The Pennsylvania State University
The Graduate School

DESIGN AND ANALYSIS OF HETEROGENEOUS NETWORKS FOR CHIP-MULTIPROCESSORS

A Dissertation in
Computer Science and Engineering
by
Asit K. Mishra

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2011

The dissertation of Asit K. Mishra was reviewed and approved* by the following:

Chita R. Das
Distinguished Professor of Computer Science and Engineering
Dissertation Advisor, Chair of Committee

N. Vijaykrishnan
Professor of Computer Science and Engineering

Mahmut Kandemir
Professor of Computer Science and Engineering

W. Kenneth Jenkins
Professor of Electrical Engineering

Onur Mutlu
Assistant Professor of Electrical and Computer Engineering
Carnegie Mellon University
Special Member

Raj Acharya
Professor of Computer Science and Engineering
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

Rarely has there been as challenging and exciting a time for research in computer architecture as now. While, the proverbial Moore's law has consistently helped architects integrate more and more silicon transistors in a single die, device constraints of power, heat, and reliability has forced the computer industry to shift focus from single processor core performance to instantiating multiple processor cores on a chip. In this quest for integrating a large number of cores on a single chip, one particular area of computer architecture that has come into prominence is the interconnection network on a chip (also called network-on-chip or NoC). NoCs seek to provide a scalable, energy-efficient and high-bandwidth communication substrate for future multi-core and many-core architectures - an aspect that critically dictates future chip designs.

Most of the prior research in NoC has focussed on optimizing the NoC considering it as a *homogeneous* system i.e. all optimizations proposed, *equally* affect all the components in the NoC substrate. However, this dissertation demonstrates that the resources in the NoC (precisely, buffers and links) are not always equally utilized, and that not all applications demand similar resources from the underlying interconnection substrate. Hence, this dissertation argues that better and smarter NoCs can be architected by considering the inherent heterogeneity in NoCs from both the network architecture perspective and from the applications' perspective. Further, considering the fact that future multicores and systems-on-chip architectures will have heterogeneous cores and compute engines, and host diverse applications, it is also inevitable that not all components will demand similar responses from the NoC and neither will these compute engines stress the NoC uniformly. Thus, it is compelling to think of heterogeneous NoCs for such heterogeneous systems. To this end, this dissertation argues in favor of NoCs that factor heterogeneity as a first-order design objective while architecting them for future multi-core systems.

In this pursuit, this dissertation investigates micro-architectural techniques that exploit heterogeneity at the network resource consumption level (following a bottom-up approach) and from applications' demand/requirement perspective (following a top-down approach). With the bottom-up approach, heterogeneity is exploited with the key observation that not all resources in an NoC are equally utilized when employing a typical network topology and a network routing protocol. With the top-down approach, heterogeneity is exploited starting from the applications' demand perspective with the key observation that not all applications require similar resources from the underlying network substrate. Based on these two approaches, this dissertation proposes four techniques with the overall goal of designing high-performance and energy-efficient NoCs.

The first scheme, called Router Architecture with Frequency Tuning (RAFT), exploits heterogeneity in the buffers of the on-chip routers and proposes a variable-frequency scheme to operate them. This design is the first of its kind to propose a distributed congestion management scheme that is based on operating individual routers at different frequency levels. The second scheme, called HeteroNoC, targets non-uniformity in both buffers and links in the on-chip networks. Using the same amount of link resources and fewer buffer resources compared to a homogeneous network, this proposal demonstrates that a carefully designed heterogeneous network can reduce average latency, improve network throughput and reduce power. The third scheme, argues in favor of designing on-chip networks by taking into account the intrinsic communication requirements of applications. This proposal is based on the observation that, in general, applications can be classified as either network bandwidth sensitive or latency sensitive. Based on this, the proposal consists of two separate heterogeneous networks in the on-chip interconnection substrate, where one network is tailored to optimize for bandwidth sensitive applications and the second network for latency sensitive applications. The fourth scheme presented in this dissertation targets heterogeneity in device technology for improving the memory subsystem performance of multi-cores. This scheme leverages the advantages of an emerging memory technology that is based on spintronics, called spin torque transfer RAM (STT-RAM), for memory subsystem design. STT-RAM can be heterogeneously integrated onto silicon and this proposal argues in favor of designing the NoC in a way that is cognizant of the presence of STT-RAM cache banks.

This dissertation investigates each of the above proposals at depth and shows that the proposed schemes have minimal overheads in terms of area and power, are simple to implement, and show significant benefits with real applications. Overall, this dissertation makes a strong case for designing heterogeneous networks for improving the performance-power envelope of future multicore processors.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This dissertation would not have been possible without the motivation, help and guidance of many individuals. This is my sincere attempt to acknowledge these individuals.

First, I would like to acknowledge my dissertation advisor, Chita R. Das. He has been a constant source of inspiration and guidance throughout my graduate student tenure. He gave me the freedom to choose my research topic, always interacted with me to make my ideas concrete, constantly kept me motivated during the formative years of my graduate student tenure, and has consistently inspired me to do quality research. He has been a caring teacher and from him I have received much more than I gave.

I wish to acknowledge professors N. Vijaykrishnan and Mahmut Kandemir for providing me with valuable advice and insights throughout my graduate student tenure. I thank both these professors for being my mentors, critiquing my work and making my ideas simpler and practical. Both of them are great teachers and I have learnt many foundational concepts used in this dissertation from their lectures.

I thank Dr. Jenkins for serving in my dissertation committee.

My sincere thanks to professor Onur Mutlu for shaping this dissertation towards the end of my graduate work, for making me more methodical and organized in doing research, and for encouraging me to strive for the best. Because of his forthcoming and helping attitude, it was great pleasure in interacting and collaborating with him.

This thesis has also been shaped in numerous ways by my interactions with people in the computer architecture industry. I would thank Mani Azimi, Akhilesh Kumar, Ravi Iyer and Joseph Hellerstein for being my internship mentors and for immensely valuable discussions on computer architecture and computer science in general. I would also extend my appreciation for my co-authors Reetuparna Das, Soumya Eachempati, Shekhar Srikantaiah, Xiaowei Jiang, Guangyu Sun and Xiangyu Dong for collaborating with me.

The most valuable part of my Penn State experience has been my lab mates

and close friends who have helped me during paper submission deadlines, worked with me on course projects, listened to my naive and impossible ideas, and my complaints and rants after my papers got rejected. In this regard, I would thank the current and former members of HPC Lab with whom my graduate student tenure has overlapped. Especially, I thank Dongkook Park, Jongman Kim, Dennis Ersoz, Pushkar Patankar and Reetuparna Das, my lab seniors, for helping me get started in this lab. Seung-Hwan Lim, Bikash Sharma, Adwait Jog, Nachiappan Chidambaram, Mahshid Sedghi and Onur Kayiran have always been of immense help and I thank all them for their friendship.

Finally, I am running short of words in thanking my parents and little sister for always having supported me and in helping me write this dissertation in more ways than one can imagine.

Thank you all.

Dedication

$\sim \sim \sim$ *To my family and friends* $\sim \sim \sim$

Chapter 1

Introduction

For the last decade in the 20th century and early years of the 21st century, single core high performance processors provided architectural abstractions that enabled applications to exploit parallelism as a means to successfully utilize the exponentially increasing on-chip transistor counts. Following this time frame, however, device constraints of power, heat, and reliability impeded innovations in single core processors and forced the computer industry to shift focus to instantiating multiple processor cores on a chip.

Currently, multicore architectures seem to be the only plausible solution to meet the performance and power requirements of a wide variety of applications targeted for a general-purpose chip multi-processor (CMP) and the special purpose system-on-chip (SoC) environments. As per the ITRS road map [1], it is projected that the performance demand would continue to grow to 300x by 2022, which in turn would require chips with 100x more cores than the current state-of-the-art designs. Better performance demand coupled with the need to minimize chip power consumption necessitates a fresh look at the design of future multicores.

The three main components of a multicore that dominate the performance and power envelope are the processor cores, memory, and the underlying on-chip interconnection fabric. While computer architects have been designing processors and caches since the single-processor era, the advent of multicores introduced a new on-chip component - the interconnection fabric. There can be numerous approaches in architecting this on-chip interconnection fabric - ad-hoc point-to-point networks or more structured network-on-chip (NoC). Poor scaling of global wires

when compared to local wires and gate delays, renders point-to-point networks unscalable. Therefore, scalable, multi-hop packet switching based NoC are widely viewed as the natural choice for connecting the multiple nodes in a CMP. It is thus foreseeable that on-chip networks will become one of the most critical resources in many-core systems and hence, design of high performance and low-power NoC is essential for sustaining the multicore growth in coming years.

An NoC can be viewed as a programmable system that facilitates the communication of information and messages between components on a chip. Alternatively, an NoC can be viewed as a system that integrates many components including channels, buffers, switches and control. Many recent works have proposed innovative solutions to optimize each of these components in the NoC. These solutions include optimizations in router micro-architectures [2–6], routing and flow control [7–9], NoC topologies [10–15], power-efficient and thermal-aware NoC designs [16–18], and fault tolerant and process variation resilient NoC [19,20]. While all these works improve the performance and power envelope of the NoC, most of these works are focussed on optimizing the NoC by considering it as a unified homogeneous system. With this design philosophy, the proposed optimizations equally affect all the components in the NoC. Although such an approach simplifies the task of an architect and chip designer in terms of design verification and correctness, this dissertation will demonstrate that the resources in an NoC are not always uniformly utilized. Further, this dissertation will also highlight the fact that dynamic resource requirements from applications are non-uniform. Hence, it is compelling to investigate NoC designs that treat the underlying NoC substrate as a heterogeneous system where both dynamic resource requirements and utilization are non-uniform. Based on this premise, this dissertation offers a unique perspective on designing high-performance, scalable and energy efficient NoC's by exploiting heterogeneity in typical NoCs.

1.1 Resources in NoC

Apart from the router nodes and channels interconnecting the routers, other building blocks comprising an NoC include: its layout topology, routing and flow control mechanism, and router microarchitecture. The network topology dictates the

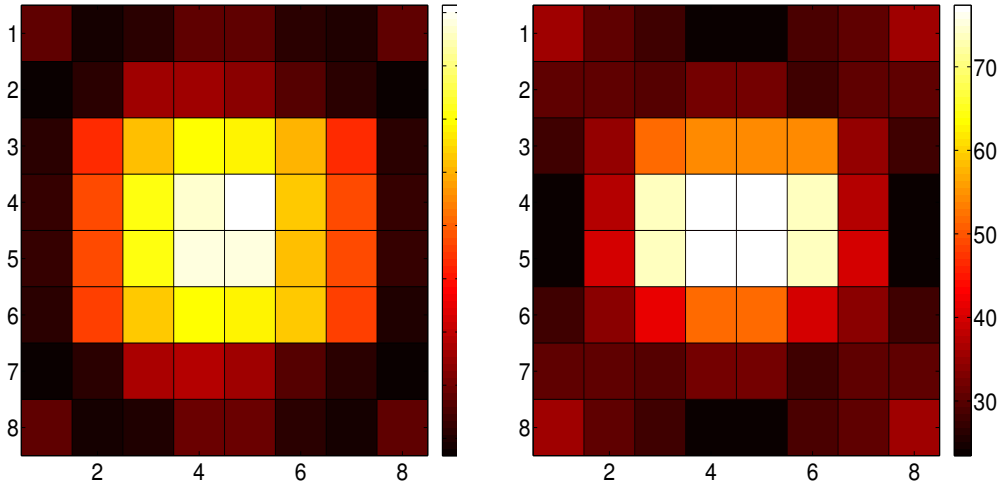
physical on-chip layout and inter-connections between nodes and channels in the network. Typical on-chip topologies include mesh [21], concentrated mesh [22], hybrid bus based topology [10, 12],express channel topologies [11, 13], and high radix topologies [14, 15, 23]. For a given topology, a routing algorithm dictates the route a message in the network will traverse from a source to its destination. Typical routing algorithms include X-Y routing [22], which belongs to a class of deterministic routing, and adaptive routing [24–26]. Flow control on the other hand, dictates the allocation of resources (buffers and links/bandwidth) in the network. Virtual-channel (VC) [27] based wormhole flow control [28, 29] is the most widely used flow control mechanism in NoC domain.

Modern state-of-the-art on-chip network designs use a modular packet-switched fabric in which network channels are shared over multiple packet flows. Such sharing of network channels (using VCs) across multiple flows provides high bandwidth utilization in the network. Packets need to compete for resources on a hop-by-hop basis, while going through a simple router pipeline before traversing the output channel/link at each intermediate node along their path. Thus, the packet energy/delay in such networks is dominated largely by contention at intermediate routers. In a mesh based topology (a typical NoC topology), each router has five input and output ports corresponding to the four neighboring directions and the local processing element (PE) port. The major components, which constitute the router microarchitecture are the input buffers, route computation logic, VC allocator, switch allocator and crossbar switch. The next two subsections will quantify the heterogeneity in resource consumption in an NoC and heterogeneity in application requirements from an NoC.

1.1.1 Non-uniform Resource Utilization

Figures 1.1 (a) and (b) show, on a heat-map scale, the average buffer and link utilization in a typical 8x8 mesh network[1] with uniform random traffic pattern. As can be observed, the routers in the center of the mesh are highly ($\backsim$75%) utilized, while the peripheral routers have low ($\backsim$35%) utilization, and the routers

[1]The network is wormhole switched and uses deterministic X-Y routing and is operated at close to saturation throughput (6% packets/node/cycle). This network has 3 virtual channels per physical channel, 5 flit buffer depth with 128 bit flit-width.



(a) Buffer utilization (b) Link utilization

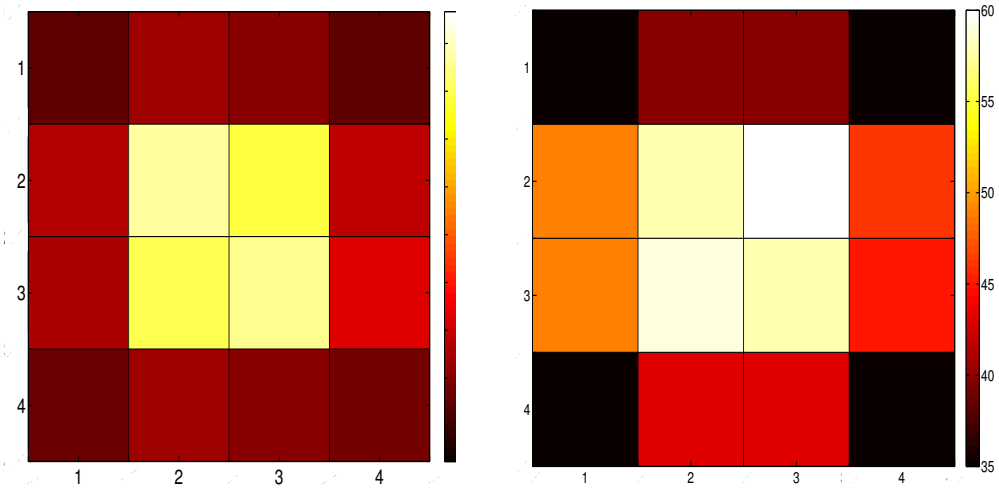
Figure 1.1. Buffer and link utilization (in percentage) across all routers in a 8x8 mesh on a heat-map scale.

around the center of the mesh have buffer utilization that lie between these two extremes. The center of a mesh usually gets more congested because it handles more traffic compared to the edge routers with deterministic X-Y routing [21]. A similar trend is seen for average link utilization as depicted in Figures 1.1(b). This behavior is consistent at both medium and high network loads and for a wide variety of traffic patterns like transpose, bit-compliment, shuffle and self-similar.

Further analysis shows that the non-uniformity in resource utilization is an artifact, primarily, in any non-edge symmetric network that employs deterministic X-Y routing. For instance, concentrated mesh [22] and flattened butterfly [14] topologies are not edge symmetric, and exhibit non-uniform resource usage as well. Figure 1.2 shows the buffer utilizations in a 4x4 concentrated mesh with concentration degree of 4 and in a 64 node flattened butterfly topology (each router is connected to 4 nodes; thus, 16 routers) with uniform random traffic. Like the mesh topology, these two topologies also exhibit non-uniform resource demands, and thus, this argument should be true for any non-edge symmetric network.

1.1.2 Non-uniform Resource Requirements

Existing NoC designs are implicitly built on the paradigm that all the hosted applications place similar demands on the underlying network. Based on this paradigm,



(a) Concentrated mesh (b) Flattened butterfly

Figure 1.2. Buffer utilization (in percentage) in concentrated mesh and flattened butterfly topologies on a heat-map scale.

a single underlying interconnect architecture caters to all applications' network demand. This dissertation, argues against this paradigm by observing how different packets (even within the same application, but particularly across different applications) have vastly differing network resource demand and how each individual network packet impacts application-level performance. This sub-section contrasts few observations that highlight the intrinsic heterogeneity in network demands across applications. These observations put together, provide the motivation for this dissertation's application-aware design of NoC, which is described in Chapter 5. This analysis starts by looking at two of the fundamental parameters in NoC: network channel bandwidth and latency.

Impact of channel bandwidth on performance scaling of applications: Channel or link bandwidth is a critical design parameter that affects network latency, throughput and energy/power of the entire network. By increasing the link bandwidth, the packet serialization latency reduces, however increase in link bandwidth adversely affects a router crossbar power envelope. To study the sensitivity of an application to variation in link bandwidth, a simple analysis is performed. For this analysis, an 8x8 mesh network is used and 64 copies of the same application are run on all nodes in the network[2].

[2]The network is wormhole switched, uses deterministic X-Y routing, has 6 virtual channels per physical channel and 5-flit buffer depth. Each router in the network tile is connected to a

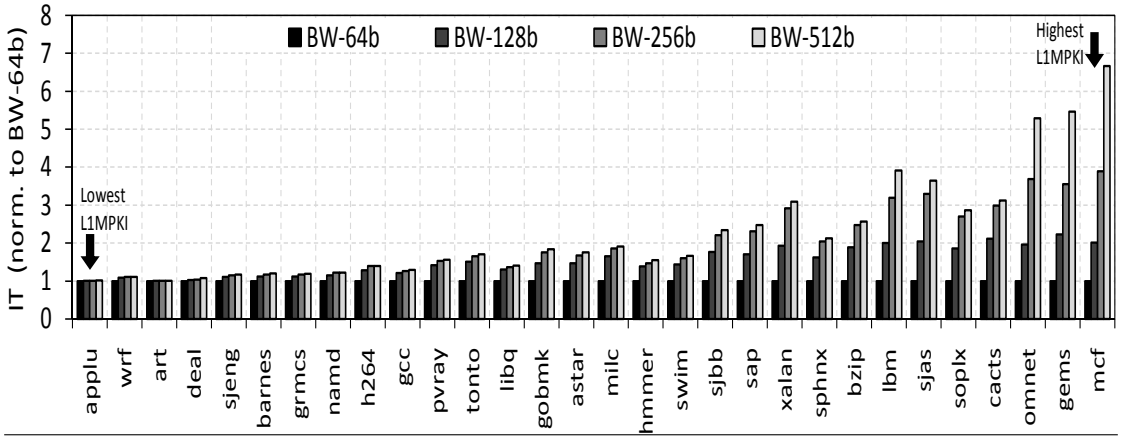


Figure 1.3. Instruction throughput (IT) scaling of applications with increase in network bandwidth.

Figure 1.3 shows the results of this analysis for 30 out of the 36 applications in the benchmark suite (6 applications are omitted to reduce clutter in the plots). This analysis considers scenarios where the network bandwidth is doubled at each step starting with 64b links to 512b links (annotated as BW-64b, BW-128b, BW-256b and BW-512b in the figure). In this figure, the applications are shown on the X-axis in order of their increasing L1MPKI (L1 misses per 1000 instructions), i.e. `applu` has the lowest L1MPKI and `mcf` has the highest L1MPKI. The Y-axis shows the average instruction throughput when normalized to the instruction throughput of the 64b network.

Observations from this analysis are as follows: (1) Out of the 30 applications shown, performance of 12 applications (the rightmost 12 in the figure after `swim`) scale with increase in channel bandwidth. For these applications, an increase in 8x bandwidth results in at least 2x increase in performance. These applications are called *bandwidth sensitive* applications. (2) The rest 18 applications (all applications to the left of `swim` and including it), show very little to no improvement in performance with increase in network bandwidth. (3) Even for bandwidth sensitive applications, not all applications' performance scale equally with increase in bandwidth. For example, while `omnet`, `gems` and `mcf` show more than 5x performance improvement for 8x bandwidth increase, applications like `xalan`, `soplex` and `cacts`

core, a private L1 cache, and an 1MB per core shared L2 cache. The network is clocked at the same frequency as the cores (2GHz). The applications used in this experiment are detailed in Chapter 2.

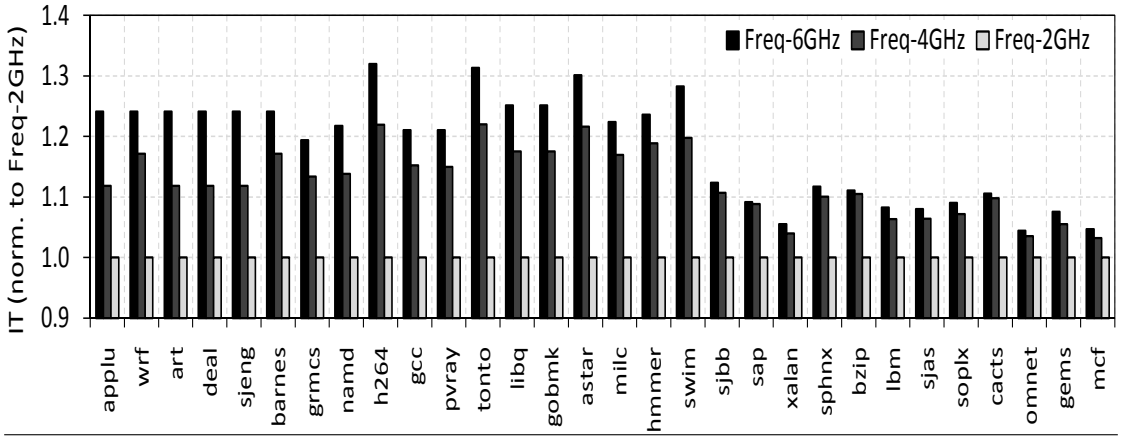


Figure 1.4. Instruction throughput scaling of applications with increase in network frequency.

show only 3x improvement for the same bandwidth increase. (4) L1MPKI is not necessarily a good predictor of bandwidth sensitivity of applications. Intuitively, applications that have high L1MPKI would inject more packets into the network, and hence, would require more bandwidth from the network. But this intuition does not hold entirely true. For instance, `bzip` in spite of having higher L1MPKI than `xalan`, is less sensitive to bandwidth than `xalan`. Thus, a better metric is needed to identify bandwidth sensitive applications.

Impact of network latency on performance scaling of applications: Next, the impact of network latency on the instruction throughput of these applications is analyzed. Network and router frequency have been advocated by few recent works [10, 30, 31] to improve performance. By increasing the frequency of the routers in a network, packet latency can be reduced, while adversely affecting the energy envelope. To study the sensitivity of applications, the frequency of the network is increased from 2GHz to 4Ghz and 6Ghz, while keeping the core frequency at 2GHz. Figure 1.4 shows the results for this analysis, where the channel bandwidth is 128b (although the observation from this analysis holds true for other channel bandwidths as well). The observations from this analysis are the following: (1) Bandwidth sensitive applications are not very responsive to frequency and on an average, for a 3x increase in frequency, there is only 8% improvement in application performance (instruction throughput) for these applications. (2) On the other hand, for all applications to the left of `swim` (and including `swim`), there

is about 25% performance improvement when the frequency increases from 2GHz to 6GHz. These applications are clearly very sensitive to network latency and these applications are called *latency sensitive* applications. (3) Additionally, this analysis shows that, increasing the frequency of the network leads to less than 1% increase in energy across both latency and bandwidth sensitive applications (results for energy with frequency scaling is omitted here but is discussed in detail in Chapter 5). (4) Further, L1MPKI is not a good indicator of latency sensitivity (`hmmer` in spite of having higher L1MPKI when compared to `h264`, does not show proportional performance improvement with increase in frequency).

1.2 Heterogeneous Network Designs

These observations of non-uniform *resource usage and requirements* compels a rethinking in the design of interconnects by questioning the rationality of uniform resource distribution across the networks, particularly across all routers. Intuitively, a network design that takes into account this inherent heterogeneity should be able to better optimize the network compared to traditional homogeneous network and router design. To this end, this dissertation investigates heterogeneity at the network and application level, and at the technology level for improving performance and reducing the overall power envelope in NoCs and memory subsystem.

In this quest, the first proposal, called Router Architecture with Frequency Tuning (RAFT), exploits heterogeneity in the buffer utilization of the on-chip routers and proposes three variable-frequency schemes to operate them. The proposed variable frequency algorithms in RAFT are targeted to reduce latency and power consumption in the NoC by distributed throttling and boosting of router frequencies depending upon network load. This proposal is the first of its kind to propose *a distributed congestion management scheme that is based on operating individual routers at different frequency levels.* It is demonstrated that the proposed techniques are not only effective in delivering better performance and reducing power consumption compared to the monolithic, single frequency design, but also can outperform other pure performance enhancement techniques such as using adaptive routing and simply increasing the operating frequency of routers

without any congestion management.

The second proposal, called HeteroNoC, targets non-uniformity in both buffers and links in the on-chip networks. HeteroNoC is a true heterogeneous network design targeted for general purpose CMPs where strategic placement of big and small routers in the network leads to improvement in performance and reduction in power. Using the *same* amount of link resources and *fewer* buffer resources compared to a homogeneous network, this proposal demonstrates that a carefully designed heterogeneous network can reduce average latency, improve network throughput and reduce power. It is shown that the proposed heterogeneous design with X-Y routing can even outperform a homogeneous network with adaptive routing. To further demonstrate the applicability of HeteroNoC approach, a co-design of HeteroNoC with heterogeneous cores is also presented in this dissertation.

The third proposal is based on the analysis in sub-section 1.1.2 which suggests that a single monolithic network is not the best option for catering various application demands. Therefore, an alternative approach to designing an on-chip interconnect is to explore the feasibility of *multiple networks* each of which is specialized for common application requirements, and dynamically steer requests of each application to the network that matches the application's requirements. Based on Figures 1.3 and 1.4, a wider and a low frequency network is suitable for bandwidth sensitive applications, while a narrow and high frequency network is best for latency sensitive benchmarks. However, a designer also requires a mechanism to classify applications at runtime to one of the two categories: *bandwidth and latency sensitive* for guiding them to the appropriate network. In addition, since not all applications are equally sensitive to bandwidth or latency, this dissertation proposes a fine grain prioritization of applications within the bandwidth and latency optimized sub-networks. This prioritization scheme is shown to further improve the overall application and system performance.

The fourth proposal discussed targets heterogeneity in *device technology* for improving the memory subsystem performance in multi-cores. This proposal argues in favor of spin torque transfer RAM (STT-RAM) as a universal replacement for memory on the die due to the 4x storage benefits of STT-RAMs when compared to SRAM based designs. However, STT-RAMs suffer from a systemic write latency problem and this proposal looks at a scheme in the NoC to mitigate this

drawback. The scheme consists of facilitating prioritization in a 3D network by partitioning the STT-RAM layer into a number of logical regions and restricting the path diversity for accurate prediction of a cache bank's busy/idle status. This proposal is the first to show how the on-chip network can be leveraged to optimize and integrate STT-RAM based cache architectures for CMPs.

The rest of this dissertation is organized as follows. Chapter 2 provides preliminaries for NoC architectures and the simulation framework used in all the analysis in this dissertation. Then, the RAFT scheme is presented in Chapter 3. HeteroNoC architecture is discussed in Chapter 4, followed by the description of the multiple networks technique in Chapter 5. Chapter 6 looks at the microarchitectural optimization schemes when STT-RAMs are integrated in the memory hierarchy. Chapter 7 concludes this dissertation and presents few proposals for future work.

Chapter 2

Network On Chips: A Brief Primer

This chapter first provides preliminaries for NoC architectures and then details the simulation framework used in all the analysis in this dissertation. Finally, this chapter ends with a discussion of prior works most closely related to those ideas presented in this dissertation.

2.1 Network and Router Architecture

As mentioned earlier in Chapter 1, network topology dictates the physical on-chip layout and inter-connections between nodes and channels in the network. NoC topology affects the number of routers a message traverses and thus, influences network latency. An $N \times N$ mesh interconnect is the most widely studied NoC topology due to its scalability, regularity and ease of implementation in silicon. Various variations of the mesh architecture have been proposed recently [10,13,22]. Routers and links are the basic building blocks of an NoC. A generic NoC router has P input and P output channels/ports. In most implementations, P=5; four inputs from the north, east, south and west directions, and one from the local Processing Element (PE). A router's microarchitecture can be decomposed into five pipeline stages: buffer write, route computation, virtual-channel allocation, switch allocation and crossbar traversal [21, 32]. Speculation [2] and look-ahead routing [33] can reduce the per hop pipeline latency and based on Peh and Dally's work [2], this dissertation uses state-of-the-art two-stage router microarchitecture. Further, deterministic X-Y routing algorithm [21], finite input buffering, wormhole

switching [28, 29], and virtual-channel (VC) flow control [27] is employed in the network architecture.

In wormhole switching and VC flow control, a head flit, on arriving at an input port, first gets decoded and buffered according to its input VC in the buffer write (BW) pipeline stage. In the next stage, the routing logic performs route computation (RC) to decide the output port for the packet. The header then arbitrates for a VC corresponding to its output port in the VC allocation (VA) stage. Upon successful allocation of a VC, the head flit proceeds to the switch allocation (SA) stage where it arbitrates for the switch input and output ports. On winning the output port, the head flit then proceeds to the switch traversal (ST) stage, where it traverses the crossbar. This is followed by link traversal (LT) to travel to the next node. Body and tail flits follow a similar pipeline except they do not go through RC and VA stages, instead inheriting the VC allocated to the head flit. The tail flit, on leaving the router, de-allocates the VC reserved by the header. To remove the serialization delay due to routing, prior work has proposed lookahead routing [33] where the route of a packet is determined one hop in advance, thereby enabling the head flit of a packet to compete for VCs immediately after the BW stage.

Typical router implementations require a clock cycle for each of the stages within the router. Lower latency router architectures parallelize the VA and SA using speculative allocation [2], which predicts the winner of the VA stage and performs SA based on that. Further, look-ahead routing [33] can also be employed to perform routing of node i+1 at node i. These two modifications result in two-stage, and even single-stage routers [6], which parallelize the various stages of operation.

2.1.1 Evaluation Methodology

For evaluation, a detailed trace-driven cycle-accurate hybrid NoC/cache simulator for CMP architectures is used. The memory hierarchy implemented is governed by a two level directory cache coherence protocol. Each core has a private write-back L1 cache. The L2 cache is shared among all cores and split into banks. The coherence model includes a MESI-based protocol with distributed directories, with

Table 2.1. Baseline processor, cache, memory and network configuration

Processor Pipeline	2 or 3 GHz processor, 128-entry instruction window
Fetch/Exec/Commit width	2 instructions per cycle in each core, only 1 can be a memory operation
L1 Caches	32 KB per-core (private), 4-way set associative, 128B block size, 2-cycle latency, write-back, split I/D caches, 32 MSHRs
L2 Caches	1MB banks, shared, 16-way set associative, 128B block size, 3-cycle bank latency, 32 MSHRs
Main Memory	4GB DRAM, up to 16 outstanding requests for each processor, 320 cycle access, 4 on-chip Memory Controllers
Network Router	2-stage wormhole switched, VC flow control, 6 VC's per Port, 5 flit buffer depth
Network Topology	8x8 mesh, each node has a router, processor, private L1 cache 4 Memory controllers (1 at each corner node) For 3D network: 128b TSBs except region TSBs, which are 256b.

each L2 bank maintaining its own local directory. The simulated memory hierarchy mimics SNUCA [34]. The sets are statically placed in the banks depending on the low order bits of the address tags. The network timing model simulates all kinds of messages: invalidates, requests, replies, write-backs, and acknowledgments.

The baseline CPU and network configurations are detailed in Table 2.1. The CPU model consists of a fetch/issue/commit out-of-order pipeline with a reorder buffer. The trace format for the applications consists of load/stores and the number of non-memory instructions between them. The non-memory instructions are executed in a single-cycle in a detailed processor model and occupy the reorder buffer until they are ready for commit (become head of the re-order buffer). The interconnect model implements a 2-stage packet-based NoC router mentioned above. For the network routers, the power estimates extracted from the synthesized implementations and Orion [35], are fed to the simulator to compute overall power consumption. Most of the router hardware logic mentioned in this dissertation are implemented in structural Register-Transfer Level (RTL) Verilog and then synthesized in Synopsys Design Compiler using a TSMC 90nm standard cell library. The library uses appropriate wire-load approximation models to accurately capture wire loading effects. The area and power numbers then obtained were scaled down to the correct technology node [36].

2.1.2 Application Suite

Both synthetic and real world benchmarks are used for performance and power consumption analysis. When simulating synthetic traffic, the network is initially warmed up with 1000 packets and the statistics are collected for 100,000 packets. Uniform Random (UR), Transpose (TP), Nearest Neighbor (NN), Bit-Complement and Self-Similar traffic patterns are used for simulating synthetic traffic (although most results in this dissertation are shown with UR traffic, the results are found to be consistent with other traffic patterns also and wherever anomalies exist, they are pointed out). With real benchmarks, a diverse set of application workloads comprising scientific, commercial, and desktop applications are used. These applications are chosen from SPEC OMP [37], SPLASH-2 [38], SPEC 2006 [39], TPC-C [40], SPEC-JBB [41], SJAS [42], SAP [43] and PARSEC suite [44].

2.1.3 Evaluation Metrics

With synthetic workloads, average flit and packet latency, average power consumption, network throughput and energy-delay product (EDP) as a function of input load are used as evaluation metrics.

With applications, the primary performance evaluation metrics are IPC (for multi-threaded benchmarks), instruction throughput and weighted speedup (for multi-programmed workloads). Instruction throughput is defined to be the sum total of the number of instructions committed per cycle (IPC) in the entire CMP (Eq. (2.1)) and is considered an application throughput metric [45, 46]. The weighted speedup metric [47], sums up the slowdown experienced by each application in a workload, compared to its stand alone run under the same configuration (Eq. (2.2)) and is widely regarded as a *system throughput* metric [48]. For some case studies using the multi-programmed workloads, *application-level* fairness using the maximum application slowdown [45, 49] metric (Eq. (2.3)) is also evaluated. Harmonic-mean speedup (Eq. (2.4)) is considered to balance performance and fairness and is used in some of the analysis. For multi-programmed workloads (SPEC 2006 benchmarks), 64 copies of the same application are run on 64 cores and improvements are reported for the slowest copy. Similarly, with multi-threaded applications, the improvements reported are with the slowest threads.

$$\text{Instruction throughput} = \sum_i IPC_i \tag{2.1}$$

$$\text{Weighted speedup} = \sum_i \frac{IPC_i^{shared}}{IPC_i^{alone}} \tag{2.2}$$

$$\text{Maximum slowdown} = max_i \left(\frac{IPC_i^{alone}}{IPC_i^{shared}} \right) \tag{2.3}$$

$$\text{Harmonic speedup} = \frac{Number of Applications}{\sum_i \frac{1}{IPC_i^{shared}/IPC_i^{alone}}} \tag{2.4}$$

2.2 Summary of Prior Work

Network-on-Chip is widely viewed as a de-facto solution to wire-delay problems with future technology scaling [21,50–53]. However, due to the resource constrained nature of an NoC substrate, most researches have focused on two major themes - improving the performance and reducing power consumption in NoC. This section briefly summarizes those NoC approaches aimed at improving performance and reducing power which are closely related to this dissertation. This section ends with a summary of related work in the field of STT-RAMs.

Router design: One of the primary design challenge which NoC architects face is to design efficient routers that meet the latency and throughput requirements amidst tight area and power constraints. To this end, one of the early proposals for NoC routers was to use virtual channels (VCs) for improving the bandwidth utilization and throughput [21]. In fact, VCs were first proposed for macro-networks in multicomputer systems as a solution for deadlock avoidance [27], but is primarily used in NoC routers to remove the head of line blocking problem [5].

Pipelining the various stages (buffer write, route computation, virtual channel allocation, switch arbitration and crossbar traversal) of a NoC router can improve throughput of the NoC further [4, 21, 32]. To remove the route computation stage from the critical path, lookahead routing [33] has been proposed, where the route of a packet is determined one hop in advance. Pre-computing the route one hop in advance, allows the flits of a packet to arbitrate for VCs immediately after

the buffer write stage. The route computation required at the next hop can be computed in parallel with the VC allocation stage.

Bypassing is another technique that is used to further shorten the router pipeline [2–4] delay. With bypassing, critical path is shortened when a flit speculatively enters the crossbar traversal stage if there are no flits ahead of it in the input buffer queue. If pipeline bypassing is not possible because router ports are busy, aggressive speculation can be used to reduce down the critical path [2, 4, 6]. With speculation, a flit enters the switch arbitration stage speculatively after being written into the input buffer and arbitrates for the switch port while simultaneously trying to acquire a free VC. If the speculation is successful, the flit enters the crossbar traversal stage directly. Else, if speculation fails, the flit goes through some of these pipeline stages again, depending on where the speculation failed.

Unified buffer organization has been proposed in [5] as a means to improve the performance and to avoid head of line blocking. Bufferless deflection routing cuts down the on-chip buffer requirements of the NoC completely and has been proposed in [8]. However, bufferless designs usually add complexity to the control logic of a router, and to this end [54] proposes a simplified router microarchitecture which eliminates in-router buffers and the crossbar.

Routing algorithm: While numerous routing algorithms have been proposed in literature, the most widely used routing algorithm in NoC is dimension-ordered routing owing to its simplicity [21]. With dimension-order routing (which is an example of deterministic routing), all messages from a source to a destination will always traverse the same path. X-Y routing is an example of dimension-ordered (deterministic) routing. With X-Y routing, a message traverses the network first along X-direction, reaching the ordinate matching its destination, before switching to the Y-direction. Another class of routing algorithm is oblivious routing [55, 56], where messages traverse different paths from source to destination. A third class of routing is called adaptive routing [4, 24, 57–59], where a message takes different paths from source to destination but the route chosen depends on network congestion [4, 60]. Routing algorithms that dynamically switch between adaptive and deterministic routing have also been proposed [7]. In terms of implementation, table-based implementations of routing algorithms have been explored in [61, 62].

Power/Thermal aware NoCs: To reduce power and thermal profiles in

on-chip networks, several prior works have proposed communication aware topologies [10–12], and, power and thermal management in routers [18, 63]. Bufferless routing [8, 54] can also significantly reduce network power by eliminating router buffers. Prior works have also proposed DVFS for links [16,64] to manage power in off-chip networks. In PowerHerd [65], throttling a flit traversal in a router has been shown as a means to manage peak power constraints in off-chip networks. Also, in ThermalHerd [18], a collaborative run-time thermal management scheme is proposed for on-chip networks that uses distributed throttling and thermal correlation based routing to tackle thermal emergencies.

Heterogeneous networks: Work in [66] showed the effectiveness of using two kinds of routers in a mesh-based NoC for GPGPU [67] applications. This design is based on the observation of many-to-few-to-many traffic pattern in manycore accelerators. This observation is exploited by employing a checkerboard network organization that uses routers with limited connectivity and full-connectivity on top of a variant of 2-phase ROMM routing [68]. The limited connectivity routers allow traffic to be routed only in certain directions and hence is inefficient for handling large number of cache to cache transfers (e.g. data and coherence messages). Thus, such limited connectivity and restricted routing works well in the context of accelerators, and it may not be effective for general purpose CMPs, where traffic patterns are not deterministic and there are large number of inter-cache messages.

A polymorphic on-chip strata was proposed in [69]. On this strata various interconnect topologies can be mapped statically to suit individual application requirements. The proposal leads to flexible design options but incurs significant area (hence, power) overheads (40% and higher).

Since this dissertation looks at designing heterogeneous NoCs, which are quite well studied in the context of SoCs, the following paragraph summarizes few related works in SoC domains that have investigated heterogeneous interconnection networks.

Prior work in the SoC domains have proposed customization of on-chip resources like buffers [70], links [71–73] and topology [72, 74]. All these works adopt a static approach, where optimal buffer sizes and link widths are pre-determined at design time based on a detailed analysis of application-specific traffic patterns. The sizing is optimal for one particular application or similar application suites

and one hardware mapping. To this end, the proposal in [70] targets application-specific (ASIC) environments, where for known traffic patterns, the scheme varies the buffer depth in each router. Thus, application characteristic needs to be known a-priori and the network is customized for each target application. Similarly, works like [71–73], require the communication demands (generation rates, packet-sizes and patterns) are known a-priori. In contrast, this dissertation proposes heterogeneous interconnection networks where total *buffer resources* and *link widths* are apportioned by exploiting the non-uniform resource usage in routers in a typical NoC. It is independent of application type and is targeted for general purpose CMPs that can host a variety of application, where customizing the NoC for one application is prohibitive. Further, the delay, area and power constraints of NoCs for SoCs are fairly different from those of CMPs. This is largely because of the different market segment where SoCs typically target the low-end mobile embedded market. In such market segments, clock frequencies, area and power budgets are not as aggressive as those in CMPs.

Multiple networks: Few works that have proposed multiple networks for NoCs include TRIPS [75], RAW [76], Tilera [77], and IBM cell [78]. The motivation for including multiple networks in all these designs is entirely different than investigated in this dissertation - in TRIPS, multiple networks are used to connect operand networks, RAW has two static networks (routes specified at compile time) and two dynamic networks (one for trusted and other for untrusted clients) and Cell's EIB has a set of four unidirectional concentric rings (arranged in groups of four and interleaved with ground and power shields) primarily to reduce coupling noises. Even DASH multiprocessor [79] had multiple networks (request and reply meshes), but the design was meant to eliminate request-reply deadlocks. Tilera's iMesh network consists of five separate networks to handle memory access, streaming packet transfers, user data, cache misses, and interprocess communications. Among these five networks, there is only one network where the processor (user) gets to send data from cores to caches (and vice-versa). Each of the five networks are based on sizes of packets from a source to a destination.

The idea of classifying applications into latency and bandwidth sensitive classes (which this dissertation also does) was investigated in TCM [46]. Based on this classification, TCM proposes a heterogeneous memory scheduling policy and it is

shown that such an application aware scheduler can improve both system performance and fairness. System performance is improved by allocating a share of the main memory bandwidth for latency-sensitive applications. Fairness is achieved by shuffling scheduling priorities of memory-intensive applications at regular intervals to prevent starvation of any single application. Further, the notion of latency and bandwidth sensitive applications, is also exploited in [80], where a heterogeneous DRAM architecture is proposed.

STT-RAM aware cache design: To address the latency/energy overhead associated with the write operations in STT-RAM (a problem which this dissertation also targets), researchers have proposed various mitigation techniques at both circuit-level and architectural level. For instance, circuit-level approaches such as eliminating redundant bit-writes [81] and data inverting [82] have been proposed to reduce the write energy; architectural techniques such as read-preemptive write-buffer design and hybrid cache/memory architecture [83,84] can also help mitigate the latency and energy overhead in STT-RAM. Work in [85, 86] explored on-chip toggle-mode MRAM as a replacement of DRAM to improve the memory bandwidth and latency. The proposal in [87] studied the STT-RAM circuit design and presented a performance, power, and area model for STT-RAM caches and analyzed the benefits of 3D-stacked STT-RAMs at an architecture level. A scheme proposed in [83] tackles the STT-RAM write problem by using a read-preemptive write buffer technique. This technique allows read operations to terminate ongoing write operations. Circuit designs for STT-RAM early write termination to reduce STT-RAM write overhead is presented in [81].

Chapter 3

RAFT: A Router Architecture with Frequency Tuning

This chapter introduces a network architecture that is based on operating the individual routers in the NoC at different frequencies. The dynamic frequency for each router is decided based on the buffer utilization in the router and the overall goal of this proposal is to design an energy proportional network i.e. a network whose energy and average power consumption is proportional to the load in the network.

3.1 Introduction

Router frequency is one of the critical design parameter that directly affects both performance and power, albeit in a contradictory fashion. With a sophisticated design of a router pipeline, it is possible to increase the operating frequency [6,31], but higher router frequency leads to higher power consumption. On the other hand, a mismatch between processor and router/network frequency can result in significant performance penalties [88]. Thus, a prudent control of the router frequency can help in optimizing both performance and power. In this context, a new design philosophy for NoCs is presented in this chapter where, unlike the traditional single-frequency routers, this design proposes to dynamically modulate the frequencies of routers in a network for effectively managing network congestion and energy consumption. The proposed variable frequency router design, RAFT,

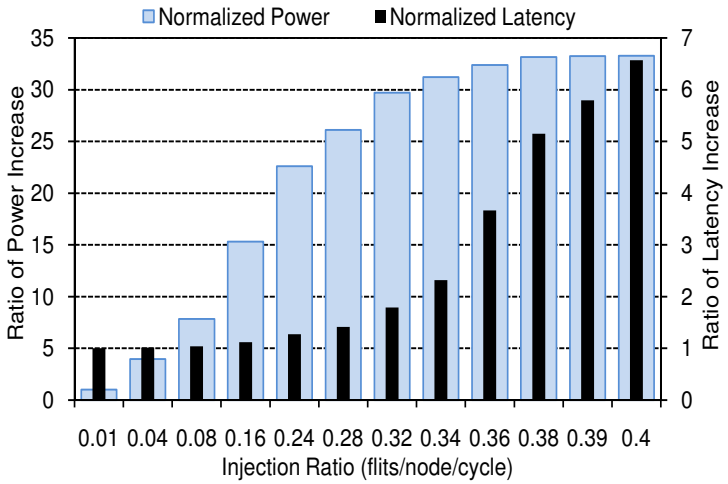


Figure 3.1. Average network latency and power behavior of an 8x8 mesh network.

uses the well-known DVFS technique and complements the use of DVFS in the processor cores for power management.

Figure 3.1 motivates the fine balance that exists between power and performance in an on-chip network with a relative power-performance trade-off analysis with respect to offered network load. Figure 3.1 shows the relative growth of network power versus network latency for an 8x8 mesh with a synthetic traffic mixture of Uniform Random, Transpose, Nearest-Neighbor and Self Similar traffic. The bars indicate network latency/power normalized with respect to the network latency/ power at no load (idle network). At low load, the network power consumption is less. However, the rate of growth of network power is much higher as compared to the rate of growth of network latency. For example, as shown in Figure 3.1, the network power grows to 30x as the injection rate varies from 1% to 40%, whereas the network latency grows only 7x. This work leverages insights from these trends to optimize the network at low load for performance and at high load for power.

Since performance and power are directly proportional to frequency, RAFT dynamically modulates the router frequency in response to network load to facilitate these optimizations, and demonstrate the advantages at system level. Specifically, at low load the routers are operated at peak frequency. At high load, the network dynamically determine the operating frequency of individual routers. The dynamic schemes that determine the operating frequencies of the routers are designed to (a)

reduce power consumption and (b) manage congestion in the network, by selectively stepping up and down the frequency of a *subset of routers* in the congested regions of a network. This proposal proposes a two-prong approach to vary the baseline router frequency: *clock scaling* and *time-stealing*. Dynamic Voltage and Frequency Scaling (DVFS) [16, 89] is employed to scale up and down the router clock frequency below the nominal frequency by switching the operating voltage levels. The time stealing technique is employed to boost the baseline router frequency by exploiting the timing imbalance between router pipeline stages, such that a router can operate at the average cycle time of all the pipeline stages in contrast to the delay of the worst case pipeline stage.

RAFT explores three techniques for dynamic frequency tuning to simultaneously address power-performance trade-offs. The first technique, called *FreqBoost*, initially employs time-stealing to operate all routers at a boosted frequency. This helps in enhancing the performance at low load conditions, while slightly increasing the power consumption. However, as the network gets congested, power consumption becomes a key challenge. Hence, it throttles the frequency/voltage of selected routers using DVFS. The second mechanism, called *FreqThrtl*, initially operates all routers at the baseline frequency and selectively employs time-stealing and DVFS to either increase or decrease the frequency at the onset of congestion. This scheme, unlike *FreqBoost*, can modulate frequency of routers bi-directionally (higher or lower) and consequently can help reduce power and manage congestion at high load more effectively. Using this technique, the frequency of a congested router is boosted at the onset of congestion and the frequency of a router adjacent to this congested router is throttled. *FreqTune* is a hybrid of the above two schemes that dynamically switches between *FreqBoost* and *FreqThrtl* as the network load varies from low to high.

3.2 Frequency Tuning Rationale

RAFT uses a congestion metric (buffer utilization) per port in a router to decide whether this port of the router is likely to get congested in the next few cycles, and if so, it signals the upstream router to throttle. The intuition behind such an approach comes from the fact that if a router is getting congested, it is due

to the pressure from its neighboring routers. The congested router is unable to arbitrate and push out its flits fast enough compared to the rate of flit injection into its buffers. To handle this mismatch and reduce the contention in the congested router, the upstream router is throttled by lowering its frequency. This decrease in frequency of the upstream router leads to a lower rate of arrival of the flits into the congested router, giving the congested router some leverage to push out its flits, and hence, reduce the overall blocking latency in the network. Based on these premises, RAFT consists of three techniques: *FreqBoost*, *FreqThrtl* and *FreqTune*.

Table 3.1. Settings for $FreqBoost$ and $FreqThrtl$

F_{base}	F_{boost}
2.20 GHz	2.75 GHz
$Threshold_{congestion}$	$Threshold_{low}$
0.60	0.40

Table 3.2. $Threshold_{throttled}$ Settings for FreqBoost, FreqThrtl and FreqTune. The table shows throttling frequency that a neighboring router uses based on its buffer utilization(BU) after receiving a *congested_high* signal.

	$BU > 0.60$	$0.50 < BU < 0.60$	$0.40 < BU < 0.50$	$BU < 0.40$
$FreqBoost$	F_{boost}	$0.9{*}F_{boost}$	$0.85{*}F_{boost}$	$0.8{*}F_{boost}$
$FreqThrtl$	F_{base}	$0.9{*}F_{base}$	$0.85F_{base}$	$0.8{*}F_{base}$
$FreqTune$	F_{boost}	$0.85{*}F_{boost}$	F_{base}	$0.8 * F_{base}$

3.2.1 FreqBoost Technique

In the *FreqBoost* technique, RAFT operates the network at a higher frequency, F_{boost} (2.75 GHz), compared to the nominal operating frequency, F_{base} (2.2GHz), right from the beginning using time-stealing technique and employs DFS to throttle a router for congestion management. This technique and all other techniques proposed later in this chapter use two thresholds, $Threshold_{congestion}$ and $Threshold_{low}$ (shown in Table 3.2), for triggering the proposed schemes. $Threshold_{congestion}$ is used to decide pro-actively whether a particular port is likely to get congested in the next few cycles. Upon detection of such an onset of congestion, this port signals a *congested_high* to the upstream router. Upon receipt of the *congested_high* signal, the upstream router compares its *overall* buffer utilization with the $Threshold_{throttled}$ settings, shown in Table 3.2, and depending on its overall buffer utilization, the

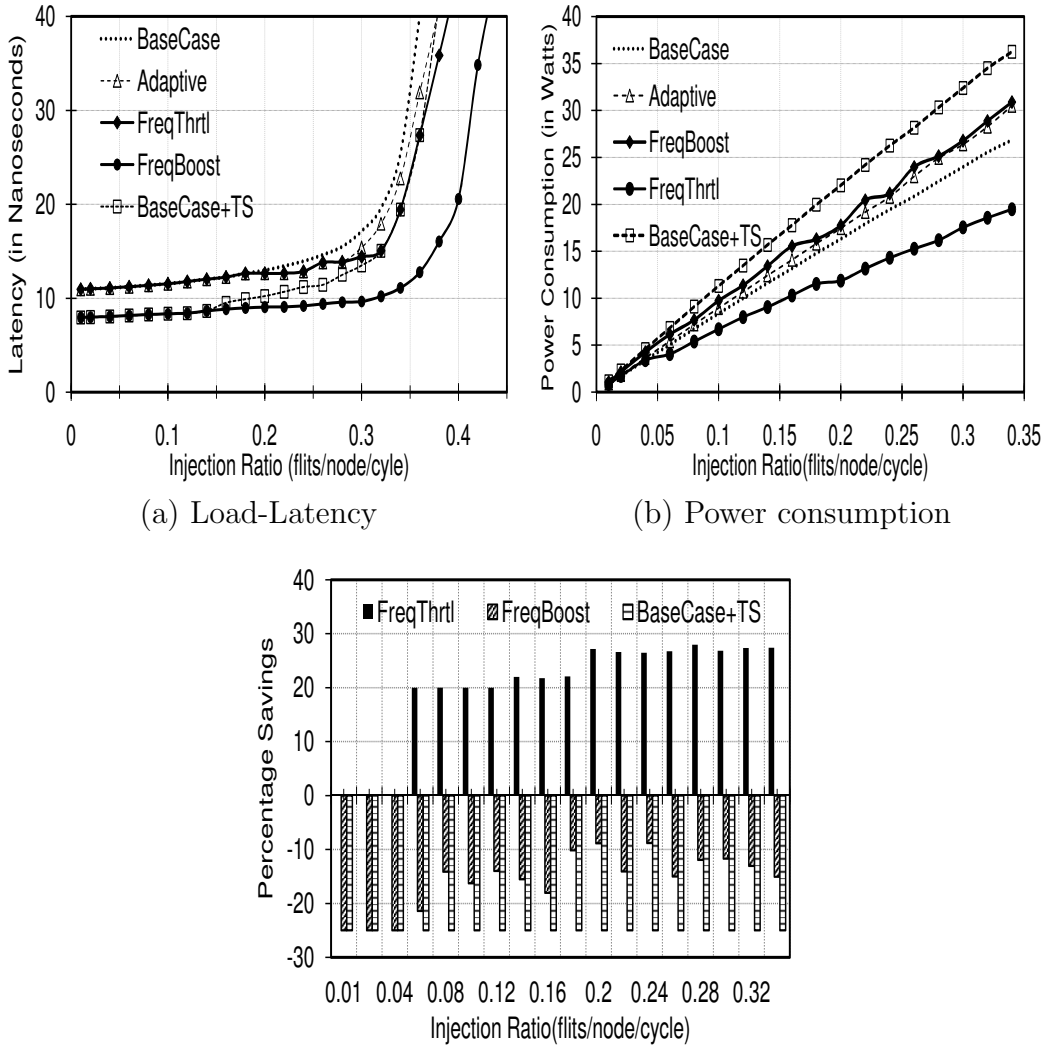
upstream router decides its operating frequency. Table 3.2 also shows the different frequency settings for each level of buffer utilization in the router.

If the total buffer utilization in the router that received the *congested_high* signal is high, then it will not throttle itself by a big margin, whereas, if the buffer utilization is low, then it would throttle itself aggressively. The reason behind using overall buffer utilization as a metric in choosing the throttled frequency settings and not a port's buffer utilization, is to handle asymmetric traffic, where buffers across some ports are heavily utilized compared to other ports. This scenario arises in X-Y routing, where buffers in the X-direction are heavily utilized compared to buffers in the Y-direction. The $Threshold_{throttled}$ settings ensure that the buffer utilization of a router to be throttled is not close to the congestion threshold, and thereby, the slowed-down router can sustain throttling, without itself getting congested. When a congested router's buffer utilization goes below $Threshold_{low}$, it signals*congested_low* to its upstream router. After receiving a *congested_low* signal, the throttled router increases its frequency back to F_{boost}.

3.2.2 FreqThrtl Technique

With $FreqThrtl$, RAFT increases the frequency of a congested router to F_{boost} only at the onset of congestion and throttles the upstream router to manage congestion, otherwise the base frequency,F_{base} (2.2GHz), is not enhanced during normal operation. Increasing the frequency of the congested router helps in servicing the flits faster through higher rate of arbitration and flit traversal. Additionally, slowing down upstream routers helps to reduce the pressure of injections and helps to ease out the traffic in the congested router.

Figure 3.2 shows the load-latency and power consumption/ savings in an 8x8 network for Uniform Random (UR) traffic. $BaseCase$, in the figures depicts a network, where no time-stealing or DVFS techniques are applied. For comparison purposes, latency curves with a minimally-adaptive routing algorithm (shown as *Adaptive* in the figure) and a case where time-stealing alone (no DVFS) is employed on top of the base case to boost the performance of the network (shown as $BaseCase + TS$) are also plotted. As can be seen, $FreqBoost$ and $FreqThrtl$ increase the throughput of the network at higher injection rates when $BaseCase$



(a) Load-Latency

(b) Power consumption

(c) Savings in power w.r.t. BaseCase

Figure 3.2. Performance and network power analysis of *FreqBoost* and *FreqThrtl* with UR traffic.

starts saturating. *FreqBoost* always gives the best performance compared to all of the schemes. With *FreqBoost*, RAFT operates the network at a 25% higher frequency due to time-stealing, and hence, consumes 25% more power at low-injection rates. However, at low injection rates, absolute power consumption in the network is low (less than 7W till 12% injection rate) and *FreqBoost* does not, therefore, increase the absolute power envelope significantly. At higher injection rates, *FreqBoost* starts throttling routers to manage congestion and thus, power consumption in the network decreases. The power curves for *FreqBoost*

lie in between *BaseCase+TS* and *BaseCase*. Comparison of performance and power curves for *BaseCase+TS* and *FreqBoost* shows that simply increasing the frequency of a router does not lead to significant performance benefits. Since *FreqBoost* employs intelligent throttling of routers along with frequency boosting, the performance difference between *FreqBoost* and *BaseCase+TS* can solely be attributed to the congestion management schemes, where the frequency of individual routers are tuned depending on load conditions.

With *FreqThrtl*, RAFT always saves power (Figure 3.2(b)), which comes mostly when few routers under high network load operate at lower than nominal voltage and frequency. However, when compared with *FreqBoost*, *FreqThrtl* saturates earlier. Therefore, while *FreqBoost* helps in performance enhancement and consumes more power, *FreqThrtl* helps in reducing power consumption while providing smaller performance enhancement margins. Figure 3.2(c) depicts that simply boosting the network frequency by 25% (*BaseCase+TS*) leads to a consistent 25% more power consumption. The figures also show that both *FreqBoost* and *FreqThrtl* outperform adaptive routing in average latency and *FreqThrtl* provides much better power savings due to congestion management in the network. Simulations show a similar power and performance trend with other non-uniform traffic patterns as well. Hence, in subsequent sections, evaluation results comparing adaptive routing to RAFT are not presented.

3.2.3 FreqTune Technique

Leveraging insights from the above results, this proposal proposes an adaptive technique that utilizes *FreqBoost* and *FreqThrtl*. It uses *FreqBoost* at low load for performance enhancement and switches to *FreqThrtl* at high load to save power and manage congestion. The EDP results shown in later sections reinforce this choice. This hybrid technique is called *FreqTune*, since using this technique, the network dynamically *tunes* to the traffic conditions. With FreqTune, each router selects its operating mode using information from its neighbors. This leads to distributed throttling of routers in the network with a mix of schemes, wherein some regions operate in the nominal frequency mode, some in FreqBoost mode and others operate in FreqThrtl mode. Thus, with *FreqTune*, a router transitions

from using high-frequency, F_{boost}, at low load to using nominal frequency, F_{base}, at medium load and again to high frequency at high load if it gets congested. This distributed dynamic frequency transition, based on network load, manages congestion as well as saves power in the network.

3.3 RAFT Architecture

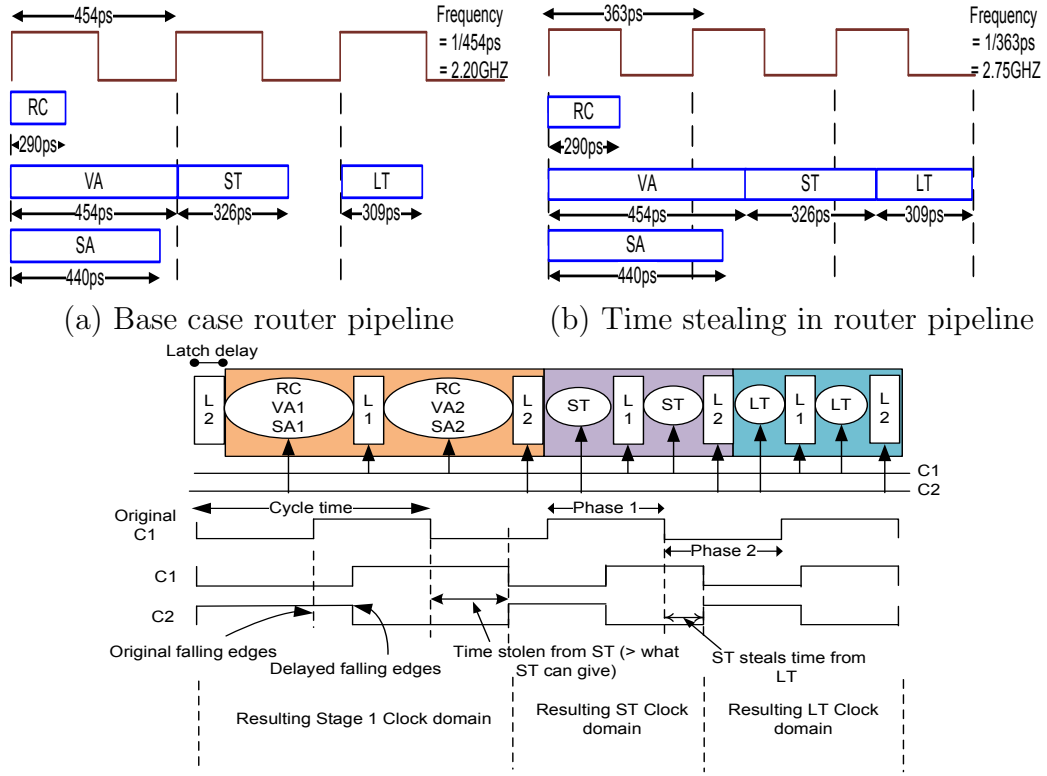
This section discusses the architectural enablers for RAFT : on-chip DVFS in routers and time-stealing.

3.3.1 Frequency Scaling

RAFT extends the concept of per-core on-chip DVFS [90] to per-router DVFS in NoCs for congestion and power management. RAFT adopts the two-step voltage regulator configuration as proposed by Kim et al. [90]. A multi-phase buck converter that can provide three voltage and frequency levels and a programmable voltage controlled ring oscillator [91] are used. The programmable ring oscillator is required for different frequencies at the same voltage to support the time-stealing technique described next. The on-chip regulators operate at 125MHz switching frequency and provide voltage transitions between 1V to 0.8V. The overhead in every transition is primarily the voltage settling time which is 13ns for every 100mV change [90]. Hence, due to higher than required voltage during step-down (and lower frequency during step-up), the power consumed by the router during a transition lies between the values before and after the scaling. In order to minimize the overhead of supporting multiple power domains in the network, this proposal also explores the notion of one regulator being shared among a group of routers.

The power consumption (at activity factor of 0.5) for the regulators with conversion efficiencies similar to that in [90] is given in Table 3.3. The area overhead in an 8x8 mesh for 64 on-chip regulators is $4mm^2$ which is about 25% of the area. The average power overhead is around 8% (at 1V) of all routers in the network. This area and power overhead reduces by 4x when RAFT uses per-column regulators (described later in Section 3.4.3).

3.3.2 Time Stealing in Router Pipeline



(a) Base case router pipeline (b) Time stealing in router pipeline

(c) Time stealing with a 2-phase clocked pipeline

Figure 3.3. Time stealing in a generic router pipeline with a 2-phase clock.

A generic on-chip router pipeline stage delays are quite imbalanced unlike the processor pipeline [2, 6, 31, 88, 92]. In order to boost the router frequency from the nominal frequency, RAFT employs time stealing techniques, where a slower stage in the router gains evaluation time by *stealing* it from successive or previous router pipeline stages. A cycle may steal time from subsequent stages by delaying the triggering edge of the clock to all the subsequent latches.

The baseline router considered is a 2-stage speculative router in which the first stage performs the routing computation (RC), the virtual channel allocation (VA) and the switch allocation (SA) in parallel and the second stage is the switch transfer (ST) as shown in the Figure 3.3(a). The link traversal (LT) stage takes an additional cycle. The router stage delays (shown in Figure 3.3(a)) are obtained after synthesis using the Synopsys Design Compiler, and the LT stage delay is

Table 3.3. On chip regulator power consumption

Output Voltage (V)	Power overhead (mW)
1	52.3
0.9	41.5
0.85	37.9
0.8	34.1

obtained using wire models from PTM [93]. A non-overlapping symmetric two-phase clock is used for boosting the router frequency (shown as C1 and C2 in 3.3(c)). In the 2-stage router, since the VA stage (454ps) is the bottleneck, it steals time from the ST stage. Since the time required by the VA stage is greater than the slack available in the ST stage, ST stage will need to steal time from LT as shown in Figure 3.3(b). Consequently, the active clock edge of C1 and C2 are delayed for both first and second stages (shown in Figure 3.3(c)). Let the new enhanced clock time after time stealing be T_c ($=\frac{T_{VA}+T_{ST}+T_{LT}}{3}$). The active clock edge for the VA stage is delayed by $S_1 = T_{VA} - T_c$. This is the extra time that is required by VA (stolen from ST and LT put together). The slack time ST can provide is $S_2 = T_c - T_{ST}$. The remaining time of $S_1 - S_2$ is stolen from the LT stage by the ST stage. Thus, the clock edge to the ST stage is delayed by $T_{VA} + T_{ST} - 2 * T_c$.

3.3.3 Hardware Support for Frequency Adaptation

Asynchronous Communication: In order for the network routers to operate at different frequencies, they should be able to communicate asynchronously with each other. The router control logic, switching logic and arbitration logic remain unaffected. This architecture essentially *mimics* a Globally Asynchronous Locally Synchronous (GALS) design within the network. To support the communication between routers operating at different frequencies, RAFT utilizes the dual clock I/O buffer design from [64] for the router buffers. In this design, the buffers use independent read and write clock signals along with control to prevent synchronization problems when read and write pointers approach each other. Note that, with the proposed distributed frequency tuning schemes, the entire network does not become asynchronous, only neighboring routers that are operating at different frequencies communicate asynchronously.

Hardware Implementation: All of the proposed techniques require only buffer

utilization (BU) as the input. This information is already gathered in conventional on-chip networks for the credit-based flow control. The threshold values for triggering the techniques are stored in registers and the comparison operations can be implemented using simple combinational logic.

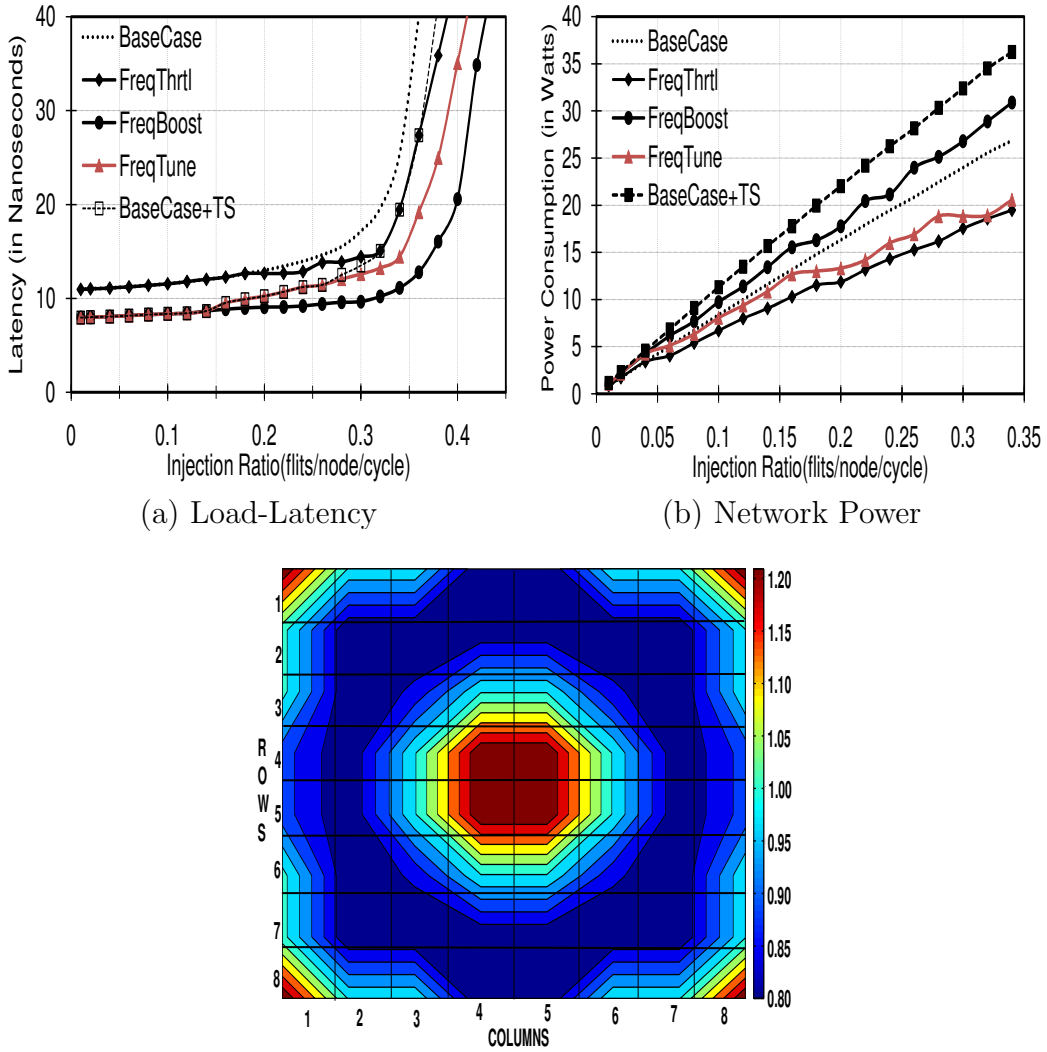
3.4 Performance Evaluation

3.4.1 Experimental Platform

A 64-node network (laid out as an 8x8 mesh) is used as an experimental platform to demonstrate the RAFT concept. The base case router has 5 physical channels (PCs) including the local PE-to-router port and 4 virtual channels (VCs) multiplexed on to each PC. A message (packet) consists of six 128-bit flits and a buffer depth of 4 flits per VC is used. The baseline router design operates at a clock voltage of 1V and 2.20 GHz (2.75 GHz using time-stealing).

3.4.2 Results with Synthetic Workload

This section presents the latency and power consumption characteristics of the three techniques with UR traffic pattern. *BaseCase* in the discussion corresponds to the standard design without any frequency tuning and congestion management. *BaseCase+TS* represents the case, where time-stealing is used on top of *BaseCase* to boost the frequency of the network, but no congestion management techniques are used. Figure 3.4(a) shows the load-latency curves for the three schemes with UR traffic. The saturation bandwidth is measured when the average latency per flit is three times the zero-load latency [60]. All three of the proposed techniques outperform the *BaseCase* as well as the *BaseCase+TS* network, and as expected, *FreqTune*'s performance envelope lies between *FreqThrtl* and *FreqBoost*. At low load, *FreqTune*'s performance is similar to that of *FreqBoost*, and at high load, *FreqTune*'s performance gets close to *FreqThrtl*. On an average, there is 24% (up to 31%) increase in throughput using *FreqTune* when compared to the *BaseCase*, and up to 21% increase in throughput when compared to the *BaseCase+TS*. As the network load increases leading to an onset of congestion, *FreqTune* switches to the *FreqThrtl* mode, where the frequency of a congested router is boosted and the



(a) Load-Latency

(b) Network Power

(c) Contour plot

Figure 3.4. Performance and network power consumption with UR traffic and a snapshot of relative frequencies of routers at high load.

neighboring routers are throttled. The congested router can thus service its flits at a faster rate compared to the rate at which it is receiving flits from neighbors. At high injection rates, while some of the routers are congested, some routers are still un-congested. Thus, using *FreqTune*, the congested regions use *FreqThrtl* and the un-congested regions use *FreqBoost*. Figure 3.4(c) shows a contour plot of a snapshot with UR traffic depicting relative operating frequencies of the routers compared to *BaseCase* at high load (0.36 injection rate). The frequencies range from the lowest possible value ($0.8*F_{base}$), shown as deep-blue color in the plot, to

the highest possible value (F_{boost}), shown as dark-red color. Since the routers in the center of a mesh are more congested with X-Y traffic, they operate mostly at F_{boost} ($FreqThrtl$), while the routers around the center operate on an average at lower frequencies. The routers at the periphery are less congested, and hence, operate at F_{boost} ($FreqBoost$). This contour plot demonstrates the flexibility and the distributed frequency modulation capability of $FreqTune$ in adapting to network congestion. Comparison of all the three proposed techniques with *BaseCase+TS* shows the benefits of a prudent congestion management scheme through frequency tuning when compared to a simple increase in frequency of the network.

Figure 3.4(c) shows the absolute power consumption with UR traffic for the three schemes. *FreqBoost* does not give any power benefit since it starts at a 25% higher frequency and gradually reduces the frequency based on the congestion scenarios. At low injection rates,*FreqTune* exhibits identical power behavior as that of*FreqBoost*, and as the load increases, power saving becomes positive with an average reduction of 14.5%. Figure 3.5 shows the savings in power and EDP with the three techniques for UR traffic. In addition, the overhead (in terms of percentage network power w.r.t. BaseCase) due to voltage-frequency transitions of the additional controllers in the network for congestion management is shown. In terms of the EDP metric, *FreqThrtl* provides no gain at light load but at higher load it provides the maximum EDP benefit. Both *FreqBoost* and *FreqTune* start with more than 20% saving in light traffic conditions primarily due to latency reduction. At higher network traffic, when buffer utilization increases beyond the specified thresholds, $FreqTune$ starts distributed throttling of routers using the $FreqThrtl$ technique, leading to reduction in EDP. By throttling the upstream router, the peak power around hot-spots is lowered as the crossbar traversal in the upstream router, traversal in the link to the downstream router and buffer write in the downstream router are all throttled. At medium load, overall buffer utilization lies in the intervals 0.55 to 0.35 and $FreqTune$ dynamically transitions between $FreqThrtl$ and $FreqBoost$. This effect is seen as fluctuations in EDP reduction during medium load in the network.

Similar trends in performance, power and EDP savings are also observed with other traffic patterns like bit-compliment, transpose, self-similar (except nearest-neighbor). A detail analysis with these traffic patterns can be found in [30, 94].

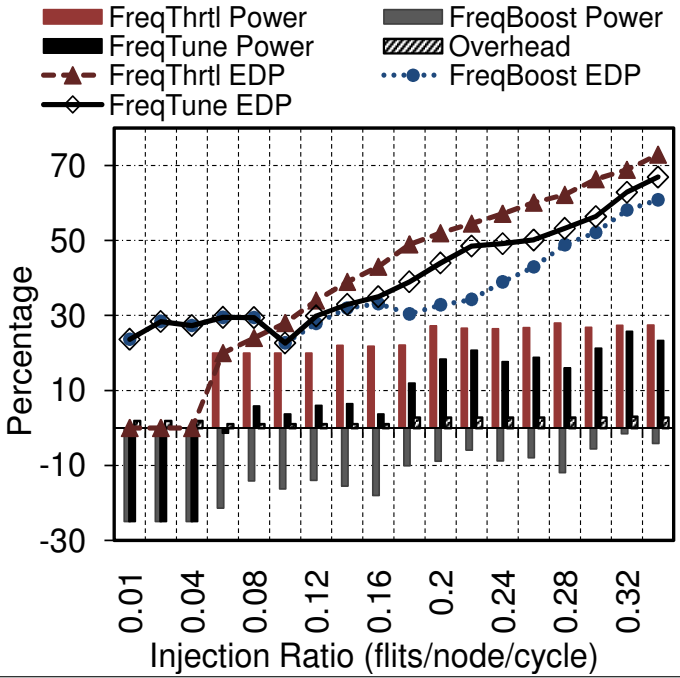


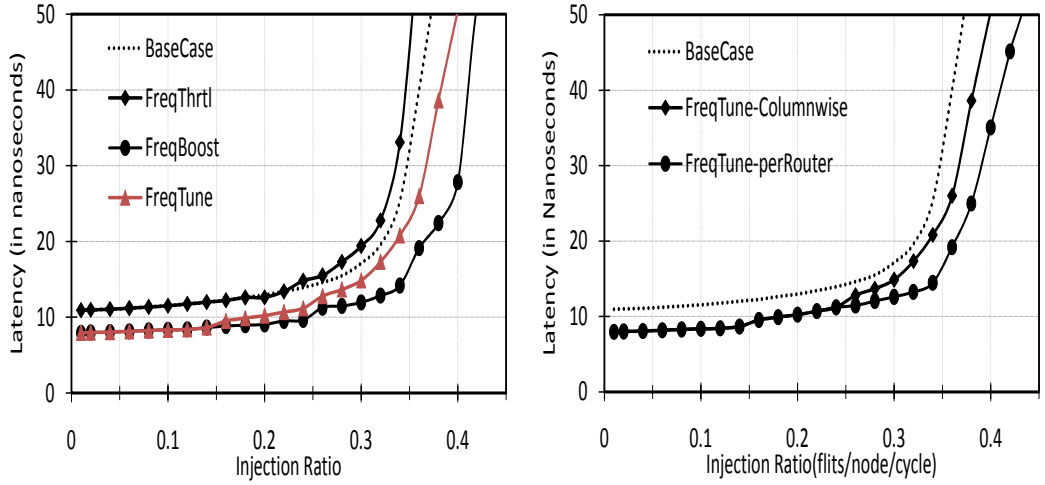
Figure 3.5. Power and EDP reduction with UR traffic for *FreqTune* and controller power overheads.

Overall, *FreqTune* is able to satisfy the two requirements discussed earlier - reduce latency at low load and conserve power at high load. However, depending on the specific requirements of a system, a designer can choose to implement any of the other two techniques if only performance enhancement or power conservation is a concern.

3.4.3 Results with Column-wise Controllers

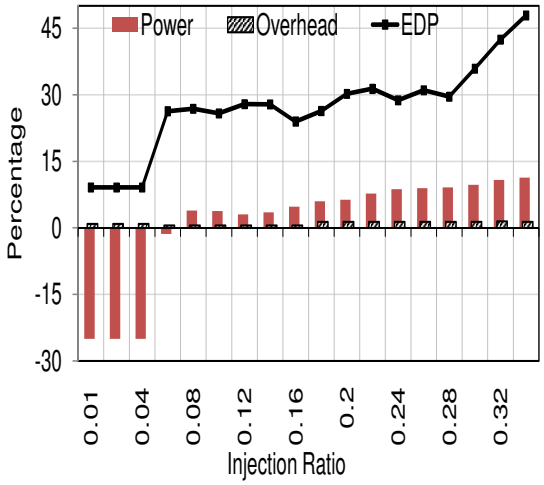
To further reduce the area and power overheads in having a per-router DVFS controller, this section investigates the performance and power behavior by reducing number of DVFS controllers. A column-wise scheme is chosen since evaluations show that such a scheme is quite agnostic toward many routing algorithms (e.g. adaptive, X-Y,Y-X routing). A network designer can choose many other schemes, e.g. having one controller for the routers in the center of a mesh network that have similar utilization with X-Y traffic (as is evident from the contour plot in Figure 3.4(c)) or have one controller for routers around a hot-spot region (e.g. around a memory controller), etc. In this study, the number of controllers are

reduced from one per router to one controller per half-column, i.e. a column in the network is divided into two halves and a controller is assigned to each half. A controller in a particular half now modulates the frequency of all the routers in half of the column. This reduces the performance gains compared to the fine-grained frequency modulation. Thus, compared to 64 controllers, 16 controllers (2 controller per column), reduce the area and power overheads by 4x.



(a)UR performance with Column-wise *Freqtune*

(b) Comparison of per-router DVFS vs. Column-wise DVFS



(c) UR Power with Column-wise *FreqTune*

Figure 3.6. UR traffic performance using Column-wise controllers.

Figure 3.6(a) shows that *FreqThrtl* performs worse than the *BaseCase* with column-wise controllers. This is because of the coarser granularity of frequency

adjustments and adaptivity. However, *FreqBoost*'s performance is not affected as much because of using high-frequency routers and thus, *FreqTune*'s latency curve is still better than the *BaseCase*. Figure 3.6(b) shows how latency of *FreqTune* is affected when using column-wise controllers compared to *FreqTune* with per-router controller. Figure 3.6(c) shows the percentage reduction in power and EDP with *FreqTune* using column-wise controllers over the *BaseCase*. With column-wise controllers, there is on an average 12% reduction in latency, 13% savings in network power and 27% reduction in EDP with *FreqTune*. Hence, although congestion management with column-wise controllers now happens at a much coarser granularity, there are still savings in power and EDP without sacrificing performance.

3.4.4 Results with Application Benchmarks

This section examines the impact of the frequency tuning schemes on application performance for a 64-tile CMP. Each tile in the CMP consists of a core with a private write-back L1 cache and a L2 cache bank. The detailed experimental set-up is mentioned in similar to that discussed earlier in Chapter 2.

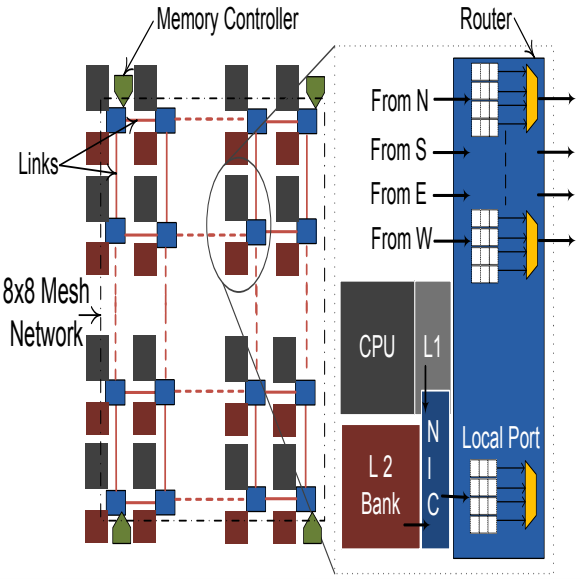


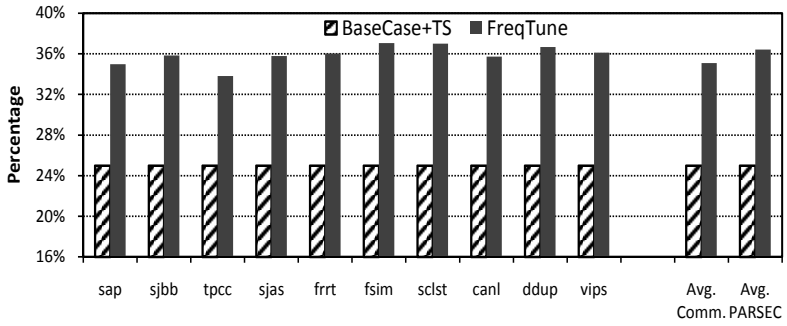
Figure 3.7. CMP layout.

Figures 6.6(a)-(d) show the results with the application suite. Having a faster network with congestion management, reduces the average load/store latency and

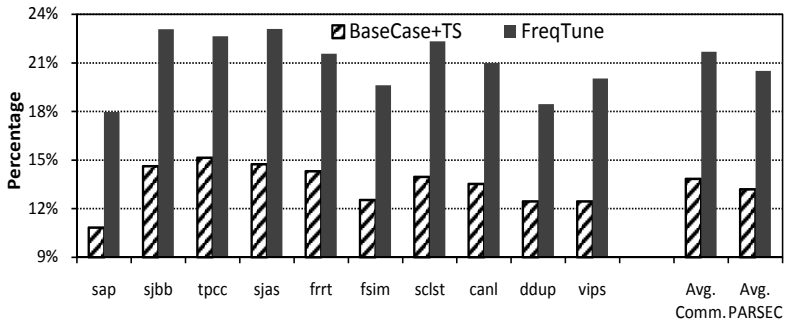
directly impacts system performance in terms of instructions per cycle (IPC). On an average, there is 35.7% (up to 37.16% with `facesim`) reduction in latency (shown in Figure 6.6(a)), which translates to 21.2% (up to 23.1% for `sjas`) improvement in IPC (Figure 6.6(b)) across all benchmarks. Correspondingly, there is on an average 15.17% (up to 19.57% for `sap`) reduction in network power (Figure 6.6(c)) for these applications. Figure 6.6(c) shows the overhead due to the controllers deployed in the network for the *FreqTune* scheme. *BaseCase +TS* is omitted in this plot since it always consumes 25% more power, and thus, incurs higher network power. There is a modest 2.4% increase in power overhead, however, there is up to 23.1% increase in IPC. Further, the overheads in power consumption due to the controllers are counter-balanced by the 15.17% savings in network power. Even with the column-wise scheme, there is on an average 16% reduction in latency, translating to 12% improvement in IPC and 7.23% reduction in power (Figure 6.6(d)).

3.5 Chapter Summary

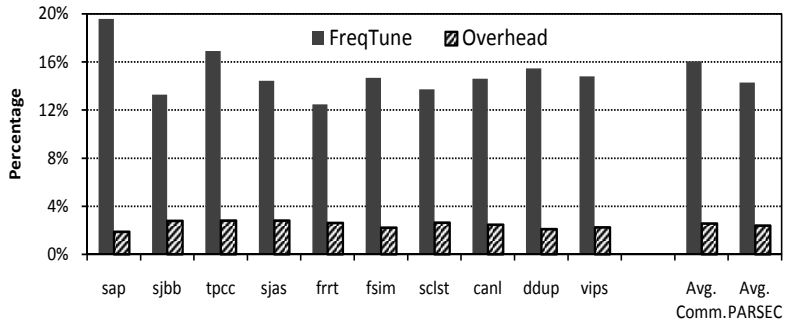
NoC power is going to dominate design decisions in future systems. Additionally, performance implications of NoCs are likely to influence CMP/SoC design decisions as well. Towards this end, this chapter proposes a variable frequency router architecture, called RAFT, for dynamically controlling the performance and power behavior of on-chip interconnects by effective congestion management. Three dynamic congestion management techniques are discussed - *FreqBoost*, *FreqThrtl* and *FreqTune*. *FreqBoost* gives the highest performance throughout but consumes more power at low load. *FreqThrtl* gives the best power behavior across all injection rates. *FreqTune* is a hybrid technique that uses *FreqBoost* at low load to enhance performance and *FreqThrtl* during high load to minimize network congestion and power consumption. Furthermore, these techniques are much more effective than using adaptive routing, simple router frequency scaling for performance enhancement, and peak power reduction techniques.



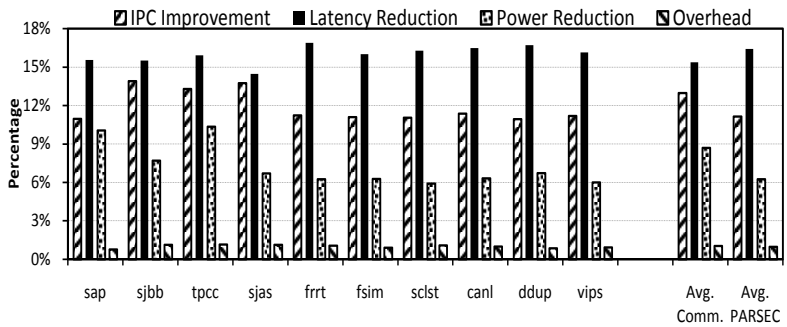
(a) Latency reduction



(b) IPC improvement



(c) Power reduction



(d) FreqTune with Column-wise scheme

Figure 3.8. Application level benefits with RAFT.

Chapter 4

HeteroNoC: Heterogeneous On-Chip Interconnects for CMPs

While the previous chapter introduced a scheme where, individual routers operated at frequencies that are proportional to the load and exploited the non-uniform buffer utilization in routers, this chapter exploits the inherent heterogeneity in both buffer and link utilization in the routers. The approach presented in this chapter is quite agnostic to the applications hosted on the cores in the multicore chip and is a bottom-up approach for designing heterogeneous networks.

4.1 Introduction

An $N \times N$ mesh interconnect is the most widely studied NoC topology due to its scalability, regularity and ease of implementation in silicon. While different variations of the mesh architecture have been proposed recently [10, 13, 22], all of them are centered around the same homogeneous router design, where all routers are provisioned with the same silicon real estate. On the contrary, it was shown earlier in Chapter 1 that the center of a mesh usually gets more congested because it handles more traffic compared to the edge routers with deterministic X-Y routing. It was demonstrated in that chapter how the buffer and link utilizations varied across the central and peripheral routers in a mesh network with deterministic X-Y routing, and how this variation in utilization is an inherent artifact of any non-edge symmetric network.

These observations of non-uniform resource utilizations compel a re-thinking of the traditional mesh interconnect design. Intuitively, since the central routers in a mesh handle more traffic compared to the peripheral routers, performance can be enhanced by allocating more buffer and link bandwidth to the center routers compared to the peripheral routers. The same argument may also be true for the corner routers. This implies that, different types of router architectures could be beneficial: the peripheral routers can employ a small router with fewer buffers (virtual channels (VCs)) and narrow link width, while the central and other highly utilized routers employ relatively large number of VCs and wider link width (*big* routers). This essentially moves the design of the interconnect from a homogeneous NoC to heterogeneous NoC domain, an emerging area that has remained unexplored in the context of performance-power trade-offs.

Based on this, the proposal in this chapter argues against the rationality of uniform resource distribution across all routers in the mesh network and advocates for heterogeneous resource redistribution in the NoC. The heterogeneous NoC concept is further supported by the fact that future multicores/SoCs will have heterogeneous cores/components [95–97]. However, it is not intuitively clear if such heterogeneous cores would benefit more from a heterogeneous NoC. Towards this end, this chapter examines the rationality of resource redistribution and proposes a heterogeneous mesh architecture, called HeteroNoC, for optimizing both performance and power. In this context, this chapter attempts to answer the following questions: *(i) Should a heterogeneous NoC be designed with two types (small and big) of routers? (ii) If yes, how many such big/small routers does the NoC need and how to redistribute the buffer and link width between these routers without changing the original bisection width and buffer resources? (iii) Is there an optimal placement of big routers that would maximize the performance and power benefits compared to the baseline homogeneous mesh? and (iv) How else can a HeteroNoC design be leveraged to achieve better performance/power in a CMP?*

Starting with an $N \times N$ homogeneous mesh network, this chapter explores the design space of the proposed HeteroNoC with two types of routers with different link widths and buffer organizations, and placement of such routers, while keeping the bisection bandwidth the same.

Table 4.1. Comparison of homogeneous and heterogeneous routers

		Power	Area	Frequency
Homogeneous net.	3VCs/5 buffer depth/192b	0.67W	$0.290mm^2$	2.2 GHz
Heterogeneous net.	2VCs/5 buffer depth/128b (small router)	0.30W	$0.235mm^2$	2.25 GHz
	6VCs/5 buffer depth/256b (big router)	1.19W	$0.425mm^2$	2.07 GHz
Total buffers in **homogeneous network** = 64 (routers) * 3 (VCs) * 5 (PCs) * 5 (buffer depth) = 4800 buffers @ 192 bits/buffer = 921,600 bits				
Total buffers in **heterogeneous network** = 48 (routers) * 2 (VCs) * 5 (PCs) * 5 (buffer depth) + 16 (routers) * 6 (VCs) * 5 (PCs) * 5 (buffer depth) = 4800 buffers @ 128 bits/buffer = 614,400 bits (33% reduction over the homogeneous case)				

4.2 HeteroNoC Architecture

Without loss of generality, a 5x5 baseline router with 3VCs per physical channel (PC), and 192b links/flit-width is chosen as the baseline design. Table 4.1 shows the architectural level details of the homogeneous routers. The baseline 8x8 homogeneous network is configured using these routers. For keeping the design space of the HeteroNoC simple, the network is composed of two types of routers - *small* power efficient routers, and relatively *big* performance boosting routers. Since buffers and links are the primary building blocks of a 5x5 router, this proposal considers redistributing these resources to leverage non-uniform resource utilization in the network. There were two primary constraints in designing the small and big routers - *the total number of VCs and network bi-section bandwidth is kept the same in the baseline and heterogeneous networks.*

HeteroNoC Link Re-distribution: Since it was decided to use only two types of routers, a natural choice was to have two kinds of links connecting these routers. Link width significantly affects router power (precisely buffer and crossbar power), and since the goal is to design the small routers to be power efficient, their crossbar width (and hence, link width) should be less than that of big routers. This design principle and constant bisection bandwidth constraint led to the formulation of the link width equation as: $W_{homo} \times n = W_{hetero} \times N_{narrow} + 2W_{hetero} \times N_{wide}$; where W_{homo} is the link width in the homogeneous network, n is the number of links crossing the bisection-cut in one direction in the homogeneous network, W_{hetero} is the link width of small routers, and N_{narrow} and N_{wide} are the number of narrow and wide links, respectively, crossing the bisection-cut in the heterogeneous network. In this case, with an (8x8) baseline network with 192b links, and considering an equal size (= 8x8) heterogeneous network bisection-cut to consist of equal number

of wide and narrow links, one gets: $192 \times 8 = W_{hetero} \times 4 + 2W_{hetero} \times 4$; implying $W_{hetero} = 128$. Hence, narrow links are chosen to be 128b and wide links to be 256b.

One other design guideline that dictated wide-link width to be double of the narrow-link width is that, narrower links (compared to baseline) in small routers dictate the flit-width to be less than the baseline network (128b vs. 192b), and hence, transfer latency increases in the heterogeneous network. Later in this chapter, a novel scheme is proposed to merge two flits together and transmit them simultaneously for reducing the transfer latency (discussed later in Section 4.3.2). This simultaneous transmission of two flits, requires the link width of big routers to be double of the flit-width.

HeteroNoC Buffer Re-distribution: Buffers consume about 35% of router power [35,98] and hence, having more VCs in a router increases the network power consumption although it helps in enhancing performance. In HeteroNoC, more VCs are allocated to big routers and fewer VCs to small routers by stripping a few VCs from the small routers, while keeping the total number of VCs constant with respect to the homogeneous network. Experimental analysis shows that, having 1VC/PC in a router degrades performance due to the inability of the router in multiplexing packet flows. Hence, 2VCs/PC are employed in the small routers and 3 VCs are stripped from three baseline routers (to make them small routers) and allocated to another baseline router to make it a big router with 6VCs/PC. The power and area profiles of the big and small routers are shown in Table 4.1.

Number of Small and Big Routers: The design goal is to keep the network area and power consumption of the heterogeneous network less (or equal) than that of the homogeneous network. This constrained the number of big routers in the heterogeneous network. To calculate the number of small and big routers in the network, consider the inequality: $0.67 \times N^2 \geq 0.3 \times n_s + 1.19 \times (N^2 - n_s)$; where 0.67, 0.3 and 1.19 are the power consumption (in Watts) of the baseline, small and big router respectively, N^2 is the total number of routers in the mesh network and n_s is the number of small routers[1]. Simplifying the inequality gives

[1]This inequality serves as an empirical guideline. The power profile of each router changes with its utilization levels, and the numbers present in the inequality represent the power profile of a router at 50% activity factor. However, ***all simulations use actual utilization*** of the router to calculate its power consumption.

us $1.71 \geq \frac{N^2}{n_s}$.

In HeteroNoC design, $N = 8$, implying $n_s \geq 37.4$. Thus, should be a minimum of 38 small routers in the network to guarantee that the heterogeneous network is power efficient than the homogeneous network. Symmetry considerations led to the selection of 48 small routers and 16 (= 2N) big routers, where for every 3 small routers, there is a big router in the network (this was the reason for stripping a VC from three baseline routers to make them small routers and allocating the 3 VCs to a big router).

Placement of Routers in HeteroNoC: The next obvious question is how to place these two types of routers in the HeteroNoC. Here, this chapter discusses two design options - first, the design space with the big and small router is considered by redistributing the buffers only 6VCs/PC and 2VCs/PC for big and small routers, respectively) without changing the original link width (192b). Next, the design space is analyzed by redistributing both the buffer and link width for the two routers (6VC/PC and 256b link width for big routers, and 2VCs/PC and 128b width for small routers). Figure 4.1 shows various layouts considered in the evaluations. The buffer only redistribution configurations are annotated with a +B extension and combined buffer with link redistribution with a +BL extension after the names of the configurations. Figure 4.1 (a) shows the baseline network layout and Figures 4.1 (b), (c) and (d) show the three network layouts with only buffer re-distribution. Figures 4.1 (e), (f) and (g) depict the HeteroNoC layouts with combined buffer and link redistribution. These heterogeneous configurations were the best six configurations among thousands of configurations evaluated[2] , and hence, for experimental evaluations, only these six layouts are discussed.

The **Center+B** (Figure 4.1 (b)) layout is motivated by non-uniform buffer utilization (Figure 1.1) in the network, where more buffers are apportioned in the central regions of the network and buffers in the peripheral routers are reduced.

[2]Apart from the heterogeneous network configurations discussed here, a *comprehensive design space exploration* was conducted on a smaller network size and the results were extrapolated for an 8x8 network size. Specifically, with a 4x4 network, all possible placements of *(4 small, 12 big)*, *(6 small, 10 big)* and *(8 small, 8 big)* routers were evaluated. In this case case, the number of simulations were tractable since, symmetry and constant bisection bandwidth constraints limited the number of cases to be evaluated to few thousands (1820, 8008 and 12870 configurations in each case respectively). Such an exhaustive analysis in an 8x8 network is infeasible because the design space bloats (e.g. the number of ways to place 48 small routers and 16 small routers in a 64 node network is $\binom{64}{48}$ = 4.89E+14).

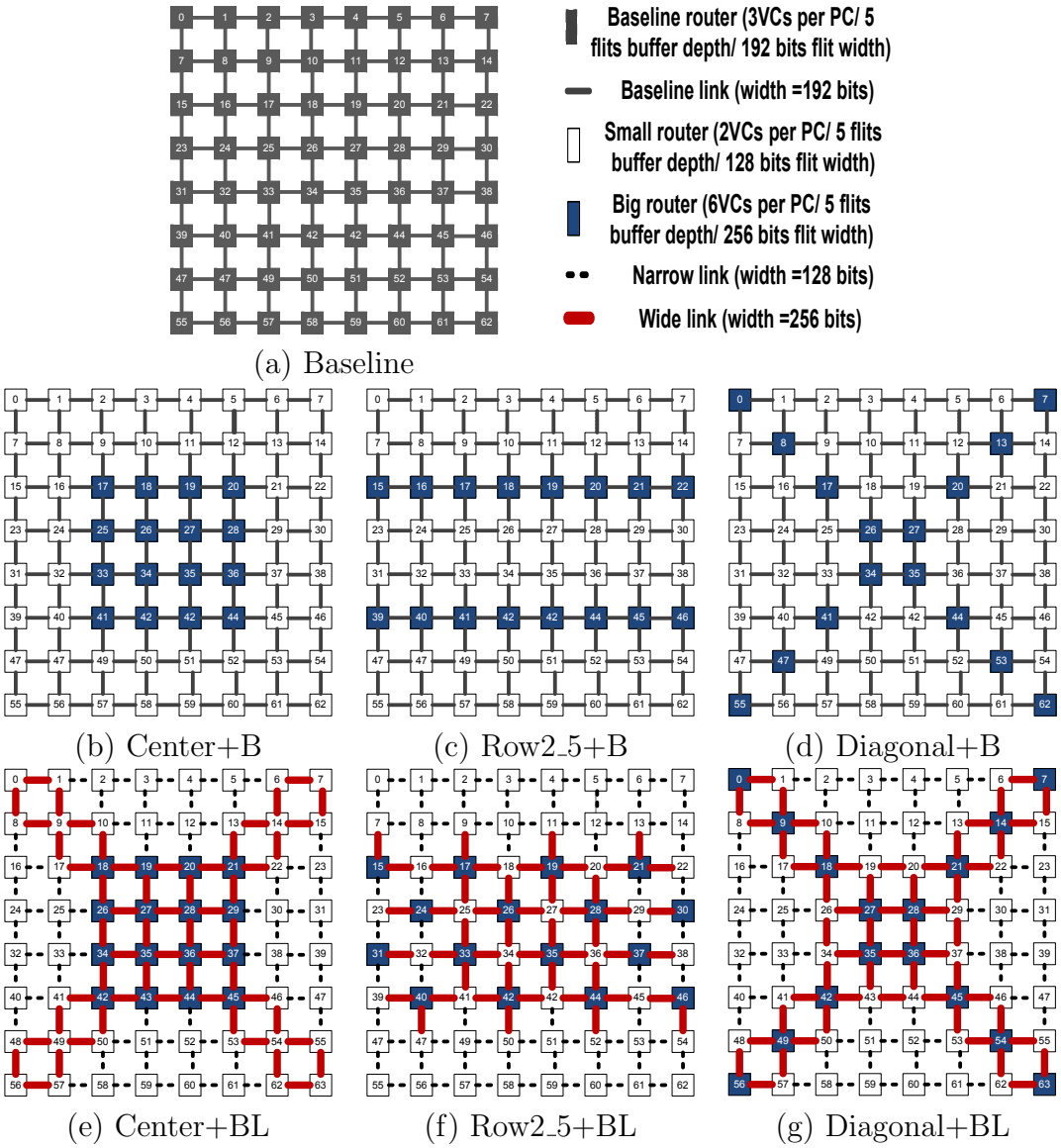


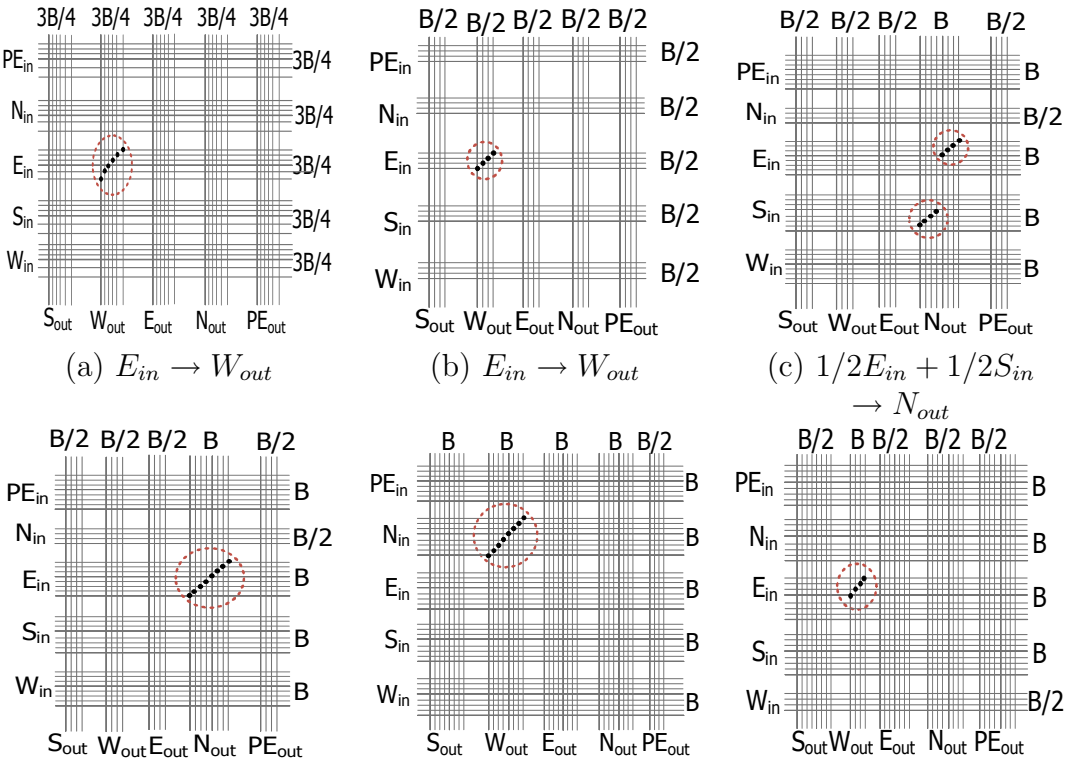
Figure 4.1. Various layouts of an (8x8) HeteroNoC

In the **Row2_5+B** layout (Figure 4.1 (c)), the big routers are arranged in the second and fifth row of the mesh network for minimizing the average hop count to reach a big router. In the **Diagonal+B** configuration, the big routers are arranged along the network diagonals to spread the big routers as far as possible and also to have a few big routers in the center of the mesh network. Particularly, placing a few big routers in each row and column helps most of the flows to use the big routers. Also, having big routers at the corner of the mesh network helps alleviate

resource contention in these routers. With both the buffer and link re-distribution approach, more buffers and links are allocated to the highly utilized routers in the network. **Center+BL**, **Row2_5+BL** and **Diagonal+BL** are similar to the Center+B, Row2_5+B and Diagonal+B layouts, respectively, with the exception that the big routers have wider links (256b) and small routers have narrow links (128b).

4.3 HeteroNoC Design Details

4.3.1 Impact on Crossbar Design



(a) $E_{in} \rightarrow W_{out}$ (b) $E_{in} \rightarrow W_{out}$ (c) $1/2E_{in} + 1/2S_{in} \rightarrow N_{out}$

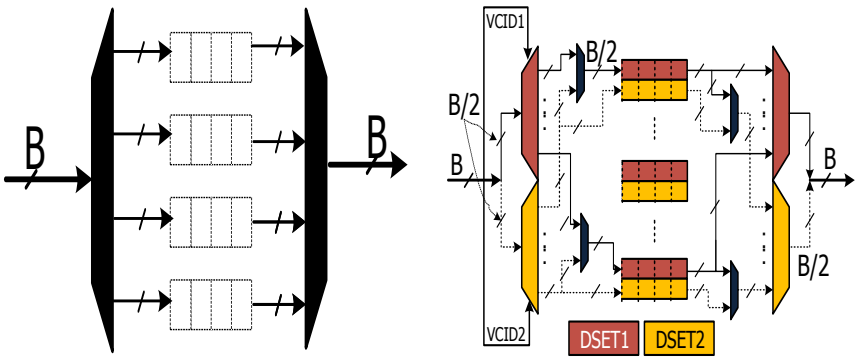
(d) $E_{in} \rightarrow N_{out}$ (e) $N_{in} \rightarrow W_{out}$ (f) $1/2E_{in} \rightarrow W_{out}$

Figure 4.2. Crossbar architecture details (B=256 bits).

HeteroNoC design requires minor modifications to the buffering, switch arbitration and crossbar stages of the routers. In the HeteroNoC architecture with combined buffer and link re-distribution, both 128b and 256b links are used with flit size being 128b. Hence, when communication takes place between a small and

a big router (or two big routers) between which a 256b link exists, two 128b flits can be combined to be simultaneously sent over the wider link. This is depicted in Figures 4.2(b)-(f), where the possible crossbar architectures for HeteroNoCs are shown. The figure shows in each case a mapping (shown by the highlighted circle) from i/p port(s) to an o/p port; (a) Baseline xbar (192b wide); (b) Small router (128b wide) when connected to small routers on all sides e.g. router 11 in Fig. 4.1(g); (c) Small router when connected to a big router on N_{out} (shows two VC's from two different i/p ports of small router being mapped to N_{out}) e.g. router 52 in Fig. 4.1(f); (d) Small router when connected to a big router on N_{out} (shows two VC's from the same i/p port of small routers being mapped to an o/p port) (e) Big router (256b wide) when connected to big routers on all sides e.g. router 35 in Fig. 4.1(e) and; (f) Big router when connected to a small router on W_{out} e.g. router 26 in Fig. 4.1(e). Essentially, in all the HeteroNoC configurations, when a small and big router communicate, rather than having a single input port mapped to a single output port, there exists a possibility of two input ports to be able to map to a single output port so that flits from these input ports can simultaneously be sent to the output.

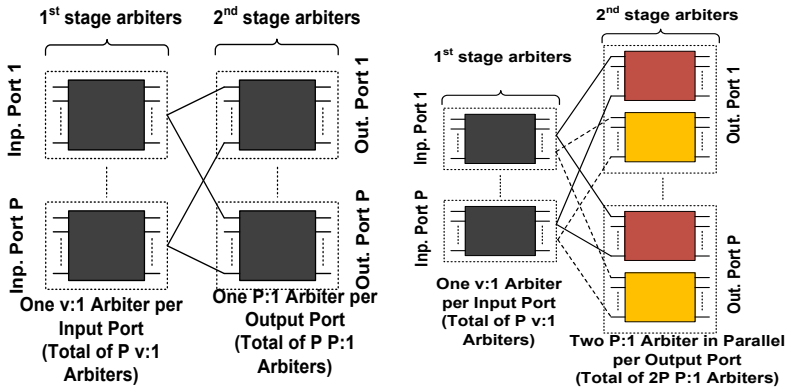
4.3.2 Impact on Buffer Read/Write Stage



(a) Baseline design (b) HeteroNoC

Figure 4.3. Baseline and HeteroNoC input buffer organization (B=256 bits).

In HeteroNoC design, a 256b link exists between a small router and a big router and between two big routers. Figure 4.3 (b) shows the modifications done to the buffering stage in the router compared to a standard design shown in Figure 4.3



(a) Baseline design (b) HeteroNoC

Figure 4.4. Baseline and HeteroNoC SA stage organization

(a), where B is the channel width. Our design is similar to the XShare technique proposed in [10], where two *small* flits (e.g. coherence traffic whose flit sizes are very small compared to data flits) are combined and sent over the link to the next router. However, in HeteroNoC, whenever possible, two 128 data flits can be combined and transmitted simultaneously over the wider link.

4.3.3 Impact on SA Stage

Figure 4.4(a) shows the switch arbitration (SA) stage of the baseline router. The SA stage can be broken into two sub-stages. In the first sub-stage, (SA stage 1), a v:1 arbiter for each input port chooses which of its virtual channels (v_i) can bid for an output port. In the second stage, (SA stage 2), a p:1 arbiter for each output port (p_o) chooses one of the input ports(p_i). Hence, at the end of the SA stage, a (p_i, v_i) pair is chosen for each output port p_o and a crossbar mapping is enabled between p_i and p_o. Two 128 bit flits can be combined in the HeteroNoC only when - (a) Two input virtual channels (VCs) within a *single* input port request the *same* output port and (b) Two input VCs from *different* input ports request the *same* output port. In case (a), a combined request for both input VCs can be sent to SA stage 1, and if the combined request wins in the SA stage 2, both the flits can be sent together. In case (b), the requests can be combined in SA stage 2.

4.3.4 Impact on Router Frequency

A critical parameter that is affected as a result of increasing the buffers and the crossbar width in larger routers is the router operating frequency. Increasing the number of VCs in a router increases the cycle time, and hence, reduces the router frequency [30]. In HeteroNoC design, the VA stage is the dominating stage and the frequency of the big routers consisting of 6 VCs is reduced by 6% compared to the frequency of the baseline router (=2.2 GHz) with 3 VCs. On the other hand, the frequency of the small routers with 2 VCs is boosted by 2% compared to the baseline frequency. For simplicity, in all the analysis in this chapter, the heterogeneous network is considered to be operated at the worst case operating frequency, i.e. frequency of the big routers.

4.3.5 Impact on Area

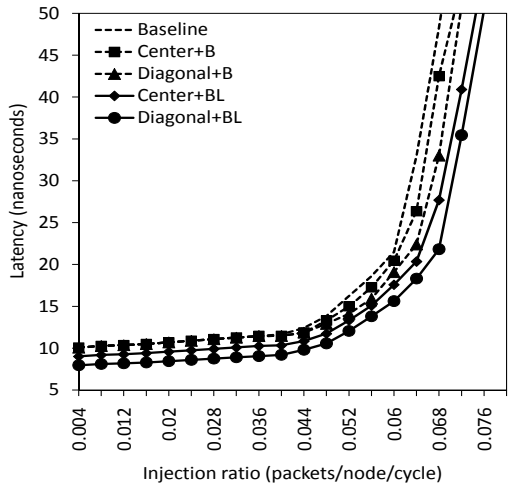
Analysis, using Synopsys design compiler, shows that the area of the big routers increase by 46% and the area of the small routers decreases by 18% compared to the baseline design. Such heterogeneous router layouts can be handled with floor-planning constraints that combine multiple processors and router blocks to create larger tiles. Hence, it is assumed that the increase in area of a router can be accommodated within the *tile* width of the core to which the router is connected and, thus, it does not increase the cycle time of the link stage. In fact, factoring out link area (since they are the same in both designs), the total area of HeteroNoC routers is 18.08 mm^2 (=0.235 mm^2*48 + 0.425mm^2*16) which is less than the total router area of the homogeneous network (= 18.56mm^2 = 0.290mm^2*64 routers).

4.4 Experimental Results

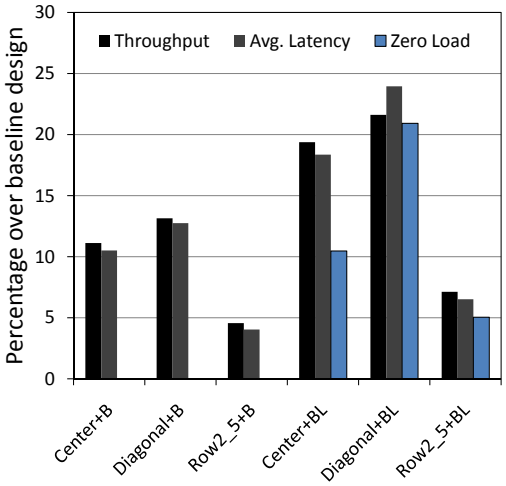
4.4.1 Experimental Setup

Most experiments are carried out with a 64-node network laid out as an 8x8 2D-mesh. The experimental setup is similar to that mentioned in Chapter 2. A data packet consists of 1024b (= cache line size) and is decomposed into 6 flits in the baseline design (with 192b buffer/crossbar/link width) and 8 flits in the HeteroNoC

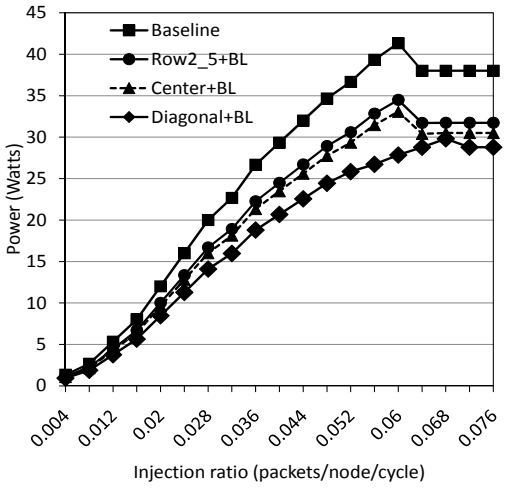
design (with 128b buffer/crossbar width). Address packet is composed of 1 flit in both cases.



(a) Load-latency



(b) Throughput/latency

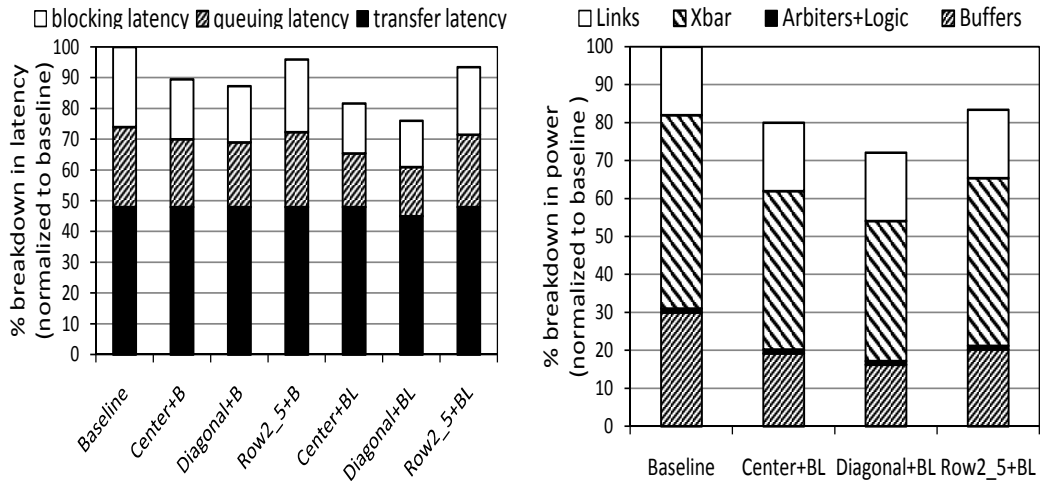


(c) Power

Figure 4.5. Performance and network power behavior with UR traffic.

4.4.2 Results with Synthetic Traffic

Figure 4.5(a) shows the load-latency plots of HeteroNoC configurations with the UR traffic pattern. To preserve clarity, the performance curves for Row2_5+B and Row2_5+BL are not shown. However, Figure 4.5(b), summarizes the benefits across all six configurations with respect to the homogeneous design. As can be



(a) Latency breakdown (b) Power breakdown

Figure 4.6. Latency and power breakdown.

seen in Figures 4.5(a) and (b), all HeteroNoC configurations outperform the baseline design. The performance improvements with a simple buffer-only redistribution brings on an average 9% improvement in latency. Center+B and Diagonal+B configurations, in particular, outperform Row2_5+B configuration. In Center+B, more buffers are allocated to the central routers in the mesh and this helps to reduce latency by 10.5% and improve throughput by 11%. The Diagonal+B configuration helps packet flows in the center and corners of the mesh and shows an additional 3% latency reduction and 4% throughput improvement over the Center+B layout. Although Row2_5+B helps reduce latency by reducing the average hop count to a big router, not having big routers at the center of the mesh in this layout, only leads to 4% reduction in latency and 4.5% increase in throughput. This analysis shows that the *placement* of big and small routers is non-trivial and a poor layout choice may lead to network under-performance.

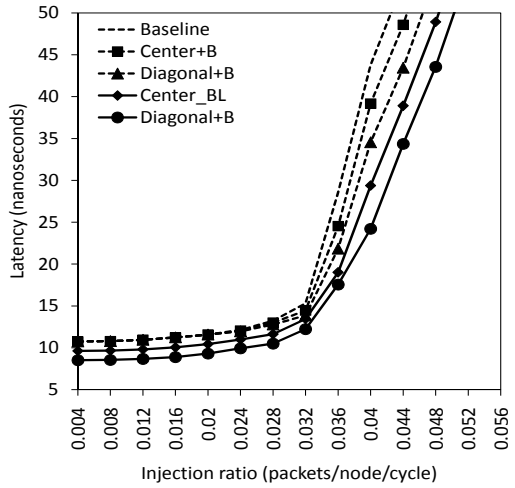
Diagonal+BL is the best HeteroNoC configuration compared to all cases, and reduces latency on an average by 24% and increases throughput by 22% over the baseline design with UR traffic. With combined buffer and link re-distribution in the network, there is on an average, 12% reduction in zero-load latency with the HeteroNoC configurations. Reducing zero-load latency helps lower end-to-end latency at low loads, and hence, HeteroNoC design helps reduce latency at both low and high injection rates. Figure 4.5(c) shows the power curves with the three

optimal HeteroNoC designs for UR traffic. HeteroNoC reduces power by balancing the number of big and small routers in the network. Buffer only redistribution does not reduce the overall power in the network significantly compared to baseline design (hence omitted in all the power plots), since, with this kind of redistribution, neither crossbar width nor buffer resources are reduced in the network. However, with the combined buffer and link re-distribution, the total buffer resources are reduced by 33% (shown in Table 4.1). Additionally, in HeteroNoC design, there are 48 small routers and 16 big routers compared to all 64 medium sized routers in the baseline design. Many small routers help cut down the power consumption further. On an average, there is 21.5% (28% with Diagonal+BL) power reduction for UR traffic across all HeteroNoC designs with the combined buffer and link re-distribution.

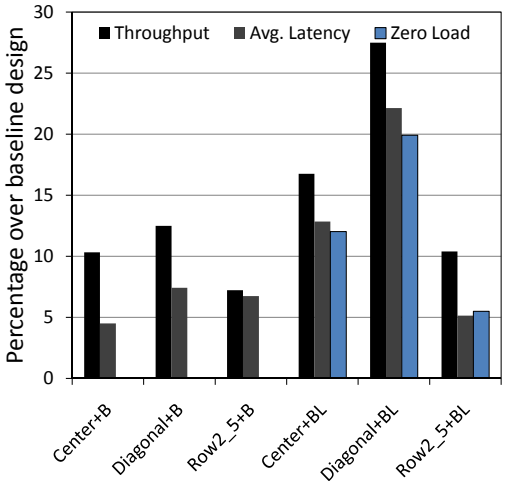
The detailed latency and power breakdowns across all HeteroNoC configurations with UR traffic are depicted in Figure 4.6. The network latency is divided into transfer latency, queuing delay at the source node and blocking delay at intermediate hops. The overall power consumption in the routers is composed of the power consumptions in the router buffers (to read/write), arbiters and logic, crossbar transfer and link transmission. With the HeteroNoC design, latency reduction results primarily by reducing the queuing and blocking latency (Figure 4.6 (a)). Wider links in big routers and combining two flits for simultaneous transmission helps reduce these latency components. Power reduction comes primarily from reduction in buffers (33%) and crossbar power (Figure 4.6 (b)).

Figure 4.7 shows the load-latency and power consumption curves with TP traffic. The performance and power trends are almost similar to UR traffic pattern. Additionally, the load-latency and power consumption curves with bit-complement and self-similar traffic patterns (not shown here due to brevity) are also similar in trend to those obtained with UR traffic pattern, and this re-enforces the fact that our HeteroNoC design is not limited to a particular traffic pattern or layout. Rather, by redistributing resources intelligently, HeteroNoC is able to provide power-performance benefits across various traffic patterns (except NN traffic).

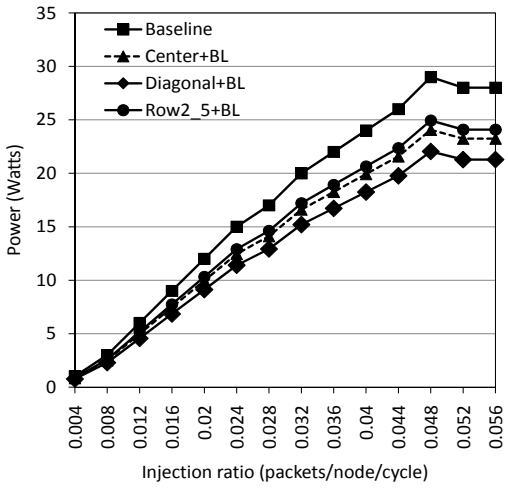
Figure 4.8 shows the performance-power curves with NN traffic. With NN traffic pattern, communication occurs between neighbor nodes and in HeteroNoC design, the buffer and link resources are reduced in most of the network routers. As



(a) Load-Latency



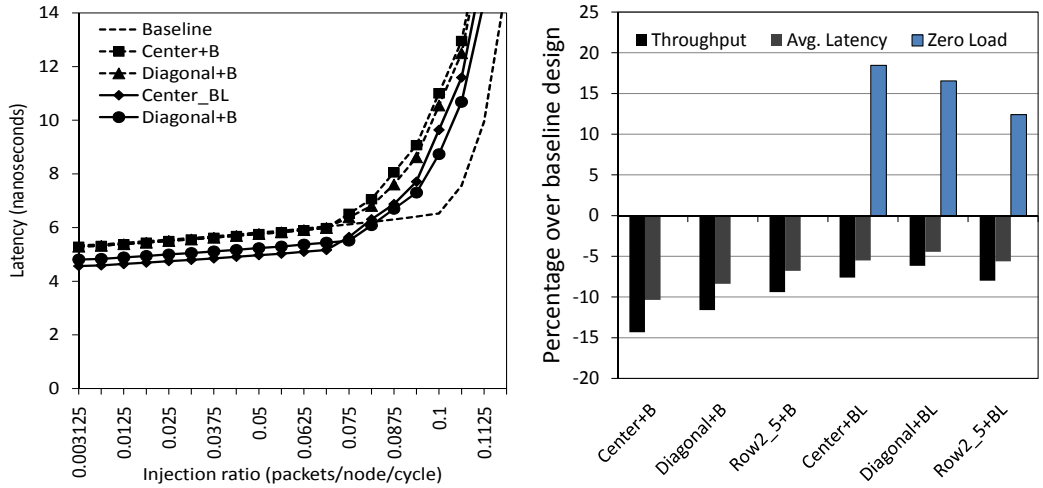
(b) Throughput improvement and latency reduction



(c) Power

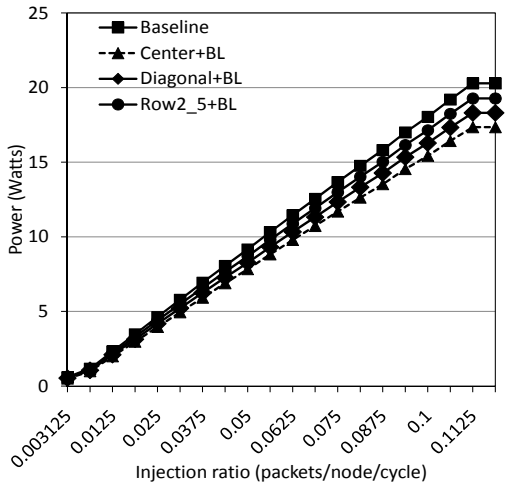
Figure 4.7. Performance and network power behavior with TP traffic.

a result, the network saturates earlier compared to the baseline design. Although, the zero load latency reduces by 16% on an average in the combined buffer and link redistribution designs, the average network latency increases by 7% and throughput reduces by 9.5% as well. Power benefits are also minimal (7%) with NN traffic. With NN traffic, Center+BL performs better than Diagonal+BL, since having *all* big routers in the center aids nearest neighbor communication between the central routers.



(a) Load-Latency

(b) Throughput improvement and latency reduction



(c) Power

Figure 4.8. Performance and network power behavior with NN traffic.

4.4.3 Scalability Analysis

Since Diagonal+BL is our best HeteroNoC design, scaling numbers are reported only for this design. Figure 4.9 shows the performance, throughput and power behavior of Diagonal+BL with UR and TP traffic as the network size scales from 64 (8x8) to 256 (16x16) nodes. All HeteroNoC designs show performance, throughput and power scaling with an increase in network size. With network scaling, the big routers get even more resources compared to small routers. For instance, in a

(16x16) mesh layout, the big routers are apportioned 10 VCs compared to 6 VCs in an (8x8) network. This leads to even higher percentage reduction in latency, power and increase in throughput. For Diagonal+BL, the latency savings increase from 24% (22%) to 28% (27%), power savings increase from 28% (24%) to 35% (30%) and throughput increases from 22% (27%) to 30% (35%) with UR (TP) traffic as the network scales to 256 nodes.

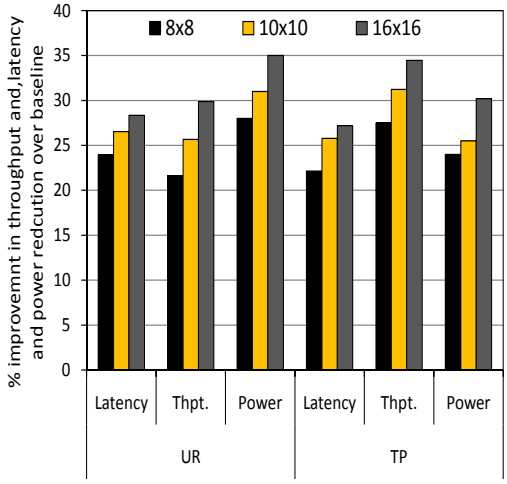
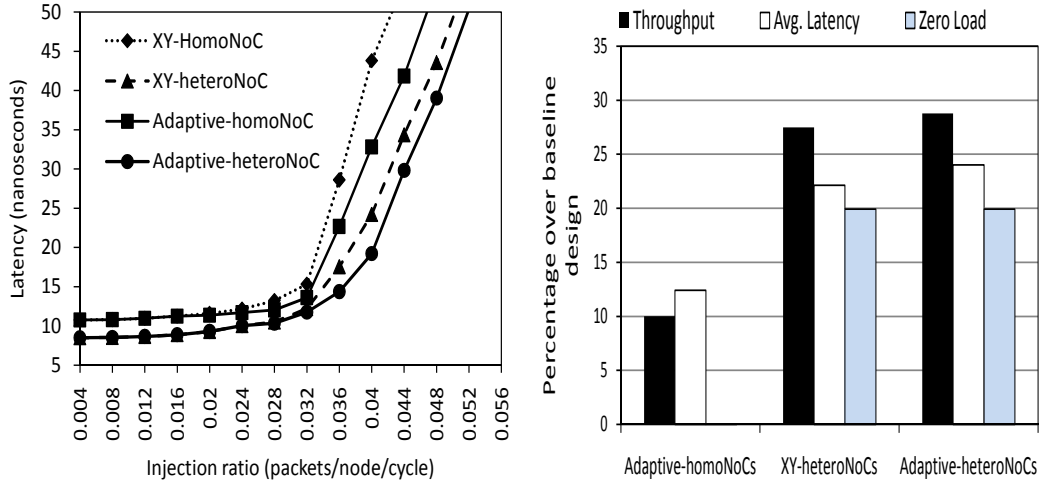


Figure 4.9. Scalability of HeteroNoC with UR and TP traffic.



(a) Latency

(b) Throughput improvement and latency reduction

Figure 4.10. Results with adaptive routing using TP traffic.

4.4.4 Adaptive Routing

The primary motivation behind HeteroNoC is to handle non-uniform resource usage in a network due to asymmetric design and X-Y routing protocol. The analysis in this subsection shows that HeteroNoC is in fact better when compared with adaptive routing. A minimal adaptive routing [4] is implemented in HeteroNoC as well as the baseline network. Figure 4.10 shows the performance curves of X-Y and adaptive routing with TP traffic pattern, where HeteroNoC with X-Y routing further reduces latency by 12%, improves throughput by 17% and reduces zero load latency by additional 19% (Figure 4.10 (b)) with respect to the homogeneous NoC with adaptive routing. Adaptive routing in the HeteroNoC does not yield much benefit compared to X-Y routing since, adaptive routing tends to use the big routers more compared to the small routers, leading to further contention in them. Re-distribution of resources and apportioning more resources to routers that face more contention helps HeteroNoC with X-Y achieve superior performance when compared to adaptive routing in the baseline.

4.4.5 Results with Applications

Figure 4.11 (a) shows the impact of using various HeteroNoC layouts on network latency. Re-distributing resources appropriately helps reduce contention in the network and reduces end-to-end delay. Simultaneous transmission of flits reduces serialization latency and thus, transfer latency (shown in Figure 4.11 (b)). Overall, there is 18.5% average latency reduction with the Diagonal+BL layout across the whole application suite.

Figures 4.12 (a) and (b) show the reduction in power consumption and power breakdown, respectively with various HeteroNoC designs. Reduction in crossbar power (using 128b wires vs 192b wires in baseline) and buffer power (since HeteroNoC uses 33% fewer buffers) are the primary reasons for reduction in overall network power. On an average, there is 18% (22% with Diagonal+BL) network power reduction with HeteroNoC designs across all application suites evaluated.

Figures 4.13 (a) and (b) depict the percentage improvement in IPC of commercial and PARSEC applications, respectively, over the baseline case, and again Diagonal+BL has the best results with 12% and 10% average improvements in

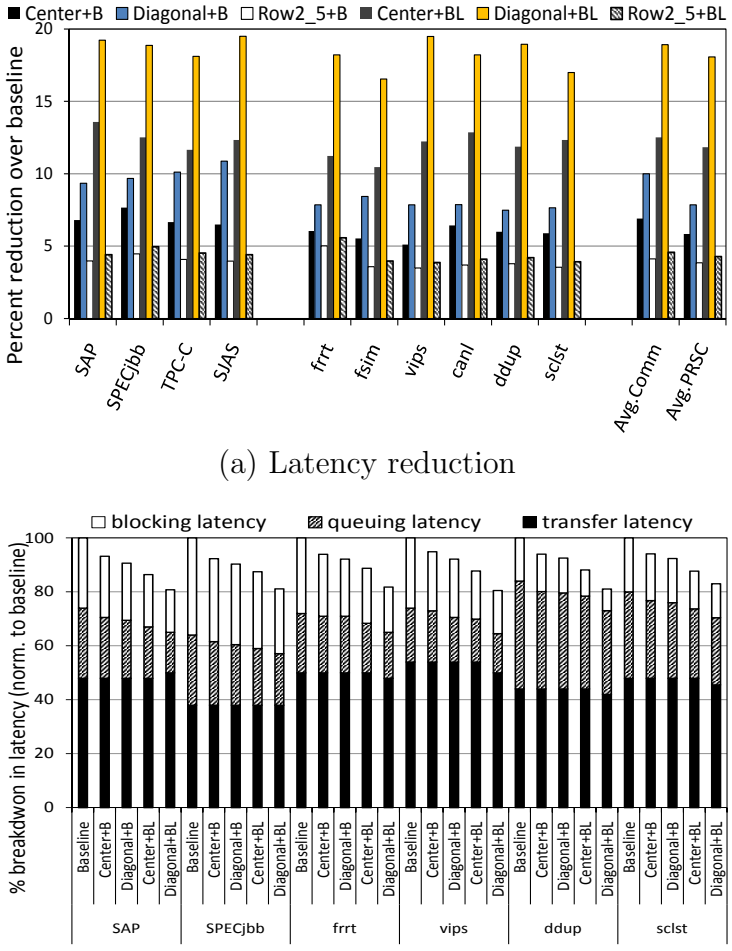
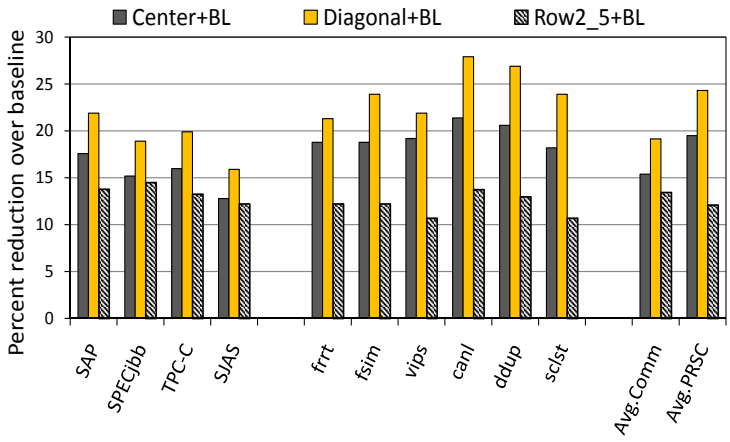


Figure 4.11. Latency reduction with applications.

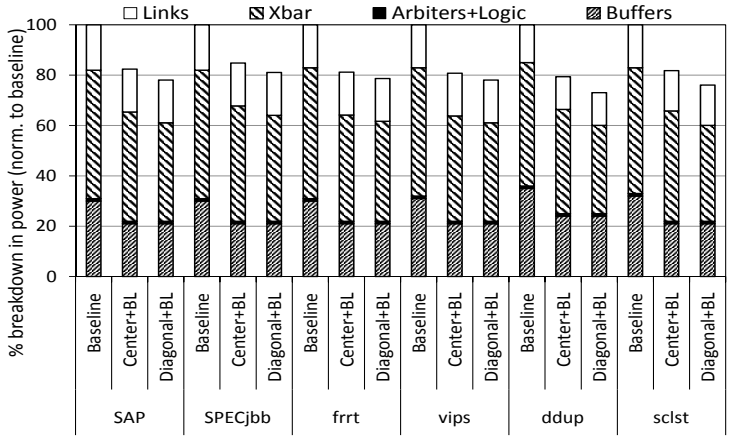
IPC for commercial applications and PARSEC benchmarks, respectively.

4.5 Co-evaluation of HeteroNoC with Heterogeneous Cores

Recent research has shown the potential of heterogeneous CMPs in providing high single-threaded performance when thread parallelism is low, and high throughput when thread parallelism is high [95–97, 99]. They do so by hosting the low thread-level parallel (TLP) application on a powerful core and the high TLP application on many simple cores. Clearly, with asymmetric cores sharing the same CMP sub-

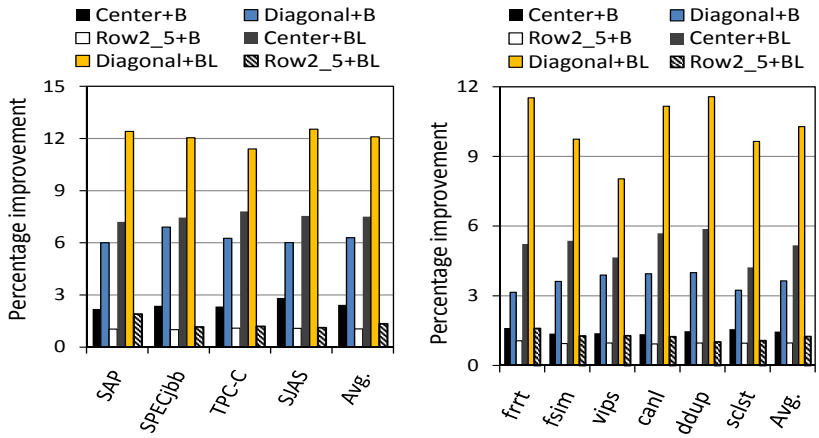


(a) Power reduction



(b) Power breakdown

Figure 4.12. Power reduction with applications.



(a) Commercial applications (b) PARSEC applications

Figure 4.13. IPC improvement with HeteroNoC.

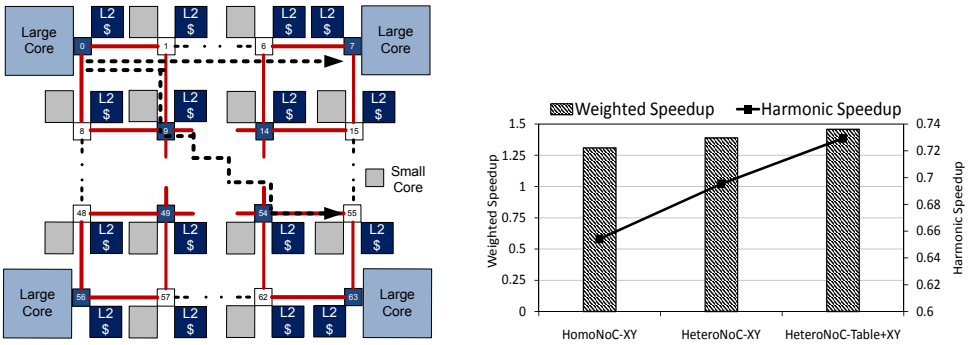
strate, the interconnect will cater to different demands - the high TLP applications may have significant on-chip shared traffic and can *tolerate* network delays, while the latency sensitive applications require a low latency interconnect.

It is compelling to envision that asymmetric CMP platform will have heterogeneous interconnect. Figure 4.14(a) shows the asymmetric CMP and the HeteroNoC platform used in this evaluation. The large cores are out-of-order powerful cores with configuration shown in Table 2.1 (a) and the small cores are in-order cores with similar private cache configuration as the large cores. A large core is assumed to occupy 4x the area of a small core. An asymmetric CMP platform consisting of 4 large cores placed at the corners of the heterogeneous mesh network (Diagonal+BL) and 60 small cores placed at the remaining nodes of the mesh network is considered. The reason for placing the large cores far apart is that these cores will consume the most power and host single threaded applications that require minimal communication with other large cores.

To show the impact of a heterogeneous network in an asymmetric platform, three scenarios are evaluated - (a) *HomoNoC-XY*, where all routers in the network have 3 VCs/PC and 192b link, (b) *HeteroNoC-XY*, where a Diagonal+BL HeteroNoC design is used and X-Y routing is employed in the network, and (c) *HeteroNoC-Table+XY*, where two routing protocols are used - table based routing for traffic originating from and destined to large cores and X-Y routing for the remaining small cores. The table based routing leverages the performance of big routers in the heterogeneous network to aid the latency critical nature of packets originating from (and destined to) large cores.

With table-based routing, packets to/from large cores are routed such that they leverage the big routers maximally. Figure 4.14(a) shows two paths that packets originating from core 0 use to reach cache banks connected to routers 7 and 55. When packets originating from core 0 (and hence router 0) want to reach a cache bank connected to router 7, they are routed similar to X-Y routing. However, when these packets want to access a cache bank connected to router 55, they are routed in a zig-zag X-Y-X-Y fashion so as to maximally use the big routers laid out along the diagonals. Table-based routing mandates maintaining source/destination pair information in all routers in the network. However, since in HeteroNoC table-based routing is used only for packets to/from the four large

cores, the table sizes are minimal when compared to maintaining table information for all 64 cores in the network. Since table-based routing approach can lead to deadlock scenarios, reserved escape VCs were used in the big routers to resolve deadlocks. 60 threads of `SPECjbb` are scheduled on the small cores and one instance of `libquantum` benchmark on each large core. `libquantum` is a latency sensitive benchmark, whereas the multithreaded `SPECjbb`, with each thread simulating one instance of a warehouse in a 3-tier JAVA server, represents a throughput-oriented application having high TLP.



(a) Layout of an asymmetric CMP (b) Performance improvement of SPECjbb and libquantum

Figure 4.14. Co-Evaluation of HeteroNoC with an asymmetric CMP.

Figure 4.14 (b) shows the weighted speedup and harmonic speedup of the system with the three network layouts. Table-based routing expedites `libquantum` packets by routing them through big routers. It also helps to accelerate `SPECjbb` packets by reducing contention in small routers, since small routers are now less frequently used by `libquantum` packets. Table-based routing along with X-Y routing in a heterogeneous network provides the maximum weighted speedup demonstrating the advantages of selectively expediting `libquantum` packets. HeteroNoC+XY and HeteroNoC-Table+XY show 6% and 11% improvement, respectively, in weighted speedup over the HomoNoC-XY design. The harmonic speedup also increases with HeteroNoC and table based-routing showing that selective expedition of `libquantum` packets does not lead to system unfairness. With HeteroNoC-Table+XY, there is 11.5% improvement in harmonic speedup over the HomoNoC-XY design.

4.6 Chapter Summary

This chapter proposes to leverage the non-uniform resource usage in a homogeneous mesh interconnect for designing a heterogeneous network, composed of big and small routers, by redistributing the buffer and link bandwidth. It presents an exploration of the design space in choosing the *size, number* and *placement* of these big and small routers and shows that HeteroNoC design with big routers placed along the network diagonals performs significantly better than the traditional homogeneous network under a variety of traffic patterns. Moreover, analysis shows that the non-uniformity in resource utilization is an artifact of any non-edge symmetric network design and deterministic X-Y routing, and this inherent artifact can be leveraged to better customize the network.

Chapter 5

Application-Driven Design of On-Chip Networks

This chapter proposes a scheme that factors the heterogeneity of applications hosted on the cores in a multicore chip. The scheme proposed in this chapter follows a top-down approach - applications are characterized in detail and their network demands are studied carefully. Based on this characterization, the network is then designed such that the applications' requirements are satisfied appropriately.

5.1 Introduction

Network-on-Chips (NoCs) are envisioned to be a scalable communication substrate for building multicore systems, which are expected to execute a large number of different applications and threads concurrently to maximize system performance. The NoC is a critical shared resource among these concurrently-executing applications, significantly affecting each application's performance, system performance, and energy efficiency. Applications that share the NoC are likely to have diverse characteristics and performance requirements, resulting in different performance demands from the network. The design parameters and algorithms employed in the NoC critically affect the latency and bandwidth provided to each application, thereby affecting the performance and efficiency of each application's execution. Therefore, devising NoCs that can efficiently satisfy diverse characteristics of dif-

ferent applications is likely to become increasingly important.

Traditionally, NoCs have been designed in a monolithic, one-size-fits-all manner, agnostic to the needs of different access patterns and application characteristics. Two common solutions are to design a single NoC for 1) common-case, or average-case, application behavior or 2) near-worst case application behavior, by overprovisioning the design as much as possible to maximize network bandwidth and to minimize network latency. However, applications have widely different demands from the network, e.g. some require low latency, some high bandwidth, some both, and some neither. As a result, both design choices are suboptimal in terms of either performance or efficiency. The "average-case" network design cannot provide good performance for applications that require more than the supported bandwidth or benefit from lower latency. Both network designs, especially the "overprovisioned" design, is power- and energy-inefficient for applications that do not need the provided high bandwidth or low latency. Hence, monolithic, one-size-fits-all NoC designs are either low performance or energy-inefficient for different applications [100].

Ideally, one would like an NoC design that can provide just the right amount of bandwidth and latency for an application such that the application's performance is maximized, while the system's energy consumption is minimized. This can be achieved by dedicating each application its own NoC that is dynamically customized for the application's bandwidth and latency requirements. Unfortunately, such a design would not only be very costly in terms of die area, but also requires innovations to dynamically change the network bandwidth and latency across a wide range. Instead, if one can categorize applications into a *small* number of classes based on similarity in resource requirements, and design multiple networks that can efficiently execute each class of applications, then one can potentially have a cost-efficient network design that can adapt itself to application requirements [100].

Building upon this insight, this chapter proposes a new approach to design an on-chip interconnect that can satisfy the diverse performance requirements of applications in an energy efficient manner. It is observed that applications can be divided into two general classes in terms of their requirements from the network: bandwidth-sensitive and latency-sensitive [46]. Two different NoC designs, each

of which is customized for high bandwidth or low latency can, respectively, satisfy requirements of the two classes in a power efficient manner. This dissertation, therefore, proposes designing two separate, heterogeneous networks on a chip, dynamically monitoring executing applications' bandwidth and latency sensitivity, and steering/injecting network packets of each application to the appropriate network based on whether the application is deemed to be bandwidth-sensitive or latency-sensitive. Later in this chapter, it is shown that such a heterogeneous design can achieve better performance and energy efficiency than current average-case or overprovisioned one-size-fits-all NoC designs.

To this end, based on extensive application profiling, this chapter first shows that a high-bandwidth, low frequency network is best suited for bandwidth sensitive applications and a low-bandwidth but high frequency network is apt for latency sensitive applications. Next, to steer packets into a particular sub-network, a packet's sensitivity to latency or bandwidth is identified at runtime. To do this, this dissertation propose a novel packet classification scheme that uses an application's network episode length and height information to dynamically identify the communication requirements (latency/bandwidth criticality). Further, observing the property that not all applications are equally sensitive to latency or bandwidth, this dissertation proposes a fine grain prioritization of applications within the bandwidth and latency customized sub-networks. Thus, the mechanism consists of first classifying an application as latency or bandwidth sensitive and then prioritizing each application's packet based on its criticality for improving the overall system/application performance.

Evaluations on a 64 core 2D architecture considering 9 design alternatives with 36 diverse applications, shows that the proposed two-layer heterogeneous network architecture outperforms all competitive monolithic network designs in terms of system/application performance and energy/energy-delay envelope. Overall, the primary contributions of this work are the following:

• This work identifies that a monolithic network design is sub-optimal when hosting applications with diverse network demands. As a step further, with extensive application level profiling, this work identifies that applications can be divided into two general classes in terms of their requirements from the network: bandwidth-sensitive and latency-sensitive.

• Therefore, this dissertation proposes a two-tier heterogeneous network architecture suitable for bandwidth and latency sensitive applications. For steering packets to an appropriate network, a novel dynamic mechanism is proposed that utilizes the communication episodes of an application, called episode length and height, not only to classify applications, but also to provide finer granularity for customized prioritization of packets within a class. It is shown that application packets can be classified into 9 sub-categories. This dynamic ranking/prioritization scheme is shown to perform better than two other schemes, proposed recently.

• This work shows that the two-layer NoC design provides 34% and 24% system and application throughput improvement, respectively, over a competitive network design, and consumes 59% and 47% lower energy when compared to a high-bandwidth network and an iso-resource network, respectively.

In Chapter 1 it was shown that a single monolithic network is not the best option for catering various application demands. Therefore, an alternative approach to designing an on-chip interconnect is to explore the feasibility of multiple networks each of which is specialized for common application requirements, and dynamically steer requests of each application to the network that matches the application's requirements. Further, based on Figures 1.3 and 1.4, it was also shown that a wider and a low frequency network is suitable for bandwidth sensitive applications, while a narrow and high frequency network is best for latency sensitive benchmarks. In addition, since not all applications are equally sensitive to bandwidth or latency, this chapter investigates a fine grain prioritization of applications within the bandwidth and latency customized sub-networks to further improve the overall application/system performance.

5.2 Application-Driven Approach for Designing NoCs

This section elaborates on the dynamic application classification scheme and proposes a mechanism for intra-class ranking and prioritization of packets for better performance/energy tuning.

Identifying application class: In the following subsection, a mechanism is proposed to dynamically classify application packets to be either bandwidth sensitive or latency sensitive so that these packets can be steered to the proper network. Ideally, this should be done as soon as packets are injected to a network so that they are not routed to a wrong network. For this, the proposed scheme uses two novel heuristics, *episode length* and *episode height*, that capture the application's latency and bandwidth demand from a network. Once determined, the application packets are then steered into a sub-network that is customized either for latency sensitive application packets or bandwidth sensitive application packets.

Application packet ranking: Application ranking within a sub-network is done to improve the application-level throughput further. This ranking is based on the *latency criticality* of a packet compared to other packets in the same sub-network. The criticality of a network packet belonging an application is identified using the same *episode length* and *episode width* heuristics that help to identify an application class. A packet's rank is determined at the network interface (NI) just before a packet is injected into a sub-network. The packet is then tagged with its rank and individual routers in a network use this information to determine which packets are prioritized at any given cycle.

5.2.1 Dynamic Classification of Applications

The goal of identifying an application's sensitivity to latency/bandwidth, is to enable the network interface (NI) to inject or steer packets into a sub-network that has been customized for either latency or bandwidth. This dissertation proposes two novel metrics, called *episode length* and *episode height* [100], that effectively capture the latency and bandwidth demands of an application and help the NI to classify an application as either bandwidth or latency sensitive. This chapter contrasts the new metrics against two heuristics (L1MPKI [45] and Slack [101]), which have been proposed to estimate a packet's criticality in the network. These heuristics are computed at runtime and is used by the NI to steer packets to one of the two sub-networks.

Episode length and height: During an application's life cycle, the application alternates between two kinds of episodes (shown in Figure 5.1): (1) *net-*

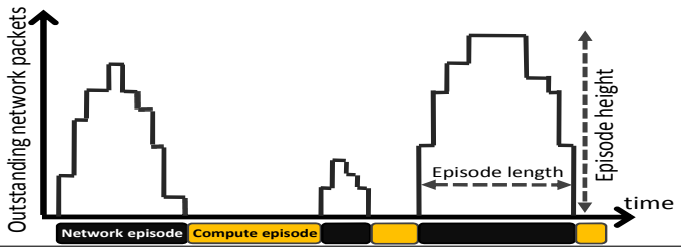


Figure 5.1. Network and compute episodes.

work episode, where the application has at least one packet (to L2 cache or to DRAM) in the network, and (2) *compute episode*, where there are no outstanding cache/memory requests by the thread. During the network phase, there may be multiple outstanding packets from the application in the network owing to various techniques that exploit memory-level parallelism (MLP) [102–104]. During this network phase, the processor is most likely to be stalling for the L2 and memory requests to be serviced. Because of this, the instruction throughput of the processor is low during this episode. During the compute episode, however, the instruction throughput is high. In this dissertation, a network episode is quantified by its length and height. Length is the number of cycles the episode lasts starting from when the first packet is injected into the network till there are no more outstanding packets belonging to that episode. Height is the average number of packets (L1 misses) injected by the application during the network episode. To compute this average height, the processor hosting the application keeps track of the number of outstanding L1 misses (when there is at least 1 L1 miss) in the re-order buffer on a per-cycle basis. For example, if the episode lasts for 3 cycles and there are 2, 3 and 1 L1 misses in each of those cycles, then the average episode height is $\frac{2+3+1}{3} = 2$.

If an episode lasts for a very few cycles, intuitively it reflects that all packets belonging to this episode are very critical for the application to make progress. Any delay of packets belonging to this short lasting episode will delay the start of the following computation phase, and, thus the performance of the application will degrade. Hence, these packets are latency sensitive. On the other hand, if an episode is long lasting, the application is most likely tolerant to this long episode length, and delaying any packets belong to this episode will not degrade

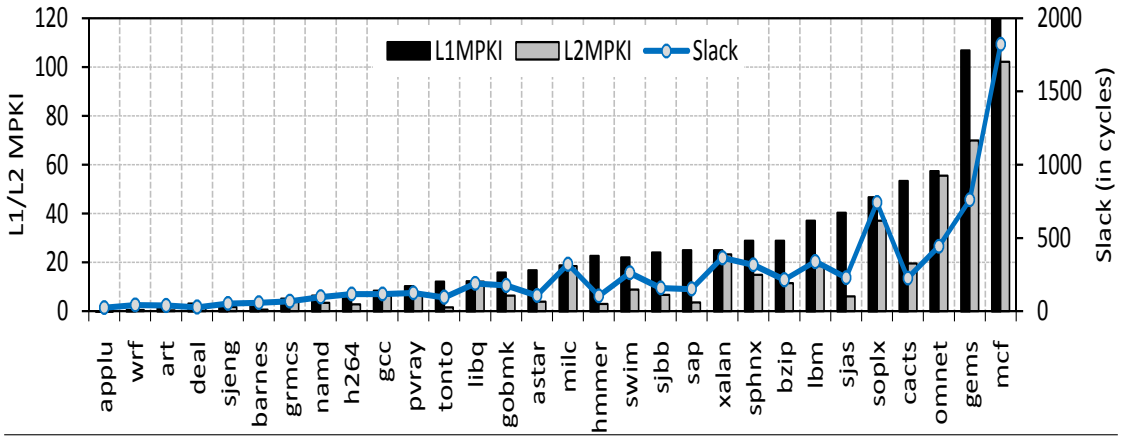


Figure 5.2. L1MPKI, L2MPKI and slack in applications.

the performance much.

If an episode's height is short, it suggests that the application is likely to have low MLP in this episode and hence, its requests are likely to be very critical for the application to make progress. The packets belonging to this phase are likely to be latency sensitive. On the other hand, if an episode height is high, then the application has a large number of requests in the network, and the network latency of all those packets are overlapped. Large number of packets in the network means that the application most likely needs more bandwidth, but the network latency is not very critical for the application. Analysis shows that, these two heuristics are least affected by the system state or network characteristics such as interference from other applications in the network. Therefore, these two metrics provide an intuitive, easy-to-compute, accurate and stable characterization of an application's network demand.

Private cache misses per instruction (MPI): This metric captures an application's network intensity. If the network intensity is lower, the application has low MLP and hence, its request are latency sensitive as opposed to bandwidth sensitive. Figure 5.2 shows the L1MPKI and L2 MPKI of several applications. It is found that, MPI (or MPKI) can help in identifying latency sensitive applications from bandwidth sensitive ones. In Figure 5.2, all applications to the left of `sjbb` have a lower MPKI than `sjbb`'s MPKI. Since these applications are latency sensitive, empirically one can think of having a threshold in MPKI (equal to `sjbb`'s MPKI) to classify applications as bandwidth or latency sensitive. However, as

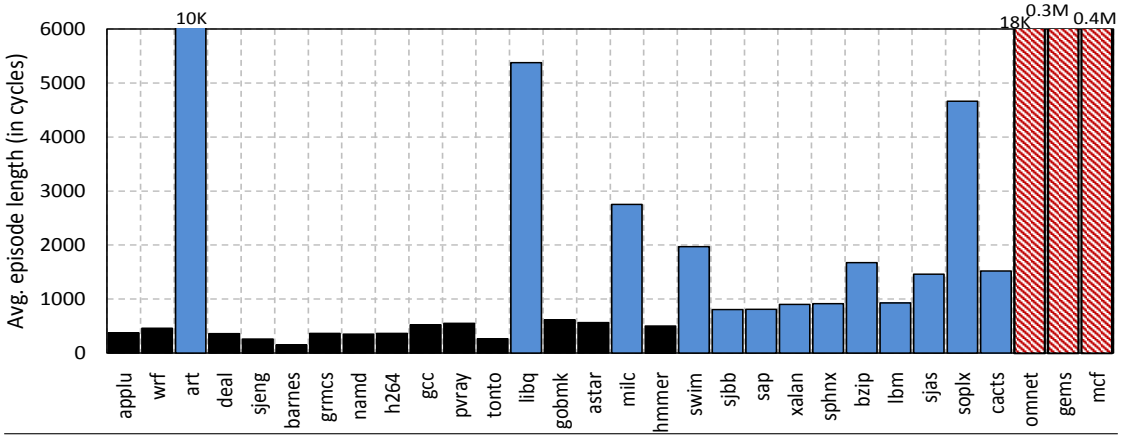


Figure 5.3. Average episode length (in cycles) across applications.

mentioned earlier, this metric is not accurate in estimating the criticality of applications *within* the latency sensitive class or bandwidth sensitive class. For instance, `bzip` in spite of having higher L1MPKI than `xalan`, is less sensitive to bandwidth than `xalan`. Similarly, `hmmer` and `swim`, in spite of having higher L1MPKI when compared to `gobmk` and `astar`, do not show proportional performance improvement with increase in bandwidth as the later applications show.

Packet slack: Slack, as a metric, was investigated by Das et al. in [101] to identify a packet's criticality in the network. An instruction's slack is measured from when it enters the re-order buffer (ROB) to when the instruction actually becomes the oldest in the ROB and is ready to commit. Figure 5.2 shows how slack varies across applications. Intuitively, slack of a L1-miss instruction directly translates to the instruction's criticality in the network. Based on this, applications that have a longer slack are more tolerant to network delays when compared to applications that have smaller or no slack. Unfortunately, slack does not capture the MLP of an application and has low correlation in identifying increase in performance with increase in bandwidth/frequency. Furthermore, slack is influenced by contention in the network and fluctuates significantly.

5.2.2 Analysis of Episode Length and Height

To avoid short term fluctuations, running averages of the episode height and length is used to keep track of these metrics at runtime. Further, episode height is quantified as *high*, *medium* or *short* and episode length as *long*, *medium* and *short*. This

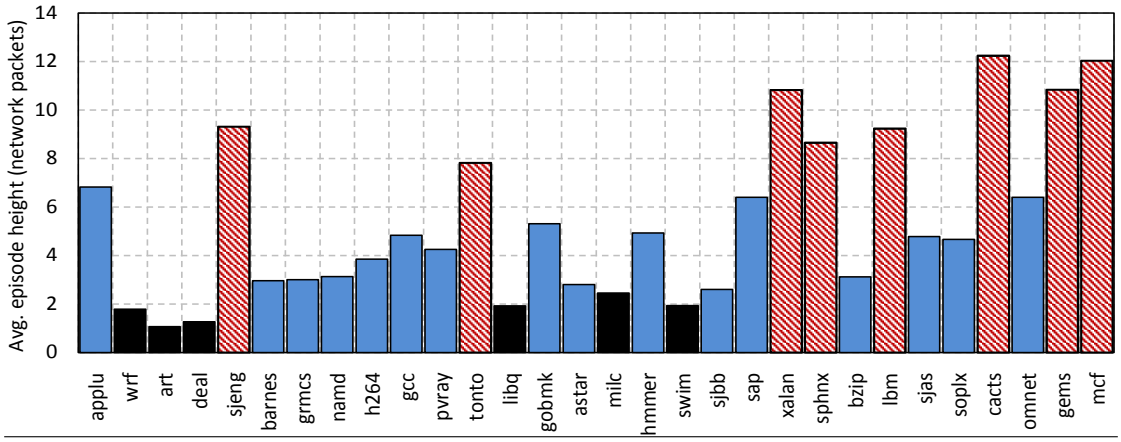


Figure 5.4. Average episode height (in packets) across applications.

allows a fine grain application classification to be performed based to episode length and height, and to classify applications as either latency sensitive or bandwidth sensitive. Section 5.2.3 provides empirical data to support such a classification scheme. Figures 5.3 and 5.4 show these metrics for 30 applications in our benchmark suite. Based on Figures 1.3 and 1.4, all applications whose episode length and height are shorter than `sjbb`'s episode length and height, respectively, are classified to be short in length and height (shaded black in the figures). Applications whose average episode is larger than `sjbb`'s episode height but lower than 7 (empirically chosen) are classified as medium (shaded blue in the figures) and the remaining as high episode heights (shaded with hatches in Figure 5.4). Empirically, a cut-off of 10K cycles is chosen to classify applications as having medium episode length.

Figure 5.5 shows the classification of applications based on their episode height and length. The figure also shows the bandwidth sensitive applications and the latency sensitive applications based on such a classification. In general, applications having high episode height are classified as bandwidth sensitive and vice-versa for latency sensitive.

5.2.3 Ranking of Applications

The fine-grained classification scheme discussed above is used to rank applications for providing application-based prioritization in the network. Essentially, applications whose episode length lasts longer, are prioritized the least in the network over other applications. The discussion below mentions the intuition behind steering of

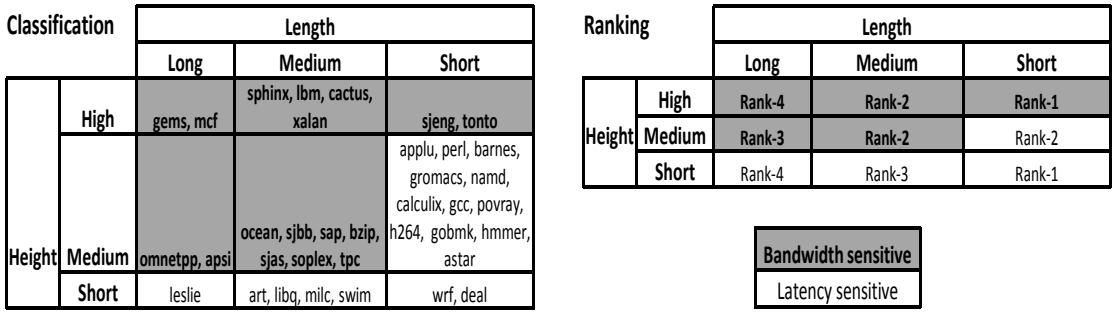


Figure 5.5. Application classification and ranking based on episode length and height.

the applications into a particular sub-network and the ranking of the applications within each sub-network.

(1) Episode length is short and height is short: Applications belonging to this category have very low MPKI and since their episode lasts for a very short period, delaying any packet is most likely to delay the start of the computation phase. This makes these applications highly latency sensitive and hence packets belonging to these applications are steered into the latency customized sub-network. Further, the proposed mechanism ranks these packets the highest (rank 1) in the latency customized sub-network.

(2) Episode length is short and height is high: These applications are bursty, but for a very short period of time. Because of this burstiness, the packets' network latency are overlapped and hence, these applications are classified as bandwidth sensitive but the proposed scheme ranks them the highest in the bandwidth customized sub-network (owing to their criticality to network latency because of a very short episode length).

(3) Episode length is long and height is short: These applications are still latency sensitive, but are relatively latency tolerant compared to applications having medium/short episode length. So, these applications are prioritized the least (rank 4) in the latency customized sub-network.

(4) Episode length is long and height is high: These applications are the most bandwidth sensitive applications, and owing to their large episode height, are the most delay tolerant applications as well. Thus, these applications are classified as bandwidth sensitive and the proposal prioritizes them the least in the bandwidth customized network.

Applications that do not belong to the above classes, have either latency or

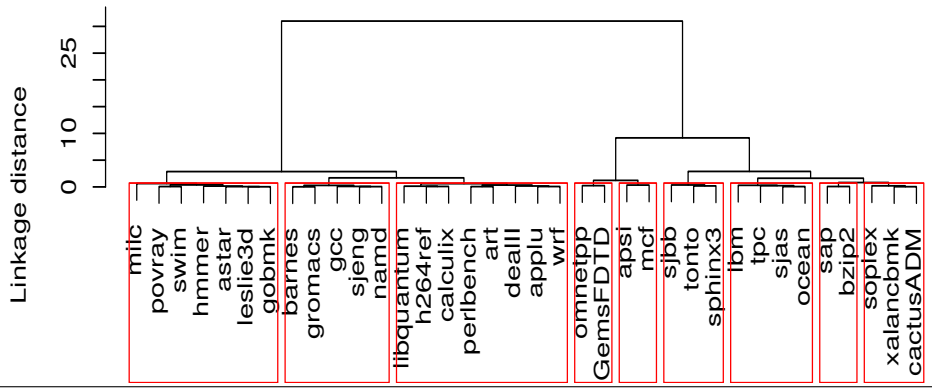


Figure 5.6. Hierarchical clustering of applications. The input to the clustering algorithm consists of improvement in IPC with bandwidth scaling (from 64b to 512b) and improvement in IPC with frequency scaling (2GHz to 6GHz).

bandwidth sensitivity that lie within the extremes and are prioritized based on their relative tolerance to network delays when compared to others. Figure 5.5 shows the ranking of the applications in their respective sub-networks.

Two critical decisions are taken in the above classifications - (1) choosing `sjbb`'s episode length and height as a threshold for short lasting episodes and episodes with smaller heights, and (2) choosing 9 smaller sub-classes after classifying the applications as bandwidth or latency sensitive. Next, this chapter outlines the empirical results that led to these decisions.

Rationality of our classification: Figure 5.6 shows the results of a hierarchical clustering of all the applications in the benchmark suite. Hierarchical clustering incrementally groups objects that are similar, i.e., objects that are close to each other in terms of some distance metric. In the current case, the input to the clustering algorithm consists of the improvement in IPC with bandwidth scaling (from 64b to 512b) and improvement in IPC with frequency scaling (from 2GHz to 6GHz) i.e. values from Figures 1.3 and 1.4. The hypothesis behind this is to observe whether a clustering algorithm perceives noticeable difference between applications' performance with frequency and bandwidth scaling. Various linkage distance metrics like Euclidean distance, Pearson correlation and average distance between the objects were also tried, and in all cases the clustering was consistent with that shown in Figure 5.6 (shown for Euclidean distance). Although the eventual hierarchical cluster memberships are different from that shown in the proposed classification matrix, the broader classification of how hierarchical

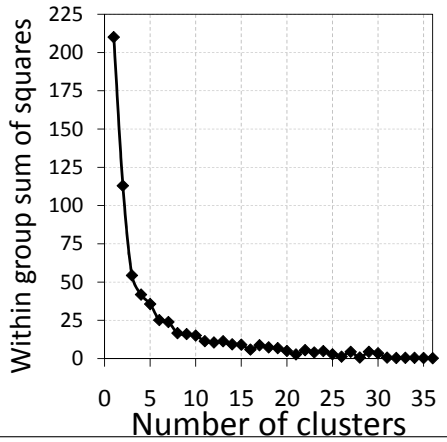


Figure 5.7. Reduction in within-group sum-of-squares with increase in number of clusters.

clustering groups applications in bandwidth and latency sensitive clusters matches exactly with the proposed classification scheme, which is based in episode height and length (with the exception of `sjeng`). The reason for `sjeng`'s misclassification is because its performance does not scale with bandwidth and hence, hierarchical clustering classifies it as a latency sensitive application. However, `sjeng`'s episode has a high episode height but a short episode length on average, meaning it is very bursty (and hence, high MLP) during a small interval of time. Because of this, it is classified as the highest ranking application in the bandwidth customized sub-network.

Why 9 sub-classes? To answer this question, the total within-group sum-of-squares (WG-SS) of the clusters resulting with hierarchical clustering is measured. Figure 5.7 shows this metric as the number of clusters increase. The total WG-SS is a measure of the total dispersion between individual clusters and often regarded as a metric to decide the optimal number of clusters from a hierarchical or K-means algorithm [105, 106]. When all clustering objects are grouped into one cluster, the total WG-SS is maximum, whereas, if each object is classified as a separate object, the WG-SS is minimum (=0). Figure 5.7 suggests that 8 or 9 clusters have similar WG-SS and, 8 or 9 clusters reduce the total WG-SS by 13x compared to a single cluster. Based on this, 9 classes are chosen for the proposed application classification and hence, episode height and length are sub-divided into three quantitative class each.

5.3 Design Details

A 2D mesh network is used in evaluating the proposed mechanisms. Chapter 2 discusses the router and network designs used as the baseline design in this study. This section focus on the critical design aspects for supporting the proposed classification and prioritization schemes.

Computing episode characteristics: To filter out short-term fluctuations in episode height and length, and adapt the proposed techniques to handle long-term traffic characteristics, running averages of these metrics are used, i.e. on every L1 miss, the NI computes the running average episode length/height. To compute episode height, outstanding L1 miss count is obtained from the miss-status handling registers (MSHRs). Counting the number of cycles the L1 MSHRs are occupied, gives the information to compute the episode length. An M-bit counter is used to count this. This counter is reset every batching interval, B (discussed below).

When an NI of a local router receives a packet, it computes the episode length and height, and based on the classification scheme mentioned in Section 5.2, decides which sub-network this packet is to be steered into. Further, the NI also tags the packet with it's rank (2-bits) and it's batch-id (3-bits). Note that, although the classification is static, each applications' rank and network sensitivity is decided at runtime. Thus, no central co-ordination is required in the proposed technique to decide a consistent central ranking across all the applications in the system. Moreover, once a packet's ranking has been decided, it is consistently prioritized across the entire sub-network until it reaches its destination. At each router, the priority bits in the header-flit are utilized by the priority arbiters in a router to allocate VCs and the switch. Fast priority arbiters can be designed using high speed adders as comparators within the arbiters and our estimates (based on [107]) show that priority arbiters do not skew the pipeline latencies of a router.

To prevent priority inversion due to virtual channels (VCs) in routers, where a packet belonging to an older batch or higher rank is queued behind a lower ranked packet, atomic buffers [5] are employed. With atomic buffers, a head-flit of a packet cannot occupy a particular VC unless the tail-flit of a packet occupying that VC has released it. Atomic buffers can lead to network under-utilization, but

experiments show that the performance loss due to this is very minimal for this proposal.

Handling starvation: Prioritizing high ranked packets in a network may lead to starvation of low ranked packets. To prevent starvation, application-aware prioritization scheme is combined with a "batching mechanism" [45]. Each packet is added to a batch; and packets belonging to older batches are prioritized over packets from younger batches. Only if two packets belong to the same batch, they are prioritized based on their applications' rank order that is based on episode height/length. A batch also provides a convenient granularity in which the ranking of the applications is enforced. To support batching, each node keeps a local copy of a batch-ID (BID) register containing the current (injection) batch number and maximum supported batch-ID register containing the maximum number of batching priority levels (L). BID is simply incremented every B cycles, and thus, BID values across all nodes are the same. Due to batch-ID wrap-around, a router cannot simply prioritize packets with lower batch-IDs over others with higher batch-IDs, and this proposal uses schemes suggested in [45, 101] to handle relative priorities inside a router.

5.4 Evaluation Methodology

Design scenarios: Starting with a monolithic network, this proposal shows the benefits of having two sub-networks each customized for either bandwidth or latency. This proposal also shows the benefits of the techniques when compared to an iso-resource network (similar bandwidth as two sub-networks). Following are the nine design scenarios evaluated on the experimental platform:

• **1N-128**: In this configuration, there is a single homogeneous 128b link network. This configuration is assumed to be the *baseline* network design since, starting with this monolithic network, the network bandwidth is increased to create a bandwidth customized sub-network, and the bandwidth reduced (and frequency increased) to design a latency customized sub-network.

• **1N-256**: In this configuration, there is a single homogeneous network with 256b link width.

• **2N-128x128**: This design has two parallel sub-networks, each with 128b

link width. The buffer resources in each sub-network is half that of the baseline 1N-128 network and each of the sub-networks operate at the same frequency as that of the processors (= 2GHz). Packets are steered into each sub-network with a probability of 0.5 i.e., there is load balancing across the sub-networks.

• **1N-512**: This design has a single network with 512b link width. This design is called a a *high-bandwidth* configuration and is analyzed to see how the proposed scheme fares when compared to a very high bandwidth network.

• **2N-64x256-ST**: In this design, there are two parallel sub-networks, one with 64b link width and the other with 256b link width. The buffering resources in each sub-network is half that of a single network, so that the total buffering resources are constant across this design and a design that has a single network. Further, in this configuration, the bandwidth sensitive packets are steered (hence, the annotation **ST**) into the 256b sub-network and the latency sensitive packets are steered into the 64b sub-network. Each sub-network in this configuration is clocked at 2GHz (= the frequency of the processors).

• **2N-64x256-ST+RK(no FS)**: This design is the same as 64x256-steering network except that, in addition to steering the application packets into the appropriate sub-network, the network also prioritize applications based on their ranks (hence, the annotation **RK**) at every cycle in a router.

• **2N-64x256-ST+RK(FS)**: This design is similar to above configuration except that the 64b sub-network is clocked at 6GHz (3x the frequency of processor). The 256b sub-network is still clocked at 2GHz. This configuration is analyzed to see the benefits of frequency scaling (hence, the annotation **FS**) the latency customized network.

• **1N-320(no FS)**: In this design, there is a single network with 320b (= 64b + 256b) bandwidth per link. The network operates at 2GHz. This configuration is iso-resource configuration when compared to all our 64x256 networks and is analyzed to see the benefits of our proposal over an equivalent iso-bandwidth configuration.

• **1N-320(FS)**: This design is similar to the above design, except that the network is now clocked at 6GHz. This design is analyzed to see the effectiveness of our scheme over a scheme that is iso-resource as well as over-clocked to help latency sensitive applications. This design is called as a *high-frequency* configuration.

Experimental setup: The proposed schemes are evaluated on a trace-driven, cycle-accurate x86 CMP simulator. Table 2.1 provides the configuration of the baseline, which contains 64 cores in a 2D, 8x8 mesh NoC. Each core has private write-back L1 caches. The memory hierarchy uses a two-level directory-based MESI cache coherence protocol. The network connects the cores, L2 cache banks, and memory controllers. Each router uses a state-of-the-art two-stage pipeline. Deterministic X-Y routing algorithm, finite input buffering, wormhole switching, and virtual-channel flow control is employed across all the above designs. A data packet consists of 1024b (= cache line size) and is decomposed into 8 flits in the baseline design (with 128b links). Since wiring resources on die are abundant [22, 51, 75, 77], when simulating parallel networks, it is assumed that the sub-networks can be implemented in the same 2D substrate as the cores. The dynamic and leakage energy numbers for the network were extracted using Orion 2.0 [108] and incorporated into the simulator for detailed network energy analysis. Based on Orion 2.0 estimates, the area of two sub-networks (router and links) consisting of 256b and 64b links is just 1% larger than an iso-resource 320b links network (2.4X larger area when compared to baseline 128b link network), and the power envelope of these two sub-networks is 20% lower than the iso-area network (2.3X higher power when compared to baseline 128b link network). The various counter bits and parameters used in our techniques are: (1) counter size for number of cycles in a network phase, $M = 14$bits (2) batching interval, $B = 16{,}000$ cycles (3) batching levels, $L = 8$.

Application characteristics: A diverse set of multiprogrammed application workloads comprising scientific, commercial, and desktop benchmarks are used. These applications comprise SPEC CPU 2006 benchmarks, applications from the SPLASH-2 and the SPEC-OMP benchmark suites, and four commercial workloads traces (`sap`, `tpcc`, `sjbb`, `sjas`) totalling 36 applications. Representative execution phases were chosen for each application using PinPoints [109] excluding commercial traces, which were collected over Intel servers. All the experiments analyze multiprogrammed workloads, where each core runs a separate application. At least 320 million instructions are simulated across 64 processors (minimum 5 million instructions per core). Table 6.2 characterizes the application suite. The reported parameters are for the applications running alone on the baseline CMP system

Table 5.1. Application characteristics when run on the baseline (Load: High or Low depending on network injection rate, Episode height: High or Medium/Short, Episode length: Long, Medium or Short, Net-fraction: Fraction of execution time spent in network episodes.)

#	Benchmark	Load	Episode height	Episode length	Net-fraction
1	applu	Low	Medium	Short	8.23%
2	wrf	Low	Short	Short	9.42%
3	perlbench	Low	Medium	Short	8.78%
4	art	Low	Short	Medium	82.33%
5	dealII (deal)	Low	Short	Short	27.92%
6	sjeng	Low	High	Short	28.41%
7	barnes	Low	Medium	Short	72.51%
8	gromacs (grmcs)	Low	Medium	Short	48.58%
9	namd	Low	Medium	Short	51.60%
10	h264ref (h264)	Low	Medium	Short	61.45%
11	calculix	Low	Medium	Short	48.21%
12	gcc	Low	Medium	Short	47.55%
13	povray (pvray)	Low	Medium	Short	59.56%
14	tonto	Low	High	Short	52.99%
15	libquantum (libq)	Low	Short	Medium	99.00%
16	gobmk	Low	Medium	Short	64.88%
17	astar	Low	Medium	Short	82.78%
18	milc	Low	Short	Medium	88.16%
19	ocean	Low	Medium	Medium	90.10%
20	hmmer	Low	Medium	Short	66.03%
21	swim	Low	Short	Medium	41%
22	sjbb	High	Medium	Medium	87.29%
23	sap	High	Medium	Medium	88.89%
24	xalancbmk (xalan)	High	High	Medium	89.86%
25	sphinx3 (sphnx)	High	High	Medium	83.92%
26	bzip2 (bzip)	High	Medium	Medium	84.90%
27	lbm	High	High	Medium	81.10%
28	sjas	High	Medium	Medium	89.47%
29	soplex (soplx)	High	Medium	Medium	81.23%
30	tpc	High	Medium	Medium	86.82%
31	cactusADM (cacts)	High	High	Medium	82.33%
32	leslie3d	High	Short	Long	99.70%
33	omnetpp	High	Medium	Long	92.62%
34	GemsFDTD	High	High	Long	97.26%
35	apsi	High	Medium	Long	95.15%
36	mcf	High	High	Long	99.18%

without any interference. The table shows application characteristics based on network load intensity (high/low), episode height (high/medium/short), episode length (long/medium/short) and the fraction of execution time spent in network episodes. All the results are aggregated across *25 workload combinations*. In each of these workload combinations, 50% (32) of the applications are latency sensitive and 50% (32) of the applications are bandwidth sensitive. This provides a good mix of bandwidth/latency sensitive applications that is likely to be a common mix for future multicore systems. Within each of these two categories, applications are

randomly picked to form the workload. In Section 5.5.1 the sensitivity of the proposed scheme is analyzed when the percentage of latency/bandwidth applications vary in a workload.

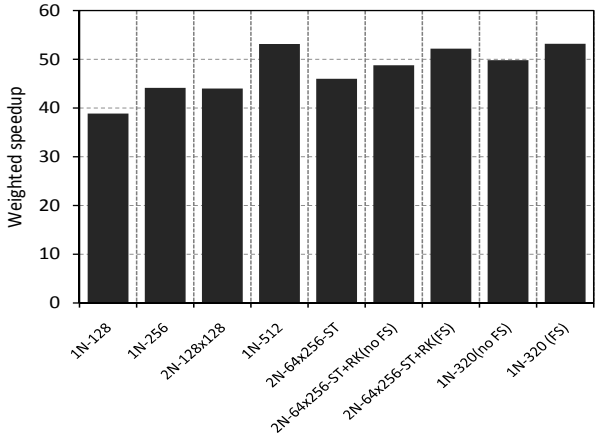
Evaluation metrics: The primary performance evaluation metrics are instruction throughput and weighted speedup. Instruction throughput is a common measure of application performance and weighted throughput is a common measure of system performance (for details on these metrics, refer to Chapter 2, Eq. (2.1) and Eq. (2.2)). When comparing with a prior work, STC [45], harmonic speedup (Eq. (2.4)) is also analyzed.

5.5 Analysis of Results

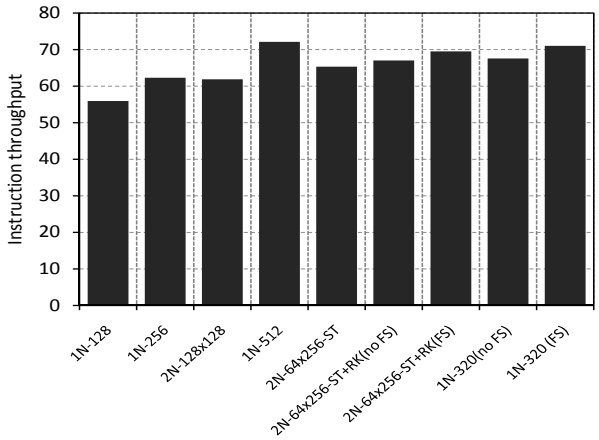
Performance comparison: Figure 5.8 shows the performance comparison across the various network designs. The following observations are in order:

• Two 128b sub-networks (2N-128x128) provide similar performance (both system and application throughput) as compared to a bandwidth equivalent single monolithic network with 256b link width (1N-256). This is in spite of the increase in packet serialization in the sub-networks. The primary reason for this performance improvement is reduction in congestion across each sub-network when compared to a monolithic wider network.

• Bandwidth and latency customized parallel sub-networks operating at the same frequency as the processor along with steering of packets based on their bandwidth and latency sensitivity (2N-64x256-ST) provides 18.3% and 16.9% system and application throughput improvement, respectively, over the baseline (1N-128) design. By providing bandwidth sensitive applications more bandwidth and reducing the congestion when compared to a monolithic network, the performance of both bandwidth and latency sensitive applications are improved. Prioritizing and ranking packets based on their criticality after steering them into a sub-network (2N-64x245-ST+RK(no FS)) provides an additional 7%/3% improvement in system/application throughput, respectively, over the 2N-64x256-ST design. This is because, the proposed ranking scheme prioritizes the more (relatively) network-sensitive applications in each sub-network, and ensures, using batching, that there is no starvation.



(a) Weighted speedup (WS) (system throughput)



(b) Instruction throughput (IT) (application throughput)

Figure 5.8. Performance comparison across various network designs with multiprogram mixes.

• Frequency scaling the latency/bandwidth sub-network along with steering and ranking the applications (2N-64x245-ST+RK(FS)) provides the maximum performance improvement among the proposed schemes (34% and 24% system and application throughput improvement, respectively) over the baseline network. With frequency scaling, the latency customized sub-network is clocked at a higher frequency, accelerating the latency sensitive packets and this brings an additional 4.4% overall improvement in application throughput.

• Frequency scaling the sub-networks and steering along with ranking of applications (2N-64x245-ST+RK(FS)) is better than an iso-resource network (1N-320(no FS)) by 5%/3% in weighted/instruction throughput. The performance

of 2N-64x245-ST+RK(FS) is within 2.0%/2.2% (system/application throughput) of the high frequency iso-resource network with frequency increased by 3x (1N-320(FS)). Frequency scaling the 320b link width network helps latency sensitive applications and more bandwidth (when compared to 256b link width) helps the bandwidth sensitive applications. But as will be shown shortly, the energy consumption of such a network is higher when compared to the proposed schemes.

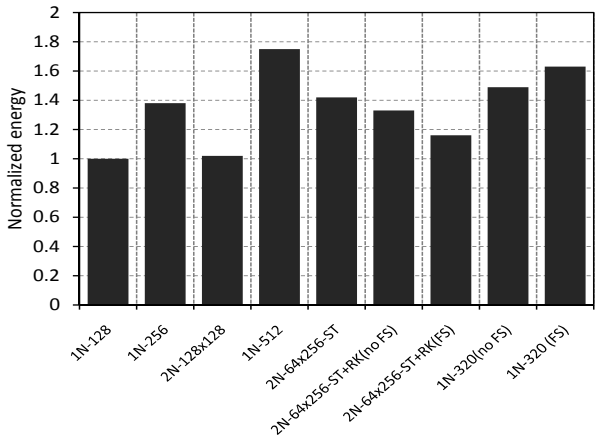
• The proposed network's (2N-64x245-ST+RK(FS)) system performance is within 1.8% of a very high bandwidth network (1N-512). A high bandwidth network helps bandwidth sensitive applications, but provides little benefit for latency sensitive applications. Additionally, as will be shown next, a wide-channel network's energy consumption is very high (about 75% higher than a 128b link width network). Hence, although the proposed network provides similar performance as a high bandwidth network, it does so at a lower energy envelope.

Energy and EDP comparison: Increasing the channel bandwidth decreases the serialization (and zero-load) latency and hence, end-to-end latency is reduced. However, increasing the channel bandwidth also affects router crossbar power. Figure 5.9 shows the energy and energy-delay product (EDP) of the applications across the 9 designs. It is found that:

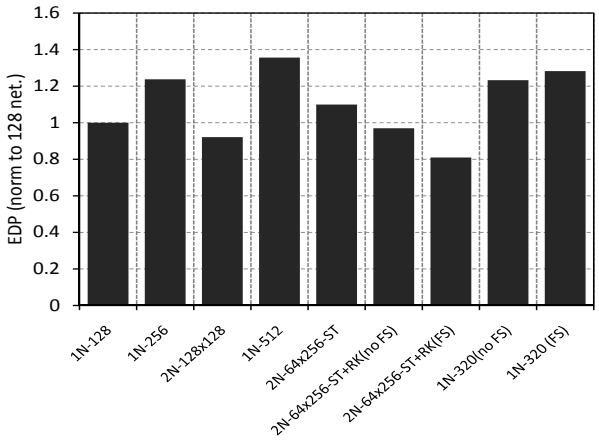
• The average energy consumption of a 256b link network (1N-256) is 38% higher than a 128b link network (1N-128). However, the two 128b sub-networks design (2N-128x128) has similar energy consumption as a single 128b link monolithic network. The energy reduction going from one network to two sub-networks comes primarily from reduction in network latency (by reducing the congestion in each sub-network). In fact, it is observed that the energy consumption of two parallel sub-networks, each with channel width $\frac{N}{2}$, is always lower than a single network with channel width N.

• The average energy consumption of a high bandwidth network with 512b links (1N-512) is 75% higher than a 128b link network. When link width increases, although serialization latency reduces, the crossbar power starts to dominate the energy component and, thus the overall energy consumption increases.

• Steering packets along with application prioritization in the routers (2N-64x256-ST-RK(no FS)) reduces energy consumption by 6.7% when compared to just steering packets (2N-64x256-ST). Amongst the proposed designs, steering



(a) Energy consumption in the network



(b) Energy-delay product (EDP) of applications

Figure 5.9. Energy and EDP comparison across various network designs (all results normalized to 128 network).

along with ranking in frequency scaled sub-networks (2N-64x256-ST-RK(FS)), consumes only 16% more energy than the baseline 1N-128 network. This is 59% lower energy when compared to a high-bandwidth network (1N-512) and 47% lower energy than an iso-resource network which is frequency scaled (1N-320(FS)). Overall, the proposed scheme consisting of heterogeneous parallel sub-network architecture always consumes lower energy than a high-bandwidth network (1N-512) and an iso-resource 320b link width network.

• When comparing EDP metric, steering along with ranking in frequency scaled sub-networks (2N-64x256-ST-RK(FS)) design is 19% better than the baseline design. This is because, the proposed scheme reduces network latency significantly

and this lowers the delay component in EDP metric. Even without frequency scaling, the 2N-64x256-ST-RK(no FS) design has 3% lower EDP than the baseline design. Again, the proposed schemes always have lower EDP than a high-bandwidth network (1N-512) or an iso-resource 320b link network.

Reply packets from L2 cache (DRAM) to L1 cache (L2 cache): In all the above evaluations, the L2 cache (DRAM) replies were routed to the L1 cache (L2 cache) in either the 64b or the 256b sub-network depending on where the request packet traversed the network: if the request packet was bandwidth sensitive, the matching reply is sent on the 256b sub-network and vice-versa. Reply packets are L1/L2 cache line sized packets (1024b) and transmitting them over the 64b network increases their serialization latency. However, the 64b sub-network is relatively less congested when compared to the 256b sub-network (because of lower injection ratio of latency sensitive applications) and since the 64b sub-network is clocked at 3x frequency, the network latency in this sub-network is lower. Analysis shows that, transmitting *all* the reply packets in the 256b network increases the system/application throughput by an additional 1.6%/2.4% and reduces energy consumption by an additional 4% when compared to the baseline 1N-128 network. Also, since coherence packets are latency sensitive packets, they are always routed in the 64b high frequency sub-network.

In conclusion, it is found that having two separate networks (each customized either for latency or for bandwidth), is beneficial from both system and application performance perspective while consuming minimally higher energy when compared to a monolithic network.

Comparison with prior works : A previous work by Das et al. [45] proposed a ranking framework, called STC, that is based on criticality of a packet in the network. In their work, the authors use L1MPKI as a heuristic to estimate the criticality of a packet and based on this, propose a ranking framework which ranks applications with lower L1MPKI over applications with higher L1MPKI. Further, in their work, a central decision logic periodically gathers information from each node, determines a global application ranking and batch boundaries, and communicates these information to each node. Apart from performance benefits, the authors also show that STC is better in terms of fairness when compared to the round-robin arbitration often employed in routers. Since, this proposal also priori-

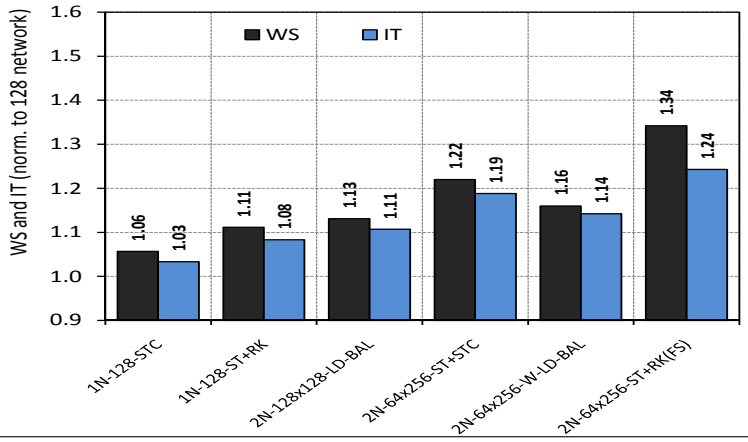


Figure 5.10. Weighted speedup (WS) and instruction throughput (IT) when compared to state-of-the art design (all results normalized to 1N-128 network).

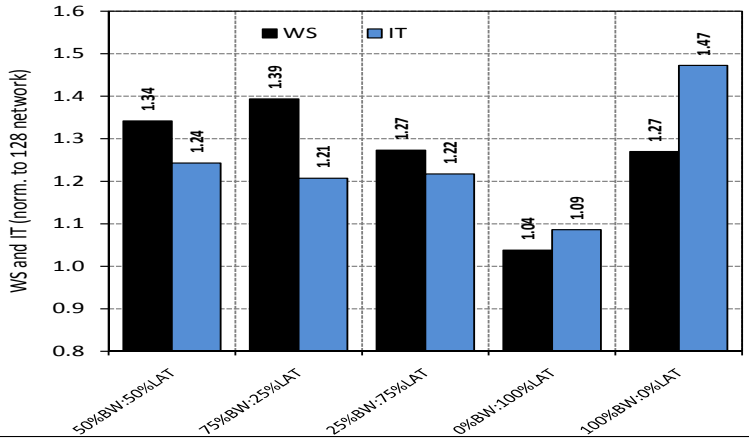


Figure 5.11. Performance comparison when varying proportion of bandwidth and latency intensive applications in each workload.

tize applications in the network, the analysis below compares the proposed scheme with STC. When comparing with STC for a single network design, a 2-level ranking scheme is utilized when using the proposed technique. The first level ranking prioritizes latency sensitive applications over bandwidth sensitive applications, and then among the latency and bandwidth sensitive applications, episode width and height is used to rank the applications (based on ranking in Figure 5.5).

Another recent work by Balfour and Dally [22] showed the effectiveness of load-distributing traffic *equally* over two parallel sub-networks. In their work, each of the sub-networks is a concentrated mesh with similar bandwidth. With detailed layout/area analysis, the authors found that a second network has no impact on

the chip area since the additional routers can reside in areas initially allocated for wider channels in the first network. Since, this proposal investigates parallel sub-networks (although the proposed heterogeneous design is shown to be better than a homogeneous design), an analysis is done comparing the proposed scheme with a similar load-balancing scheme proposed by Balfour and Dally [22].

Figure 5.10 shows the results, where the performance and fairness of the proposed schemes are compared against the two prior proposals mentioned above. All numbers in these plots are normalized to that of a 128b link network with no prioritization (i.e the baseline network, 1N-128). The STC schemes are annotated as **-STC** with a given network design and the load-balancing schemes are annotated as **-LD-BAL** in the figures. The overall performance improvement with STC is 6%/3% (system/application) in a single 128b link monolithic network when compared to 1N-128. Compared to this, the proposed 2-level ranking scheme shows 11%/8% (system/application) throughput improvement over 1N-128 design. Since STC uses L1MPKI to decide rankings, and as shown earlier, L1MPKI is not a very strong metric to decide the latency/bandwidth criticality of applications. Moreover, when using L1MPKI, STC does not take into account the *time* factor i.e how long in cycles does an application has this L1MPKI. The proposed episode length metric captures this factor, and hence, can differentiate between two applications having similar episode height (L1MPKI in the context of STC) from each other. Based on this, the proposed design ranks an application with shorter episode length higher than an application with longer episode length, hence capturing the *true* criticality of these packets. Even when comparing the proposed scheme (2N-64x256-ST-RK(FS)) with that of STC in a two parallel network design (2N-64x256-ST-STC) (where applications are first steered into the appropriate network and then ranked using STC), an additional 12%/5% (system/application) benefit is observed over the STC based design. Moreover, in terms of fairness (harmonic speedup results omitted for brevity), the proposed scheme is 4% and 2% better than STC in a single and multiple parallel network design, respectively. Further, in our scheme the rankings are determined dynamically when the packet enters into each sub-network and there is no requirement of a dynamic co-ordination scheme to decide rankings as is required by STC scheme.

Since this chapter proposes heterogeneous sub-networks, when load balancing

between two sub-networks, packets are steered in the weighted-ratio of $\frac{256}{256+64}$ and $\frac{256}{256+64}$ between the 256b and the 64b sub-network. This scheme is annotated as **-W-LD-BAL** in the Figure 5.10. Evaluations show that, steering packets with equal probability into each network leads to more congestion the 64b link sub-network and under-utilizes the 256b sub-network. It is found that, the proposed design (2N-64x256-ST-RK(FS)) has an additional 18%/10% (system/application) throughput improvement over the weighted load-balancing scheme (2N-64x256-W-LD-BAL). Load balancing scheme is oblivious to the sensitivity or criticality of packets. With this scheme, a latency sensitive packet is steered into the bandwidth customized network with a probability of 0.8 and bandwidth sensitive packet is steered into the latency sensitive network with a probability of 0.2 and, thus in both these cases performance either does not improve (for the former) or degrade (with the later). Further, with weighted load-balancing, there is negligible improvement in fairness whereas, in the proposed scheme the fairness of the system improves by 19% over the baseline network. Overall, with heterogeneous sub-networks, load balancing is a sub-optimal scheme and intelligently steering packets based on their sensitivity and criticality can lead to significant performance benefits.

5.5.1 Sensitivity to Distribution of Bandwidth-Latency Applications in the Workload

All results shown till now had a multiprogram mix with equal percentage of latency and bandwidth sensitive applications. To analyze the sensitivity of the proposed scheme across various application mixes, the bandwidth sensitive application mix in a workload is varied from 100%, 25%, 50%, 75% to 0%. Figure 5.11 shows the results of this analysis. The proposed scheme, in general, has higher system/application throughput across the entire spectrum of workload mix. However, the benefits are small (4%/9% system/application throughput improvement over baseline) when the system has 100% latency sensitive applications. When the application mix is skewed (i.e. system has *only* bandwidth *or* latency sensitive applications), an oracle knowledge is assumed, and weighted-load balancing is done in both the sub-networks. As such, with 100% latency sensitive applications in the workload mix, benefits arise only due to load distribution and the benefits are

minimal in this case. Without this load balancing, the benefits of the proposed scheme will only be because of ranking. A scheme that can dynamically measure this skew (by measuring that a particular sub-network is over-provisioned) and can then steer packets to the second sub-network is an interesting avenue of research that has not been explored in this study.

5.6 Chapter Summary

Recently, design and analysis of NoCs has gathered significant momentum because of the criticality of the communication substrate in designing scalable, high performance and energy efficient multicore systems. However, most NoCs have been designed in a monolithic manner without considering the actual application requirements. This proposal argues that such an approach is sub-optimal from both the performance and energy standpoints and propose an application driven approach to designing NoCs. Based on the characterization of several applications, it is observed that a heterogeneous NoC consisting of two separate networks, one customized for bandwidth and other for latency, can cater to the applications' requirement more effectively.

The effectiveness of the proposed two-layer network is evaluated over a range of monolithic designs. Evaluations with 36 benchmarks on a 64-core 2D architecture indicate that the proposed two-layer heterogeneous network approach consisting of a 256b-link (64-link) bandwidth (latency) customized network provides 34%/24% (system/application) throughput improvement over a 128b-link network, and is 5%/3% better in weighted/instruction throughput while consuming 47% lower energy when compared to an iso-resource (320b-link) single network (59% lower energy when compared to a very high bandwidth 512b-link network). In a combined performance-energy design space, the proposed application-driven NoC outperforms all competitive designs. In conclusion, while multiple on-chip networks have been proposed in the literature, none of these are based on a systematic, application-driven approach like this proposal. Also, the proposed communication episode based classification and ranking schemes are significantly better than state-of-the-art NoC prioritization mechanisms.

Chapter 6

Architecting NoCs for Stacked 3D STT-RAM Caches in CMPs

The last three chapters looked at optimizing the NoC for performance and power. This chapter looks at designing an NoC that is cognizant of the presence of cache banks constructed out of a new memory technology, called Spin-Torque Transfer RAM (STT-RAM). After introducing STT-RAMs, this chapter highlights the performance degradation when using traditional NoCs in the presence of STT-RAM cache banks and then looks at schemes in overcoming this performance degradation by advocating a co-design of cache and NoC.

6.1 Introduction

Emerging memory technologies such as Magnetic RAM (MRAM), Phase-change RAM (PCRAM), and Resistive RAM (RRAM) are being explored as potential alternatives to existing memories such as SRAM and DRAM [81, 83, 84, 110–113]. Among these technologies, Spin-Torque Transfer RAM (STT-RAM) [114,115] combines the speed of SRAM, the density of DRAM, and the non-volatility of Flash memory, with excellent scalability. Furthermore, it has been shown that with 3D stacking [116, 117], STT-RAM can be integrated with conventional CMOS logic [87]. Thus, STT-RAM is potentially attractive to replace the traditional on-chip SRAM, with benefits such as higher density and lower leakage compared to traditional SRAM-based cache architecture.

Even though STT-RAM based cache architecture has many advantages, it suffers from a longer write latency and higher write energy consumption compared to SRAM. The write energy and duration is higher since, to write a "0" or "1" into an STT-RAM cell, a strong current is necessary to force the storage node (Magnetic Tunnel Junctions, or MTJ) to reverse the magnetic direction [114, 118]. The latency and energy overheads associated with the write operations of these emerging memories have become the major obstacles of adopting these memory technologies.

In this chapter, the integration of STT-RAM in a 3D multi-core environment is explored and this proposal proposes solutions at the on-chip network level to circumvent the write overheads problem related to STT-RAM based cache architecture for CMPs. The scheme is based on the observation that instead of using the network for sending subsequent requests to a write-busy STT-RAM bank, where they will be queued until the bank is free, the network should serve requests to idle cache banks to hide the write latency. Thus, the network *selectively grants network resources* to the cache requests for accessing idle STT-RAM banks by prioritizing these requests over requests to busy banks. The cache access distributions of different commercial and server workloads are studied and it is observed that on an average, it is possible to delay 17% of accesses to an STT-RAM bank for giving priority to other cache accesses and thereby hide the memory latency.

Despite this simple observation and preliminary analysis showing the potential benefits of selective prioritization of requests in a network, determining which cache banks are busy and which banks are idle in a distributed environment is non-trivial. Thus, the main body of this proposal presents a mechanism for accurately predicting the busy/idle status of a cache bank a few hops away from the destination and a request prioritization mechanism in the routers for effectively utilizing the idle cache banks. The idea is demonstrated using a two-layer stacked architecture of cores and STT-RAMs in different layers using both multithreaded and multiprogrammed workloads across a diverse set of 42 applications.

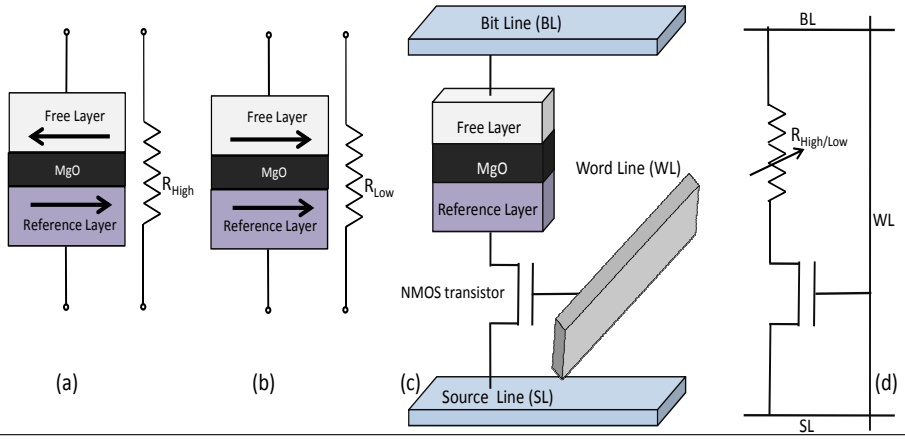


Figure 6.1. MTJ and STT-RAM cell (a) Anti-parallel (high resistance), indicating "1" state (b) Parallel (low resistance), indicating "0" state (c) STT-RAM Structural view (d) STT-RAM Schematic.

6.2 Background

This section gives a brief background on STT-RAM, on-chip networks, and 3D integration technology.

STT-RAM: Unlike the traditional SRAM and DRAM technologies that use electric charges as the information carrier, STT-RAM is based on magnetic characteristics of Magnetic Tunnel Junctions (MTJs) and uses MTJ for binary storage. As shown in Figure 6.1, a MTJ contains two ferromagnetic layers and one tunnel barrier layer (MgO). The direction of one ferromagnetic layer (called reference layer) is fixed, while the direction of the second layer (called free layer) can be changed by forcing a driving current. The relative magnetization direction of two ferromagnetic layers determines the resistance of MTJ. If two ferromagnetic layers have the same directions, the resistance of MTJ is low, indicating a "0" state and vice-versa for a "1" state.

MTJ is the storage element of STT-RAM and a memory cell can be designed using a *one-transistor-one-MTJ* ("1T1J") structure [114, 115]. As illustrated in Figure 6.1, each MTJ is connected in series with an NMOS. The gate of the NMOS is connected to the word line (WL), and the NMOS is turned on if it's connected MTJ needs to be accessed during read or write operations. The source of the NMOS is connected to the source line (SL), and the free ferromagnetic

layer is connected to the bit line (BL). The STT-RAM data read mechanism uses sense amplifiers to sense the voltage difference caused by the resistance difference of MTJ in "0" and "1" status. The read latency of STT-RAM can be as short as the SRAM read latency and the read energy of STT-RAM is also comparable to the SRAM read energy. However, the write duration as well as the write energy of STT-RAM is significantly higher than that of a SRAM write access.

Network-on-Chip (NoC) Architectures: Preliminaries on router architectures are presented in Chapter 2. In this section, only the details relevant to arbitration in routers are re-visited. A VA unit arbitrates amongst all packets requesting access to the same VCs and decides on winners. The SA unit arbitrates amongst all VCs requesting access to the crossbar and grants permission to the winning packets/flits. The winners are then able to traverse the crossbar and are placed on the output links. The two arbitration stages (VA and SA), where a router must choose one packet/flit among several packets/flits competing for either a common (a) output VC or (b) crossbar output port, play a major role in selecting packets for transmission. Current router implementations use simple, local arbitration policies such as round robin (RR) to decide which packet should be scheduled next. This proposal proposes to modify these local and architecturally oblivious arbiters to prioritize packets for hiding the STT-RAM write memory latency.

3D Integration Technology: 3D stacking is a technology that stacks multiple active silicon die on top of each other, and connects them through wafer bonding. The multiple dies communicate through Through Silicon Vias (TSVs) [119]. Among various 3D architectures, stacking cache or memory chips directly on top of a 2D multicore chip [117, 120, 121] has gained its popularity because such architecture provides fast and high memory bandwidth access between the core layer and memory layer. In this paper, the 3D stacking technology is exploited for putting the STT-RAM based cache banks in one layer of the CMP configuration.

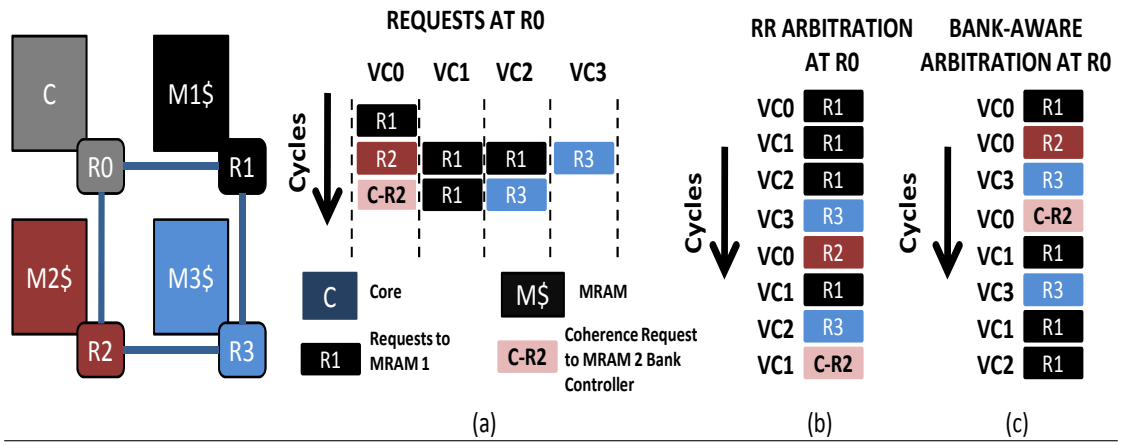


Figure 6.2. (a) Example request sequence at a router (annotated as R0 in the figure) in a 2x2 mesh topology. The resulting arbitration sequence is shown when using (b) simple round-robin arbiter and (c) an STT-RAM bank aware arbiter.

6.3 STT-RAM Aware NoC Design

6.3.1 A Case for STT-RAM Aware Router Arbitration

To motivate the importance of STT-RAM aware router arbitration, consider the example illustrated in Figure 6.2. In this example, the network consists of 4 routers connected in a 2x2 mesh, where router R0 is connected to a processing node (P) and the other routers (R1-R3) are connected to STT-RAM cache banks. Consider the case where R0 receives multiple write requests from P over time as shown in Figure 6.2(a). The resulting arbitration sequence at R0 that employs the traditional RR policy routes multiple requests to R1 before forwarding requests to R2 and R3 (see Figure 6.2(b)). Since writes to STT-RAMs have a long latency, subsequent accesses within a short duration of the first write to the STT-RAM module connected to R1, are queued at the STT-RAM module interface (possibly at the network interface). This STT-RAM oblivious arbitration, not only leads to degraded performance (since banks connected to R2 and R3 remain idle) but also, leads to unwanted network congestion at R1 (which might affect other flows going through R1 in a larger network).

In contrast, with an STT-RAM aware arbitration, R0 can prioritize requests to STT-RAM modules (R2 and R3) over a request to R1. This can be done by being cognizant of the scenario that the module connected to R1 would be busy with the

write request that was sent to it recently. Such a scheme can yield performance improvement by (a) prioritizing requests to idle banks and (b) shifting the buffering of requests to busy STT-RAM modules from the module interface to the network router buffers. An example schedule resulting from the STT-RAM aware arbitration is shown in Figure 6.2(c). This prioritization scheme is specifically beneficial for STT-RAM structures (and possibly for other memory technologies with long bank access times) but not attractive for conventional SRAM cache banks since, write latencies in SRAMs are typically of the order of router and network latencies and delaying a write-request to a busy SRAM bank would not overlap with the service time in the bank and hence, may hurt performance.

6.3.2 The Proposal: Re-ordering Accesses to STT-RAM Banks

Since write-latencies for STT-RAMs (33 cycles) are about 11 times larger than that of router hop latency (3 cycles), requests to banks other than the one recently forwarded with a write request are prioritized. The key intuition behind this is - the bank to which a write request was sent would be busy servicing the request and sending more requests to this bank would only queue them. Thus, not all requests become equally critical from a network stand-point and a router can schedule requests prioritizing few over the others. Another aspect of this re-ordering and selective prioritization can be used to *prioritize packets to memory controllers and coherence traffic*. For example, if the destination of all cache request packets in a particular router are to an already busy cache bank, coherence packets and packets destined to memory controllers can be prioritized (see Figure 6.2(c)) to boost performance.

Next, one needs to address two critical questions (1) *How* long should a packet be delayed? and (2) *Where* should a packet be re-ordered? i.e. how far from its destination. Ideally, a packet to a busy bank should be delayed such that it arrives at the busy bank immediately after the previous write request has been serviced. Thus, the network and queuing delays of the subsequent (delayed) packets should overlap with the service time of the first packet to avoid performance degradation. The ability to detect the busy duration of STT-RAM banks in the network is

critical to achieving this overlap. If one were to prioritize and re-order requests in a router far away from the requests' destination, then estimating the congestion in the network and busy time of destination cache bank becomes difficult. In contrast, if one were to do the re-ordering of packets from a router very close to the destination cache bank, then this particular router would not have many requests to re-order and hence, would be forced to route a packet to a busy cache bank (only to be later queued at that bank). After extensive sensitivity analysis, the proposed scheme chooses to selectively delay request packets in a router whose destination is a busy STT-RAM 2-hops away. Qualitatively, there are two reasons behind this choice: (1) Estimating busy time of a destination bank that is 2-hops from the current router is relatively easy (shown later in Section 6.4.2)) and (2) It gives us significant opportunity to prioritize coherence, memory-controller destined traffic and any other packets destined to routers more than 2-hops away.

Two factors that decide the success of the re-ordering/prioritizing scheme are - (1) How separated in time (cycles) are two consecutive accesses to a cache bank. For example, if the inter-arrival gap of successive writes to the same bank is much larger than 33 cycles, there is not enough scope for our approach. (2) Number of different cache banks to which requests are buffered in a router indicates the potential of the re-ordering scheme (and also dictates our decision in choosing 2 hops as the distance for re-ordering). For example, if all requests in a router are destined to a particular bank, then re-ordering those requests is not useful. The following subsection analyzes these two factors.

6.3.3 Cache Access Distribution

For different applications, the distribution (in cycles) of consecutive accesses to STT-RAM banks and the average number of request packets buffered in a router in the cache layer, whose destination is 2-hops away from the current router is analyzed. In this analysis, there are 64 cores in one layer and 64 STT-RAM banks in the other layer Figure 6.3 shows the result of this analysis, where the vertical axis represents the percentage of all accesses to a cache bank following a write access to that bank and the horizontal axis corresponds to latency in cycles. The graphs are plotted with 33 cycle intervals (except the first bin), with the leftmost

bar indicating the percentage of accesses that are separated by $0 <$ latency < 16 cycles after a write request. The rightmost bar represents the percentage of all accesses that are separated by more than 165 cycles.

This plot quantitatively shows the burstiness in applications. For `ferret`, around 8% of the accesses occur within 16 cycles after a write request is initiated to a cache bank and 10% of the accesses within 33 cycles after a write request is initiated. Considering that write service times in STT-RAMs last 33 cycles (Table 6.1), all these subsequent requests inevitably get queued in the network-interface of the router or the bank controller before getting serviced. Requests that are separated by at least 33 cycles (after a write request to a cache bank), are not queued and can directly proceed to be serviced. These requests are represented in the 66, 99, 132 and 165+ bins in the histograms shown. The figure also shows that not all applications are bursty. For instance, requests in `x264` are spread out and the percentage of requests following a write request to a bank that can potentially get queued (i.e. in bins 16 and 33) is only 4%. However, across all applications that are analyzed, on an average, 17% of requests are always queued behind a write operation. Figure 6.3 also shows the number of request packets in a router in the cache layer whose destination is exactly 2-hops away and that follows a write access request. It is also observed that, almost always there are about *3 requests* in a router following a write packet, which if scheduled soon after the write request would end up getting queued in the STT-RAM banks. These are the requests packets that can potentially be *delayed* to *hide* the long latency write operations of STT-RAMs.

6.3.4 Facilitating Prioritization

To prioritize requests to idle banks and delay requests to busy banks, a router should be able to know which banks in its vicinity are idle and which banks are busy. One way of achieving this, is to have a global network that transmits this information to every router in the network on a cycle by cycle basis. This approach would, however, be highly expensive for resource and power constrained NoCs. An alternate approach, could be to route all packets destined to a particular cache bank through a particular router in the network - a serialization point. This serialization

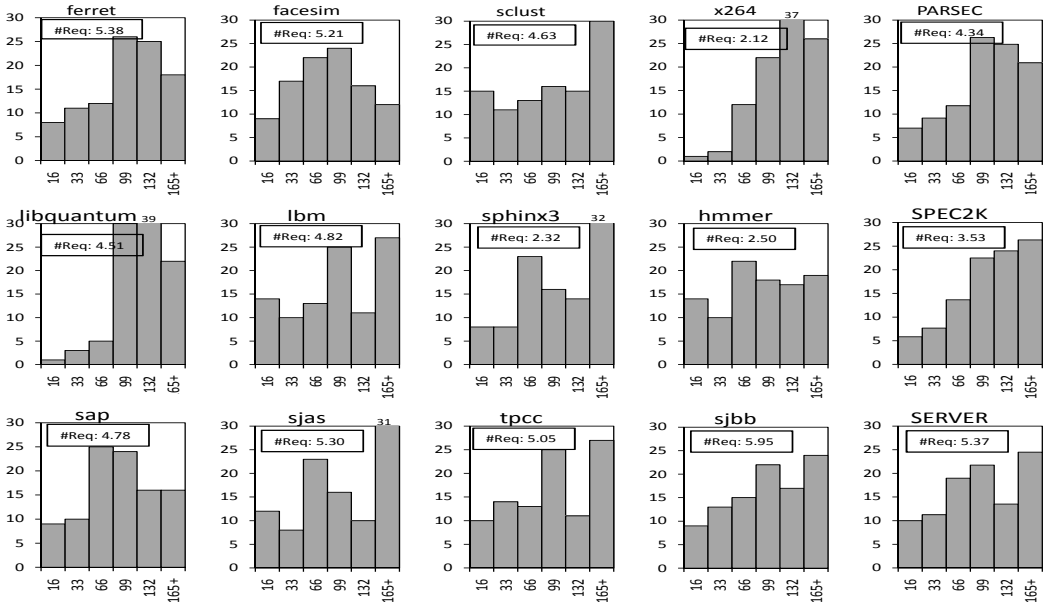


Figure 6.3. Plots showing the distribution of consecutive accesses to STT-RAM banks in different applications following a write access (the last column shows the average across the whole benchmark suite). The horizontal axis represents the access latencies in cycles and the vertical axis represents the percentage of access. The inset in each plot mentions the average number of request-packets in a router in the cache layer to 2-hop away STT-RAM locations.

point can serve as a secondary source for the cache bank to which all other nodes wanting to communicate with the cache bank would send their requests. However, this is not the case in current implementations of 3D NoC. In a typical 3D network, each router in the core layer is connected to a router in the cache layer below. When using a deterministic routing algorithms like Z-X-Y or X-Y-Z routing, owing to the path-diversity, each cache may receive requests from various cores along different routes. For example, consider the illustration shown in Figure 6.4 with the core layer being on top of the cache layer and core 0 connected to router 0, core 1 connected to router 1 and so on. If core 0 wants to send a request to cache node 64, the request packet would be routed vertically downwards. If core 63 wants to send a request to cache node 0, assuming Z-X-Y routing, the request packet is routed vertically to router 127, followed by X-direction routing from router 127 to router 120 and then Y-direction routing to router 0. Since these two routes never overlap, there is no single point in the network, where a router can re-order requests if cache bank 64 were to be busy.

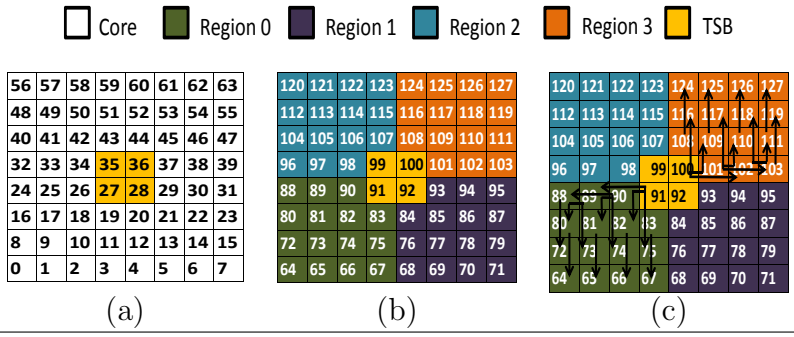


Figure 6.4. Two layers of the 3D CMP: (a) Core Layer (b) Cache Layer (c) Cache layer showing the child nodes of parent nodes.

This chapter introduces a novel scheme to provide such a serialization point in the network. The proposed scheme involves (1) dividing the cache layer into logical regions and (2) limiting the path-diversity in the 3D NoC.

Partition the cache layer for reducing path-diversity: The cache layer consisting of 64 cache banks is partitioned into a few logical regions - 4 to demonstrate the concept. One vertical through-silicon bus (TSB)[1] is designated in each region through which all cores can communicate their request packets with any cache bank in the corresponding region in the cache layer. Figure 6.4 shows the logical partition and the 4 TSBs connecting one router in the core layer to one router in the cache layer. Use of X-Y routing together with these TSBs, serializes packets to routers in each region. Hence, a packet routed from the core layer to the cache layer is first routed using X-Y routing to a particular router in the core layer, followed by a TSB traversal in the vertical direction to a router in the cache region. Finally, the packet follows X-Y routing again in the cache layer to the destination cache bank. No such path restriction is imposed when communicating between the cache layer to the core layer i.e. all 64 TSBs can be used in this case. Also, coherence traffic is not constrained to go through the 4 TSBs alone and can use all the 64 TSBs.

Note that the innermost corner router in a region does not form a serialization point for *all* traffic in that region. Each router manages traffic for all 2-hop away routers in the region. The router that manages traffic for the 2-hops away destination STT-RAM bank is called a *parent node* and the routers connected to the

[1]We call a collection of TSVs to be a TSB

STT-RAM bank 2-hops away from the parent node are called *child nodes*. Thus, few routers in each region serve as the parent nodes, where the prioritization of core to cache packets occur and these parent nodes have a predicted estimate of the busy times of 2-hops away bank from them (Section 6.3.5 describes this prediction schemes).

Clearly, restricting the routes from the core layer to the cache layer would increase the hop count and hurt performance. To reduce the performance penalty of increase in hop count, instead of sending one flit at a time through the core to cache layer TSBs, the proposal is to combine few flits and transmit them simultaneously. The design is similar to the XShare technique proposed in [10], where a NoC router combines two small flits (coherence and header flits) and sends them over the link to the next router. However, in this proposal, whenever possible, two 128b data flits are combined and transferred simultaneously over the high density TSB (256b in our design).

Putting it all together: Figure 6.5 shows our resulting 3D NoC design. In this design, all communications from the core to the cache layer occur through the 4 high density TSBs (256b), while communication from the cache layer to the core layer can use all of the 64 128b TSBs.

This figure shows the path taken for request generating at cores 7, 46 and 48 to communicate with cache bank 89, 82 and 75, respectively. For all these requests, since the destination cache node lies in region 0, they are first routed using X-Y to node 27 in the same layer, followed by a vertical TSB transaction, and finally being buffered in the secondary source router 91. The router 91 now sees all the requests and since it is the only router through which all processor requests to routers 75,82 and 99 pass, it has a estimate of which of these cache banks are currently busy or idle and thus, can selectively prioritize these requests to them. The next section describes how each parent router knows an estimate of busy times of its child nodes. This helps the parent node delay a request packet to the child node as long as the child node is busy.

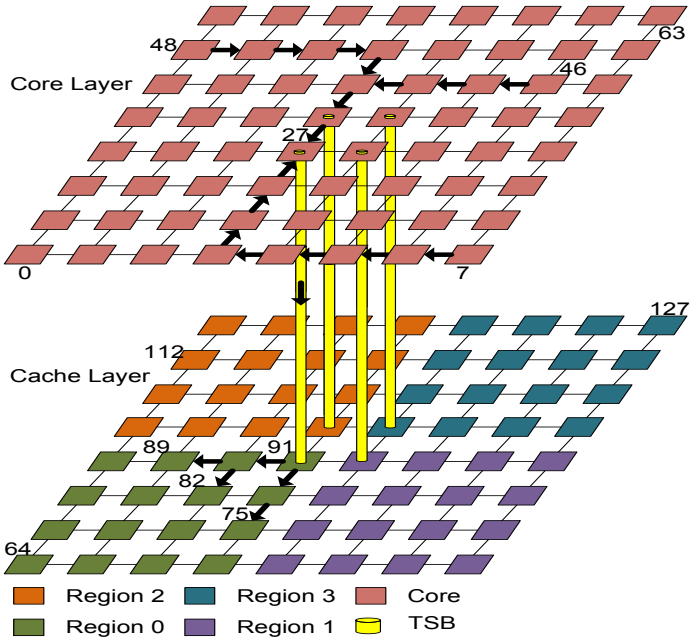


Figure 6.5. Proposed 3D architecture with cores in the top layer and STT-RAM banks (partitioned into 4 logical regions) in the bottom layer. The bold arrows show the route taken by requests from core to cache bank.

6.3.5 Estimation of Busy Time

Restricting the path diversity by allowing only 4 TSBs to be used when communicating from core to the caches and using X-Y routing in the cache regions, helps each parent node estimate the busy times of its child nodes. This is because, each child node only receives request from its parent nodes except the coherence traffic, which can be received from any router in the network. Also, ideally a parent node should delay a request to its child node such that, as soon as the child node is done servicing a request another request arrives from its parent node. The latency of a packet from a parent router to the 2-hop away destination consists of router delay, link traversal delay and delay due to congestion. Now, since the destination is two hops away, there is one intermediate router with 2 cycles delay (we assume a 2-stage router) and 2 cycles link traversal (1 cycle each) delay. The only unknown component is the congestion at the intermediate and destination nodes. Thus, each parent node should delay a request by 4 cycles + estimated congestion cycles + write service time in the STT-RAM bank (= 33 cycles) *following* a write-request. The delay is kept track using counters and busy-bits are maintained for each child

nodes to identify a busy cache bank. This proposal looks at three heuristics for estimating this congestion:

Simplistic Scheme (SS): In the simplistic scheme, the parent node delays a request packet to a *busy* cache bank for 33 cycles following a write-request by ignoring the congestion. While delaying packets for this duration, the parent node prioritizes packets destined to other parent nodes, coherence traffic and traffic to memory controllers. Clearly, in this scheme, since congestion is not modeled, a packet is not sufficiently delayed when congestion at the destination bank is significant and arrives at its destination only to be queued.

Regional Congestion Aware (RCA) scheme: In the RCA scheme, information from neighboring nodes is used to estimate congestion. This scheme is based on Grot et al.'s [60] scheme, where a coarse view of regional congestion can be obtained by aggregating congestion estimates from neighboring routers. The RCA scheme requires additional wires for propagating the congestion estimates among neighbors and based on [60], 8 bit extra wires are used between each node. Among the three schemes presented here for estimating congestion, RCA provides the best estimate albeit at the cost of additional wires.

Window Based (WB) scheme: The third scheme is a novel window based scheme that does not require any back wiring overheads. In this scheme, for every N packets, the parent node tags a packet with a B-bit time stamp and starts a B-bit counter before despatching it to the destination node. The destination (child) node, after receiving the tagged packet, sends an ACK packet together with the time stamp it received back to the parent node. The B-bit counter is updated every cycle until an ACK packet with the B-bit time stamp is received by the parent node from the child node. After receiving the ACK packet, the parent node estimates the congestion as half of the difference between the current time and the time stamp received. This scheme is similar to the window based scheme used in TCP/IP protocol except that the size of the window is just 1. In this proposal, N=100 and B=8 is chosen and the time stamp is appended with the header flit. A header usually carries source-destination information and is typically shorter (64b) than data flits (128b) and appending 8 bit time stamp to the header flit does not require additional wires nor introduces any significant overheads. Congestion can vary with program phases and analysis shows that updating the congestion

Table 6.1. Comparison of SRAM and STT-RAM's architectural characteristics

	Area (mm^2)	Read Energy (nJ)	Write Energy (nJ)	Leakage Power at 80°C (mW)	Read latency (ns)	Write latency (ns)	Read @3GHz (cycles)	Write @3GHz (cycles)
1MB SRAM	3.03	0.168	0.168	444.6	0.702	0.702	3	3
4MB STT-RAM	3.39	0.278	0.765	190.5	0.880	10.67	3	33

information every 100 packets provides reasonably accurate congestion estimates. The overhead in WB scheme is that of maintaining B-bit counters in each router and communicating ACK (1-bit) messages. A synthesized implementation of the counter scheme implemented in Verilog shows minimal gate count increase in each router and hence, the WB scheme is simpler to *implement* in the network.

6.4 Experimental Evaluation

6.4.1 Experimental Platform

Architecture Model: The proposed schemes are evaluated using a trace-driven, cycle-accurate x86 CMP simulator discussed earlier in Chapter 2. A message (packet) consists of eight 128-bit flits and one header flit. 3.5 mm^2 area for each core is assumed at 32 nm technology based on a scaled version of Sun's Rock core [122]. The STT-RAM die, placed directly below the core die, is partitioned into 64 banks. Based on estimates from CACTI 6.0 [123], a 1 MB SRAM cache bank and 4MB STT-RAM bank and associated router/controller have an area roughly equal to the area of one core as shown in Table 6.1. The 3D TSV model parameter is based on [83].

STT-RAM Model: To model the STT-RAM based cache memory, the technology parameters reported in [83, 87, 114, 115] are used. Because the critical current will increase dramatically when write pulse duration is shorter than 10ns [114], a write pulse duration is constrained to be within 10ns.

Benchmarks: 42 applications consisting of *multi-threaded* PARSEC, four commercial workloads and, *multi-programmed* SPEC 2006 benchmarks are used in the analysis. For each application, at least 50 million instructions (3200 million instructions across the 64 processors) are simulated. Table 6.2 shows the characterization

Table 6.2. Benchmark Table: l1mpki: L1 misses per 1000 instructions, l2mpki: L2 misses per 1000 instructions, l2wpki: L2 writes per 1000 instructions l2rpki: L2 reads per 1000 instructions, Bursty: (High/Low) based on latency between 2 consecutive requests to a L2 bank.

No.	Benchmark	l1mpki	l2mpki	l2wpki	l2rpki	Bursty
1	tpcc	51.47	6.06	40.9	10.57	High
2	sjas	41.54	4.48	35.06	6.48	High
3	sap	29.91	3.84	23.57	6.15	High
4	sjbb	25.52	7.01	19.42	6.09	High
5	streamcluster(sclust)	29.28	8.34	15.23	14.05	High
6	vips	13.51	8.07	6.61	6.89	High
7	canneal	12.8	5.47	6.52	6.27	Low
8	dedup	12.8	4.59	7.42	5.36	High
9	ferret	11.62	9.16	6.39	5.22	Low
10	facesim	10.62	6.82	6.15	4.46	Low
11	swaptions (swptns)	5.47	6.35	2.46	3.00	Low
12	blackscholes (bscls)	5.29	3.73	2.80	2.48	Low
13	bodytrack (bdtrk)	5.62	5.71	2.81	2.81	Low
14	raytrace (rtrce)	5.65	4.98	3.62	2.03	Low
15	x264	4.17	4.62	1.87	2.29	Low
16	fluidanimate (fldnmt)	4.89	4.41	2.68	2.2	Low
17	freqmine (frqmn)	2.29	3.96	1.31	0.98	Low
18	gemsfdtd	104.04	94.62	0.8	103.23	Low
19	mcf	99.81	64.47	5.45	94.37	Low
20	soplex	48.54	16.88	19.59	28.95	Low
21	cactus	43.81	15.64	18.65	25.16	Low
22	lbm	36.49	18.88	30.76	5.73	High
23	hmmer	34.36	3.31	12.5	21.86	High
24	xalancbmk	29.7	21.07	3.02	26.68	Low
25	leslie	26.09	18.06	7.65	18.45	Low
26	sphinx	25.55	10.91	0.97	24.58	High
27	gobmk	22.81	8.68	8.02	14.79	High
28	astar	20.03	4.21	6.11	13.92	Low
29	bzip2	19.29	10.02	2.66	16.63	High
30	milc	19.12	18.67	0.05	19.06	Low
31	libquantum	12.5	12.5	0	12.5	Low
32	omnetpp	10.92	10.15	0.25	10.67	Low
33	povray	9.63	7.86	0.88	8.75	High
34	gcc	9.39	8.51	0.06	9.34	High
35	namd	8.85	5.11	0.65	8.19	High
36	gromacs	5.36	3.18	0.32	5.05	High
37	tonto	5.26	0.55	3.52	1.74	High
38	h264	4.81	2.74	2.03	2.78	High
39	dealII	4.41	2.36	0.35	4.06	High
40	sjeng	3.93	2	0.92	3.01	Low
41	wrf	1.8	0.75	0.88	0.92	Low
42	calculix	0.33	0.23	0.03	0.29	Low

of the application suite. The reported parameters are for the applications running alone on the baseline CMP system with the L2 composed of STT-RAM cells.

Design Scenarios: To show the benefits of our proposals, the following six design scenarios are analyzed in the experimental platform:

- **SRAM-64TSB:** This is the baseline scheme where all L2 cache banks are

composed of SRAM cells and where there is no restriction on path diversity i.e. all 64 TSBs are used for communication from core layer to cache layer.

• **STT-RAM-64TSB:** This scheme is similar to SRAM-64TSB except that now all L2 banks are composed of STT-RAM cells.

• **STT-RAM-4TSB:** This scheme is similar to STT-RAM-64TSB scheme, except that only 4 TSBs are used when communicating between the core layer and the cache layer i.e. request packets. All 64 TSBs are still used when communicating from the cache layer to the core layer (response packets). This scheme is analyzed to quantify the performance degradation solely due to restriction in path-diversity.

•**STT-RAM-4TSB-SS:** In this scheme, all L2 cache banks are composed of STT-RAM cells, only 4 TSBs are used for request packets from the core to cache layer and we employ selective prioritization of requests to idle banks. This scheme employs the Simplistic Scheme (SS) for estimating congestion in the network.

• **STT-RAM-4TSB-RCA:** This scheme is the same as STT-RAM-4TSB-SS with the exception that RCA is employed to estimate congestion in the network.

• **STT-RAM-4TSB-WB:** This scheme is similar to STT-RAM-4TSB-SS with the exception that the window based (WB) scheme is employed to estimate congestion[2].

6.4.2 Experimental Results

Performance Comparison: Figure 6.6 shows the IPC and instruction throughput improvements for a subset of applications evaluated (the average, however, is taken across all applications). For clarity in the figures, all the reported numbers are normalized to the IPC (or instruction throughput) with the baseline design (SRAM-64TSB). It is seen that, with STT-RAM-64TSB, benchmarks having a high percentage of reads to the L2 benefit in performance. This benefit comes due to the 4x increase in size of an L2 bank when SRAM is simply replaced by STT-RAM. However, for benchmarks that have a high percentage of writes to the L2, simply replacing an SRAM bank with an STT-RAM bank, hurts performance. This, in spite of the 4x capacity increase of the STT-RAM banks, is because of the high write latency of the STT-RAM cells. In the analysis, all server bench-

[2]For clarity, STT-RAM is annotated as MRAM in all of the plots.

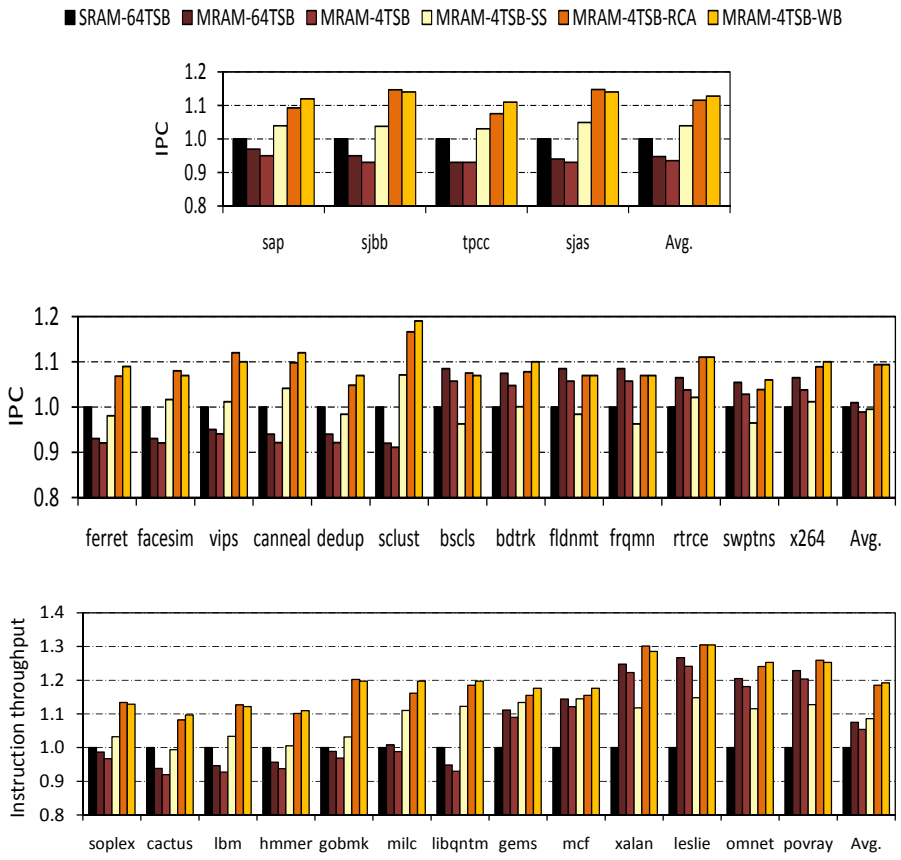


Figure 6.6. System throughput of the benchmarks normalized to the SRAM-64TSB case (from top to bottom: IPC for server and PARSEC benchmarks and instruction throughput with multi-programmed SPEC-2006) .

marks, six benchmarks from the PARSEC suite and seven benchmarks from the SPEC-2006 suite show performance degradation when SRAM banks are replaced by STT-RAM banks. With STT-RAM-4TSB, due to restriction on path diversity for request packets, performance further degrades when compared to STT-RAM-64TSB. However, for L2 read intensive benchmarks, performance is still better compared to SRAM-64TSB (solely due to 4x capacity increase with STT-RAM banks). With STT-RAM-64TSB, there is 6% IPC degradation across all server benchmarks and only 1% performance improvement with PARSEC benchmarks. Additionally, with the SPEC 2006 workloads, there is just 7% improvement in instruction throughput. Clearly, simply replacing SRAM with STT-RAM cells does not translate to significant performance improvement in spite of the 4x capacity increase.

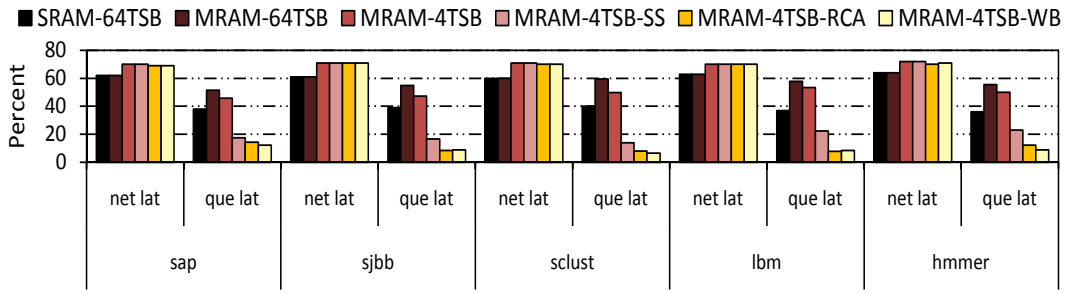


Figure 6.7. Packet latency breakdown into network latency (net lat) and queuing latency at memory banks (queue lat). SRAM-64TSB are exact percentages. All other values are normalized to that of SRAM-64TSB.

It is observed that, if a benchmark is not very bursty, then the STT-RAM-4TSB-SS scheme, in spite of using only 4 TSBs for sending requests from the core to cache layer, on an average improves performance. However, if an application is bursty, STT-RAM-4TSB-SS scheme is not able to manage the requests to busy and idle banks well enough due to the lack of congestion estimation among parent and child nodes in this scheme. On an average, the STT-RAM-4TSB-SS scheme improves performance by 2.5% across benchmarks evaluated. If an application is read intensive, STT-RAM-4TSB-SS shows performance improvement over simply using STT-RAM-64TSB, since, even though STT-RAM-4TSB-SS has no built-in congestion estimation, selectively delaying the accesses to a bank following a write request to the same bank in the application helps improve performance. Performance is substantially improved when STT-RAM-4TSB-RCA and STT-RAM-4TSB-WB schemes are employed. These schemes by virtue of better congestion estimation and prioritization of requests to idle banks, improve the performance significantly over SRAM-64TSB. STT-RAM-4TSB-RCA proves to be slightly (1%) better than the STT-RAM-4TSB-WB scheme. Since, the RCA scheme has overheads of backwiring, the WB scheme works well across all applications with performance close to the RCA scheme. On an average, there is 14.5% and 9% IPC improvement with server and PARSEC benchmarks, respectively, and 19.5% improvement in instruction throughput with SPEC-2006 benchmarks.

Network Packet Latency: To further investigate the reason behind performance benefits, the reduction in network packet latency for a few benchmarks is analyzed. Figure 6.7 shows the results of this analysis, where a packet's latency is broken

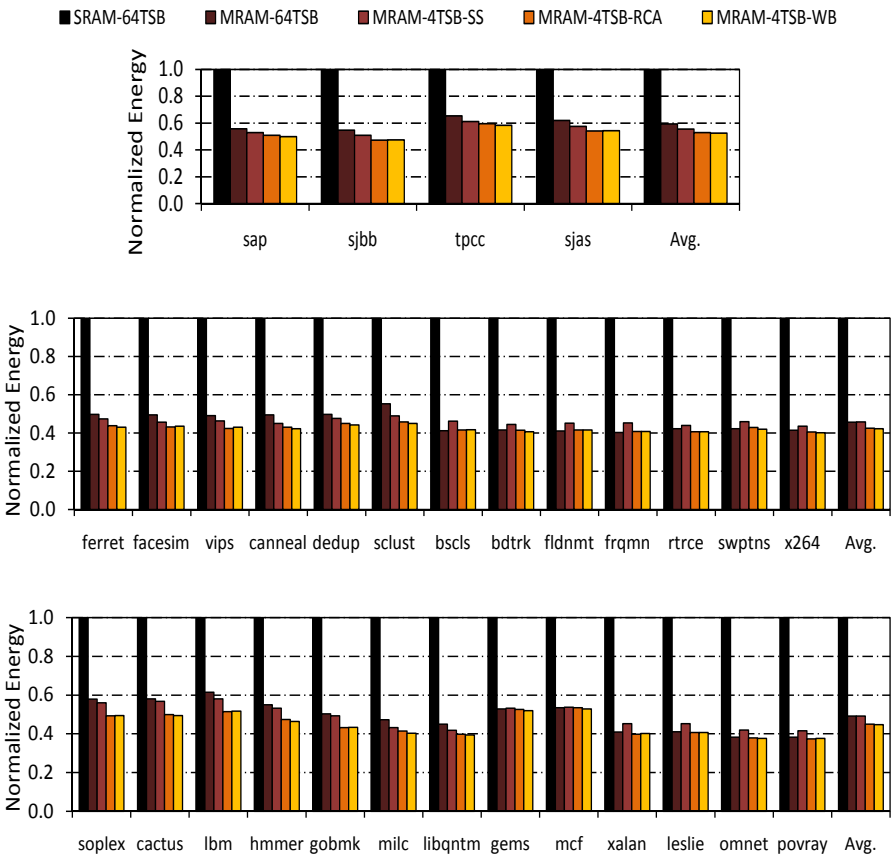
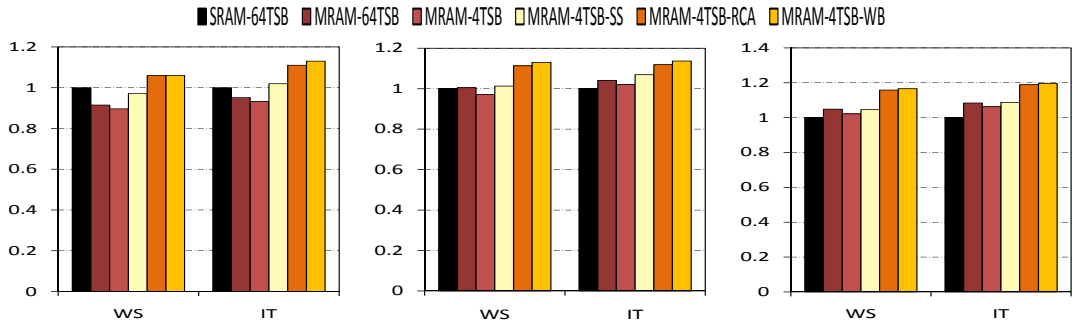


Figure 6.8. Energy of the benchmarks normalized to that of SRAM-64TSB.

down into network latency (router latency and network congestion latency) and queuing latency at memory banks. As can be observed, that queuing latency constitutes a significant portion of a packet's latency and this component worsens (average 8%) by replacing the SRAM banks with STT-RAM banks due to the increased write latency. However, by prioritizing requests to idle banks and estimating congestion accurately within a buffered request's vicinity, all of the three schemes reduce the queueing delay component up to 35% in the network.

Energy Saving: Figure 6.8 shows the savings in un-core (interconnect and cache) energy with the three schemes. The energy numbers are normalized to the energy of the baseline design (SRAM-64TSB). Energy is reduced solely due to the use of low leakage energy STT-RAM cells and all of the three schemes show similar energy savings. On an average, there is 54% reduction in energy across all workloads with the STT-RAM-4TSB-WB scheme. Reduction in energy consumption depends on



(a) Case 1 (b) Case 2 (c) Aggregate across 32 cases

Figure 6.9. Weighted speedup (WS) and instruction throughput (IT) for the multiprogrammed workloads.

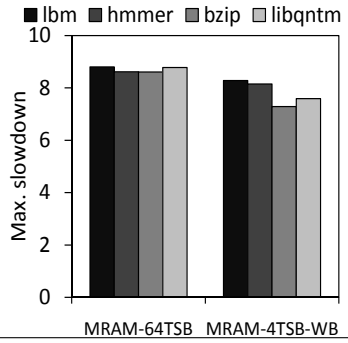


Figure 6.10. Maximum slowdown of the applications in Case-2.

the write intensity of the applications - applications with high write intensity to L2 show less reduction in energy compared to less write intensive applications.

Performance analysis with multi-programmed workloads: Next, the effectiveness of the schemes is evaluated using three diverse multi-programmed workload combinations:

• **Case-1:** 16 copies each of 4 write intensive applications (`soplex`, `cactus`, `lbm` and `hmmer`) are mixed - a worst case scenario when SRAM banks are replaced by STT-RAM banks.

• **Case-2:** It consists of 16 copies each of 2 bursty and write intensive (`lbm` and `hmmer`), and 2 read intensive (`bzip` and `libqntm`) application mixes.

• **Case-3:** It is a realistic case consisting of an aggregate of 32 workload mixes, where each workload has a mix of 8 applications (8 copies each); each of the 32 workload mix is spread out across the design space: 8 workloads represent read

intensive applications, 8 workloads represent write intensive applications and 16 workloads represent combination of read, write and compute intensive applications. Within each of these three categories, applications are *randomly picked* to form the mixes.

Figure 6.9 shows the weighted speedup and instruction throughput improvement of our schemes across the three cases compared to the baseline. Observations from these evaluations are:

• In Case-1, when all write intensive applications are co-scheduled on the same CMP, there are no speedup improvements when SRAM is simply replaced by STT-RAM. On the other hand, WB scheme by selectively prioritizing only those packets for which *useful* work gets done, helps achieve 6% and 13% improvement in weighted speedup and instruction throughput, respectively. This case study highlights the importance of this proposal: in a worst case but quite probable scenario, the long write latency of STT-RAM cells can degrade performance by as much as 9% (weighted speedup with STT-RAM-64TSB), however, by intelligently scheduling requests on to the banks using the NoC, this performance degradation can not only be amortized, but also can lead to performance improvement (7% weighted speedup with STT-RAM-4TSB-WB).

• In Case-2, when write and bursty, and read intensive applications are co-scheduled, the performance improvements with the schemes are again better. However, an important observation made in this case study is that: when SRAM is replaced by STT-RAM, the bursty applications hog the network resources and consequently the read intensive applications are unfairly slowed down. This is highlighted in Figure 6.10, which plots the maximum slowdown experienced by each application in the workload (smaller max. slowdown is better) for the STT-RAM-64TSB and STT-RAM-4TSB-WB schemes. The slowdown experienced by the read intensive applications (`bzip` and `libqntm`) is similar to what the write-intensive applications experience despite the fact that the read intensive benchmarks have lower miss-rates than the write intensive applications. This is because, the write intensive applications in this workload are bursty and owing to longer write latency of their request packets, the read intensive applications' packets are not granted network resources by an un-oblivious STT-RAM router. However, using the WB scheme, when the destination STT banks are busy serving writes, the

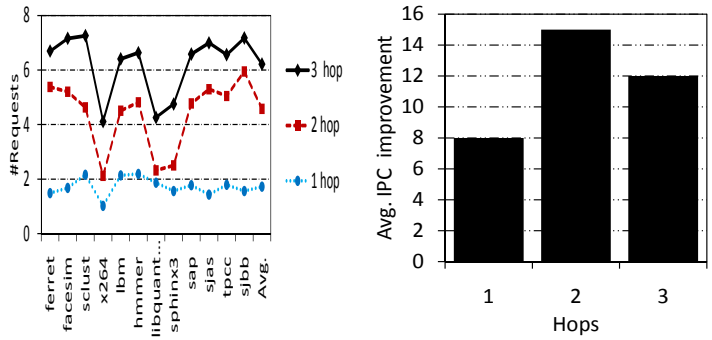


Figure 6.11. Average number of request-packets in a router in the cache layer to 1-hop, 2-hop and 3-hop away STT-RAM destination and sensitivity to hop distance.

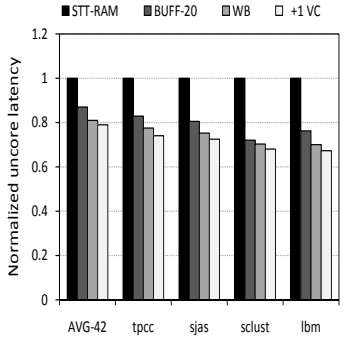


Figure 6.12. Latency reduction comparison with write-buffering (normalized to STT-RAM with no write-buffering).

read packets from read intensive applications are prioritized over write packets and this decreases their max. slowdown by 14%. Thus, the proposed schemes apart from improving performance, also aid in establishing a certain level of *fairness* across applications co-scheduled in the CMP.

• Finally, in Case-3, the aggregate results obtained are consistent with the previous performance numbers. When averaged across 32 multiprogrammed workloads, WB improves system throughput by 16% and instruction throughput by 19% compared to SRAM based cache design.

In summary, from among the six design alternatives, the WB scheme provides the best system performance and energy efficiency over a wide variety multi-threaded and multi-programmed workloads. In addition, it also introduces a level of fairness among the applications which neither SRAM nor STT-RAM banks can provide.

Sensitivity to distance between parent and child nodes: For this analysis,

the sensitivity of prioritizing requests from parent routers H hops away from the child nodes is analyzed in detail. Figure 6.11(a) shows the number of potential requests in a parent node H-hop away from the child node when H varies from 1-3 hops across a few applications from the benchmark suite. It is seen that, if a parent node is too close to child nodes, then the opportunity for re-ordering reduces since the potential number of requests that can be re-ordered is few. When H is 3, each parent node has four child nodes, and hence, the potential number of requests that can be re-ordered increases. However, if re-ordering is done from 3 hops away (or farther) routers, then congestion estimation is not very accurate and this affects the delay of individual requests to busy child nodes (shown in Figure 6.11(b)). After an extensive sensitivity analysis, it is found that for $H = 2$, the number of requests to re-order is ideal for accurate congestion estimation, and all the aforementioned results are thus based on 2-hop congestion calculation.

6.4.3 Comparison with a SRAM Write Buffer Scheme

A previous work by Sun et al. in [83] evaluated the benefits of adding a small SRAM write buffer to *each* STT-RAM bank for hiding long latency writes. In their implementation, the cache bank controller services a write request by writing it into a SRAM buffer and when the bank is idle, the writes are written back to the STT-RAM bank. On a read request, the address is searched in parallel in the write-buffer and also in the STT-RAM bank for a hit. The authors found a 20 entry write-buffer as the optimal SRAM buffer sizes for STT-RAM banks. A large entry write buffer increases the design complexity as well as the area overhead. Additionally, the larger the buffer is, the longer it takes to check whether there is a hit in the buffer and then to access it. Large entry write buffers prove beneficial only when a benchmark is bursty. For comparison purposes, the benefits of using WB scheme with a 20-entry write buffer for a few bursty and write intensive benchmarks are evaluated. Figure 6.12 depicts the results of this comparison, where the normalized latency of the un-core part of processor requests (i.e. the round-trip latency from a core to a cache bank through the network and back to the core) is shown. Observations from this analysis follow in order:

- The uncore latency of our scheme (annotated as WB in the figure) is, on

an average, 6% better than that with STT-RAM banks having a 20-entry write buffer (annotated BUFF-20 in the figure). With the write-buffering scheme, the 1-cycle overhead in detecting if a request is a read/write before being written into the write-buffer is in the critical path of every request.

• A write-buffer is not beneficial when the benchmark is read intensive and/or is not bursty. In such a case, the write-buffer contributes to area and latency (for 1-cycle detection) overhead and there is negligible performance improvement. In fact, when averaged across 42 applications (annotated as AVG-42 in the figure), BUFF-20 only reduces uncore latency by 12.5% (only 2.25% with read intensive benchmarks), whereas the WB scheme reduces latency by 19%. Moreover, for bursty and write intensive benchmarks (e.g. tpcc, sjas, sclust and lbm), the WB scheme is 6% better than BUFF-20 since the WB scheme does a better job of managing the writes and burstiness in the network.

• Coupled with the write-buffering scheme, Sun at al. proposed a read preemption scheme that can preempt a long-latency write request. In this case, an un-finished write-request is stored in an additional buffer. We find that, with bursty applications, preemption occurs more frequently than desired, thereby becoming an overhead.

• Instead of having a 20 entry write-buffer for every STT-RAM bank, if the number of VCs in the routers is increased by one (result annotated as +1 VC in the figure), there is an opportunity for an additional 1.6% latency reduction (compared to the WB scheme) with a 97% area reduction over the BUFF-20 scheme. This is because more flows are benefited when resources in the network are increased as opposed to increasing resources at the end-points.

• Buffering at the end points eliminates any possibility of providing application level fairness which the WB scheme can provide as shown earlier.

In conclusion, buffering request intelligently in the network is more beneficial (and practical to implement since on-chip routers are already equipped with buffers) than buffering them at the end-points.

6.5 Chapter Summary

STT-RAM posses many attractive characteristics such as fast access time and low standby power. Along side, the emerging 3D integration technology provides a cost-efficient way to integrate STT-RAM with multicore architectures/CMPs. However, one of the systemic disadvantage of STT-RAM technology is the latency associated with the write operations. This work proposes an elegant network level solution to alleviate this problem. The proposed network level solution centers around prioritizing packets to idle banks and delaying accesses to an STT-RAM bank currently servicing a write request. Studies show that by accurately estimating the busy status of a cache bank two hops away using the WB estimation, a router can prioritize requests to idle banks for effectively hiding the long write latency. Another important conclusion of this study is that the proposed network tailored solution is more efficient (in terms of performance and resource overheads) compared to a recently proposed write-buffer mechanism for hiding write latency.

Chapter 7

Conclusions and Future Work

With many-core processors becoming mainstream, one of the challenges going forward is on interconnecting these processors in an efficient way. As such, on-chip networks are going to have a significant impact on performance and power of the overall chip. The premise of this dissertation is to exploit *heterogeneity* at the network, application and technology level for improving performance and reducing the overall power envelope.

7.1 Summary of Dissertation Contributions

The research proposed in this dissertation has five major contributions. First, this dissertation starts with a preliminary notion of heterogeneity in on-chip networks, and it quantifies the non-uniform resource consumption in NoCs. Specifically, non-uniform buffer and link usages are quantified for a non-edge symmetric network that uses X-Y routing. The analysis is further extended to make the observation that not all routers in the network are equally stressed or utilized by applications. This analysis then looks at the inherent network demands of applications and makes the observation that not all applications require similar resources from the underlying interconnection substrate. Based on these observations, the first chapter proposes to factor heterogeneity at the network, application, and at the technology level for improving performance and reducing the overall power envelope in NoCs and memory subsystem.

Second, the RAFT design is proposed, which exploits heterogeneity in the

buffers of the on-chip routers. This design is the first of its kind to use a variable-frequency scheme that helps alleviate congestion in the network and also reduce average network power [30, 94]. The key idea in this design is to operate individual routers in the network at frequency levels that is proportional to the load in the routers. This approach throttles low utilization routers and boosts the frequency of highly utilized routers int the network. After extensive analysis, it is shown that RAFT and few of its variants can significantly lower the network power consumption and improve system throughput by efficiently managing congestion.

Third, the HeteroNoC design is introduced, which targets non-uniformity in both buffers and links in the on-chip networks and proposes a heterogeneous network for future CMPs. The key idea in this proposal is to allocate more resources to routers that are highly utilized in the network and reduce resource allocations from routers that are less utilized in the network. The dissertation discusses the design space of heterogeneous networks in terms of choosing the size, number and placements of heterogeneous routers in the network. Further, it is observed that, it is not always sufficient to allocate more resources to highly utilized routers, rather more resources should be allocated to those routers in the network which on an average are more used by network flows. To further demonstrate the applicability of HeteroNoC design in CMPs, a co-design of HeteroNoC with heterogeneous cores is also discussed in detail in this dissertation. A co-design of HeteroNoC with multiple memory controllers can be found in [124]

Fourth, a top-down approach is proposed for designing NoC. After investigating the inherent communication requirements of applications from an NoC, it is observed that, applications have widely different demands from the network, e.g. some require low latency, some require high bandwidth and some both [100]. Building upon this insight, this dissertation proposes multiple on-chip networks to satisfy the communication requirements of all applications simultaneously in an energy-efficient manner. It first shows that a high-bandwidth, low frequency network is best suited for bandwidth sensitive applications and a low-bandwidth but high frequency network is well suited for latency sensitive applications. Next, to steer packets into a particular sub-network, a packet's sensitivity to latency or bandwidth is identified at runtime. The effectiveness of the proposed two-layer network is evaluated over a range of designs and the proposed approach is found

to boost the system and application level performance while keeping the energy envelope of the system lower than a monolithic network design.

Finally, properties of a new device technology is exploited in this dissertation for improving the memory subsystem performance in multi-cores. This dissertation argues in favor of STT-RAM as a universal replacement and proposes an approach that targets one of the systemic disadvantage of STT-RAM technology: long write latency [125]. The dissertation then investigates network level solution to alleviate this problem. The proposed network level solution centers around prioritizing packets to idle banks and delaying accesses to busy STT-RAM banks. This dissertation also discusses the benefits of using distributed buffers in the network (in the virtual channel of the routers) and shows that the proposed scheme is better than the conventional write-buffering schemes in the STT-RAM banks. Overall, this dissertation makes a strong case for replacing SRAM cache banks with STT-RAM caches to lower the energy/power and improve the performance envelope and of the entire CMP.

7.2 Future Research Directions

Extensions to RAFT: While RAFT is very effective in managing network power and performance, the design framework and methodology described in this dissertation could have broader potential to address other energy and power issues such as, hot-spot and thermal control as well as QoS provisioning and fault-tolerance in NoCs. For example, the proposed frequency boosting techniques can be used to selectively expedite time-constrained traffic in the network (satisfying SLA's for these traffic classes), while the throttling technique can complement such an effort. Similarly, traffic congestion due to router and link failures can be mitigated by employing both these techniques (boosting and throttling) with different types of routing schemes. Additionally, it is worth co-evaluating DVFS schemes, typically implemented in processor cores, with DVFS schemes in the NoC as proposed in this dissertation.

Extensions to HeteroNoC: HeteroNoC is a very generic concept and, in general, the design of heterogeneous networks is very promising, especially when future CMPs are likely to have asymmetric cores. In particular, an interesting

research avenue could be to evaluate the effectiveness of heterogeneous networks when big cores, small cores, GPGPUs and accelerators are co-existing on the same die (a likely scenario in future multicore chips). It is compelling to expect that with heterogeneity in compute engines on a single die, a homogeneous network design might fall short in terms of providing the expected level of differing bandwidth/latency/QoS demands from them. Thus, a simple but effective HeteroNoC like network architecture might prove to be the right candidate for such heterogeneous systems.

Extensions to multiple-networks concept: The general notion of classifying applications into smaller groups and customizing network for such groups of applications can be very effective in managing system wide power/energy and maximize performance. Researchers are starting to look into this area [126] and this dissertation's proposal is only a first step in this direction. The idea of having multiple sub-networks can be extended to include networks that are customized for: accelerator based traffic, real-time traffic, long-haul communication, short and nearest-neighbor type traffic, power, QoS metrics, etc. Additionally, it is worth investigating multiple sub-networks for various kinds of on-chip traffic: prefetching, write-backs, coherence, data from memory controllers, etc. With 3D integration it might be a cost-effective approach to have the sub-networks in a separate layer than the compute engines, effectively leveraging additional wire bandwidth in each additional layer of the 3D chip. Further, the idea of using network episode length and height can be extremely useful in classifying applications into sub-categories before designing customized networks for them. Although this dissertation has looked only at designing on-chip networks for different application classes, the key ideas in this dissertation can be extended to designing an operating system (OS) scheduler (which prioritizes each application based on its criticality), asymmetric cores for latency and bandwidth sensitive applications, asymmetric caches [127] and even memory controllers [46, 49, 128].

Extensions to STT-RAM aware NoCs: As CMOS scales, leakage power is starting to limit the microprocessor performance. Alternative technologies will be necessary if Moore's Law is to continue beyond the 2020 time frame. As such, technologies like STT-RAM have a huge potential going forward. As a part of future work in this area, researchers could look into exploring the trade-offs in designing

STT-RAM cells that sacrifice non-volatility in favor of faster write times. The key idea is to have STT-RAM banks which apart from offering 4x improvements in storage (compared to SRAM), are also optimized for long latency writes. Research shows that this comes at the cost of losing the retention duration in the STT-RAM cells, and one research avenue could be to explore architectural solutions to handle this. A possible solution in this direction could be to identify the data blocks whose loss might affect performance. These blocks can be copied over to a small buffer and re-written back into the refreshed STT-RAM cell. Such an *out-of-place* refresh mechanism is, however, only feasible if the number of data blocks to be copied over and re-written are less in number so that a small intermediate buffer is feasible. Further, as observed in the multiple networks chapter, bandwidth sensitive applications are tolerant to uncore latency. STT-RAMs can provide significant uncore bandwidth (in terms of cache capacity) but have a long write latency overhead. As such, it seems very intuitive that STT-RAMs are a perfect fit for bandwidth sensitive applications. Latency sensitive applications on the other hand, do not require bandwidth from the uncore hierarchy and thus, SRAM based caches seem to be a perfect fit for such applications. Hence, an investigation of hybrid STT-RAM and SRAM caches could be worthwhile in this context.

To conclude, the era of many core processors has already dawned upon us. One of the most challenging task in this new paradigm is to efficiently connect the compute and memory components on the many core processor. Network-on-Chip (NoC) architectures are widely viewed as the de facto solution in this regard and are projected to play an increasingly dominant role as we move forward. Hence, it is very critical to investigate this design aspect of future many core processors. To this end, this dissertation investigates design and analysis of heterogeneous on-chip networks with the overall goal of improving the performance and power envelope of the entire many core processor.

Bibliography

[1] International Technology Roadmap for Semiconductors (ITRS), 2009 edition, http://www.itrs.net/.

[2] Peh, L.-S. and W. J. Dally (2001) "A Delay Model and Speculative Architecture for Pipelined Routers," in *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)*.

[3] Kim, J., C. Nicopoulos, D. Park, V. Narayanan, M. S. Yousif, and C. R. Das (2006) "A Gracefully Degrading and Energy-Efficient Modular Router Architecture for On-Chip Networks," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pp. 4–15.

[4] Kim, J., D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das (2005) "A Low Latency Router Supporting Adaptivity for On-Chip Router," in *42nd Design Automation Conference (DAC)*, pp. 559 – 564.

[5] Nicopoulos, C. A., D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das (2006) "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 333 –346.

[6] Mullins, R., A. West, and S. Moore (2004) "Low-Latency Virtual-Channel Routers for On-Chip Networks," in *Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA)*, pp. 188–.

[7] Hu, J. and R. Marculescu (2004) "DyAD: Smart Routing for Networks-on-Chip," in *Proceedings of the 41st Annual Design Automation Conference (DAC)*, pp. 260–263.

[8] Moscibroda, T. and O. Mutlu (2009) "A Case for Bufferless Routing in On-Chip Networks," in *36th International Symposium on Computer Architecture (ISCA)*, pp. 196–207.

[9] Kumar, A., L.-S. Peh, and N. K. Jha (2008) "Token Flow Control," in *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 342–353.

[10] Das, R., S. Eachempati, A. Mishra, V. Narayanan, and C. Das (2009) "Design and Evaluation of a Hierarchical On-Chip Interconnect for Next-Generation CMPs," in *IEEE 15th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 175–186.

[11] Grot, B., J. Hestness, S. Keckler, and O. Mutlu (2009) "Express Cube Topologies for On-Chip Interconnects," in *IEEE 15th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 163–174.

[12] Udipi, A. N., N. Muralimanohar, and R. Balasubramonian (2010) "Towards Scalable, Energy-Efficient, Bus-Based On-Chip Networks," in *IEEE 16th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1–12.

[13] Kumar, A., L.-S. Peh, P. Kundu, and N. K. Jha (2007) "Express Virtual Channels: Towards the Ideal Interconnection Fabric," in *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA)*, pp. 150–161.

[14] Kim, J., W. J. Dally, and D. Abts (2007) "Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks," in *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA)*.

[15] Kim, J., W. Dally, S. Scott, and D. Abts (2008) "Technology-Driven, Highly-Scalable Dragonfly Topology," in *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA)*.

[16] Shang, L., L.-S. Peh, and N. K. Jha (2003) "Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks," in *9th International Symposium on High Performance Computer Architecture (HPCA)*.

[17] Wang, H.-S., L.-S. Peh, and S. Malik (2003) "Power-Driven Design of Router Microarchitectures in On-Chip Networks," in *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 105 – 116.

[18] Shang, L., L.-S. Peh, A. Kumar, and N. K. Jha (2004) "Thermal Modeling, Characterization and Management of On-Chip Networks," in *Proceedings of the 37th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 67–78.

[19] Nicopoulos, C., S. Srinivasan, A. Yanamandra, D. Park, V. Narayanan, C. Das, and M. Irwin (2010) "On the Effects of Process Variation in Network-on-Chip Architectures," *Dependable and Secure Computing, IEEE Transactions on*, **7**(3), pp. 240 –254.

[20] Hernandez, C., F. Silla, and J. Duato (2010) "A Methodology for the Characterization of Process Variation in NoC Links," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pp. 685 –690.

[21] Dally, W. J. and B. Towles (2003) *Principles and Practices of Interconnection Networks*, Morgan Kaufmann.

[22] Balfour, J. and W. J. Dally (2006) "Design Tradeoffs for Tiled CMP On-Chip Networks," in *Proceedings of the 20th Annual International Conference on Supercomputing (ICS)*.

[23] Kim, J., W. J. Dally, B. Towles, and A. K. Gupta (2005) "Microarchitecture of a High-Radix Router," in *Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA)*, pp. 420–431.

[24] Duato, J. (1991) "Deadlock-Free Adaptive Routing Algorithms for Multicomputers: Evaluation of a New Algorithm," in *Proc. 3rd IEEE Symp. on Parallel and Distributed Processing*, pp. 840–847.

[25] Duato, J. and T. M. Pinkston (2001) "A General Theory for Deadlock-Free Adaptive Routing Using a Mixed Set of Resources," *IEEE Trans. Parallel Distrib. Syst.*, **12**(12), pp. 1219–1235.

[26] Duato, J. (1995) "A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Transactions on Parallel and Distributed Systems*, **6**(10), pp. 1055–1067.

[27] Dally, W. J. (1992) "Virtual-Channel Flow Control," *IEEE Trans. on Parallel and Distributed Systems*, **3**(2), pp. 194–205.

[28] Ni, L. M. and P. K. McKinley (1993) "A Survey of Wormhole Routing Techniques in Direct Networks," *IEEE Computer*, pp. 62–76.

[29] Dally, W. J. and C. L. Seitz. (1986) "The Torus Routing Chip," *Journal of Distributed Computing*, **1**(3), pp. 187–196.

[30] Mishra, A., R. Das, S. Eachempati, R. Iyer, V. Narayanan, and C. Das (2010) "A Case for Dynamic Frequency Tuning in On-Chip Networks," in *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*.

[31] Kumar, A., P. Kundu, A. Singh, L.-S. Peh, and N. K. Jha (2007) "A 4.6Tbits/s 3.6GHz Single-cycle NoC Router with a Novel Switch Allocator in 65nm CMOS," in *25th International Conference on Computer Design (ICCD)*.

[32] Jerger, N. D. E. and L.-S. Peh (2009) *On-Chip Networks*, Synthesis Lectures on Computer Architecture, Morgan & Claypool Publishers.

[33] Galles, M. (1996) "Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI SPIDER Chip," in *Symposium on High Performance Interconnects (Hot Interconnects)*, pp. 141–146.

[34] Beckmann, B. M. and D. A. Wood (2004) "Managing Wire Delay in Large Chip-Multiprocessor Caches," in *Proceedings of the 37th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 319–330.

[35] Wang, H., X. Zhu, L.-S. Peh, and S. Malik (2002) "Orion: a power-performance simulator for interconnection networks," in *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*.

[36] Borkar, S. (1999) "Design Challenges of Technology Scaling," *IEEE Micro*, **19**(4), pp. 23–29.

[37] Aslot, V. and R. Eigenmann (2001) "Performance Characteristics of the SPEC OMP2001 Benchmarks," *SIGARCH Computer Architecture News*, **29**(5), pp. 31–40.

[38] Woo, S. C., M. Ohara, E. Torrie, J. P. Singh, and A. Gupta (1995) "The SPLASH-2 Programs: Characterization and Methodological Considerations," in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*, pp. 24–36.

[39] Systems Performance Evaluation Cooperation,SPEC 2006 Benchmarks, www.spec.org/.

[40] "TPC-C Design Document," Http://www.tpc.org/tpcc/.

[41] "SPECjbb2005 Java Business Benchmark," Http://www.spec.org/jbb2005.

[42] "SPECjAppServer Java Application Server Benchmark," Http://www.spec.org/jAppServer.

[43] "SAP Sales and Distribution Benchmark," Http://www.sap.com/solutions/benchmark/index.epx.

[44] Bienia, C., S. Kumar, J. P. Singh, and K. Li (2008) "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques (PACT)*.

[45] Das, R., O. Mutlu, T. Moscibroda, and C. Das (2010) "Application-Aware Prioritization Mechanisms for On- Chip Networks," in *IEEE/ACM 42nd International Symposium on Microarchitecture (MICRO)*.

[46] Kim, Y., M. Papamichael, O. Mutlu, and M. Harchol-Balter (2010) "Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior," in *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 65–76.

[47] Snavely, A. and D. M. Tullsen (2000) "Symbiotic Jobscheduling for A Simultaneous Multithreaded Processor," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[48] Eyerman, S. and L. Eeckhout (2008) "System-Level Performance Metrics for Multiprogram Workloads," *IEEE Micro*.

[49] Kim, Y., D. Han, O. Mutlu, and M. Harchol-Balter (2010) "ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers," in *IEEE 16th International Symposium on High Performance Computer Architecture (HPCA)*.

[50] Olukotun, K., B. A. Nayfeh, L. Hammond, K. Wilson, and K. Chang (1996) "The Case for a Single-chip Multiprocessor," in *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 2–11.

[51] Dally, W. J. and B. Towles (2001) "Route Packets, Not Wires: On-Chip Inteconnection Networks," in *Proceedings of the 38th Conference on Design Automation (DAC)*, pp. 684–689.

[52] Benini, L. and G. D. Micheli (2002) "Networks on Chips: A New SoC Paradigm," *IEEE Computer*, **35**(1), pp. 70–78.

[53] Marculescu, R., U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote (2009) "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives," *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, **28**, pp. 3–21.

[54] Fallin, C., C. Craik, and O. Mutlu (2011) "CHIPPER: A low-complexity bufferless deflection router," in *IEEE 17th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 144–155.

[55] Towles, B. and W. Dally (2002) "Worst-Case Traffic for Oblivious Routing Functions," in *Proceedings of the Fourteenth Annual ACM symposium on Parallel Algorithms and Architectures (SPAA)*, pp. 1–8.

[56] Seo, D., A. Ali, W.-T. Lim, N. Rafique, and M. Thottethodi (2005) "Near-Optimal Worst-Case Throughput Routing for Two-Dimensional Mesh Networks," in *Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA)*, pp. 432–443.

[57] Vaidya, A., A. Sivasubramaniam, and C. R. Das (1999) "LAPSES: A Recipe for High Performance Adaptive Router Design," in *Proc. of IEEE Symposium on High Performance Computer Architecture (HPCA-5)*, pp. 236–243.

[58] Vaidya, A. S., A. Sivasubramaniam, and C. R. Das (1997) "Performance Benefits of Virtual Channels and Adaptive Routing: An Application-Driven Study," in *Proceedings of the 11th international conference on Supercomputing (ICS)*, New York, NY, USA, pp. 140–147.

[59] Jiang, N., J. Kim, and W. J. Dally (2009) "Indirect Adaptive Routing on Large Scale Interconnection Networks," in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*, pp. 220–231.

[60] Gratz, P., B. Grot, and S. Keckler (2008) "Regional Congestion Awareness for Load Balance in Networks-on-Chip," in *Proceedings of the 14th International Symposium on High-Performance Computer Architecture (HPCA)*.

[61] Rodrigo, S., J. Flich, J. Duato, and M. Hummel (2008) "Efficient Unicast and Multicast support for CMPs," in *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 364–375.

[62] Flich, J., S. Rodrigo, and J. Duato (2008) "An Efficient Implementation of Distributed Routing Algorithms for NoCs," in *Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, pp. 87–96.

[63] Hoskote, Y., S. Vangal, A. Singh, N. Borkar, and S. Borkar (2007) "A 5-GHz Mesh Interconnect for a Teraflops Processor," in *IEEE Micro*, vol. 27,5, pp. 51–61.

[64] Kim, E.-J., G. Link, K. H. Yum, V. Narayanan, M. Kandemir, M. J. Irwin, and C. Das (2005) "A Holistic Approach to Designing Energy-Efficient Cluster Interconnects," in *IEEE Trans. on Computers*, vol. 54, 5, pp. 660–671.

[65] Shang, L., L.-S. Peh, and N. K. Jha (2003) "PowerHerd: Dynamic Satisfaction of Peak Power Constraints in Interconnection Networks," in *Proceedings of the 17th Annual International Conference on Supercomputing (ICS)*, pp. 98–108.

[66] Bakhoda, A., J. Kim, and T. M. Aamodt (2010) "Throughput-Effective On-Chip Networks for Manycore Accelerators," in *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 421–432.

[67] GPGPU. General-Purpose Computation Using Graphics Hardware. http://www.gpgpu.org/.

[68] Nesson, T. and S. L. Johnsson (1995) "ROMM Routing on Mesh and Torus Networks," in *Proceedings of the 7th Annual ACM symposium on Parallel Algorithms and Architectures (SPAA)*, pp. 275–287.

[69] Kim, M. M., J. D. Davis, M. Oskin, and T. Austin (2008) "Polymorphic On-Chip Networks," in *Proceedings of the 35th Annual International Symposium on Computer Architecture (ISCA)*.

[70] Hu, J. and R. Marculescu (2004) "Application-Specific Buffer Space Allocation for Networks-On-Chip Router Design," in *Proceedings of the 2004 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 354–361.

[71] Murali, S., M. Coenen, A. Radulescu, K. Goossens, and G. De Micheli (2006) "Mapping and Configuration Methods for Multi-Use-Case Networks on Chips," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*.

[72] Marculescu, R., U. Y. Ogras, and N. H. Zamora (2006) "Computation and Communication Refinement for Multiprocessor SoC design: A System-Level Perspective," *ACM Trans. Des. Autom. Electron. Syst.*, **11**(3).

[73] Guz, Z., I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny (2007) "Network Delays and Link Capacities in Application-Specific Wormhole NoCs," *VLSI '07: Journal of VLSI Design*.

[74] Mejia, A., M. Palesi, J. Flich, S. Kumar, P. López, R. Hoismark, and J. Duato (2009) "Region-Based Routing: A Mechanism to Support Efficient Routing Algorithms in NoCs," *IEEE Trans. Very Large Scale Integr. Syst.*, **17**(3).

[75] Sankaralingam, K., R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. W. Keckler, and C. R. Moore (2003) "Exploiting ILP, TLP, and DLP with The Polymorphous TRIPS Architecture," in *Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA)*, pp. 422–433.

[76] Taylor, M. B., J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffmann, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal (2002) "The Raw Microprocessor: A Computational Fabric for Software Circuits and General Purpose Programs," *IEEE Micro*, **22**(2), pp. 25–35.

[77] Wentzlaff, D., P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. Brown, and A. Agarwal (2007) "On-Chip Interconnection Architecture of the Tile Processor," *Micro, IEEE*.

[78] Kahle, J. A., M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy (2005) "Introduction to the Cell Multiprocessor," *IBM J. Res. Dev.*, **49**(4/5), pp. 589–604.

[79] Lenoski, D., J. Laudon, K. Gharachorloo, W.-D. Weber, A. Gupta, J. Hennessy, M. Horowitz, and M. Lam (1992) "The Stanford Dash multiprocessor," *Computer*.

[80] Phadke, S. and S. Narayanasamy (2011) "MLP Aware Heterogeneous Memory System," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pp. 1 –6.

[81] Zhou, P., B. Zhao, J. Yang, and Y. Zhang (2009) "Energy Reduction for STT-RAM Using Early Write Termination," in *International Conference on Computer Aided Design (ICCAD)*.

[82] Joo, Y., D. Niu, X. Dong, G. Sun, N. Chang, , and Y. Xie (2010) "Energy and Endurance-Aware Design of Phase Change Memory Caches," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*.

[83] Sun, G., X. Dong, Y. Xie, J. Li, and Y. Chen (2009) "A Novel Architecture of the 3D Stacked MRAM L2 Cache for CMPs," in *IEEE 15th International Symposium on High Performance Computer Architecture (HPCA)*.

[84] Qureshi, M. K., V. Srinivasan, and J. A. Rivers (2009) "Scalable High Performance Main Memory System Using Phase-Change Memory Technology," in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*.

[85] Desikan, R., C. R. Lefurgy, S. W. Keckler, and D. Burger (2002) *On-chip MRAM as a High-Bandwidth Low-Latency Replacement for DRAM Physical Memories*, *Tech. Rep. TR-02-47*, UT-Austin.

[86] Desikan, R., S. Keckler, and D. Burger (2001) *Assessment of MRAM Technology Characteristics and Architectures*, *Tech. Rep. TR-01-36*, UT-Austin.

[87] Dong, X., X. Wu, G. Sun, Y. Xie, H. Li, and Y. Chen (2008) "Circuit and Microarchitecture Evaluation of 3D Stacking Magnetic RAM (MRAM) as a Universal Memory Replacement," in *Design Automation Conference (DAC)*.

[88] Das, R., A. K. Mishra, C. Nicopoulus, D. Park, V. Narayanan, R. Iyer, M. S. Yousif, and C. R. Das (2008) "Performance and Power Optimization Through Data Compression in Network-on-Chip Architectures," in *Proceedings of the 14th International Symposium on High-Performance Computer Architecture (HPCA)*.

[89] Xie, F., M. Martonosi, and S. Malik (2003) "Compile-time Dynamic Voltage Scaling Settings: Opportunities and Limits," in *Conference on Programming Language Design and Implementation (PLDI)*.

[90] Kim, W., M. S. Gupta, G.-Y. Wei, and D. Brooks (2008) "System Level Analysis of Fast, Per-Core DVFS Using On-Chip Switching Regulators," in *Proceedings of the 14th International Symposium on High-Performance Computer Architecture (HPCA)*.

[91] Hausman, K., G. Gaudenzi, J. Mosley, and S. Tempest (1990), "US Patent 4978927 - Programmable Voltage Controlled Ring Oscillator," .

[92] Muralimanohar, N. and R. Balasubramonian (2006.) "The Effect of Interconnect Design on the Performance of Large L2 Caches," in *3rd IBM Watson Conference on Interaction between Architecture, Circuits, and Compilers (P=ac2)*.

[93] 65 nm PTM Technology Model, http://www.eas.asu.edu/ ptm/.

[94] Mishra, A. K., A. Yanamandra, R. Das, S. Eachempati, R. R. Iyer, N. Vijaykrishnan, and C. R. Das (2011) "RAFT: A Router Architecture With Frequency Tuning for On-Chip Networks," *J. Parallel Distrib. Comput.*, **71**(5), pp. 625–640.

[95] Hill, M. D. and M. R. Marty (2008) "Amdahls Law in the Multicore Era," *IEEE COMPUTER*.

[96] Kumar, R., D. M. Tullsen, N. P. Jouppi, and P. Ranganathan (2005) "Heterogeneous Chip Multiprocessors," *Computer*, **38**(11).

[97] Kumar, R., D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas (2004) "Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance," *SIGARCH Comput. Archit. News*, **32**(2).

[98] Wang, H., L.-S. Peh, and S. Malik (2005) "A Technology-Aware and Energy-Oriented Topology Exploration for On-Chip Networks," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*.

[99] Morad, T. Y., U. C. Weiser, A. Kolodny, M. Valero, and E. Ayguade (2006) "Performance, Power Efficiency and Scalability of Asymmetric Cluster Chip Multiprocessors," *IEEE Computer Architecture Letters*.

[100] Mishra, A. K., O. Mutlu, and C. R. Das (2011) *An Application Driven Approach for Designing Heterogeneous On-Chip Networks*, *Tech. Rep. CSE-11-007*, Department of Computer Science and Engineering, The Pennsylvania State University.

[101] Das, R., O. Mutlu, T. Moscibroda, and C. R. Das (2010) "Aergia: Exploiting Packet Latency Slack in On-Chip Networks," in *International Symposium on Computer Architecture (ISCA-37)*.

[102] Fields, B., S. Rubin, and R. Bodík (2001) "Focusing Processor Policies via Critical-Path Prediction," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*.

[103] Srinivasan, S. T. and A. R. Lebeck (1998) "Load Latency Tolerance in Dynamically Scheduled Processors," in *Proceedings of the 31st annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*.

[104] Subramaniam, S., A. Bracy, H. Wang, and G. H. Loh (2009) "Criticality-Based Optimizations for Efficient Load Processing," in *IEEE 15th International Symposium on High Performance Computer Architecture (HPCA)*.

[105] Krzanowski, W. J. and Y. T. Lai (1988) "A Criterion for Determining the Number of Groups in a Data Set Using Sum-of-Squares Clustering," *Biometrics*, **44**(1).

[106] Tibshirani, R., G. Walther, and T. Hastie (2001) "Estimating the Number of Clusters in a Data Set via the Gap Statistic," *Journal of the Royal Statistical Society.*, **63**(2).

[107] V. G. Oklobdzija and R. K. Krishnamurthy (2006) *Energy-Delay Characteristics of CMOS Adders (Chapter-6), High-Performance Energy-Efficient Microprocessor Design*, Springer.

[108] Kahng, A. B., B. Li, L.-S. Peh, and K. Samadi (2009) "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*.

[109] Patil, H., R. Cohn, M. Charney, R. Kapoor, A. Sun, and A. Karunanidhi (2004) "Pinpointing Representative Portions of Large Intel Itanium Programs with Dynamic Instrumentation," in *Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*.

[110] Guo, X., E. Ipek, and T. Soyata (2010) "Resistive Computation: Avoiding the Power Wall with Low-Leakage, STT-MRAM Based Computing," in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*.

[111] Lee, B. C., E. Ipek, O. Mutlu, and D. Burger (2009) "Architecting Phase Change Memory as a Scalable DRAM Alternative," in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*.

[112] Xie, Y. (2010) "Modeling, Architecture, and Applications for Emerging Memory Technologies," *IEEE Design and Test of Computers, Special Issues on Memory Technologies*.

[113] Qureshi, M. K., J. P. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali (2009) "Enhancing Lifetime and Security of PCM-Based Main Memory with Start-Gap Wear Leveling," in *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*.

[114] Hosomi, M., H. Y. Yamagishi, T., et al. (2005) "A Novel Nonvolatile Memory with Spin Torque Transfer Magnetization Switching: Spin-RAM," in *IEEE International Electron Devices Meeting. IEDM Technical Digest.*

[115] Kawahara, T., R. Takemura, K. Miura, et al. (2007) "2Mb Spin-Transfer Torque RAM (SPRAM) with Bit-by-Bit Bidirectional Current Write and Parallelizing-Direction Current Read," in *IEEE International Solid-State Circuits Conference. ISSCC Digest of Technical Papers.*

[116] Xie, Y., G. H. Loh, B. Black, and K. Bernstein (2006) "Design Space Exploration for 3D Architectures," *J. Emerg. Technol. Comput. Syst.*, **2**(2).

[117] Black, B., M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCaule, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb (2006) "Die Stacking (3D) Microarchitecture," in *IEEE/ACM International Symposium on Microarchitecture (MICRO).*

[118] Zhao, W., E. Belhaire, Q. Mistral, et al. (2006) "Macro-Model of Spin-Transfer Torque Based Magnetic Tunnel Junction Device for Hybrid Magnetic-CMOS Design," in *BMAS.*

[119] Woo, D. H., N. H. Seong, D. L. Lewis, and H.-H. S. Lee (2010) "An Optimized 3D-Stacked Memory Architecture by Exploiting Excessive, High-Density TSV Bandwidth," in *IEEE 16th International Symposium on High Performance Computer Architecture (HPCA).*

[120] Kgil, T., S. D'Souza, A. Saidi, N. Binkert, R. Dreslinski, T. Mudge, S. Reinhardt, and K. Flautner (2006) "PicoServer: Using 3D Stacking Technology to Enable a Compact Energy Efficient Chip Multiprocessor," *SIGOPS Oper. Syst. Rev.*

[121] Madan, N., L. Zhao, N. Muralimanohar, A. Udipi, R. Balasubramonian, R. Iyer, S. Makineni, and D. Newell (2009) "Optimizing Communication and Capacity in a 3D Stacked Reconfigurable Cache Hierarchy," in *IEEE 15th International Symposium on High Performance Computer Architecture (HPCA).*

[122] Tremblay, M. and S. Chaudhry (2008) "A Third-Generation 65nm 16-Core 32-Thread Plus 32-Scout-Thread CMT SPARC Processor," in *IEEE International Solid-State Circuits Conference. ISSCC Digest of Technical Papers.*

[123] Muralimanohar, N., R. Balasubramonian, and N. Jouppi (2007) "Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0," in *IEEE/ACM International Symposium on Microarchitecture (MICRO).*

[124] Mishra, A. K., N. Vijaykrishnan, and C. R. Das (2011) "A Case for Heterogeneous On-Chip Interconnects for CMPs," in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*, pp. 389–400.

[125] Mishra, A. K., X. Dong, G. Sun, Y. Xie, N. Vijaykrishnan, and C. R. Das (2011) "Architecting On-chip Interconnects for Stacked 3D STT-RAM Caches in CMPs," in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*, pp. 69–80.

[126] Chien, A. A. (2011) "10x10: Taming Heterogeneity for General-purpose Architecture: A New Optimization Paradigm Unlocks Customization Benefits," in *2nd Workshop on New Directions in Computer Architecture (NDCA-2)*.

[127] Jiang, X., A. K. Mishra, L. Zhao, R. Iyer, Z. Fang, S. Srinivasan, S. Makineni, P. Brett, and C. R. Das (2011) "ACCESS: Smart scheduling for asymmetric cache CMPs," in *IEEE 17th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 527–538.

[128] Mutlu, O. and T. Moscibroda (2008) "Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*.

[129] Mishra, A. K., S. Srikantaiah, M. T. Kandemir, and C. R. Das (2010) "CPM in CMPs: Coordinated Power Management in Chip-Multiprocessors," in *Conference on High Performance Computing Networking, Storage and Analysis, (SC)*, pp. 1–12.

[130] Park, D., S. Eachempati, R. Das, A. K. Mishra, Y. Xie, N. Vijaykrishnan, , and C. R. Das (2008) "MIRA: A Multi-layer On Chip Interconnect Router Architecture," in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*.

[131] Srikantaiah, S., R. Das, A. K. Mishra, C. R. Das, and M. T. Kandemir (2009) "A Case for Integrated Processor-Cache Partitioning in Chip Multiprocessors," in *Conference on High Performance Computing Networking, Storage and Analysis, (SC)*.

[132] Mishra, A. K., J. L. Hellerstein, W. Cirne, and C. R. Das (2010) "Towards Characterizing Cloud Backend Workloads: Insights from Google Compute Clusters," *SIGMETRICS Performance Evaluation Review*, **37**(4), pp. 34–41.

[133] Mishra, A. K., S. Srikantaiah, M. T. Kandemir, and C. R. Das (2010) "Coordinated Power Management of Voltage Islands in CMPs," in *Proceedings of the 2011 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pp. 359–360.

[134] Sharifi, A., S. Srikantaiah, A. K. Mishra, M. T. Kandemir, and C. R. Das (2011) "METE: Meeting End-to-End QoS in Multicores through System-Wide Resource Management," in *Proceedings of the 2011 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pp. 13–24.

[135] Saripally, V., A. K. Mishra, S. Datta, N. Vijaykrishnan, and C. R. Das (2011) "An Energy-Efficient Heterogeneous CMP based on Hybrid TFET-CMOS Cores," in *48th Design Automation Conference (DAC)*.

Vita

Asit K. Mishra

Asit K. Mishra is a Ph.D candidate in the Department of Computer Science and Engineering at the Pennsylvania State University. Asit joined Penn State in 2006 after completing his Bachelor in Technology (with Honors) in Electrical Engineering from National Institute of Technology, Rourkela, India. His primary research interests are computer architecture (especially on-chip interconnection networks), 3D architectures, new memory technologies, and Internet-scale data centers. Asit was awarded the *Graduate Student Research Assistant Award* by the CSE Department for outstanding research in spring 2011. His research has been published in major computer architecture conferences (ISCA, MICRO, HPCA, SIGMETRICS). He has served as a technical reviewer for many journals and conferences including IEEE TPDS, IEEE TVLSI, IEEE TNSM, ISCA, HPCA and MICRO. He also served as the *Submissions Chair* for HPCA 2010. Asit has worked at Intel Corporation during summers 2008 and 2010, and at Google Incorporation in summer 2009. As a graduate student, Asit was also a photographer at The Daily Collegian, a student run newspaper daily.