulation yield information on how to calibrate the DTM's temperature margins so as to better track hotspot temperatures.

Figure 3.21(a) shows a $1350\mu\mathrm{m}\times850\mu\mathrm{m}$ die with a single PE and its caches. The ideal temperature sensor location is marked with a $+$ symbol, while candidate locations for sensor placement are indicated with labels A through F. Locations shown inside the caches are situated in wiring tracks with unused active regions. Figure 3.21(b) plots the rise in PE temperature as a function of time, and also the perceived temperature rise at each sensor location. Due to their distance from the PE, sensors at C and F exhibit a delayed response, and report temperature readings $2K$ lower than the actual. A DTM using sensors at these locations must account for this inaccuracy in its temperature margins in order to effectively control system temperature.

(iv) Thermal Impact of Software Workloads The thermal behaviour of processors in SoCs is largely determined by the workloads that execute on them. Minor changes to the software algorithm can have a drastic impact on both execution performance as well as system power and thermal profiles. To illustrate this, we extend the *first sum* workload of design case 2 with an additional kernel option (B). The original kernel (A) performs the first sum computation in a column-first manner, while the new kernel (B) follows a row-first approach. The kernels are precluded by an initialization of the multidimensional arrays, which executes for approximately $2.75M$ cycles, following which the computation begins. In order to emulate the iterative nature of optimizations and thermal evaluations for workloads, each kernel's evaluation was repeated 10 times. Two sets of runs were performed, one using the conventional continuous thermal simulation, and another using Ctherm's checkpointed simulation.

The conventional approach without checkpointing requires the simulation to be restarted from the beginning of execution following each optimization to the kernel. This means that in addition to the kernel under test, all other sections of the program must also be re-simulated. Thus, the initialization section is simulated a total of 20 times, 10 times for each kernel. Note that since each simulation in the conventional approach causes the entire program to be executed, the runtime of the initialization section is included within that of the kernels. With the checkpointed approach, on the other hand, the initialization is simulated only once and its resulting thermal

Table 3.5. Comparison of conventional and checkpointed thermal simulation runtime

	ITERATIONS		RUNTIME (SECONDS)	
	Conv	*Checkpointed*	*Conv*	*Checkpointed*
Initialization	20	1	-	159
Kernel A	10	10	2060	760
Kernel B	10	10	4640	3340
Total Runtime (seconds)			*6700*	*4259*
Improvement over Conventional Approach				**36%**

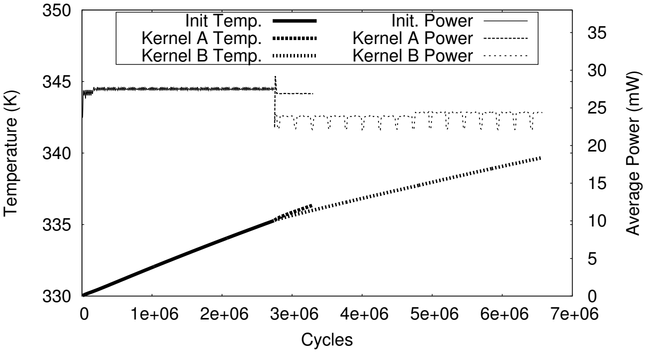


Figure 3.22. Temperature and power profiles for initialization, kernel A and kernel B of the extended *first sum* workload

map saved as a checkpoint. This checkpoint is subsequently used as a starting point for each kernel's thermal simulation, thereby allowing the initialization to be fast-forwarded. Total runtime is consequently decreased by 36% as observed in Table 3.5. The temperature and power profiles for the initialization and the two kernels are reported in Figure 3.22. The row-first approach of kernel B results in a high miss-rate observable from the repeated fluctuations in its power trace. This consequently increases runtime, resulting in a higher leakage energy consumption, and a peak temperature over 4K higher than that of kernel A.

3.4.3 Additional Media

Ctherm enables the generation of videos depicting transient variations in on-chip temperature due to power dissipation in components. Two such videos are available for demonstration purposes. The first video [109] depicts the transient response of a PE and its 32KB data cache during the first 2.3 million cycles of the execution of the Lee circuit routing algorithm. The second video [110] depicts the fine-grained internal temperature map of a 32KB data cache.

3.4.4 Conclusions

Ctherm is an integrated framework that enables the thermal-aware design space exploration of systems-on-chip. It enables the characterization of internal component thermal behaviour by using fine-grained physical models for components, automatically generated from input system specifications. Ctherm's fine-grained modelling improves accuracy of hotspot temperature resolution by up to 70% as compared to the conventional approach that abstracts component internals. Furthermore, by introducing thermal checkpointing, the framework enables discrete thermal simulations which reduce the runtime of post-optimization evaluation runs by up to 36% over the conventional continuous simulation approach.

Summary

The performance and integration density of modern ICs is constrained by their physical characteristics and the thermal efficiency of their cooling interfaces. The complex thermal characteristics of 3D ICs further aggravate this issue due to their non-uniform heat transfer characteristics. In this chapter, we investigated the steady-state thermal behaviour of 3D ICs using a high-level characterization flow, and determined the impact of parameters in Nagata's equation. The characterization uncovered the critical influence of power density on hotspot magnitudes, the significance of thermal conductivity in determining temperature gradients, and the impact of stack depth on temperature profiles in 3D ICs. We further investigated how the choice of vertical interconnect topology influences electrical performance, area overheads, and thermal performance. Based on this understanding, we proposed the Ctherm framework that enables the thermal-aware exploration of the multiprocessor design space, facilitating the evaluation of system performance in light of runtime physical effects.

Associated Publications

The contents of this chapter are derived from the following publications:

1. S.S. Kumar, A. Aggarwal, R. Jagtap, A. Zjajo, R. van Leuken, ”System Level Methodology for Interconnect Aware and Temperature Constrained Power Man-

agement of 3-D MP-SOCs” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 7, pp. 1606-1619, July 2014

2. S.S. Kumar, A. Zjajo, R. van Leuken, ”Exploration of the Thermal Design Space in 3D Integrated Circuits”, *Physical Design for 3D Integrated Circuits*, CRC Press, December 2015, *Invited Book Chapter*
3. S.S. Kumar, A. Zjajo, R. van Leuken, ”Physical Characterization of Steady-State Temperature Profiles in Three-Dimensional Integrated Circuits” *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015
4. S.S. Kumar, A. Zjajo, R. van Leuken, ”Ctherm: An Integrated Framework for Thermal-Functional Co-simulation of Systems-on-Chip ” *Proceedings of the IEEE/Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pp.674-681, 2015
5. R. Jagtap, S.S. Kumar, R. van Leuken, ”A Methodology for Early Exploration of TSV Placement Topologies in 3D Stacked ICs” *Proceedings of the IEEE/Euromicro Conference on Digital System Design (DSD)*, pp. 382-388, 2012
6. A. Aggarwal, S.S. Kumar, A. Zjajo, R. van Leuken, “Temperature constrained power management scheme for 3D MPSoC,” Proceedings of the IEEE Workshop on Signal and Power Integrity (SPI), pp. 7-10, 2012

Associated Posters and Demonstrators

1. S.S. Kumar, A. Aggarwal, R. Jagtap, A. Zjajo, R. van Leuken, “Interconnect and Thermal Aware 3D Design Space Exploration”, Invited Presentation and Poster, *ICT.OPEN*, Eindhoven, The Netherlands, 2013
2. S.S. Kumar, R. van Leuken, A. Michos, A. Chahar, J. de Klerk, “Naga High-Performance Array Processor”, Poster and Demonstrator, *University Booth – Design Automation and Test in Europe (DATE)*, Grenoble, France, 2013

4

Runtime Temperature and Power Management for 3D Multiprocessors

In large scale *multiprocessor systems-on-chip (MPSoC)*, the utilization of *processing elements (PE)* varies based on the nature of the workload under execution. Heavily utilized PEs dissipate a greater amount of power and in thermally constrained systems, result in the formation of thermal hotspots. In 3D stacked multiprocessors, due to the thermal coupling between stacked dies, high activity in one tier can result in the formation of thermal hotspots on other over- and under-lying tiers. Sustained thermal gradients and high operating temperatures can however be detrimental to reliability, and result in the accelerated degradation of devices [31, 33]. *Dynamic Thermal Managers (DTM)* are therefore used to control on-chip temperatures at runtime by reducing the dynamic power dissipation of components. DTMs typically achieve such a reduction by decreasing the switching activity of components. However, this also results in degraded system performance [83].

The efficiency with which heat can be evacuated from a 3D IC is effectively a function of the system's physical characteristics and the thermal efficiency of its cooling interfaces, as previously discussed. For a tiled-multiprocessor, this is shown with an equivalent of *Nagata's* equation [32]:

$$\frac{\alpha_t(N_t N_g)E_t}{t_{pd}} \leq g \cdot \Delta T_{max} \quad where \;\; g = \kappa_{eff}\frac{A}{l_{x,y,z}} \tag{4.1}$$

where N_t represents the number of PE tiles each with N_g gates, energy dissipation E_t, average activity rate α_t and a clock period t_{pd} . ΔT_{max} represents the maximum permissible temperature difference between the components on-chip and the ambience through heat transfer surfaces of area A, situated at a distance $l_{x,y,z}$ from the power dissipation site. Note that although heat is also dissipated by the side walls, for simplicity of explanation, we only consider the primary heatsink situated at the top of the stack. κ_{eff} represents the effective thermal conductivity of the die stack and its *through silicon vias (TSV)*, while g is the effective thermal conductance between power dissipating elements and the heatsink surface. While the right hand side of (4.1) represents the thermal properties of the die-stack and its cooling surfaces, the left hand side determines the dynamic power dissipation of the system, which is of the general form:

$$P_{dynamic} = \frac{E_{dynamic}}{t_{pd}} \tag{4.2}$$

$$where \;\; E_{dynamic} = \alpha V^2 C_L \tag{4.3}$$

$$\therefore \;\; P_{dynamic} \propto \frac{\alpha V^2}{t_{pd}} \;\; or \;\; P_{dynamic} \propto \alpha V^2 f \tag{4.4}$$

where $P_{dynamic}$ is the dynamic power dissipation, $E_{dynamic}$ is the dynamic energy dissipated per clock period t_{pd}, V and f are the operating voltage and frequency levels respectively, and α is the activity rate of components with load capacitance C_L. Runtime power management techniques primarily focus on controlling the three variables in (4.4), and can be categorized as:

- **Activity Control:** Techniques that vary the temporal behaviour of components to control their activity (α). Such techniques are generally implemented at the architecture-level. Examples include thermal-aware instruction steering [111], pipeline throttling [112] and dynamic interconnect throttling [113][114][115].

- **Voltage-Frequency Control:** Techniques that vary the operating voltage(V) and frequency (f) to control power dissipation ($P_{dynamic}$). Such techniques are generally implemented at the system- and circuit-level.

- **Clock and Power Gating:** Techniques which freeze the clock signal ($t_{pd} \rightarrow \infty$) and disconnect the component from the power supply ($E_t = 0$), effectively stopping all switching activity in the gated component. While clock gating can be implemented at either system- or circuit-level, power gating is a predominantly circuit-level technique.

Clock and power gating are effective in arresting thermal emergencies. However, because they completely stall all activity in the gated component, they drastically reduce system performance [116]. For this reason, they are used as a last-resort when other techniques are ineffective [117]. Activity and Voltage-Frequency control, on the other hand, are used for power management since they enable fine-grained control of α_t, E_t and t_{pd}.

As we observed in Chapter 3.2, the magnitude of thermal hotspots and temperature gradients is directly influenced by stack power density. Effective regulation of power dissipation, and control of on-chip temperatures is therefore essential towards harnessing the performance of large, 3D stacked many-core processor arrays, and ensuring dependable operation. In this chapter, we propose two techniques that fulfil these essential requirements. These include:

- A temperature-aware *Dynamic Voltage Frequency Scaling (DVFS)* scheme that regulates power dissipation of PEs in 3D stacks, while taking into account their physical position, utilization, and thermal relationship with other PEs in the system. The proposed technique maximizes execution performance within the available temperature and power budgets.

- A temperature-aware adaptive *network-on-chip (NoC)* routing strategy that balances temperature profiles and reduces thermal gradients in 3D stacked multiprocessors by adaptively routing interconnect traffic away from hotspots.

The proposed techniques enable the many-core system to adapt to changing operating conditions, maximizing performance within the constraints of prevailing thermal conditions, and effectively managing power dissipation in order to reduce variations in on-chip temperatures.

4.1 Temperature-Aware DVFS for Stacked Die Architectures

Dynamic Voltage Frequency Scaling (DVFS) is a well known runtime technique that varies the operating voltage and frequency of PEs in order to manage their power dissipation [118–122]. DVFS schemes rely on the premise that for most applications, execution performance can be marginally decreased without violating deadlines. Consequently, the system can be continuously *adapted* to better match application performance requirements with the available power budget. DVFS schemes have also been used to manage on-chip temperatures in multiprocessor systems [83, 123, 124]. In [124] for instance, *voltage and frequency (V-F)* levels are scaled to control power dissipation based on local temperatures. However, this scheme does not account for thermal dependencies between PEs while making power management decisions, and consequently, applying it to 3D ICs is non-trivial considering the predominance of thermal coupling between stacked dies [125]. Sabry et al. highlighted this inefficacy of conventional DVFS [126] by analyzing the variation in thermal conditions between the extremities of deep stacks. They found that the complex nature of heat flow within die stacks resulted in PEs on the lowest tiers turning off more often than others. Their solution consequently focussed on improving thermal conditions in deep tiers through the use of a microchannel liquid cooling system. Ayala et al. similarly proposed thermal TSVs as a method of evacuating heat generated in deep tiers [127]. Zhu et al. [128] on the other hand used a combination of scheduling, workload migration and run-time power budgeting to improve the performance of 3D stacked multiprocessors.

The thermal relationship between components in a 3D IC is a function of the effective thermal conductance of the heat flow path between them. This conductance consists of two parts: conductance between the components, and conductance between the power dissipating component and the heat sinking surface. Consequently, the thermal impact on a victim PE is strongly influenced by the proximity of the power dissipating PE to the heat sinking surface, and by the magnitude of thermal coupling between the two PEs. The proposals of Sabry et al. [126] and Ayala et al. [127] focus on increasing conductance to the heat sink in order to mitigate heatflow between thermally coupled power dissipating elements in die stacks. Zhu's proposal [128] focuses on achieving the same result by migrating application tasks towards PEs with a lower conductance to the victim PE.

We propose a novel temperature-aware power management strategy that relies on the observation that the temperature of a PE is influenced not only by its local power dissipation, but also its position in the die stack and thermal relationship with other power dissipating elements within the system. The proposed 3D DVFS strategy evaluates the feasibility of V-F scaling on PEs at runtime by evaluating the thermal implications of such an action on other neighbouring PEs. This enables the optimal utilization PEs under a given temperature and power budget, while preventing those on deep tiers from being switched off due to thermal emergencies. 3D DVFS outperforms conventional DVFS both in its ability to maintain the temperatures of all PEs stable, as well as in its improvement of performance by increasing the aggregate system frequency.

4.1.1 3D DVFS

Figure 4.1 illustrates a *Power Manager (PM)* implementing 3D DVFS, and its control inputs and outputs. In order to determine appropriate V-F levels for PEs in a stack, 3D DVFS utilizes runtime monitoring of physical conditions such as activity, power

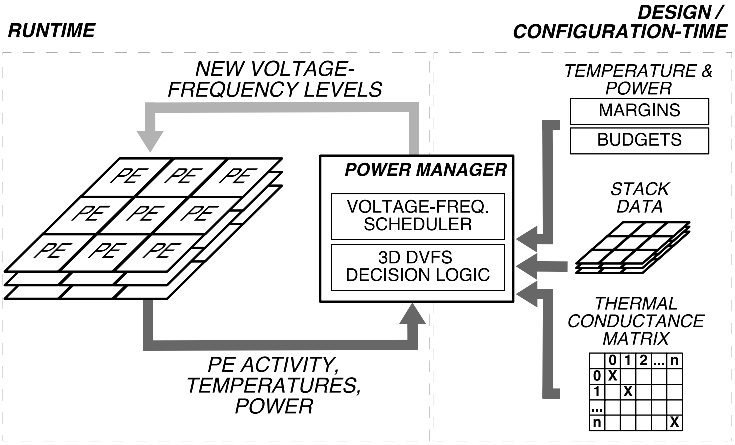


Figure 4.1. Control loop and inputs for the power management scheme

dissipation and on-chip temperatures. Coupled with a physical model of the die stack indicating the position of PEs and their thermal relationships, these vectors provide an effective information base for V-F scaling decisions.

DVFS decisions are made by the algorithm shown in Figure 4.2. The algorithm is invoked once at the beginning of each *control period*. This period essentially determines how often V-F scaling is performed in the system, and its duration depends primarily on two factors: the degree of temporal variation in PE activity, and the sampling rate of temperature sensors. The former is dependent on the behaviour of the application program, and how its characteristics vary as execution progresses. The latter is dependent on the thermal inertia of the system, which determines the timescales at which temperature variations are perceivable. In general, variations of under $0.5K$ are perceivable in the 10-100μs timescale, and temperature sensors are therefore typically sampled at intervals within this range. It is important that the control period be sufficiently short so as to enable V-F levels to be quickly adapted following changes in application activity, yet sufficiently long to minimize the overheads of frequent power management.

The algorithm shown in Figure 4.2 is divided into several stages, namely, *Initial Updates, Thermal Run-out, Convergence Check, Pull-up/ Pull-down,* and *Write-back and Reset*. No scaling decisions are made in the first two control periods after system startup. The system is initialized at maximum V-F levels, and begins execution with the maximum power dissipation.

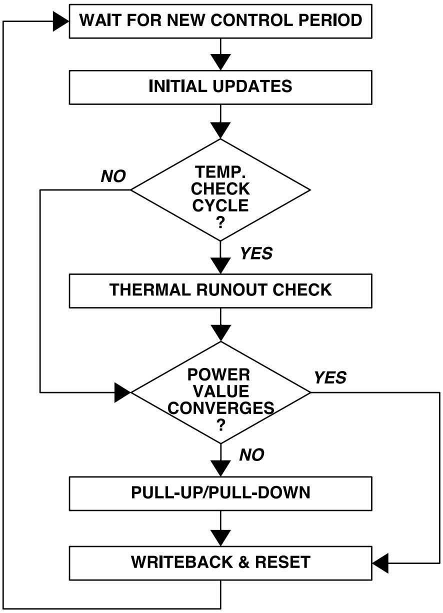


Figure 4.2. Flowchart illustrating the various stages of the power management algorithm

4.1.1.1 Initial Updates

At the beginning of each control period, the difference between the actual power dissipation and the power budget value is computed. This value represents the available budget that can be utilized by V-F upscaling. In the event that new temperature values are available, the difference between the actual and the critical temperatures of each PE is also computed.

4.1.1.2 Thermal Runout

This step ensures that the temperature of each PE is maintained within the safety margin. Each PE i is assigned a weight:

$$weight = w_a(1 - \alpha_i) + w_b \cdot G(victimPE, i) \tag{4.5}$$

where w_a and w_b are co-efficients whose values are obtained through a linear regression, once the physical structure of the stack is known. α_i is the average activity of PE_i, while G represents the normalized thermal conductance matrix. Each row in the matrix holds normalized thermal conductances between a particular PE and all the others in the system. In the event that a PE is close to its critical temperature, a thermally related PE with the highest weight is chosen for V-F downscaling. From the equation, a less active PE with a high thermal conductance with the victim is considered to have the heaviest weight, and is thus the prime candidate for downscaling. Following each scaling decision, the thermal impact of the new V-F levels of PE_i on the victim PE_j is computed in terms of the projected temperature difference δT_j.

$$\delta T_j = \frac{1}{G(i,j)} \alpha_i ({V_{i\ new}}^2 f_{i\ new} - {V_{i\ old}}^2 f_{i\ old}) \tag{4.6}$$

$$\delta T_j = \frac{1}{G(i,j)} \alpha_i \Delta ({V_i}^2 f_i) \tag{4.7}$$

where δT_j is the projected temperature difference arising at PE_j due to scaling of V-F levels from $(V_{i\ old}, f_{i\ old})$ to $(V_{i\ new}, f_{i\ new})$ at PE_i with activity rate α_i. $G(i,j)$ represents the effective thermal conductance of the heat flow path between the two PEs.

If a scaling action does not sufficiently decrease the temperature of a victim, the next candidate PE is selected and scaled down. This process continues until the aggregate δT is sufficient to bring the victim PE's temperature under the critical value. However, in the event that the victim remains at or exceeds the critical temperature, it is clock gated. It is recommended that the range of V-F values supported by the algorithm be set keeping in mind the power budget value. This ensures that even in the extreme case where all PEs are pulled down to their minimum V-F level, their power dissipation falls well within the power budget, thereby allowing the temperature of the critical PE to be brought within the safe margin.

Repeated fluctuations between V-F levels may, however, be observed in certain cases, incurring large performance and power penalties. As a means to avoid this, the V-F levels of PEs that were scaled down due to a victim are prevented from being reinstated until the victim is within the safe temperature margin. This is implemented by means of a special flag that can only be reset in the *Initial Updates* stage of the algorithm.

4.1.1.3 Convergence Check

In order to prevent frequent fluctuations in V-F levels, the algorithm considers the power value as converged when the actual power dissipation falls between *98%* and *100%* of the power budget value. On the other hand, if actual power is determined to fall below 98% of the budget, V-F levels may be scaled up in the *Pull Up/Pull Down* stage of the algorithm.

4.1.1.4 Pull Up/Pull Down

In this stage V-F levels of PEs is scaled in the event that actual power remains below 98% of the budget value. The PEs on which V-F scaling occurs is determined using the weighted equation:

$$\begin{aligned} weight_{pe} = w_c \cdot \alpha_{normalized} + w_d \cdot temperature\ margin_{normalized} \\ +w_e \cdot height_{normalized} + w_f \cdot area_{normalized} \end{aligned} \tag{4.8}$$

where w_c, w_d, w_e and w_f are weights whose values are determined through a linear regression at design time, and serve to establish the impact of their corresponding parameters. Since the height of the stack and area of PEs may be expected to remain constant even through floorplan revisions, and the temperature margin fixed within a range of operating temperatures, only the utilization can be considered as variable in this equation. However, since even the value of utilization can be generalized for a homogeneous MPSoC, these weights only need to be determined once for a given design. Following each scaling decision, the thermal impact of the candidate V-F levels is determined using (4.7), as previously explained in the Thermal Runout stage. Therefore, V-F scaling to maximize utilization of the power budget can only be performed if the temperature budget permits.

The PE with the largest weight is chosen for voltage-frequency upscaling. This upscaling is performed iteratively until no more PEs can be pulled up or if the total power reaches the *98%* window of convergence with the budget value. In the event that the budget has been exceeded, the pull down stage is invoked in order to achieve convergence. For V-F downscaling, the PE with the smallest weight is selected and the pull down is performed iteratively until no more PEs can be pulled down or until the total power falls below the budget value. At each instance of pull up and pull down, the difference between the PE's actual and critical temperatures is updated.

4.1.1.5 Write-Back and Reset:

Finally, the V-F level determined for each PE is actuated along with the gating signals if required. At this stage, internal parameters are reset, and the algorithm is suspended until the next control cycle.

4.1.2 Implementation Considerations

The use of on-chip power managers and V-F level shifters is motivated by the need for fast response times and fine-grained control of spatial power dissipation [129]. 3D integration offers the additional possibility of integrating power management hardware into the stack in the form of a dedicated die, or within an active interposer at the base of the die stack [130]. While this dissertation does not explore the hardware implementation of the power manager, this aspect of the design is discussed to provide a more complete understanding of the deeper implications of on-chip power management.

The first challenge pertains to the design and implementation of voltage domains in the system. *Dynamic Voltage Scaling (DVS)* at the core-granularity requires each PE to have a dedicated voltage regulator [129]. While this provides the fastest response times, it constitutes a significant area and power overhead in large MPSoCs. An alternative approach, used by the AsAP-2 many-core array involves using multiple supply rails operating at different voltages [131]. Rather than scaling voltage, PEs simply *attach* themselves to the supply rails corresponding to their assigned voltage level. This approach enables fast transitions between voltage levels ($2ns$ [132]) and facilitates power gating, however, at the cost of increased complexity in the power delivery network. A middle path between the two approaches involves grouping multiple PEs into a single voltage domain, or *voltage islands*. Each island therefore has a single power delivery network and a dedicated voltage regulator. Although this decreases design complexity, it results in the under-utilization of temperature and power budgets since all PEs are forced to operate at the same voltage. 3D DVFS, however, effectively mitigates such losses.

The second challenge pertains to the design and implementation of *Dynamic Frequency Scaling (DFS)* in the system. The simplest approach involves using a divider to derive a variable frequency local version of a global clock signal. However, large SoCs are increasingly migrating towards a *globally asynchronous locally synchronous (GALS)* strategy to reduce design complexity and mitigate timing issues in large systems. Local clocks can in this case be generated using a ring-oscillator with a configurable divider [131]. The costs of implementing per-core DFS are low given the relatively small size of the oscillator compared to MPSoC tiles. Consequently, DFS is most often implemented at per-core granularity, even in systems with voltage-islands [133]. By supporting multiple frequency levels for each voltage, large islands can be prevented from completely switching between voltage levels by performing only DFS on their constituent PEs.

4.1.3 Evaluation

In order to evaluate and compare the proposed 3D DVFS strategy with the state-of-the-art, a cycle accurate SystemC model of the power manager was created. A corresponding power model for the PE was generated for various voltage-frequency

Table 4.1. System Configuration

SYSTEM		POWER MANAGEMENT	
System Size	3×2×2	Cooling	Heatspreader + Sink
Processor	32b PISA	Heat Transfer Co-efficient	1E5 $Wm^{-2}K^{-1}$
Tech. Node	90nm	TSV/tile	74
Die Size	9mm × 9mm	Convergence Window	2%
Temp. Sensors	12 (1 per tile)	Temp. Margin	2K
Temp. Accuracy	1K	Critical Temp.	320K
Sampling Interval	1ms	Control Period	$50\mu s$

Table 4.2. DVS levels with corresponding DFS levels

		CLOCK GATED	FREQUENCY 1	FREQUENCY 2
Voltage 1	0.855V	0MHz	700MHz	800MHz
Voltage 2	0.956V	0MHz	900MHz	1000MHz
Voltage 3	1.048V	0MHz	1100MHz	1200MHz

operating points using *Wattch* [98] for the *PISA* instruction set architecture [61]. The *basicmath* workload of the MiBench benchmark suite [49] was used as a test application. This particular workload contains a mix of varied computations and load-store operations, incorporates transient variations in behaviour throughout its execution, and represents the activity patterns of a typical computation kernel [60]. The system configuration used for the evaluation is listed in Table 4.1, and the V-F levels supported by the power manager, in Table 4.2. The temperature sampling interval was set taking the measurement accuracy into account.

The proposed 3D DVFS strategy is compared against conventional 2D DVFS [124] at both per-core as well as island granularities. The three-tier stack and its island partitioning are illustrated in Figure 4.3.

4.1.3.1 Per-core Granularity

Figure 4.4(a) illustrates the temperature of PE0, situated on the lowest tier of the three-tier stack with both the conventional 2D DVFS as well as the presented 3D DVFS strategies. Conventional 2D DVFS schemes consider the temperatures of PEs independently and are oblivious to the physical structure of the stack. Consequently, the temperature is observed to fluctuate within the margin, and occasionally breach the critical limit resulting in the PE being clock gated and its execution stalled. The new 3D approach however is observed to maintain temperatures well below the temperature margin without any fluctuations. As a result, PE0 completes execution of its task much sooner with the new approach, than with the conventional approach. This is observed

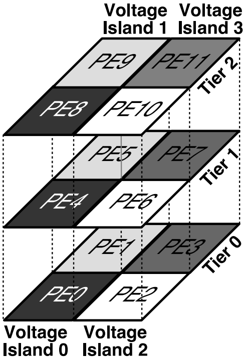
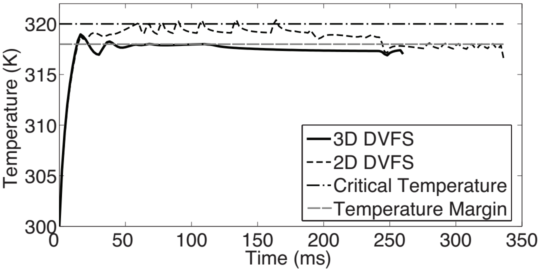
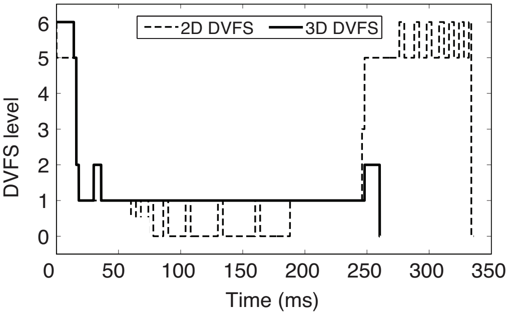


Figure 4.3. Voltage Island partitioning



(a)



(b)

Figure 4.4. (a) Operating temperature of a PE on the lowest tier of the stack illustrating the effective temperature control with the new approach. (b) Operating voltage-frequency levels on PE0

Table 4.3. Comparison of execution performance with each DVFS strategy

	2D DVFS	3D DVFS
Total Execution Time	336.5ms	260.35ms
Stall Time in Tier 0	106.5ms	0ms
Tier 0 Performance Loss	78.38%	38.48%
Tier 1 Performance Loss	29.28%	37.8%
Tier 2 Performance Loss	0%	29.34%

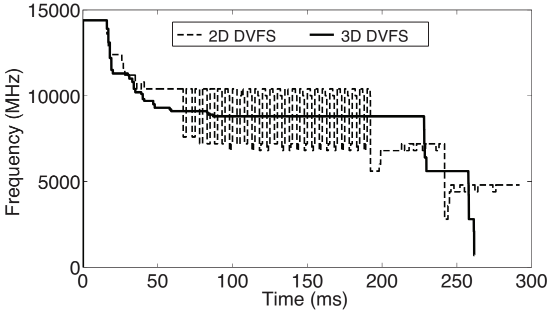
in Figure 4.4(a), with the temperature profile for the new approach terminating close to the $250ms$ mark, well ahead of the conventional approach which extends upto $340ms$.

The corresponding V-F profiles are shown in Figure 4.4(b) for PE0. In the new approach, the weighted equation used to determine candidates for scaling ensures that the temperature of PEs on the lowest tiers of the stack is considered before switching any voltage-frequency levels. As a consequence, all PEs are operated at voltage-frequency levels lower than the maximum thereby reducing their power dissipation, and ensuring that they never cross the critical temperature margin. This results in a more uniform utilization of PEs within the system, and eliminates execution stalls due to thermal runoff. This yields a performance improvement of upto 19.55%. The summary of the execution performance with both schemes is presented in Table 4.3. Furthermore, since PEs remain operation as a consequence of the lower temperatures, aggregate operating frequency of the MPSoC is increased, as indicated in Figure 4.5.

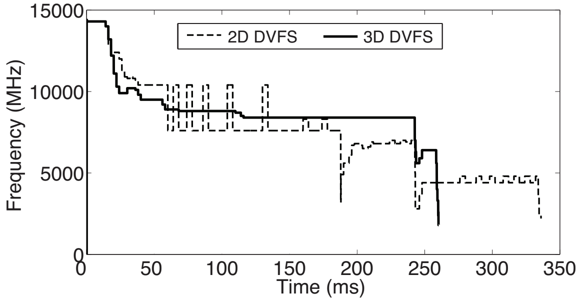
In Chapter 3.3, we explored the implications of TSV topologies on thermal performance, and consequently on execution performance with a conventional DVFS scheme. The shielded topology was found to be thermally superior due to its higher TSV count, and consequently, yielded better execution performance than the isolated topology. Figure 4.5 illustrates the sum of frequencies for the MPSoC achieved with the shielded and isolated topologies, with both power management strategies. 3D DVFS minimizes the execution performance difference between the two topologies by effectively managing the temperature and power budgets. The weighted selection of candidate PEs for scaling essentially adapts power dissipation in the stack to match the available vertical thermal conductance. Thus, when coupled with 3D DVFS, the isolated topology is more preferable than the shielded topology on account of its lower area overheads.

4.1.3.2 Island Granularity

The stack is subsequently partitioned into voltage islands as illustrated in Figure 4.3. The process essentially groups vertically adjacent PEs with strong thermal



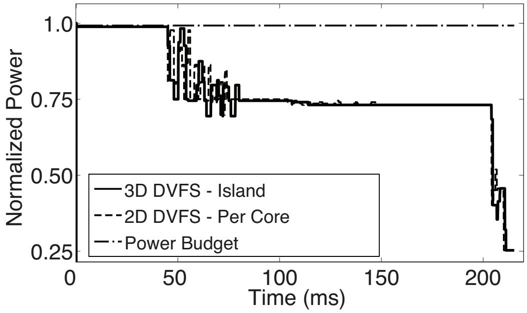
(a)



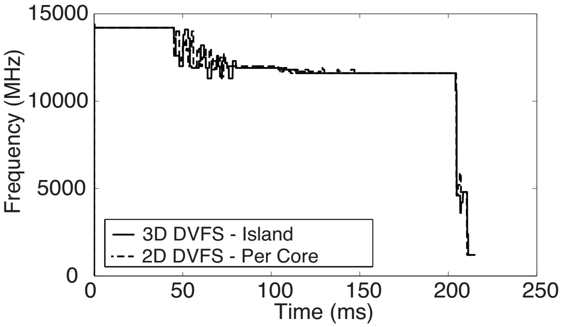
(b)

Figure 4.5. Sum of frequencies of all PEs with: (a) Shielded topology, (b) Isolated topology. Note that the sum of frequencies achieved with the shielded topology using the conventional 2D approach is higher as compared to the Isolated case, resulting in the reduction of execution time. The proposed 3D DVFS approach achieves shorter execution times than the 2D approach with both topologies.

relationships into a single voltage domain. Similar to the previous experiment, the voltage island partitioned stack with the proposed scheme is compared to a stack with a per-core 2D DVFS scheme. The total power dissipation for the two schemes is shown in Figure 4.6(a). While both schemes maintain power dissipation well within the set power budget, the proposed 3D DVFS approach is seen to result in a smoother profile with much fewer fluctuations than the 2D approach. The sum of frequencies in Figure 4.6(b) is also observed to be smoother with the new approach. Table 4.4 lists the total number of voltage-frequency transitions that occur during execution in the MPSoC, with both the conventional 2D as well as the new 3D DVFS schemes. Our scheme is seen to result in fewer transitions at each PE, as well as fewer fluctuations in the aggregate frequency of the system as compared to the conventional 2D scheme.



(a)



(b)

Figure 4.6. (a) Total power dissipation of the stack (b) Sum of frequencies with PEs grouped into islands

With the 2D scheme, PEs on the upper tiers are found to operate at the highest frequency level that their local power and temperature budgets allow. On the other hand, those on lower tiers run at much lower frequencies on account of their increased temperature. The new scheme considers the temperature of PEs on lower tiers while scaling the island voltage and the frequency of individual PEs on higher tiers. This effectively improves the performance of PEs deep within the stack. The island partitioning however, also restricts PEs in the upper tiers to voltage-frequency levels lower than that achievable with the per-core 2D approach, i.e. only a part of the available performance from the upper PEs is utilized. The two approaches are noted to present similar results in terms of execution performance. While the 2D approach offers this performance primarily from PEs in tiers closer to the heatsink, the new 3D approach achieves the same performance by ensuring that PEs even in the lower tiers

Table 4.4. Voltage-Frequency level transitions with the conventional 2D approach and the new 3D approach

		PER CORE		ISLAND	
		2D DVFS	3D DVFS	2D DVFS	3D DVFS
TIER 2	*PE 11*	1	5	16	23
	PE 10	1	6	18	5
	PE 9	1	6	19	5
	PE 8	1	7	22	11
TIER 1	*PE 7*	16	4	17	23
	PE 6	12	5	20	5
	PE 5	12	5	19	5
	PE 4	11	6	25	13
TIER 0	*PE 3*	33	6	29	33
	PE 2	31	7	28	9
	PE 1	31	5	26	9
	PE 0	33	7	31	17
AGGREGATE FREQ.		67	30	111	57

remain active. This leads to a more uniform utilization of all PEs in the MPSoC rather than the preferential utilization of only a few close to the heatsink. The similarity in temperature and frequency profiles illustrates that the performance of 2D per-core DVFS can be achieved even while using voltage islands. Since our proposed scheme achieves this performance through the uniform utilization of all PEs in the MP-SoC, devices can be expected to wear more evenly than with the conventional scheme which results in the over-utilization of cooler PEs.

4.1.4 Conclusions

The complex nature of heatflow paths within 3D MPSoCs results in non-uniform operating conditions for stacked components. Conventional 2D Dynamic Voltage Frequency Scaling (DVFS) schemes do not take the thermal characteristics of die stacks into account when performing power management. This results in the preferential utilization of cooler PEs, and degraded performance due to the under-utilization of system resources. To address the short-comings of state-of-the-art schemes, we proposed a novel temperature-aware 3D DVFS strategy that takes into account the physical composition of the die stack, positional information of PEs as well as runtime system conditions when making power management decisions. Using this information, the proposed 3D DVFS strategy evaluates the feasibility of V-F scaling on PEs at runtime by evaluating the thermal implications of such an action on other neighbouring PEs. The proposed strategy enables the optimal utilization PEs under a given temperature and power budget, while preventing those on deep tiers from being switched off due to thermal emergencies. Execution performance is improved by upto 19.55$\%$ as compared to state-of-the-art schemes for per-core DVFS granularity with uniform utilization of PEs across the stack, and smoother V-F profiles. Furthermore, when used at island-granularity, the proposed 3D DVFS scheme's performance matches that of conventional 2D DVFS at per-core granularity, facilitating a decrease in overheads of voltage regulators without any performance loss.

4.2 Temperature-Aware Adaptive Routing for Dynamically-Throttled 3D Networks-on-Chip

Multiprocessors such as Naga rely on the system interconnect to move data between tiles. Average switching activity per tile (α_t) from (4.1) can thus be broken down into its two constituent parts: activity due to functional operations (such as processing and memory load-stores), and activity due to communication over the interconnect. α_t is given as:

$$\alpha_t = \alpha_p \frac{E_p}{E_t} + \alpha_r \frac{E_r}{E_t} \tag{4.9}$$

where E_p and E_r represent the average energy dissipation of the functional components within the tile and the interconnect router respectively, with corresponding activity rates α_p and α_r. E_t is the average total energy dissipation of the tile. The *network-on-chip (NoC)* interconnect in modern MPSoCs consumes a significant amount of power, and can consequently aggravate thermal imbalances in the system. For instance, interconnect traffic routed close to highly active PEs increases power density in the region, and can thus result in elevated operating temperatures, i.e. a thermal hotspot. If traffic were instead steered away from such critical regions by a temperature-aware routing strategy, the amount of interconnect power dissipated near high-activity nodes would reduce. This would yield more balanced operating temperatures, prevent invocation of the DTM, and accordingly minimize performance losses.

In this section, we propose the *Immediate Neighbourhood Temperature (INT)* adaptive routing algorithm which balances operating temperatures across 3D NoCs by incrementally routing packets along low temperature minimal paths. INT eliminates the need for system-wide temperature awareness, and instead utilizes only local temperature information from adjacent routers to drive output port selection for in-flight packets. INT outperforms state-of-the-art proposals [134–136], yielding shorter communication latencies, lower congestion, and balanced temperature profiles. Our work demonstrates the effectiveness of localized temperature information in driving adaptive routing in 3D ICs.

4.2.1 Background

Following Skadron's early work on temperature aware microarchitectures [83], a number of proposals have addressed the issue of temperature management in MPSoCs. The proposals vary in the parameters that they use to control temperatures, and in their effect on equation (4.1). In proposals such as *ThermalHerd* [114], interconnect traffic is steered away from thermal hotspots along paths with a low thermal correlation (effective thermal conductance g) with the heat source. However, such an approach requires a comprehensive analysis of heat flow paths within the chip at design time, which is especially complicated in the case of 3D ICs due the complex nature of heat flow in multi-die stacks. Furthermore, the storage of correlation data at every node poses a considerable overhead.

Other schemes rely on an indirect method to control temperature. Thermal-aware throttling reduces the activity of routers (α_r) based on the available temperature margin ($\Delta T_{max} - \Delta T$). However, as temperature increases, routers are increasingly throttled reducing not only power dissipation but also throughput. In highly active regions of the interconnect, the decreased throughput results in congestion. *Regional Congestion Awareness (RCA)* [134] provides a means to aggregate and propagate this congestion information across the interconnect over a dedicated monitoring network. At each router, the RCA network aggregates congestion information from the two quadrants surrounding each output port together with local congestion information, and propagates this value to neighbouring routers. Congestion-aware adaptive routing on an interconnect with thermal-aware throttling effectively acts as a temperature-aware routing strategy. This is the principle of *Traffic and Thermal Awareness Routing (TTAR)* [135]. There are two main disadvantages of this approach. Firstly, it assumes that congestion always implies a thermal hotspot. However, congestion can also be a consequence of actual interconnect traffic in a relatively lower temperature region of the chip. In such a case, the path of least congestion need not necessarily form the path of least temperature. Secondly, the aggregation and propagation of congestion information across the network poses an overhead in terms of complexity as well as power.

Another competing scheme, notable for its simplicity, is *Downward Routing* [136] [1]. This scheme minimizes the amount of routing activity occurring in tiers farther

---

[1] [136] considers the heatsink to be situated at the bottom of the die stack, hence the name Downward

from the heatsink and most prone to overheating. Instead, it preferentially routes packets towards higher tiers which dissipate their heat more effectively on account of their proximity to the heatsink. By minimizing α_r in tiers with the highest distance from the heatsink (l_z), Downward Routing decreases temperatures in the stack and thus reduces the magnitude of throttling induced in routers. On the other hand, it results in the under-utilization of routing resources on lower tiers of the die stack, and increases congestion on the tiers close to the heatsink.

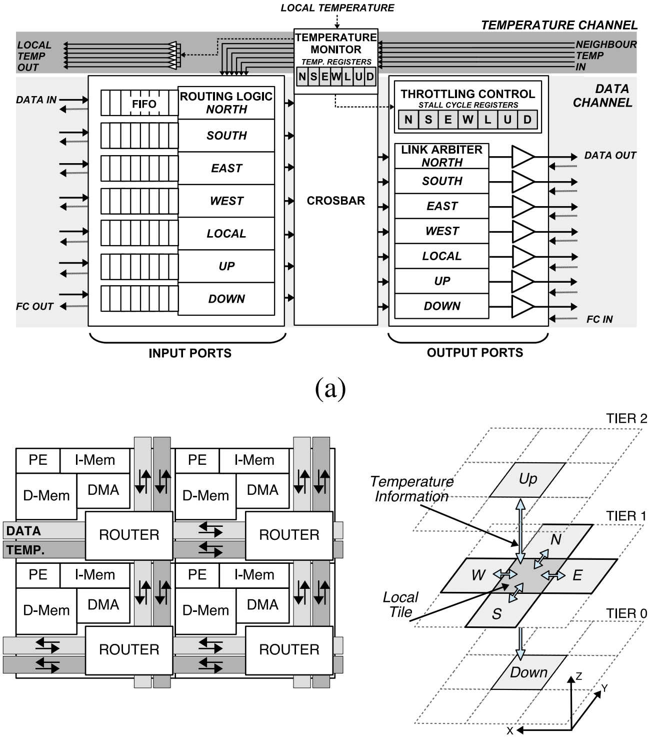
A complex derivative of TTAR is the *Traffic and Thermal Aware Beltway Adaptive Routing (TTABR)* scheme which uses a set of concentric non-minimal routing paths to bypass central regions of congestion and high-temperature. However, the proposal assumes that thermal hotspots occur only in the center of the die, and not along the critical beltway paths on the periphery. Thermal maps shown later in this chapter illustrate this to be an unrealistic assumption.

A number of proposals use unique techniques to detect and adapt traffic flows at runtime. For instance, [137] uses a dynamic programming approach to determine optimal traffic paths at runtime, while [138] proposes an artificial neural network to predict the location of thermal hotspots based on runtime monitoring. On the other hand, schemes such as [139] implement a limited proactive strategy that reverts to conventional deterministic routing once thermal hotspots have formed in the system.

4.2.2 Immediate Neighbourhood Temperature (INT) Adaptive Routing

Temperature, unlike congestion information, does not need to be propagated across the network in order to support thermal-aware routing. The physical nature of heat transfer results in thermal hotspots influencing the temperature of tiles in their immediate vicinity. This observation is even more significant in 3D IC where the magnitude of thermal conduction between the thin stacked dies results in hotspots spread across multiple tiers. Consequently, the temperature of candidate links in the direction of the thermal hotspot appear higher than others in the direction of cooler regions. This effectively removes the need for an aggregate-propagate type monitoring network, and enables temperature-aware adaptive routing to be implemented based on information available from routers in the immediate vicinity alone. In this work, interconnect activity α_r is controlled using both a thermal-aware throttling mechanism in direct response to local temperature, as well as an adaptive routing strategy that steers interconnect traffic away from regions of high temperature.

Routing. In this dissertation, we consider the heatsink to be situated on top of the stack.



(a)

(b) (c)

Figure 4.7. Illustration of (a) INT router architecture (b) MPSoC array with INT-based network-on-chip with separate temperature and data channels (c) Exchange of temperature information between immediate neighbour routers in a 3D stack.

4.2.2.1 Temperature Monitoring

Temperature monitoring is performed locally through the use of one or more thermal sensors integrated within each tile. Sensors integrate an *analog-to-digital converter (ADC)* to provide digital temperature measurements in every sampling interval. The digital temperature value is stored within the router's *Temperature Monitor*, shown in Figure 4.7(a), and is used to control thermal-aware dynamic throttling in the local router, as well as to provide a basis for adaptive routing at neighbouring routers.

4.2.2.2 Temperature Channel Considerations

Unlike in RCA where congestion information must be aggregated and propagated across the network, INT relies only on temperature information available in the

```
$Address_{local} \leftarrow (x_{local}, y_{local}, z_{local})$
$Address_{dest} \leftarrow (x_{dest}, y_{dest}, z_{dest})$
if $Address_{local} == Address_{dest}$ then
    Selected Port ← Local
else
    Candidate Ports ← Perform initial OE routing
    Fetch Temperatures of Candidates
    Preferred Candidates ← Candidate port with lowest temperature
    if Preferred Candidates >1 then
        if Up is a Preferred Candidate then
            Selected Port ← Up
        else
            Selected Port ← First Preferred Candidate
    else
        Selected Port ← First Preferred Candidate
Route packet onto Selected Port
```

Figure 4.8. INT Routing Algorithm

immediate neighbourhood. This effectively reduces the amount of logic within the monitoring network. INT's temperature monitor only consists of a few registers to store the latest temperature measurements from neighbouring routers, as well as the local temperature. The local temperature is placed on the outgoing temperature channels to the router's immediate neighbours, as shown in Figure 4.7(b) and (c). The area and power overheads imposed by this channel are mainly due to the link drivers. The magnitude of these overheads is determined by the width of the temperature vector, which in turn depends on the resolution of the temperature sensor, and the maximum operating range. However, given the non-aggregating nature of this monitoring network, transitions on the temperature channel occur only once new temperature measurements are available from the sensor (typically every $50\mu s$). The power overheads of the temperature channel are thus minimal as compared to the data channel, and RCA-type networks.

4.2.2.3 Thermal-Aware Dynamic Throttling

Throttling influences the rate at which traffic flows through an interconnect router, thereby influencing activity (α_r) and hence, power dissipation. The crossbar switch and output link drivers account for over 80% of the dynamic power dissipation of the complete router. Our throttling approach therefore focuses on controlling activity within these components, and is implemented within the *link arbiters* of output ports as shown in Figure 4.7(a). When throttling is applied, a varying number of stall cycles is introduced between the polling of successive ports by the round-robin arbiter. The amount of throttling invoked is dependent on temperature, and the range of available levels can be controlled using a set of registers within each router. These registers specify the maximum permissible throttling level, ensuring that throughput on a link never falls below a certain minimum due to throttling. However, this also limits the level of temperature control that the interconnect can exercise.

In addition to controlling α_r, throttling also affects the performance of processing elements in dataflow architectures, where execution of tasks is dependent on the presence of the necessary triggering data within the tile's message passing buffers. Since throttling decreases the throughput of routers, triggering of tasks is delayed, and PE activity (α_p) is thus decreased.

4.2.2.4 Temperature-Aware Adaptive Routing Algorithm

Temperature-aware path selection in INT consists of two steps. In the first step, an initial routing of the packet is performed using the *Odd-Even (OE)* algorithm [140], identifying the candidate output ports that could be used to reach the destination. The algorithm returns a set of candidate output port options corresponding to the available minimal paths to the destination router. The generated set encapsulates the ports through which the waiting packet can be ejected, while respecting OE's turn restrictions that guarantee deadlock freedom within the network. INT's second step consists of identifying the candidate port with the least temperature from this set. This port is referred to as the preferred candidate, and it is used to eject the packet towards its destination. INT's output port selection is therefore thermal-adaptive, steering interconnect traffic in response to temperature, within the set of available minimal paths.The INT routing algorithm is listed in Figure 4.8. For a given quantum of traffic, INT influences the following terms of (4.1):

- α_r/A: INT spreads the utilization of interconnect routers over a larger area by adaptively routing packets over multiple paths based on the available temperature margin (ΔT_{max}-ΔT).

- α_r/α_t: INT reduces interconnect traffic in regions of high activity (α_t) which typically exhibit higher operating temperatures.

- α_r/l_z: When temperatures permit, INT preferentially routes packets intended for tiers closer to the heatsink in the Z-dimension first. When the Z path is

Table 4.5. SYSTEM CONFIGURATION

MULTIPROCESSOR		NETWORK-ON-CHIP	
System Size	4×4×4	*Flit Width*	38b
D-/I-Mem	64KB/16KB	*FIFO Depth*	16 flit
DMA Buffers	8KB	*Packet Size*	4B/32B/64B
PE	32b RISC	*Port Count*	7
PHYSICAL			
Tech Node	90nm	*Frequency*	500MHz
Tile Size	2mm×1.4mm	*Heat Transfer Co-eff.*	100 $\mathrm{Wm^{-2}K^{-1}}$
Temp. Range	300-370K	*Trigger Temp.*	332K
Sensor Accuracy	0.5K	*Sampling Interval*	50 μs

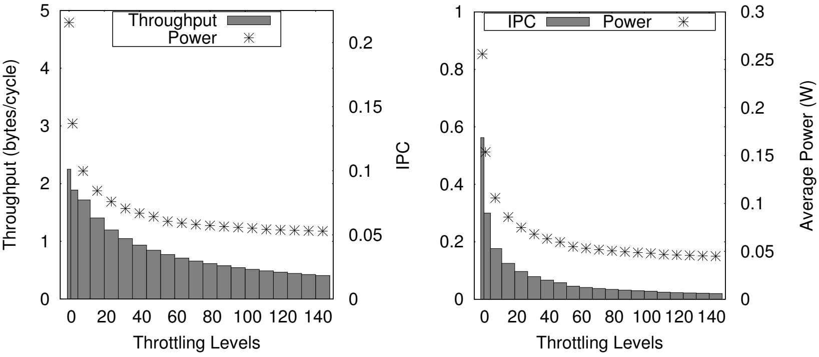
blocked by a thermal hotspot, packets are adaptively routed towards the coolest direction permitted by their minimal path.

4.2.3 Evaluation

INT is evaluated using the *Ctherm* cycle-accurate thermal-functional co-simulation framework with a *SystemC* model of the *NagaM* multiprocessor. The evaluation consists of two parts - first, the characterization of thermal-aware throttling mechanism and its effect on power and performance, and second, the evaluation of the INT routing algorithm under varying traffic conditions, and number of thermal hotspots. The system configuration is listed in Table 4.5. The test platform consists of 64 tiles arrayed over 4 tiers with 16 nodes each. Tiles incorporate a processing element, memories, a *DMA* controller implementing the *Pronto* message-passing system, and a NoC router, as illustrated earlier in Figure 4.7(b).

4.2.3.1 Characterization of Throttling

The throttling mechanism is characterized using a single router with an emulated temperature input. Figure 4.9 illustrates the relationship between average power dissipation, throughput and throttling levels. Each level on the x-axis corresponds to a $0.5K$ increment over the triggering temperature. Throttling levels thus increase with rising temperatures, enabling a steeper decrease in power dissipation to arrest temperature rise. This however has a significant impact on throughput. This observation motivates the use of a temperature-aware routing strategy that can prevent such temperature peaks from forming. To demonstrate the influence of α_r on α_p in dataflow and memory-bound systems, PEs in the characterization setup are configured to execute a fixed set of simple integer operations for every data word that arrives into the DMA's message passing buffer. When the buffer is empty, execution is stalled.



(a) (b)

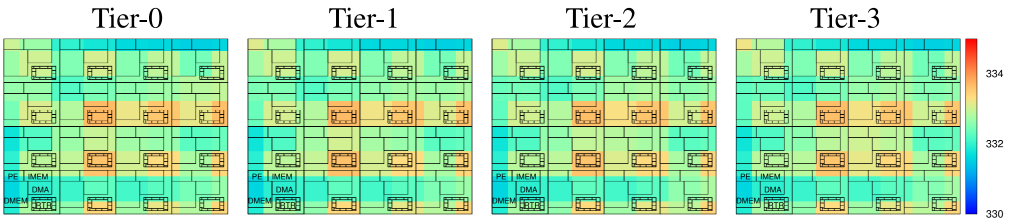
Figure 4.9. Influence of throttling on router power and (a) Throughput (b) Effective *instructions per cycle (IPC)* of PE

Figure 4.9(b) reports the average *instructions per cycle (IPC)* and power dissipation within a $50K$ cycle window corresponding to each router throttling level. This result indicates that in MPSoCs with significant inter-tile communication, the interconnect can be used to regulate α_p.

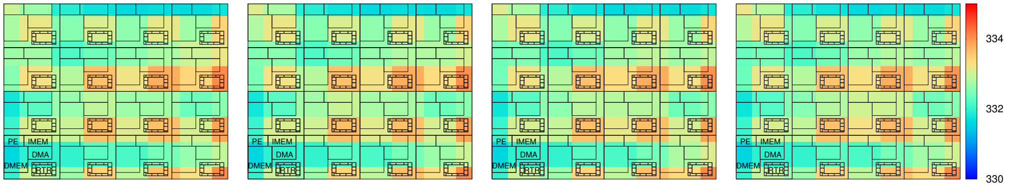
4.2.3.2 INT Evaluation

The evaluation of INT utilizes three synthetic traffic patterns: *hotspot (10%)*, *uniform random* and *bit transpose*. Packets are injected into the network by synthetic traffic generators that target destinations according to the probability density function of the traffic pattern.

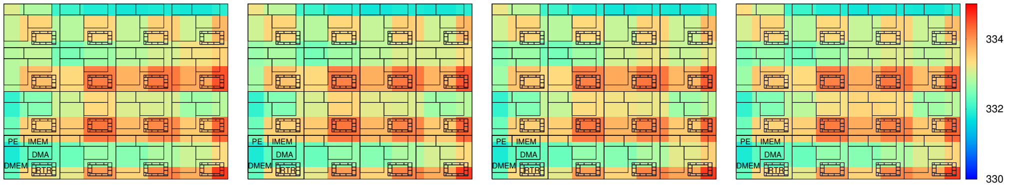
In order to test the ability of the various routing algorithms in balancing temperatures, the evaluation also uses a variable number of thermal hotspots placed at random locations within the system. These hotspots are obtained from the execution of a *load-compute-store* loop on tile PEs, resulting in a constant power dissipation that is interconnect-independent. This behaviour is characteristic of long running compute-bound workloads which are unaffected by interconnect performance once their triggering data has arrived at the tile. Although the routing strategies are evaluated in the presence of 4 and 8 such thermal hotspots inside the 3D mesh, since the general performance trends of both are similar, we only present the results from the evaluation with 8 thermal hotspots for brevity's sake. INT is compared with 3 other routing strategies: *Downward routing* [136], *TTAR* [135] and an adapted version of TTAR that additionally incorporates a dedicated temperature monitoring RCA network *(TTAR+)*. The INT and TTAR cases use an 8-bit monitoring network for



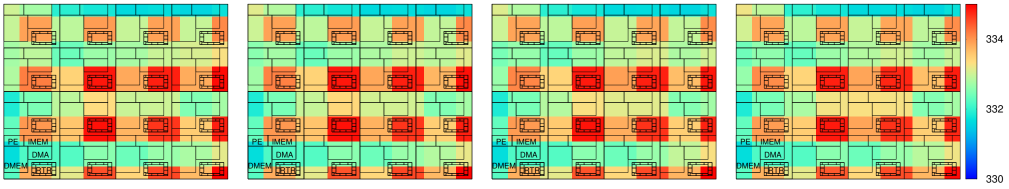
INT (proposed)



Downward Routing [136]



TTAR [135]



TTAR+

Figure 4.10. Temperature maps from *uniform random* traffic with 8 thermal hotspots with different routing strategies. Temperatures are measured in *Kelvin (K)*.

temperature and congestion, respectively. For TTAR+, temperature and congestion information are carried over independent 5-bit and 3-bit networks, respectively. In order to compensate for the loss in temperature resolution due to the decreased vector width, the monitoring temperature range was decreased to $330K$-$345K$. The congestion resolution although decreased is still sufficient to discriminate between paths.

(i) Thermal Performance Figures 4.10(a)-(d) illustrate temperature maps of tiers 0 and 3 of the die-stack with each routing algorithm following $750K$ simulation cycles at the peak injection rate with *uniform random* traffic and a total of 8 thermal hotspots. INT is observed to provide the most balanced temperature profiles, and the lowest peak temperatures amongst all the tested schemes. Downward routing provides a similar balance, however, with increased latency and congestion as observed in Figure 4.11. In the case of TTAR, the temperature imbalance occurs as a consequence of network traffic being routed through high temperature regions with low congestion. The figures also illustrate the high degree of thermal coupling between stacked dies.

The efficacy of the interconnect in balancing temperature differences in the system requires the presence of sufficient traffic. At low injection rates, the influence of network traffic on temperatures is relatively small. As a consequence, despite its temperature awareness, INT yields peak temperature differences identical to the competing schemes, evidenced in Figure 4.11. With increasing network traffic however, the activity and power dissipation of the interconnect assumes increasing significance, and the effects of temperature-aware routing become evident.

(ii) Latency and Congestion Figure 4.11 also illustrates the average packet latency and network congestion resulting from the use of each routing strategy. In a network with temperature-dependent throttling, packet latencies are influenced by operating temperatures as well as network congestion. Congestion aware routing strategies like TTAR offer significantly low latencies on account of their avoiding high traffic regions. However, this often results in the routing of packets close to interconnect-independent thermal hotspots. Consequently, operating temperatures rise, and as throttling is invoked, packet latencies increase. Downward routing on the other hand prevents the formation of high temperature regions by routing traffic towards higher tiers. This has the effect of increasing network congestion on the cooler tiers, and drastically increasing packet delivery latencies. INT's routing of packets based on operating temperatures results in a decreased chance of encountering a heavily throttled router. This ensures that packets remain on higher throughput paths, reducing the congestion caused due to slow moving traffic in the network. The latency benefits obtained as a result of INT's temperature-adaptive routing are observed in Figure 4.12, which reports the latency distribution across network nodes at the peak injection rate. In the case of TTAR+, the power overheads incurred due to the propagation of monitoring information across the system are so considerable that they result in elevated operating temperatures, yielding higher latencies.

Figure 4.12 also illustrates the preferential use of certain tiers by each routing strategy. For instance, Downward routing results in the preferential utilization of higher tiers located closer to the heatsink. Consequently, congestion in these tiers is

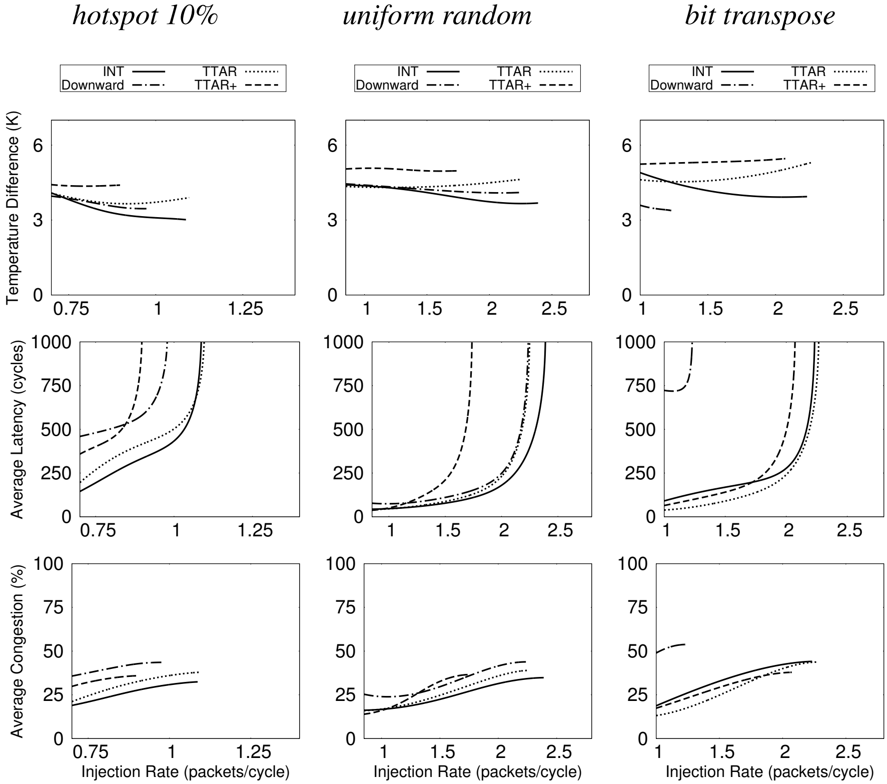


Figure 4.11. Peak temperature difference, average packet delay and average congestion with 8 thermal hotspots for different traffic patterns. In the figures, the data lines end at different injection rates corresponding with the saturation throughput of the network.

aggravated. TTAR on the other hand routes traffic towards regions of least congestion. However, its inability to directly sense temperature results in the over utilization of deeper tiers, and in the throttling of routers. Latencies are thus observed to worsen with increasing distance from the heatsink. Similar behaviour is noted for TTAR+ as well. In contrast, INT thermal-aware routing function avoids regions of high temperature, and reduces the magnitude of throttling invoked in the network. Latencies are thus lower even in the deeper tiers. The central tiers of the stack exhibit a higher latency due to the volume of passing traffic, however, this is observed even for other routing strategies.

(iii) Overheads Table 4.6 lists the area and power overheads imposed by each routing strategy. Note that the area overheads listed for RCA derivatives such as

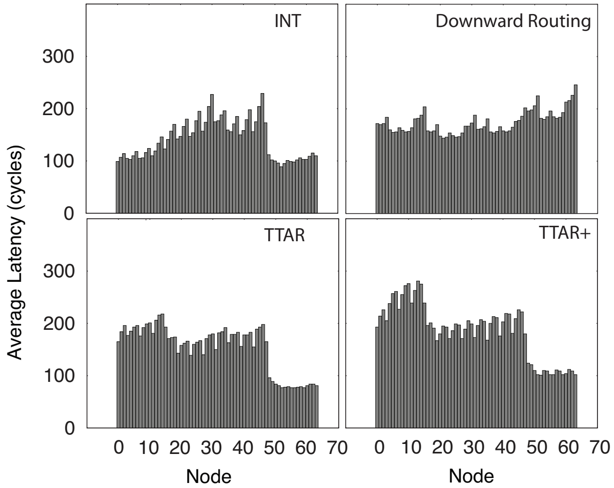


Figure 4.12. Latency distribution among network nodes for *uniform random* traffic at peak injection rate.

Table 4.6. Overheads

	WIDTH	AREA	POWER
INT	8-bit	0.9%	< 0.1%
TTAR	8-bit	10.7%	25%
TTAR+	8-bit	10.7%	61%
DOWNWARD	X	0%	0%

TTAR and TTAR+ does not include the computational resources required to perform the weighted summation. In the present work, these are considered to be integrated within the area of the crossbar switch. The aggregate-propagate network for these cases is approximated as an 8-bit point-to-point network with a single-entry input buffer per port.

4.2.4 Conclusions

In this section, we presented the *Immediate Neighbourhood Temperature (INT)* adaptive routing algorithm for dynamically-throttled networks-on-chip. Unlike state-of-the-art proposals which rely on complex aggregate-propagate networks, INT bases adaptive routing decisions solely on temperature information from neighbouring routers, thus minimizing monitoring overheads. Interconnect traffic is steered away from regions of high temperature, yielding balanced thermal profiles, with up to 25%

lower thermal gradients. Furthermore, as a consequence of lower operating temperatures and adaptive routing, communication latencies are improved, and network congestion decreased by up to 50% even in the presence of system thermal hotspots. In addition to the adaptive routing strategy, we illustrated the inter-dependence of PE and interconnect activity, and consequently demonstrated how interconnect throttling could be used to control power dissipation of PEs in dataflow multiprocessors.

Summary

While frameworks such as Ctherm enable the realization of a thermally efficient system, actual temperature profiles at runtime are a consequence of the workload that is executed on the multiprocessor. For this reason, *Dynamic Thermal Management (DTM)* schemes are used to control power dissipation through a range of schemes in order to limit temperatures to within a safe operating range. *Dynamic Voltage Frequency Scaling (DVFS)* based DTM strategies provide fine-grained control of power dissipation, allowing performance levels to be varied according to the magnitude of the thermal emergency. However, due to non-uniformities in temperature margins in 3D ICs, conventional 2D DVFS strategies result in frequent voltage and frequency level transitions, and degraded performance. In this chapter, we proposed a novel 3D DVFS strategy, that takes the thermal characteristics of 3D ICs into account, and maximizes utilization of the non-uniform temperature margins within die stacks at runtime. This yields an improvement in aggregate system performance. Furthermore, in order to control temperature gradients occurring as a consequence of activity imbalances, we proposed the INT adaptive routing algorithm that steers interconnect traffic away from regions of high temperature. Due to the spread in interconnect traffic, peak temperatures are reduced, and the magnitude of thermal gradients decreased. Finally, due to the reduction in operating temperatures, interconnect latencies are improved.

Associated Publications

1. S.S. Kumar, A. Aggarwal, R. Jagtap, A. Zjajo, R. van Leuken, ”System Level Methodology for Interconnect Aware and Temperature Constrained Power Management of 3-D MP-SOCs” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 7, pp. 1606-1619, July 2014

2. S.S. Kumar, A. Zjajo, R. van Leuken, ”Immediate Neighbourhood Temperature Adaptive Routing for Dynamically-Throttled 3D Networks-on-Chip” *IEEE*

Transactions on Circuits and Systems II (TCAS-II), *in press*

3. A. Aggarwal, S.S. Kumar, A. Zjajo, R. van Leuken, "Temperature constrained power management scheme for 3D MPSoC," *Proceedings of the IEEE Workshop on Signal and Power Integrity (SPI)*, pp.7-10, 2012

5

Conclusions

This dissertation investigates the efficient design, and dependable operation of high performance, thermal-aware chip multiprocessors. The realization of such machines is impeded by three challenges – efficiency of the architecture, thermal constraints, and runtime temperature management. Essentially, these challenges can be translated into the primary objectives of this dissertation - to realize an efficient multiprocessor, to enable its thermal-aware design, and to empower it with the ability to adapt to changing thermal conditions at runtime. This final chapter reviews the most significant conclusions that were derived in the course of achieving these objectives, and outlines how these contributions strengthen and advance the state of the art in thermal-aware multiprocessor design. This is done in the context of the research questions that were posed earlier in Chapter 1.

How can the performance and efficiency of on-chip memory operations in multiprocessors be improved?

In large multiprocessors, the management of data has a key role to play in the performance and efficiency of on-chip memory operations. At the scale required to achieve *trillion operations per second (TOPS)* performance levels, it is important to pay close attention to what data is stored in on-chip memories, where it is stored in the system, and for how long. These are essential factors that must be given due consideration when designing memory hierarchies for multiprocessors of scale. The importance of these factors was demonstrated in Chapter 2, across all three proposals - *Pronto*, *Persistence Selective Caching (PSC)*, and *CacheBalancer*.

Conventional wisdom in memory hierarchy design has focused on reducing the occurrence of cache misses, and the magnitude of their associated penalties. This dissertation showed how the converse approach, i.e. focussing on cache hits instead of misses, can yield massive performance and energy benefits. For an application with sufficient data reuse, cache hits represent the common case, and it is essential for the memory hierarchy design to leverage this fact in order to improve performance. Our proposed *Persistence Selective Caching (PSC)* scheme, demonstrated the benefits of accelerating the common case, and highlighted the significant amount of reuse present in a number of standard workloads. For certain applications from the *Mibench* benchmark suite, *average memory access time (AMAT)* was decreased by upto 59% by accelerating references to as few as 7 cache lines. PSC also demonstrated a secondary benefit of using cache assists in the memory hierarchy – the energy per memory access was decreased by upto 75%. These are significant improvements in performance and efficiency, achieved with relatively small hardware overheads.

With the growing size of multiprocessor systems, the physical location of the data in relation to its consumer becomes a matter of concern. The latency and energy costs associated with accessing data located in distant on-chip cache memory banks are of considerable magnitude. When coupled with the additional penalties incurred due to the unbalanced utilization of system resources, these costs diminish the execution performance of the multiprocessor. This observation highlights the need to apply knowledge of the system's physical organization and runtime state when managing system resources.

Finally, in order to make memory operations more predictable, it is important to look into the interaction between communication flows in the multiprocessor interconnect. It is observed that the head and tail sections of data flows within task graphs are more susceptible to jitter (variation in arrival time) in the presence of network contention, than intermediate sections. These sensitive sections tend to delay the triggering of tasks, producing latency variations that cannot be hidden by successive stages of the graph. The jitter resulting from contention in intermediate sections on the other hand, is far less pronounced. This indicates that contention sensitivity varies throughout the task graph, and that the magnitude of jitter induced in the output depends primarily on the amount of contention experienced by the head and tail edges. To ensure predictability of output data, it crucial for interconnect contention to be reduced around such sensitive sections.

How do the physical design parameters in Nagata's equation affect the thermal behavior of 3D Integrated Circuits?

Due to the relatively nascent state of 3D integration, the thermal behaviour of die stacks is not well understood. The most significant contribution of this dissertation lies in its advancement of the state of the art in this very regard. Chapter 3 presented a detailed characterization of the thermal design space of 3D ICs, and examined the specific influence of individual design parameters on the operating temperatures and thermal gradients in die stacks.

The characterization revealed the dominant role of effective thermal conductivity (κ_{eff}) in determining operating temperatures in die stacks. Dependent on the design of the *Through Silicon Vias (TSV)* based vertical interconnect, the κ_{eff} parameter affects the magnitude of heat flowing across and through dies within the stack, essentially changing the shape of operating temperature profiles. Thermal hotspots are found to be completely mitigated when κ_{eff} is raised sufficiently. In general, the range of variation observed in temperatures with changing κ_{eff} indicates the need to take the parameter into consideration within simulation models during evaluation of thermal behaviour. The parameter also holds repercussions for the runtime management of the system by impacting the placement of on-chip temperature sensors. For instance, with increasing conductivity, the size of the window within which sensors can accurately track the temperature of a hotspot center also increases. Consequently, the design rules for the placement of temperature sensors can be relaxed, allowing them to be placed farther away from the hotspot location, without losing accuracy. The placement strategy is therefore dependent on κ_{eff}, and in turn, on vertical interconnect design.

The vertical interconnect is in itself a structure with a complex design space. While evaluating the performance of this structure, it is prudent to note that seemingly insignificant differences in thermal performance can have a considerable impact on effective system performance. This was highlighted in Chapter 3, where a meagre 0.5K of additional temperature margin afforded by the Shielded TSV topology yielded a 11% improvement in the execution performance over the competing Isolated topology. This observation underlines the importance of including κ_{eff} in thermal evaluations.

From a system design perspective, power density was observed to have a similarly critical influence on operating temperature. Unlike κ_{eff}, power density is a result of floorplanning choices within the 3D stack. While the integration of power dissipating elements within a die stack inevitably raises operating temperature, an uneven spread in the dissipated power results in the formation of thermal hotspots which are detrimental to the system's reliability. The key to mitigating hotspot formation lies in adapting system floorplans to reduce power density.

Apart from the influence of these parameters, the thermal characterization also confirmed the non-uniform nature of temperature margins within die stacks. Power dissipating elements situated on deep tiers of the stack experience a decreased operating head room as compared to their counterparts on higher tiers as a result of increased thermal resistance. System performance is thus non-uniform in 3D ICs, and estimating it accurately at design time requires the inclusion of runtime thermal effects in the design flow.

How can the knowledge of thermal behaviour be effectively leveraged in the design of 3D stacked multiprocessors?

The evaluation of effective thermal performance of architectural and system design choices is essential in realizing thermal-efficient 3D stacked multiprocessors. In particular, the ability to perform closed-loop co-simulation of thermal and functional behaviour are fundamental to realistically modeling the influence of runtime physical effects, and the consequences of their mitigation. This is especially important since the variability of runtime operating conditions necessitates the constant adaptation of system parameters in order to maximize performance within physical constraints. The ability to model this is predicated on the existence of a closed-loop simulation platform such as *Ctherm*, presented in Chapter 3.

The accuracy of the modeled behaviour is linked to the granularity of spatial and temporal power models. Proposals within the state of the art exhibit serious shortcomings in this regard. Most commonly, proposals average power dissipation of components over multiple cycles in order reduce simulation time. As a result of the decreased fidelity of power data, peaks that would normally result in hotspots are not accurately modelled by the existing proposals. A more serious disadvantage lies in their abstraction of component internals when mapping dissipated power to physical locations in the floorplan. Such abstraction results in significantly lower accuracy, and yields highly optimistic estimates of thermal behaviour since it ignores the influence of power density. For instance, in the evaluation of the thermal behaviour of a cache memory, inclusion of fine-grained component internals revealed the existence of thermal hotspots at locations where state of the art approaches indicated low operating temperatures. This thermal behaviour was found to drastically vary as the internal organization was changed, primarily as a consequence of changing power density. It is therefore imperative that the internal organization of components be taken into account when evaluating thermal behaviour.

Furthermore, any architectural modification that impacts the latency of operations, effectively changes the power dissipation characteristics of the component. While the modification that yielded this latency benefit could be small in terms of hardware cost, the thermal impact of the decreased latency (thus higher power dissipation) could be significant. It is for this reason that cycle-accurate modeling of component behaviour is essential when performing thermal simulation.

It must be noted that design effort for creating these fine-grained component models manually is prohibitively high, and unrealistic in the context of production *design space exploration (DSE)* flows. However, *Ctherm* demonstrated how such a flow can be realized practically with automated model generators to simplify the evaluation of specific candidate design options.

How can the architecture and operating parameters be efficiently adapted at runtime to mitigate the severity of thermal issues, and improve execution performance?

Despite the use of thermal-aware design practices, the variability of runtime operating conditions necessitates the use of *Dynamic Thermal Management (DTM)* to control temperature. DTM strategies arrest thermal emergencies by limiting power dissipation in regions of high temperature. However, their inability to adapt to the unique thermal characteristics of die stacks limits the efficacy of conventional DTM strategies. For instance, conventional *Dynamic Voltage Frequency Scaling (DVFS)*, when applied to a 3D multiprocessor, results in the preferential utilization of *processing elements (PE)* situated closer to the heatsink over those on deeper tiers. This is a consequence of the non-uniform thermal margins prevalent in die stacks. To realize an effective DTM strategy for 3D systems, it is essential for these non-uniformities to be taken into account within the decision making algorithm. This enables system performance to be maximized within the thermal margins available at individual PEs, while reducing the chances of a DTM action pushing any part of the system into a thermal emergency. It may be observed that although such an approach results in PEs operating below their full performance level, overall system performance is effectively improved. This was demonstrated in Chapter 4, where a 19.55% improvement in execution performance was obtained by incorporating knowledge of non-uniform thermal margins into the design of the DTM strategy for a 3D multiprocessor.

Chapter 4 also proposed the *Immediate Neighbourhood Temperature (INT)* adaptive routing strategy to balance thermal profiles in 3D *networks-on-chip (NoC)*. Conventional thermal-aware NoC architectures rely on complex aggregate-propagate networks to provide a system-wide view of temperatures at every node. However, the overheads of relaying global temperature information are so high that they actually increase operating temperatures, and consequently degrade system performance when used in conjunction with dynamic throttling. On the other hand, adaptive routing based only on temperature information available in the immediate vicinity of the router allows packets to be incrementally routed along low temperature paths without the overhead of an expensive monitoring network. This approach was demonstrated to reduce thermal gradients 25% better than conventional strategies, with negligible overheads. Furthermore, this strategy has the effect of spreading interconnect activity over a large area, yielding operating temperature reductions and thus upto 50% lower network congestion in the presence of thermal-aware dynamic throttling.

Finally, it was discovered that although congestion can be used within thermal-aware dynamically throttled NoCs as an indicator of temperature, basing adaptive routing decisions on it can erroneously result in high traffic regions being mistaken for thermal hotspots. For this reason, it is prudent to drive the adaptive routing function using actual temperature measurements.

In summary, this dissertation provides insight into the architectural features, design methodologies and runtime strategies required to effectively realize high-performance thermal-aware multiprocessors. It furthers the state of the art of 3D IC design through its characterization of thermal behaviour of die stacks, and its examination of the influence of individual design parameters on operating temperatures. Importantly, it demonstrates a practical thermal-aware DSE framework that enables the co-simulation of both thermal and functional behaviour of system design options, and highlights the importance of considering physical effects such as temperature early in the architecture and system design flows. Such a holistic approach is essential in ensuring dependability and harnessing multiprocessor performance in the dark silicon era. It is hoped that this framework will serve as a template for thermal-aware *Electronic Design Automation (EDA)* tooling of the future, and provide a sense of the critical issues that designers must actively address in order to successfully realize dependable 3D systems.

Additional Contributions

In addition to the scientific contributions outlined above, this dissertation also generated a range of software tooling. These are summarized below:

- *Ctherm*: The framework described in Chapter 3 was implemented using a *Python* front-end, *SystemC*-based exploration and simulation engines, and an adapted version of the 3D ICE thermal simulator. The outputs are processed using a combination of *gnuplot* and *ffmpeg* to enable still and video-based visualizations of thermal behaviour.
- *TMFab*: This is a cycle-accurate simulation platform that models a 3D stacked chip multiprocessor incorporating the transactional memory concurrency control paradigm. *TMFab* was developed alongside the *SoCLiB* based platform

used within *Ctherm*. The platform also supports detailed power tracing and debugging of PEs. Details of the TMFab platform are available in [141–143].

- *Simulation models*: Cycle-accurate *SystemC* simulation models were developed for all the components introduced in this dissertation. These models are integrated within the *SoCLiB* component library and contain the necessary activity monitors to implement power modelling. *Pronto* is implemented as a *VHDL*-based simulation model.

- *Software Libraries*: Extensive software libraries were created to support the *Naga* multiprocessor. These include pthread-like libraries to manage execution of tasks, control their mapping, perform file and terminal I/O, and implement inter-task communication. Furthermore, additional libraries were created to implement *CacheBalancer*, and functions to control the thermal simulator from the virtual platform. These libraries also contain architecture specific code written in *Microblaze* assembly. Examples include the bootstrapper, exception handler and interrupt service routine.

At the time of writing, plans exist to release a number of these tools and simulation models into the public domain.

Future Work

From our experiences with building the Naga multiprocessor and its associated tooling, we find a number of topics worthy of future exploration. These are listed below:

Naga Multiprocessor Architecture

- The integration of the NagaM and NagaS into a single homogeneous multiprocessor forms the next step in Naga's evolution. Critical challenges in this regard pertain to the integration of the message-passing and shared-memory address spaces into a single unit, and prevention of contention between workloads of varying criticality.

- Reliability aspects such as device degradation due to aging, device failure and dynamic faults are important practical issues that should also be addressed.

- In this dissertation, we evaluated the efficacy of the PSC scheme in reducing AMAT and energy per access. However, the persistence thresholds were determined through an exploratory simulation. In order to maximize applicability, it is important to investigate simpler methods to determine the threshold. We envision two possibilities in this regard - a compiler-based technique, and a feedback-based control loop. In the first method, thresholds are determined based on static analysis of the application program's memory access patterns. This would enable thresholds to be embedded within the binary, and facilitate configuration of the assist at runtime. In the second method, thresholds are determined based on cache contention and memory access patterns at runtime. Despite its complexity, this method is promising as it provides a means to alleviate cache contention due to thrashing in multi-threaded environments.

- The CacheBalancer concept evolved rather late in Naga's development, and consequently, only a limited evaluation could be performed. Further development requires a detailed characterization of the impact of α, β, θ_{hot} and θ_{cold} parameters on temperature, and execution performance.

Thermal Modelling

- The Ctherm framework presently relies on the 3DICE thermal model, mainly due to its well structured code and excellent documentation. However, future efforts must focus on fine-grained thermal models [75] that account for heat spreading at the device level and include factors such as thermal diffusivity (variation in k with temperature). The latter acts as a source of inaccuracy at the higher end of the operating temperature range. In addition to accuracy, it is also important to consider parallelizing the model to reduce execution time.

Thermal Characterization of Die Stacks

- In this dissertation, we considered only the steady state behaviour of die stacks. However, it is also important to examine transient behaviour in relation to vertical interconnect properties and heatsink dimensions. Furthermore, although TSVs are considered critical in 3D ICs, a number of recent proposals replace them with inductive links [144]. These reduce vertical thermal conductance, and can potentially alter the heat flow characteristics that we established in this dissertation. It is prudent for other types of vertical interconnect structures to also be included in the characterization of 3D ICs.

- The TSV topology exploration in Chapter 3 considered only electrical performance, KOZ requirements and thermal behaviour of candidate topologies. From recent interactions with industry representatives, we are led to believe that TSV yield is also significantly impacted by placement topology. Consequently, the characterization framework could be extended with a set of *design rules* for TSV placement that take yield into account.

Ctherm Framework

- The current version of Ctherm only includes basic algorithms for system-level floorplanning multiprocessors. Future versions could include a more sophisticated floorplanner [101][102], and automated placement of temperature sensors. Based on this placement, a calibration plan for sensors can be generated by the floorplanner.
- While Ctherm supports the integration of custom power models, this can be further enhanced by developing a methodology for generating ALE estimators for non-standard components. This effort requires the development of a *pseudo-synthesis* methodology capable of determining an approximate gate level netlist, and estimating its area, latency and energy dissipation.
- Presently, the co-simulation setup ignores the power dissipation of temperature sensors which, however, can be significant at high accuracies ($\sim$0.1K) [105]. Aging and temperature-related reliability effects can also be integrated into the setup to better evaluate the lifetime of systems, and the reliability impact of design decisions.

Temperature and Power Management

- The influence of scheduling and task mapping strategies on operating temperatures, and techniques for mitigating thermal emergencies through such strategies must be prioritized.
- The decision logic in the 3D DVFS scheme proposed in Chapter 4 relies on a weighted equation to determine the candidate PE for V-F scaling. The weights are presently determined through an exploratory simulation. Their determination should, however, be integrated into the characterization flow. Furthermore, the proposed scheme should also be evaluated with stacked voltage islands.

- The design of the power delivery network is critical in the context of many-core multiprocessors like Naga. In particular, the complexity of power distribution, power dissipation of voltage regulators and ring oscillators and their integration into the system as part of an active interposer or dedicated die are topics that are interesting to explore in the next phase of Naga's evolution.

Acronyms

BOPS	Billion operations per second
TDP	Thermal design power
CMP	Chip multiprocessor
PE	Processing element
I/O	Input/Output
3D IC	Three-dimensional integrated circuit
TSV	Through silicon via
RAM	Random access memory
TOPS	Trillion operations per second
DTM	Dynamic thermal management
AMAT	Average memory access time
L1D	Level 1 data cache memory
NoC	Network-on-chip
PSC	Persistence selective caching
VLIW	Very long instruction word
ILP	Instruction level parallelism
MPB	Message passing buffer

QoS	Quality of service
GT	Guaranteed throughput
MPI	*Message Passing Interface standard*
FPGA	Field programmable gate array
API	Application programming interface
DMA	Direct memory access
ATT	Address translation table
FIFO	First in first out
HDL	Hardware description language
FIR	Finite impulse response
IDCT	Inverse discrete cosine transform
TI	Traffic injector
VC	Victim cache
SVC	Selective victim cache
NUCA	Non-uniform cache access
RC	Resistance-capacitance
KOZ	Keep out zone
DSE	Design space exploration
tDSE	Thermal-aware design space exploration
MPSoC	Multiprocessor system-on-chip
ALE	Area-latency-energy
SRAM	Static Random Access Memory
CPI	Cycles per instruction

ADC	Analog-to-digital converter
TDC	Time-to-digital converter
V-F	Voltage-frequency
DVFS	Dynamic voltage frequency scaling
INT	Immediate neighbourhood temperature
PM	Power manager
DVS	Dynamic voltage scaling
DFS	Dynamic frequency scaling
GALS	Globally asynchronous locally synchronous
RCA	Regional congestion awareness
TTAR	Traffic and thermal awareness routing
TTABR	Traffic and thermal aware beltway routing
OE	Odd-even routing
IPC	Instructions per cycle

Notation

t_{task}	Total execution time of a task
$t_{instructions}$	Time spent in execution of arithmetic, logic and control instructions
t_{memory}	Time spent in execution of memory load-stores
α	Activity rate
N_G	Number of gates
E	Energy dissipation per gate
t_{pd}	Clock period
g	Thermal conductance
ΔT	Temperature difference between IC and ambience
$t_{computation}$	Time spent in executing actual computation
t_{mp}	Time spent in executing message-passing library functions
$t_{transfer}$	Aggregate latency for transfer
t_{fc}	Overhead incurred for synchronizing and implementing flow control between communicating tasks
t_{AMA}	Average memory access time
μ_{hit}	Fraction of all memory references that hit in the cache
μ_{miss}	Fraction of all memory references that miss in the cache
M_A	Total number of memory accesses

t_{hit}	Hit latency
t_{miss}	Miss latency (also referred to as miss penalty)
R_{min}	Minimum number of references required for a line to be regarded as reusable
W	Reference window size
θ_{hot}	Access rate threshold for a cache bank to be considered as hot
θ_{cold}	Access rate threshold for a cache bank to be considered as cold
ϕ	Hot cache sensitivity
ψ	Access rate of a cache bank
Ψ	Set of access rates of all cache banks within a multiprocessor
β	Mid-range value of Ψ
λ	Allocation spread
$HC(p_i, m_j)$	Hop count between PE p_i and cache bank m_j
$Z(m_i)$	Intensity with which a cache bank m_i is utilized by executing tasks
$z_i(\pi_j, m_i)$	Intensity with which task π_j utilizes cache bank m_i
Π_i	Set of all executing tasks
$S(\pi_i)$	Cache sensitivity of task π_i
$h(x)$	Number of hits to data at stack position x
CI	Communication impact of a task mapping
M	Set of all cache banks in the multiprocessor
c_v	Volumetric heat capacity
$\frac{\delta T}{\delta t}$	Transient temperature response
Q	Generated heat

∇T	Steady state temperature gradient along x, y and z dimensions
κ_{eff}	Effective thermal conductivity
A	Area
$l_{x,y,z}$	Distance from heatsink surfaces along the x,y and z dimensions
A_{tsv}	Effective area of TSV
A_{mat}	Effective area of layer material
κ_{tsv}	Thermal conductivity of TSV material
κ_{mat}	Thermal conductivity of layer material
r_{tsv}	Radius of TSV
h_{die}	Length of silicon die
w_{die}	Width of silicon die
Q/A	Power density
KOZ	Dimensions of Keep Out Zone
S	Inter-TSV spacing
D	TSV diameter
α_t	Average activity rate of components within a tile
N_t	Number of tiles
E_t	Average energy per tile
ΔT_{max}	Maximum temperature margin at zero power dissipation
$\Delta T_{max} - \Delta T$	Actual temperature margin
$P_{dynamic}$	Dynamic power dissipation
V	Operating voltage
f	Frequency

Notation

C_L	Load capacitance
$w_a - w_f$	Weights for voltage-frequency scaling
G	Normalized thermal conductance matrix
δT	Projected temperature change
α_p	Activity rate of processing element
α_r	Activity rate of network-on-chip router
E_p	Energy dissipation of processing element
E_p	Energy dissipation of router

Bibliography

[1] K. van Berkel, “Multi-core for mobile phones,” in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 1260–1265, Mar. 2009.

[2] G. Xu, “Evaluation of a liquid cooling concept for high power processors,” in *Proceedings of the IEEE Semiconductor Thermal Measurement and Management Symposium*, pp. 190–195, Mar. 2007.

[3] S. Borkar, “Thousand core chips: A technology perspective,” in *Proceedings of the Design Automation Conference*, pp. 746–749, Jun. 2007.

[4] M. Butts, “Synchronization through communication in a massively parallel processor array,” *IEEE Micro*, vol. 27, pp. 32–40, Sept. 2007.

[5] B. Hutchings, B. Nelson, S. West, and R. Curtis, “Comparing fine-grained performance on the ambric mppa against an fpga,” in *Proceedings of the International Conference on Field Programmable Logic and Applications*, pp. 174–179, Sept. 2009.

[6] M. Haselman and et al., “Fpga vs. mppa for positron emission tomography pulse processing,” in *Proceedings of the International Conference on Field Programmable Technology*, pp. 231 – 238, Dec. 2009.

[7] M. Butts, B. Budlong, P. Wasson, and E. White, “Reconfigurable work farms on a massively parallel processor array,” in *Proceedings of the International Symposium on Field-Programmable Custom Computing Machines*, pp. 206 – 215, Apr. 2008.

[8] G. Panesar, D. Towner, A. Duller, A. Gray, and W. Robbins, “Deterministic parallel processing,” *International Journal of Parallel Programming*, vol. 34, pp. 323–341, Mar. 2006.

[9] G. Panesar, “One die. 300 processors. real systems.,” in *Presentation at Multicore Expo*, Mar. 2006.

[10] A. Agarwal, “The tile processor: A 64-core multicore for embedded processing,” in *Proceedings of the IEEE High Performance Extreme Computing Workshop*, Sept. 2007.

[11] S. Bell and et al., “Tile64 - processor: A 64-core soc with mesh interconnect,” in *Proceedings of the IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, pp. 588 – 598, Feb. 2008.

[12] A. Agarwal, L. Bao, and et al., “Tile processor: Embedded multicore for networking and multimedia,” in *Proceedings of the HotChips Symposium*, Aug. 2007.

[13] I. Corporation, “Seaforth 40c18 product brief,”

[14] F. Diotalevi, A. Fijany, M. Montvelishsky, and J.-G. Fontaine, “Very low power parallel implementation of stereo vision algorithm on a solar cell powered mimd many core architecture,” in *Proceedings of the IEEE Aerospace Conference*, pp. 1 – 13, Mar. 2011.

[15] S. Kyo and S. Okazaki, “Imapcar: A 100 gops in-vehicle vision processor based on 128 ring connected four-way vliw processing elements,” *Journal of Signal Processing Systems*, vol. 62, pp. 5–16, 2011.

[16] S. Kyo and S. Okazaki, “In-vehicle vision processors for driver assistance systems,” in *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 383 – 388, Mar. 2008.

[17] S. Borkar, “Designing reliable systems from unreliable components: the challenges of transistor variability and degradation,” *IEEE Micro*, vol. 25, pp. 10 – 16, nov.-dec. 2005.

[18] J. W. McPherson, “Reliability challenges for 45nm and beyond,” in *Proceedings of the Design Automation Conference*, pp. 176–181, 2006.

[19] T. Zhang, C. Xu, K. Chen, G. Sun, and Y. Xie, “3d-swift: A high-performance 3d-stacked wide io dram,” in *Proceedings of the Great Lakes Symposium on VLSI*, pp. 51–56, 2014.

[20] L. Madden, “Heterogeneous 3-d stacking, can we have the best of both (technology) worlds?,” in *Proceedings of the ACM International Symposium on Physical Design*, pp. 1–2, 2013.

[21] R. Patti, “Three-dimensional integrated circuits and the future of system-on-chip designs,” *Proceedings of the IEEE*, vol. 94, pp. 1214 – 1224, Jun. 2006.

[22] P. Franzon, W. Davis, and T. Thorolffson, “Creating 3d specific systems: Architecture, design and cad,” in *Proceedings of the Design, Automation Test in Europe Conference Exhibition*, pp. 1684 – 1688, Mar. 2010.

[23] S. W. Yoon, D. W. Yang, J. H. Koo, M. Padmanathan, and F. Carson, “3d tsv processes and its assembly/packaging technology,” in *Proceedings of the IEEE International Conference on 3D System Integration*, pp. 1 – 5, Sept. 2009.

[24] S. Goel and et al., “Test and debug strategy for tsmc cowos stacking process based heterogeneous 3d ic: A silicon case study,” in *Proceedings of the IEEE International Test Conference*, pp. 1–10, Sept. 2013.

[25] B. Black and et al., “Die stacking (3d) microarchitecture,” in *Proceedings of IEEE/ACM Symposium on Microarchitecture*, 2006.

[26] K. Puttaswamy and G. Loh, “3d-integrated sram components for high-performance microprocessors,” *IEEE Transactions on Computers*, vol. 58, pp. 1369 – 1381, Oct. 2009.

[27] H. Sun and et al., “3d dram design and application to 3d multicore systems,” *IEEE Design Test of Computers*, vol. 26, pp. 36 – 47, Sept.-Oct. 2009.

[28] T. Zhang and et al., “A customized design of dram controller for on-chip 3d dram stacking,” in *Proceedings of IEEE Custom Integrated Circuits Conference*, pp. 695 –705, Sept. 2010.

[29] E. A. Lee and T. Parks, “Dataflow process networks,” in *Proceedings of the IEEE*, pp. 773–799, 1995.

[30] J. L. Hennessy and D. A. Patterson, *Computer Architecture, Fourth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., 2006.

[31] K. Ramakrishnan and e. al., “Variation impact on ser of combinational circuits,” in *International Symposium on Quality Electronic Design*, pp. 911–916, Mar. 2007.

[32] M. Nagata, “Limitations, innovations, and challenges of circuits and devices into a half micrometer and beyond,” *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 465–472, Apr 1992.

[33] H. Kufluoglu and M. Ashraful Alam, “A computational model of nbti and hot carrier injection time-exponents for mosfet reliability,” *Journal of Computational Electronics*, vol. 3, no. 3-4, pp. 165–169, 2004.

[34] J. Joven, F. Angiolini, D. Castells-Rufas, and J. Carrabina, “Qos-ocmpi: Qos-aware on-chip message passing library for noc,” in *Workshop on Programming Models for Emerging Architectures*, 2010.

[35] N. Muralimanohar et. al, “Optimizing nuca organizations and wiring alternatives for large caches with cacti 6.0,” in *Proceedings of the International Symposium on Microarchitecture*, pp. 3–14, 2007.

[36] A. Agarwal and et al., “The mit alewife machine: A large-scale distributed-memory multiprocessor,” in *Proceedings of Workshop on Scalable Shared Memory Multiprocessors*, 1991.

[37] J. Kubiatowicz, *PhD Dissertation: Integrated Shared-Memory and Message-Passing Communication in the Alewife Multiprocessor*. Massachusetts Institute of Technology, 1998.

[38] S. Wong, T. van As, and G. Brown, “p-vex: A reconfigurable and extensible softcore vliw processor,” in *Proceedings of the International Conference on Field Programmable Technology*, pp. 369–372, 2008.

[39] C. Kim, D. Burger, and S. Keckler, “Non-uniform cache architectures for wire-delay dominated on-chip caches,” *IEEE Micro*, vol. 23, pp. 99–107, Nov 2003.

[40] O. Lempel, “Second generation intel core processor family: Intel core i7, i5 and i3,” in *Presentation at the HotChips Symposium*, 2011.

[41] S. Kumar and R. van Leuken, “A 3d network-on-chip for stacked-die transactional chip multiprocessors using through silicon vias,” in *Proceedings of the International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, pp. 1–6, Apr. 2011.

[42] J. F. Broenink, G. H. e. Hilderink, A. Duller, G. Panesar, and D. Towner, "Parallel processing — the picochip way!," in *Communicating Processing Architectures*, pp. 125–138, 2003.

[43] J. Howard and et al., "A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling," *IEEE Journal of Solid-State Circuits*, vol. 46, pp. 173 – 183, Jan. 2011.

[44] R. F. van der Wijngaart, T. G. Mattson, and W. Haas, "Light-weight communications on intel's single-chip cloud computer processor," *SIGOPS Operating Systems Review*, vol. 45, pp. 73–83, Feb. 2011.

[45] T. G. Mattson et al., "The 48-core scc processor: the programmer's view," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–11, 2010.

[46] M. P. Forum, "Mpi: A message-passing interface standard," tech. rep., 1994.

[47] M. Saldana and P. Chow, "Tmd-mpi: An mpi implementation for multiple processors across multiple fpgas," in *Proceedings of the International Conference on Field Programmable Logic and Applications*, pp. 1–6, 2006.

[48] J. Psota and A. Agarwal, "rmpi: message passing on multicore processors with on-chip interconnect," in *Proceedings of the International Conference on High Performance Embedded Architectures and Compilers*, pp. 22–37, 2008.

[49] M. R. Guthaus et al., "Mibench: A free, commercially representative embedded benchmark suite," in *Proceedings of the International Workshop on Workload Characterization*, pp. 3–14, 2001.

[50] P. Francesco, P. Antonio, and P. Mar.al, "Flexible hardware/software support for message passing on a distributed shared memory architecture," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 736–741, 2005.

[51] H. Nikolov et al., "Daedalus: Toward composable multimedia mp-soc design," in *Proceedings of the Design Automation Conference*, pp. 574–579, 2008.

[52] J. Kin, M. Gupta, and W. H. Mangione-Smith, "The filter cache: an energy efficient memory structure," in *Proceedings of the International Symposium on Microarchitecture*, pp. 184–193, 1997.

[53] N. P. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," *SIGARCH Computer Architecture News*, vol. 18, pp. 364–373, May 1990.

[54] D. Stiliadis and A. Varma, "Selective victim caching: A method to improve the performance of direct-mapped caches," *IEEE Transactions on Computers*, vol. 46, pp. 603–610, May 1997.

[55] N. Duong et al., "Revisiting level-0 caches in embedded processors," in *Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, pp. 171–180, 2012.

[56] A. Janapsatya, S. Parameswaran, and A. Ignjatovic, "Hitme: Low power hit memory buffer for embedded systems," in *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 335–340, 2009.

[57] F. J. Sanchez, A. Gonzalez, and M. Valero, "Software management of selective and dual data caches," in *IEEE Technical Committee on Computer Architecture Newsletter - Special Issue on DSM and related issues*, pp. 3–10, 1997.

[58] K. Beyls and E. H. D'holl, "Reuse distance as a metric for cache behavior," in *Proceedings of the IASTED Conference on Parallel and Distributed Computing and Systems*, pp. 617–662, 2001.

[59] G. Rivera and C. wen Tseng, "Data transformations for eliminating conflict misses," in *In Proceedings of the Conference on Programming Language Design and Implementation*, pp. 38–49, 1998.

[60] K. Hoste and L. Eeckhout, "Comparing benchmarks using key microarchitecture-independent characteristics," in *Proceedings of the International Symposium on Workload Characterization*, pp. 83–92, 2006.

[61] T. Austin, E. Larson, and D. Ernst, "Simplescalar: an infrastructure for computer system modeling," *IEEE Computer*, vol. 35, pp. 59 – 67, feb 2002.

[62] E. D. Berger, K. S. McKinley, R. D. Blumofe, and P. R. Wilson, "Hoard: A scalable memory allocator for multithreaded applications," *ACM SIGPLAN Notices*, vol. 35, pp. 117–128, Nov. 2000.

[63] S. Cho and L. Jin, "Managing distributed, shared l2 caches through os-level page allocation," in *Proceedings of the International Symposium on Microarchitecture*, pp. 455–468, 2006.

[64] A. Ros, M. Cintra, M. Acacio, and J. Garcia, “Distance-aware round-robin mapping for large nuca caches,” in *Proceedings of the International Conference on High Performance Computing*, pp. 79–88, Dec 2009.

[65] L. Tang, J. Mars, and M. L. Soffa, “Contentiousness vs. sensitivity: Improving contention aware runtime systems on multicore architectures,” in *Proceedings of the International Workshop on Adaptive Self-Tuning Computing Systems for the Exaflop Era*, pp. 12–21, 2011.

[66] T. Agarwal, A. Sharma, A. Laxmikant, and L. Kale, “Topology-aware task mapping for reducing communication contention on large parallel machines,” in *Proceedings of the International Parallel and Distributed Processing Symposium*, p. 10, Apr. 2006.

[67] S. Zhuravlev, S. Blagodurov, and A. Fedorova, “Addressing shared resource contention in multicore processors via scheduling,” *SIGPLAN Notices*, vol. 45, no. 3, pp. 129–142, 2010.

[68] C. Feenstra, “A memory access and operator usage profiler framework for hls optimization,” *MSc. Thesis, Delft University of Technology*, 2011.

[69] W. van Teijlingen, “Determining performance boundaries and automatic loop optimization of high-level system specifications,” *MSc. Thesis, Delft University of Technology*, 2014.

[70] S. A. Ostadzadeh, R. J. Meeuws, C. Galuzzi, and K. Bertels, “Quad: A memory access pattern analyser,” in *Proceedings of the International Conference on Reconfigurable Computing: Architectures, Tools and Applications*, pp. 269–281, 2010.

[71] R. L. Mattson, J. Gecsei, D. R. Slutz, and I. L. Traiger, “Evaluation techniques for storage hierarchies,” *IBM Systems Journal*, vol. 9, pp. 78–117, June 1970.

[72] G. Almási, C. Caşcaval, and D. A. Padua, “Calculating stack distances efficiently,” *SIGPLAN Notices*, vol. 38, pp. 37–43, June 2002.

[73] C.-K. Luk and et al., “Pin: Building customized program analysis tools with dynamic instrumentation,” in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 190–200, 2005.

[74] W. Dally, “Virtual-channel flow control,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, pp. 194–205, Mar 1992.

[75] A. Zjajo, N. van der Meijs, and R. van Leuken, “Dynamic thermal estimation methodology for high-performance 3-d mpsoc,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, pp. 1920–1933, Sept 2014.

[76] H. Oprins and et al., “Fine grain thermal modeling and experimental validation of 3d-ics,” *Microelectronics Journal*, vol. 42, pp. 572–578, Apr. 2011.

[77] H. Oprins and et al., “Numerical and experimental characterization of the thermal behavior of a packaged dram-on-logic stack,” in *Proceedings of the Electronic Components and Technology Conference*, pp. 1081–1088, May 2012.

[78] R. Clarke and et al., “Thermal modeling of 3-d stacked dram over sige hbt bicmos cpu,” *IEEE Access*, vol. 3, pp. 43–54, 2015.

[79] K. Matsumoto and Y. Taira, “Thermal resistance measurements of interconnections, for the investigation of the thermal resistance of a three-dimensional (3d) chip stack,” in *Proceedings of the Semiconductor Thermal Measurement and Management Symposium, 2009*, pp. 321–328, Mar. 2009.

[80] B. Vaisband, I. Savidis, and E. Friedman, “Thermal conduction path analysis in 3-d ics,” in *Proceedings of the International Symposium on Circuits and Systems*, pp. 594–597, Jun. 2014.

[81] I. Savidis and E. G. Friedman, “Thermal coupling in tsv-based 3-d integrated circuits,” in *Workshop on 3D Integration - Design, Automation and Test in Europe*, Mar 2014.

[82] A. Sridhar et. al, “3d-ice: Fast compact transient thermal modeling for 3d ics with inter-tier liquid cooling,” in *Proceedings of the International Conference on Computer-Aided Design*, pp. 463–470, 2010.

[83] K. Skadron et. al, “Temperature-aware microarchitecture: Modeling and implementation,” *ACM Transactions on Architectural Code Optimizations*, vol. 1, no. 1, pp. 94–125, 2004.

[84] S. Memik, R. Mukherjee, M. Ni, and J. Long, “Optimizing thermal sensor allocation for microprocessors,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, pp. 516–527, Mar. 2008.

[85] R. Jagtap, “A methodology for early exploration of tsv interconnects in 3d stacked ics,” Master’s thesis, Delft University of Technology, Sept. 2011.

[86] R. Jagtap, S. S. Kumar, and R. van Leuken, “A methodology for early exploration of tsv placement topologies in 3d stacked ics,” in *Proceedings of the Euromicro Conference on Digital System Design*, pp. 382–388, Sept. 2012.

[87] S. Hanson, M. Seok, D. Sylvester, and D. Blaauw, “Nanometer device scaling in subthreshold circuits,” in *Proceedings of the Design Automation Conference*, pp. 700–705, Jun. 2007.

[88] A. Mercha et al., “Comprehensive analysis of the impact of single and arrays of through silicon vias induced stress on high-k / metal gate cmos performance,” in *Proceedings of the IEEE International Electron Devices Meeting*, pp. 2.2.1 – 2.2.4, Dec. 2010.

[89] W. Huang and et al., “Hotspot: a compact thermal modeling methodology for early-stage vlsi design,” *IEEE Transactions on Very Large Scale Integrated (VLSI) Systems*, vol. 14, no. 5, pp. 501–513, 2006.

[90] G. Paci et. al, “Exploring ”temperature-aware” design in low-power mpsocs,” in *Proceedings of the Design, Automation and Test in Europe Conference Exhibition*, pp. 838–843, 2006.

[91] D. Atienza et. al, “Hw-sw emulation framework for temperature-aware design in mpsocs,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 12, pp. 26:1–26:26, May 2008.

[92] A. Bartolini et. al, “A virtual platform environment for exploring power, thermal and reliability management control strategies in high-performance multicores,” in *Proceedings of the Great Lakes Symposium on VLSI*, pp. 311–316, 2010.

[93] T. Bouhadiba et. al, “Co-simulation of functional systemc tlm models with power/thermal solvers,” in *Proceedings of the International Symposium on Parallel and Distributed Processing*, pp. 2176–2181, 2013.

[94] A. Varma, *High-speed Performance, Power and Thermal Co-simulation for SoC design*. PhD thesis, University of Maryland, 2007.

[95] S. Priyadarshi et. al, “Thermal pathfinding for 3-d ics,” *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 4, no. 7, pp. 1159–1168, 2014.

[96] D. Milojevic and et al., "Automated pathfinding tool chain for 3d-stacked integrated circuits: Practical case study," in *Proceedings of the IEEE International Conference on 3D System Integration*, pp. 1–6, 2009.

[97] J. Cong et. al, "An automated design flow for 3d microarchitecture evaluation," in *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 384–389, 2006.

[98] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *Proceedings of the International Symposium on Computer Architecture*, pp. 83–94, 2000.

[99] A. Kahng, B. Li, L.-S. Peh, and K. Samadi, "Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration," in *Proceedings of the Design, Automation, Test in Europe Conference Exhibition*, pp. 423–428, 2009.

[100] S. Kumar, A. Aggarwal, R. Jagtap, A. Zjajo, and R. van Leuken, "System level methodology for interconnect aware and temperature constrained power management of 3-d mp-socs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, pp. 1606–1619, Jul. 2014.

[101] D. Cuesta, J. Risco-Martin, J. Ayala, and D. Atienza, "3d thermal-aware floorplanner for many-core single-chip systems," in *Proceedings of the Latin American Test Workshop*, pp. 1–6, 2011.

[102] D. Cuesta, J. Risco-Martin, J. Ayala, and J. Hidalgo, "3d thermal-aware floorplanner using a moea approximation," *Integration, the VLSI Journal*, vol. 46, no. 1, pp. 10–21, 2013.

[103] SoCLiB-Project, "Soclib: an open platform for virtual prototyping of multiprocessors system on chip." Available at http://www.soclib.fr.

[104] F. Mahon, *The Livermore Fortran Kernels: A Computer Test of the Numerical Performance Range*. Lawrence Livermore National Laboratory, 1986.

[105] U. Sonmez, R. Quan, F. Sebastiano, and K. Makinwa, "A 0.008-mm2 area-optimized thermal-diffusivity-based temperature sensor in 160-nm cmos for soc thermal monitoring," in *Proceedings of the European Solid State Circuits Conference*, pp. 395–398, Sept 2014.

[106] P. Chen, C.-C. Chen, C.-C. Tsai, and W.-F. Lu, "A time-to-digital-converter-based cmos smart temperature sensor," *IEEE Journal of Solid-State Circuits*, vol. 40, pp. 1642–1648, Aug 2005.

[107] A. Zjajo, N. van der Meijs, and R. van Leuken, "A 11 μw 0°c-160°c temperature sensor in 90 nm cmos for adaptive thermal monitoring of vlsi circuits," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 2007–2010, 2012.

[108] A. Bakker, "Cmos smart temperature sensors - an overview," in *Proceedings of IEEE Sensors*, vol. 2, pp. 1423–1427 vol.2, 2002.

[109] S. S. Kumar, "Realtime thermal map of pe and data cache," 2015. Available at http://youtu.be/3r8bllNRifM.

[110] S. S. Kumar, "Realtime thermal map of a data cache," 2015. Available at http://youtu.be/vxPmUNYImes.

[111] P. Chaparro, J. Gonzalez, and A. Gonzalez, "Thermal-aware clustered microarchitectures," *Proceedings of the International Conference on Computer Design*, pp. 48–53, 2004.

[112] S.-W. Lee and J.-L. Gaudiot, "Throttling-based resource management in high performance multithreaded architectures.," *IEEE Transactions on Computers*, vol. 55, no. 9, pp. 1142–1152, 2006.

[113] L. Shang, L.-S. Peh, and N. Jha, "Powerherd: a distributed scheme for dynamically satisfying peak-power constraints in interconnection networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 92–110, Jan 2006.

[114] K. Puttaswamy and G. H. Loh, "Thermal herding: Microarchitecture techniques for controlling hotspots in high-performance 3d-integrated processors," in *Proceedings of the International Symposium on High Performance Computer Architecture*, pp. 193–204, 2007.

[115] K. K. Chang, R. Ausavarungnirun, C. Fallin, and O. Mutlu, "HAT: heterogeneous adaptive throttling for on-chip networks," in *Proceedings of the International Symposium on Computer Architecture and High Performance Computing*, pp. 9–18, 2012.

[116] H. Jacobson and et al., "Stretching the limits of clock-gating efficiency in server-class processors," *Proceedings of the International Symposium on High Performance Computer Architecture*, vol. 0, pp. 238–242, 2005.

[117] E. Le Sueur and G. Heiser, "Slow down or sleep, that is the question," in *Proceedings of the USENIX Annual Technical Conference*, 2011.

[118] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low Power Methodology Manual: For System-on-Chip Design.* Springer Publishing Company, Incorporated, 2007.

[119] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget," in *Proceedings of the International Symposium on Microarchitecture*, pp. 347–358, 2006.

[120] G. Semeraro and et al., "Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," in *Proceedings of the International High-Performance Computer Architecture Symposium*, pp. 29–40, 2002.

[121] Q. Wu, P. Juang, M. Martonosi, and D. W. Clark, "Voltage and frequency control with adaptive reaction time in multiple-clock-domain processors," in *Proceedings of the International Symposium on High-Performance Computer Architecture*, pp. 178–189, 2005.

[122] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Proc. ACM/IEEE Int Low Power Electronics and Design (ISLPED) Symp*, pp. 38–43, 2007.

[123] M. Bao, A. Andrei, P. Eles, and Z. Peng, "On-line thermal aware dynamic voltage scaling for energy optimization with frequency/temperature dependency consideration," in *Proceedings of the Design Automation Conference*, pp. 490–495, Jul. 2009.

[124] X. Wang, K. Ma, and Y. Wang, "Adaptive power control with online model estimation for chip multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 1681 – 1696, oct. 2011.

[125] S. S. Kumar, A. Zjajo, and R. van Leuken, "Physical characterization of steady-state temperature profiles in three-dimensional integrated circuits," in *International Symposium on Circuits and Systems*, May 2015 (in press).

[126] M. M. Sabry, D. Atienza, and A. K. Coskun, "Thermal analysis and active cooling management for 3d mpsocs," in *Proceedings of the International Symposium on Circuits and Systems*, pp. 2237–2240, 2011.

[127] J. L. Ayala, A. Sridhar, and D. Cuesta, "Invited paper: Thermal modeling and analysis of 3d multi-processor chips," *Integration, the VLSI Journal*, vol. 43, pp. 327–341, Sept. 2010.

[128] C. Zhu et al., "Three-dimensional chip-multiprocessor run-time thermal management," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, pp. 1479 – 1492, aug. 2008.

[129] W. Kim, M. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core dvfs using on-chip switching regulators," in *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, pp. 123–134, Feb 2008.

[130] F. Clermidy, F. Darve, D. Dutoit, W. Lafi, and P. Vivet, "3d embedded multi-core: Some perspectives," in *Proceedings of the Design, Automation Test in Europe Conference Exhibition*, pp. 1–6, Mar. 2011.

[131] D. Truong and et al., "A 167-processor computational platform in 65 nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 44, pp. 1130 – 1144, Apr. 2009.

[132] D. Truong and et al., "A 167-processor 65 nm computational platform with per-processor dynamic supply voltage and dynamic clock frequency scaling," in *Proceedings of the IEEE Symposium on VLSI Circuits*, pp. 22 – 23, Jun. 2008.

[133] N. Ioannou, M. Kauschke, M. Gries, and M. Cintra, "Phase-based application-driven hierarchical power management on the single-chip cloud computer," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, pp. 131–142, Oct 2011.

[134] P. Gratz, B. Grot, and S. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *Proceedings of the International Symposium on High Performance Computer Architecture*, pp. 203–214, Feb 2008.

[135] S.-Y. Lin and e. al., "Traffic- and thermal-aware routing for throttled three-dimensional network-on-chip systems," in *Proceedings of the International Symposium on VLSI Design, Automation and Test*, pp. 1–4, Apr. 2011.

[136] C.-H. Chao and e. al., “Traffic- and thermal-aware run-time thermal management scheme for 3d noc systems,” in *Proceedings of the International Symposium on Networks-on-Chip*, pp. 223–230, May 2010.

[137] N. Dahir, R. Al-Dujaily, T. Mak, and A. Yakovlev, “Thermal optimization in network-on-chip-based 3d chip multiprocessors using dynamic programming networks,” *ACM Transactions on Embedded Computing Systems*, vol. 13, pp. 139:1–139:25, Apr. 2014.

[138] E. Kakoulli, V. Soteriou, and T. Theocharides, “Intelligent hotspot prediction for network-on-chip-based multicore systems,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, pp. 418–431, Mar. 2012.

[139] F. Liu, H. Gu, and Y. Yang, “Dtbr: A dynamic thermal-balance routing algorithm for network-on-chip,” *Computers and Electrical Engineering*, vol. 38, pp. 270–281, Mar. 2012.

[140] G.-M. Chiu, “The odd-even turn model for adaptive routing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, pp. 729–738, July 2000.

[141] A. Michos, “A novel concurrent validation scheme for hardware transactional memory,” Master’s thesis, TU Delft, 2012.

[142] J. de Klerk, “Cachebalancer: A communication latency and utilization aware resource manager,” Master’s thesis, Delft University of Technology, 2014.

[143] S. Kumar, “Tmfab: A transactional memory fabric for chip multiprocessors,” Master’s thesis, Delft University of Technology, 2010.

[144] N. Miura and et al., “A scalable 3d heterogeneous multicore with an inductive thruchip interface,” *IEEE Micro*, vol. 33, pp. 6–15, Nov 2013.

Samenvatting

De aanhoudende stijging van de door een nieuwe generatie toepassingen geëiste rekenprestaties drijft het aantal rekeneenheden van de moderne multiprocessor-systemen omhoog. In het *dark silicium* tijdperk worden de prestaties en de integratiedichtheid van dergelijke systemen echter bemoeilijkt door thermische beperkingen. Deze beperkingen worden groter bij *driedimensionale (3D)* geïntegreerde systemen, als gevolg van de complexe thermische eigenschappen van de 3D IC's. Dit proefschrift onderzoekt het ontwerp van efficiënte, temperatuur-gestuurde multiprocessor architecturen, en presenteert methoden om gelijktijdige optimalisatie van thermisch en functioneel gedrag mogelijk te maken. Deze onderwerpen worden in drie achtereenvolgende hoofdstukken behandeld waarbij ieder hoofdstuk een specifiek thema behandelt.

Hoofdstuk 2 onderzoekt de efficiëntie van multiprocessor architecturen vanuit het perspectief van de geheugenhiërarchie en presenteert technieken die gericht zijn op het effectieve beheer en overdracht van on-chip gegevens en op minimalisatie van de tijd die gebruikt wordt voor geheugentoegang. In het geval van *shared* geheugen multiprocessor architecturen, wordt dit bereikt door *Persistence Selective Caching (PSC)* en *CacheBalancer* schema's die invloed hebben op welke gegevens worden opgeslagen in on-chip geheugens, waar ze worden opgeslagen, en voor hoe lang. Hierdoor kan de geheugenhiërarchie zich aanpassen aan veranderend functioneel gedrag, het gebruik van hardware componenten balanceren en, het allerbelangrijkste, de gemiddelde wachttijd en energie per geheugentoegang verminderen. Hoofdstuk 2 presenteert het *Pronto* systeem dat een efficiënte overdracht van gegevens mogelijk maakt bij het verwerken van berichten van en naar microprocessoren, door de rol van het verwerkingselement in het beheer van de gegevensoverdrachten te minimaliseren. Pronto vermindert op effectieve wijze de indirecte kosten bij het opzetten en beheren van gegevensverwerkingen waardoor de communicatie-wachttijden korter worden. Bovendien vereenvoudigt Pronto de semantiek van de gegevensbeweging doordat implementatie-details van berichtencommunicatie voor de software programmeur worden verborgen waardoor transfers uitsluitend op taak-niveau kunnen worden gespecificeerd.

Thermische-bewust ontwerp voor *3D Integrated Circuits (IC)* door middel van de *Nagata's* vergelijking - een wiskundige voorstelling van het dark silicium probleem – wordt onderzocht in hoofdstuk 3. Dit hoofdstuk verkent de thermische ontwerpruimte van 3D-IC's gebaseerd op deze vergelijking, en stelt een hoog niveau ontwerpproces voor om de specifieke invloed van individuele ontwerpparameters op thermisch gedrag van 3D IC's te karakteriseren. De resultaten van deze verkenning verbeteren de state-of-the-art door de verkregen nieuwe inzichten in de kritieke rol van vermogensdichtheid, thermische geleidbaarheid en 3D constructie bij de vorming van *hotspots* in 3D IC's. Voortbouwend op deze inzichten is *Ctherm* ontwikkeld ten behoeve van thermische-bewust ontwerp voor *multiprocessor systems-on-chip (MPSoC)*. Ctherm maakt de gelijktijdige beoordeling van thermische en functionele prestaties van MPSoC's mogelijk aan het begin van het systeemontwerp en vergemakkelijkt de exploratie van thermisch gedrag met behulp van een automatisch gegenereerd fijnkorrelig opgedeeld silicium oppervlak, alsmede wachttijd en energiemodellen van systeemcomponenten. De effectiviteit van dit raamwerk wordt aangetoond door de toepassing van een aantal praktische ontwerpen variërend van floorplanning en temperatuursensor plaatsing en rekentaken. De karakterisering en de Ctherm omgeving verbeteren ons begrip van het thermisch gedrag van 3D IC's, en geeft ons een praktisch sjabloon voor het realiseren van thermisch-bewuste ontwerpgereedschappen voor 3D IC's.

Het beheren van de thermische problemen die zich voordoen in 3D MPSoCs wordt onderzocht in hoofdstuk 4. Temperatuurregeling wordt typisch uitgevoerd door middel van *Dynamische Thermal Management (DTM)* door continu activiteit en energiegebruik van systeemcomponenten aan te passen. Een belangrijk nadeel van state-of-the-art DTM ligt in haar onvermogen rekening te houden met het niet-uniforme thermische gedrag van 3D IC's, wat leidt tot ineffectief beheer van temperaturen en tot slechtere systeemprestaties. In hoofdstuk 4 wordt een nieuw 3D *Dynamic Voltage Frequency Scaling (DVFS)* algoritme geïntroduceerd dat rekening houdt met deze ongelijkmatigheden, effectief temperaturen handhaaft binnen een veilig bereik, en systeemprestaties maximaliseert binnen de beschikbare thermische marges van de individuele verwerkingselementen. Dit hoofdstuk presenteert ook een adaptieve verbindingspaden-strategie om de impact van de thermische gradiënten van netwerk-on-chip gebaseerde 3D-architecturen te verlagen door dataverkeer langs verbindingen met een laag temperatuurprofiel te sturen. De beschreven *Immediate Neighbourhood Temperatuur (INT)* adaptieve verbindingspaden methode reduceert dataverkeer van gebieden met thermische hotspots, gebaseerd op temperatuur-informatie over de directe omgeving en op de warmteoverdrachtskenmerken van 3D IC's, zonder de noodzaak van een globaal temperatuur-meet netwerk. De daaruit voortvloeiende verspreiding van dataverkeer over meerdere verbindingspaden resulteert in een evenwichtiger thermisch profiel en verlaagde bedrijfstemperaturen van het systeem.

Dit proefschrift onderzoekt de kritieke problemen die het realiseren van thermisch geoptimaliseerde 3D IC's bemoeilijken, en detailleert voor de reductie van het dark silicium verschijnsel veelzijdige oplossingen.

Acknowledgements

Our legacy lies in our people – those whose lives we form a part of, and those that become an inseparable part of our own. Through all the greatness that we achieve, it is these people that define the sum total of who we are and the value of what we contribute. This dissertation is the end result of a journey that began a little over 6 years ago. I had only just started my Msc thesis, and was thrilled to be building my first multiprocessor. Little did I realize that this project would go on to seed what eventually became the Naga many-core processor. Hidden amongst its many victories, this journey also played host to innumerable setbacks and disappointments. It is through these that I came to understand and accept failure as an important element of the learning process. This final chapter of my dissertation is dedicated to honouring those that gave me the strength to stand tall through countless battles, and the courage to take the road less traveled.

First, Rene van Leuken. Rene has been my advisor since I started working at the Circuits and Systems (CAS) group as an Msc student in the winter of 2008. He gladly obliged when I went up to him looking for an extra project to do alongside my coursework. Whether he was actually glad, safe to say, we will never know. Rene is a man with a calm demeanour, and an extremely sharp mind. He is most often that person in the room who, at the end of a technical discussion, with the simplest of questions, demolishes even the most solid argument. He is capable of administering a potent dose of humility to the ambitious student drunk on success, and instilling fearlessness in those in doubt of their own abilities. He exhibits the traits of a great teacher – someone that treats his students with respect, giving them a chance to learn regardless of their background. I shall forever hold great respect for him, and remember his willingness to put his weight behind plans simply on the basis of my word. There is much more that I could say about him, but I suspect it would be inappropriate to have an acknowledgement longer than the dissertation itself. Nevertheless, Rene's efforts are as intricately woven into this dissertation as my own.

Amir Zjajo and I started working together only towards the end of the second year of my PhD. Our fascination for power and temperature management pulled us into what became an intensive two year effort that yielded some of our most exciting results. It is with Amir that I progressed from writing like an academic, to writing like a scientist. This is no coincidence, as Amir himself is a brilliant scientist. Judicious in his work, meticulous in his reviews, gentle by nature, and a good listener, Amir is the kind of teacher that every budding scientist needs. One cannot help but become a little more intelligent just by spending an afternoon with him. That this dissertation is in the form it has taken today is a testament to Amir.

It has been my privilege to have had Alle-Jan van der Veen as my promotor through the course of my PhD. He is an extremely intelligent man with standards higher than the stack of books that occupy his desk at any given point in time. I hold a deep sense of respect for his ability to delve into subjects so distinct from his own, and quickly develop a profound understanding of its most intimate concepts. His incisive comments are often all that one needs to convert a good scientific argument into an excellent one. Over the years, I have developed a sincere appreciation for his role in nurturing CAS, and leading it through what I knew were tough times.

Michel Berkelaar's role in this journey is that of a guiding light, much like Gandalf in the Lord of the Rings. His humble disposition masks an extremely sharp intellect, and his intuitive understanding of human sensitivities lends his every critique a genuine sense of encouragement. He seems to always have a clear sense of what is important, and is empowered with the ability to deliver profound clarity even in times of confusion. A lifetime of gratitude would be insufficient in matching the value of his counsel.

Alexander and Huib contributed immensely to the design of Naga, and the infrastructure for SystemC simulation and high-level synthesis. It is their invaluable experience and insight that enabled many of the system design activities that we executed during the COBRA project.

I am sincerely grateful to Dirk Stroobandt, Jose Pineda de Gyvez, Francois Pecheux, Koen Bertels and Geert Leus for taking the time to read this dissertation, and for agreeing to be part of my defense committee. It was Francois that introduced me to the SoCLiB simulation platform which quickly became our simulation framework of choice. Was it not for our meeting, Ctherm would be an order of magnitude slower in its simulation speed. It was Prof. G Indumathi and Prof. Raveesha K.H. who provided me with much inspiration and encouragement early in my academic career. I will remain eternally grateful to both of them.

Like a parent, my greatest joys came to me through my students. It was my privilege to serve both formally and informally as advisor to a large group of bright students working on their thesis projects, and in the process watch them develop into

independent, fearless, critical-thinking engineers. It is through the eyes of Radhika and Arnica (my dearest friends and the two first ladies of COBRA), Tasos, Mitzi, Anupam, Jayesh, Jurrien, Kim and Ram that I got to explore subjects which I otherwise would never have had enough time for. Martijn, Milovan, Keke, Jaco, Wouter and Kiki were kind enough to involve me in decisions at important junctures in their work, and made for great company in the 17th floor design room. It is this set of people that defines to a great extent what I take away from my PhD, and who I became in the course of it. Their street signs are not the only mark they leave behind.

Minaksie Ramsoekh, the glue that holds everything together. CAS is a diverse group in both its technical focus as well as its staff. Minaksie is that smiling force that transcends all this diversity, and turns it into the friendly cohesive unit that it is. Her personal connection with every member of the group is abundantly clear in the sheer number of inside jokes she has running with most of us. Since I left CAS, I've missed seeing her in the morning, listening to her singing old Hindi songs. I now look forward to getting yelled at by her for not coming to visit more often. CAS would be very different without her. The same holds for Rosario whose work keeps our many projects ticking like clockwork.

Antoon Frehe's efforts in keeping our compute infrastructure running smoothly meant that I never spent a day without access to my machine. Despite having troubled him for over 7 years, he's still as friendly as the day I first met him. Always ready to help, never without a solution, and proactive to the bone, Antoon is one cool system admin that I'm very grateful to have worked with. I must also thank Snits for tirelessly working with me through all these years, and for being a faithful companion on the 17th floor. A large friendly bunch of students and staff members made CAS a lively and truly memorable place to be. Jorge and Andrea Simonetto brought our social activities to life with their enthusiasm and energy. Raj, Chocka, Venkat, Sundeep, Augusto, Esteban, Seyran, Shahzad, Mohammad Karami, Hadi, Hamid, Rocio, Sina, Francesco, Georg, Yuki, Shingo, Matt, Dony, Tao, Mu, Hyung-June, Qin, Yu Bi, Shahrzad, Andrea, Harald, Millad, Adib, Jeroen and Yan Xie made for wonderful colleagues. Wim, thank you for the many wonderful conversations.

There were also many others outside of CAS that made my experience so memorable. The ever sharply dressed Said Hamdioui was the one that chaperoned me at my first conference (DATE 2011), and graciously introduced me to the rockstars of our field. Koen, Stephan, Zaid and Carlo served as committee members for the defenses of a majority of our Msc students, and in doing so, provided important feedback that influenced the course of our work. Many thanks go out to Roel Seedorf and Anthony Brandon for their willingness to help out with the Rvex VLIW core. On a daily basis, Roy and Mariska provided the friendliest conversations at the Service Point, as did

the Sodexo ladies who, in addition, made sure that what I wanted to eat was always available. I sincerely thank Nininha Tavares for being so friendly and for taking care of our working environment so well.

Hannelore, Judith, Germaine, Willem, Mary, Marlies, Ismail and Ineke deserve special mention for giving me an opportunity to do something different every summer since 2009. The Introduction Programme is something that has become an inseparable part of me, and has given me wonderful friends and the best of memories. Not to forget, it was where I learnt the art of bartending. Young Mi, thank you for getting me involved in the programme's equivalent for PhD students.

Belgian beer, good food and genuine conversations are essential towards successfully overcoming the daily challenges of exploratory research. I thankfully find myself surrounded by dear friends that are in no short supply of any of these three. Adi, Rohan and Sarah, Yash, Raj, Puneet, Ghazaleh, Renu, Ashima, Arvind, Despina, Valia, Niraj, Suriya, Karthik, Phadnis, Elisa and Eduard, Chocka, Sachin, Manju, Sridevi and Rein, Axy and Ravi, Luisa, Trivik, Nundlall, Eleonora, Mallika, and Kajal, are some of the dear people that brought me much happiness and fulfilment outside of work. It is in their company that I found my life enriched with music, dance, theater, drama, magic, meditation, startups, philosophy, tiramisu, American candy, travel, and most importantly, love. Yash also served as my ”external consultant” on thermodynamics. The fun group of people at bootcamp, the lindyhop group, and the folks at Delftsbleau made for excellent companions.

Growing up as the youngest in my family, I was always awestruck by the stellar, larger than life personalities of my sisters. I often found myself wondering how they became as amazing as they are. Two decades have passed, and I'm still in search of that answer. Their strength and concrete will are inspiring, and their unrelenting concern for me serves as a constant reminder of what unconditional love really is. My parents taught me the importance of hard work and sincerity by example. They taught me the importance of fighting for principle, of being fearless in the face of every adversity if you believed your actions to be just, and right. They instilled in me the values that I hold dearly to this day – compassion, kindness, selflessness, unconditional respect for people and animals, courage, and faith. Everything good in me, I've received from them. And I shall consider it my fortune if I can give to my children as much as they've given me. I am but a reflection of my family.

I've always believed in the adage – “A rough road leads to the stars”. However, prolonged travel over rough roads, just as in real life, can leave you disoriented, and wondering why you chose that road in the first place. At such times, it was Sharmishta that was my voice of reason. It was her reassurance that gave me peace even in the middle of the fiercest storms, and the strength to persevere. In moments of uncertainty

she gave me clarity, and in those of weakness, she gave me strength. Through the ups and downs, she kept me anchored, never letting complacence or failure get the better of me. And most importantly, she constantly reminded me of the importance of being a good human being. Her support and encouragement has been truly limitless.

To all of these people, and many more that I could not mention, I am sincerely grateful. Their contributions are deeply embedded within this dissertation, and forever within myself.

Curriculum Vitae

Sumeet Kumar was born on 18th September 1986 in Kuwait, and grew up in the Indian city of Bangalore. He received the Bachelor of Engineering degree in Electronics and Communications from the CMR Institute of Technology (affiliated to the Visveswaraiah Technological University) in 2008. In the first half of the same year, he led a team developing energy-efficient instruction set extensions for a sensor control network processor at Indrion Technologies, Bangalore. He received the Master of Science degree in Microelectronics for his work on the TMFab transactional memory fabric for chip multiprocessors, and the R3 router architecture for 3D networks-on-chip. From February 2011, Sumeet worked as a PhD researcher with the Circuits and Systems group at the Delft University of Technology on the CATRENE funded Computing Fabric for High Performance Applications (COBRA) project. He is a Graduate Student Member of the IEEE, IEEE Circuits and Systems Society and the IEEE Computer Society, and serves on the Program Committee of the International Symposium on Networking and Computing (CANDAR), and as reviewer for the ACM Transactions on Embedded Computing Systems (TECS). In his free time, he enjoys bouldering and dancing the lindy hop.

Since March 2015, Sumeet works with the Imaging and Camera Technologies Group (ICG) of Intel Corporation in Eindhoven, The Netherlands as Processor Tools Engineer, designing domain-specific tools for the development of next-generation media processing architectures.

Propositions

1. The intelligent application of relatively small amounts of hardware can greatly improve system performance and efficiency. [Chapter 2]
2. Seemingly insignificant variations in temperature margins within die stacks have a considerable impact on the effective performance of 3D stacked multiprocessors. [Chapter 3]
3. The accurate simulation of operating temperatures is predicated on the granularity of spatial and temporal power models used during thermal-aware design space exploration. Though abstraction is a system architect's friend, it is reality that sours their relationship. [Chapter 3]
4. Thermal constraints in 3D stacked ICs are far more severe than those in conventional single-die ICs. Adaptability is key to sustaining performance in the dark silicon era. [Chapter 3,4]
5. The wide adoption of 3D stacking is hindered by the lack of clearly defined roles for vendors in the semiconductor supply chain. The definition of an industry standard dictating electrical interconnect, power and thermal specifications for *plug-and-play* 3D integration will facilitate the unfettered development of innovative systems without the hassles of intellectual property licensing.
6. We are inherently sequential in our thinking. The successful parallel programming model will respect this limitation.
7. Learning through practice results in the creation of deep-seated knowledge. It is essential to incorporate the element of practice within all aspects of education.
8. In a constructive democratic system, it is imperative for stakeholders to distinguish the rationality of opinions from that of the individuals that hold them.

9. That our perceptions are coloured by our experiences is both our fortune, as well as our misfortune.
10. The strength of one's character lies in one's ability to do that which is right instead of that which is good.

These propositions are regarded as opposable and defendable, and have been approved as such by the promotors prof.dr.ir. A.-J. van der Veen and dr.ir. T.G.R.M. van Leuken.

Stellingen

1. De intelligente toepassing van relatief kleine aantallen van hardware componenten kan systeem prestaties en efficiëntie sterk verbeteren. [Hoofdstuk 2]
2. Schijnbaar onbeduidende variaties van de temperatuur marges in 3D IC's heeft aanzienlijke invloed op de effectieve prestaties van de aanwezige multiprocessors. [Hoofdstuk 3]
3. Nauwkeurige simulatie van bedrijfstemperaturen tijdens het onderzoeken van thermisch gevoelige ontwerpruimte wordt gegeven door de compositie van de op ruimte en tijd gebaseerde vermogensmodellen. Alhoewel abstractie veelal een vriend van de systeemarchitect is, maakt de werkelijkheid hun relatie moeizaam. [Hoofdstuk 3]
4. Thermische beperkingen in 3D IC's zijn veel ernstiger van aard dan die in conventionele IC's. De sleutel tot het behoud van prestatie in het 'dark' silicium tijdperk is aanpassingsvermogen. [Hoofdstuk 3,4]
5. De brede invoering van 3D IC's wordt belemmerd door het ontbreken van duidelijk omschreven rollen voor leveranciers in de semiconductor productieketen. De definitie van een industriestandaard voor verbindingen, vermogen en thermische specificaties voor plug-and-play 3D-integratie zal de onbelemmerde ontwikkeling van innovatieve systemen vergemakkelijken, zonder het gedoe over gebruiksrechten van intellectueel eigendom.
6. Wij zijn inherent sequentieel in ons denken. Een succesvol parallel programmeer model zal deze beperking respecteren.
7. Leren door te doen resulteert in de creatie van diepgewortelde kennis. Het is essentieel om het element van praktijk te integreren in alle aspecten van het onderwijs.

8. In een constructief democratisch systeem is het noodzakelijk voor de stakeholders om de rationaliteit van meningen te scheiden van de rationaliteit van de individuen die deze mening hebben.
9. Dat onze waarnemingen worden gekleurd door onze ervaringen is zowel ons geluk als ons ongeluk.
10. De kracht van iemands karakter ligt in zijn vermogen om dat wat juist is te doen in plaats van dat wat goed is.

Deze stellingen worden opponeerbaar en verdedigbaar geacht en zijn als zodanig goedgekeurd door de promotoren prof.dr.ir. A.-J. van der Veen en dr.ir. T.G.R.M. van Leuken.