

BOSTON UNIVERSITY
COLLEGE OF ENGINEERING

Dissertation

**ENERGY-EFFICIENT ELECTRICAL AND
SILICON-PHOTONIC NETWORKS IN MANYCORE
SYSTEMS**

by

CHAO CHEN

B.E., Shanghai Jiao Tong University, 2005
M.S., Pohang University of Science and Technology, 2007

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2014

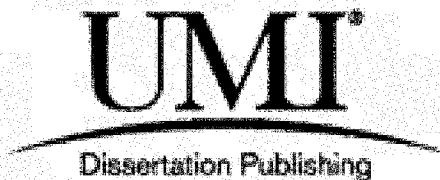
UMI Number: 3581011

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 3581011

Published by ProQuest LLC 2014. Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

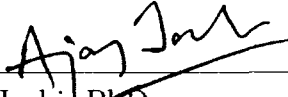
All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Approved by

First Reader



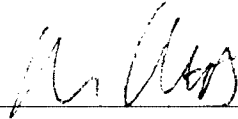
Ajay Joshi, PhD
Assistant Professor of Electrical and Computer Engineering

Second Reader



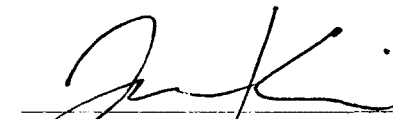
Ayse K. Coskun, PhD
Assistant Professor of Electrical and Computer Engineering

Third Reader



Martin Herbordt, PhD
Associate Professor of Electrical and Computer Engineering

Fourth Reader



Jonathan Klankin, PhD
Assistant Professor of Electrical and Computer Engineering

The interconnection network is the heart of parallel architecture

Chuan-Lin and Tse-Yun Feng
Interconnection Networks for Parallel
and Distributed Processing (1984)

Acknowledgments

I would like to express my deepest appreciation to the committee members for reviewing this dissertation and providing me precious feedback. Then I want to thank my adviser, Prof. Ajay Joshi, who led my research and worked with me for many interesting research topics. Also it is my pleasure to work with Dr. Abellan Jose, the post-doctoral fellow in our group, who provided many experimental results and valuable suggestions. I want to thank members in our group, Zafar takhirov, Mahmoud Zangeneh, and Schuyler Eldridge for making our research in the lab enjoyable.

Also I wish to thank Prof. Ayse Coskun and Prof. Jonathan Klamkin for their suggestions in my research. They helped me overcome many technical obstacles when we were writing papers together. I also want to thank several members in their groups, including Tiansheng Zhang, Dr. Jie Meng, and Dr. Pietro Contu, for all their contributions when we were working on research projects together.

In addition, a thank you to thank Prof. Martin Herbordt and Prof. Roscoe Giles. I worked as a teaching fellow for their classes in the first year at Boston University. I enjoyed when helping undergraduate students with their homework and lab projects, and I also obtained a lot of useful knowledge for my later research.

Finally, I want to thank my father, Shunjin Chen, my mother, Wenin Liu, and my wife, Xiaodan Zhang for their continuous support and encouragement. They helped me focus on the research and work hard for achieving my education and career goals. Especially, my wife gave up a very good job opportunity in China and chose to stay with me in the United States. I hope to provide my parents and wife better lives after graduation and spend more time with them.

I dedicate this dissertation to my parents and my beloved wife.

ENERGY-EFFICIENT ELECTRICAL AND SILICON-PHOTONIC NETWORKS IN MANYCORE SYSTEMS

CHAO CHEN

Boston University, College of Engineering, 2014

Major Professor: Ajay Joshi, PhD,
Professor of Electrical and Computer Engineering

ABSTRACT

During the past decade, the very large scale integration (VLSI) community has migrated towards incorporating multiple cores on a single chip to sustain the historic performance improvement in computing systems. As the core count continuously increases, the performance of network-on-chip (NoC), which is responsible for the communication between cores, caches and memory controllers, is increasingly becoming critical for sustaining the performance improvement. In this dissertation, we propose several methods to improve the energy efficiency of both electrical and silicon-photonic NoCs. Firstly, for electrical NoC, we propose a flow control technique, Express Virtual Channel with Taps (EVC-T), to transmit both broadcast and data packets efficiently in a mesh network. A low-latency notification tree network is included to maintain the order of broadcast packets. The EVC-T technique improves the NoC latency by 24% and the system energy efficiency in terms of energy-delay product (EDP) by 13%. In the near future, the silicon-photonic links are projected to replace the electrical links for global on-chip communication due to their lower data-dependent power and higher bandwidth density, but the high laser power can more

than offset these advantages. Therefore, we propose a silicon-photonic multi-bus NoC architecture and a methodology that can reduce the laser power by 49% on average through bandwidth reconfiguration at runtime based on the variations in bandwidth requirements of applications. We also propose a technique to reduce the laser power by dynamically activating/deactivating the L2 cache banks and switching ON/OFF the corresponding silicon-photonic links in a crossbar NoC. This cache-reconfiguration based technique can save laser power by 23.8% and improves system EDP by 5.52% on average. In addition, we propose a methodology for placing and sharing on-chip laser sources by jointly considering the bandwidth requirements, thermal constraints and physical layout constraints. Our proposed methodology for placing and sharing of on-chip laser sources reduces laser power. In addition to reducing the laser power to improve the energy efficiency of silicon-photonic NoCs, we propose to leverage the large bandwidth provided by silicon-photonic NoC to share computing resources. The global sharing of floating-point units can save system area by 13.75% and system power by 10%.

Contents

1	Background and Introduction	1
1.1	Trends in VLSI Computing System	1
1.2	Electrical NoC	2
1.3	Silicon-photonic NoC	4
1.3.1	Laser Power Management	4
1.3.2	On-chip Laser Source	8
1.3.3	The Use of Large Bandwidth	9
1.4	Contribution and Organization	10
2	Broadcast through Express Virtual Channel with Taps in Electrical NoC	14
2.1	Introduction	14
2.2	Target System	16
2.3	Notification Trees for Broadcasting Order	18
2.4	EVC-T Flow Control Mechanism	22
2.4.1	Express Virtual Channels (EVC)	22
2.4.2	Express Virtual Channels with Taps (EVC-T)	24
2.4.3	Flit Organization for EVC-T	28
2.4.4	Credit Channels for EVC-T	29
2.5	Evaluation	29
2.5.1	Simulation Methodology	31
2.5.2	Evaluation Results with Synthetic Benchmarks	33

2.5.3	Evaluation Results with NAS Benchmarks	36
2.6	Summary	38
3	Bandwidth Reconfiguration for Managing Laser Power in Silicon- photonic NoC	40
3.1	Introduction	40
3.2	Target System	43
3.3	Photonic Technology	44
3.4	Multi-bus NoC Architecture	47
3.5	Laser Power Management	52
3.5.1	Runtime Network Reconfiguration	53
3.5.2	Bandwidth Weight Calculation	55
3.6	Evaluation	58
3.6.1	Simulation Methodology	58
3.6.2	Evaluation Results with NAS Benchmarks	59
3.7	Discussion	66
3.7.1	Large Core Counts	66
3.7.2	Simultaneously Executing Applications	67
3.7.3	Distributed L2 Cache vs. Private L2 Cache	67
3.7.4	Alternate NoC Architectures	68
3.8	Summary	69
4	Runtime Cache Reconfiguration for Managing Laser Power in Silicon- photonic NoC	70
4.1	Introduction	70
4.2	Target System, Simulation tools	72
4.3	Runtime Reconfiguration	74
4.3.1	Reconfiguration Controller	75

4.3.2	Reconfiguration Decision Process	76
4.3.3	L2 Bank Activation/Deactivation Process	79
4.4	Evaluation	81
4.4.1	Simulation Methodology	81
4.4.2	Reconfiguration Opportunities	82
4.4.3	Reconfiguration Benefits	84
4.5	Summary	88
5	Sharing and Placement of On-chip Laser Sources for Managing Laser Power in Silicon-photonic NoC	89
5.1	Introduction	89
5.2	Target System	91
5.3	Laser Sources	93
5.3.1	Modeling	94
5.3.2	Optical Power, WPE and Temperature Tradeoffs	96
5.3.3	Broadband vs Single-band Laser Sources	101
5.3.4	Methodology to Determine Optimal Sharing and Placement	102
5.4	Case Studies	104
5.5	Summary	109
6	Sharing of Computing Resources through Large Bandwidth Provided by Silicon-photonic NoC	111
6.1	Introduction	111
6.2	Target System	115
6.2.1	Logic Layer	116
6.2.2	Photonic Layer	118
6.3	The EUCloud-based Manycore System	119
6.3.1	Processor Core Design	119

6.3.2	Core-to-EUCloud NoC Architecture	122
6.3.3	Execution in EUCloud	126
6.3.4	Execution in EUCloud using Instruction Bundling	128
6.3.5	Workload Allocation	132
6.4	Evaluation	133
6.4.1	Simulation Methodology	133
6.4.2	EUCloud and NoC Design	134
6.4.3	Single-Application Performance	137
6.4.4	Multi-programmed Workload Performance	139
6.4.5	EUCloud-based Manycore Benefits	141
6.4.6	Clustered vs. Global EUCloud	143
6.5	Summary	144
7	Conclusion and Future Work	146
7.1	Conclusion	146
7.2	Future Work	149
7.2.1	Laser Power Management through Core Reconfigurations . . .	149
7.2.2	Runtime Selection and Sharing of On-chip Laser Sources . . .	149
7.2.3	Performance Improvement through Silicon-photonic NoCs . .	150
	References	151
	Curriculum Vitae	161

List of Tables

2.1	Micro-architectural parameters of the 64-core system	17
2.2	Timing analysis of the notification tree architecture	20
2.3	Network architecture details	30
3.1	Micro-architectural parameters of the 64-core system	43
3.2	Energy and Power Projections for Photonic Devices	46
3.3	Optical Loss per Component	46
3.4	L_{low} for all bandwidth weight at various L_{high}	56
4.1	Micro-architectural parameters of the 64-core system	72
5.1	Architectural-level parameters for 5 NoCs under consideration	105
6.1	Architectural parameters of the 256-core system	117
6.2	Multi-programmed workloads composed of four 64-thread applications that are classified depending on demand of FPUs	139

List of Figures

1.1	Trends in transistor count, performance, core count and power over the past decades	2
1.2	Power breakdown of manycore network	6
2.1	Physical layout of our 64-core system with electrical NoC.	18
2.2	Example of Notification trees	19
2.3	Example of broadcast packets	20
2.4	The router architecture for EVC	23
2.5	The C-EVC network: physical cmesh layout with EVC	24
2.6	The router architecture supporting EVC-T	25
2.7	The C-EVC-T network: the physical cmesh layout with EVC-T	26
2.8	Flit organization of an EVC-T packet	27
2.9	Network latency and power vs. offered bandwidth	34
2.10	Comparison of C-VC and C-EVC-T networks	37
3.1	Photonic Link Components	45
3.2	Physical layout of the silicon-photonic multi-bus NoC architecture . .	48
3.3	Timing diagram for token stream arbitration	50
3.4	Physical layout of the silicon-photonic Clos and butterfly NoC architecture	51
3.5	Flow chart for runtime laser power management using weighted TDM	55
3.6	Methodology to determine threshold L_{low} for a bandwidth weight . .	56
3.7	Power and IPC for various NoCs with the same laser power budget .	60

3.8	IPC for various baseline bandwidth	61
3.9	IPC and bandwidth weight for various reconfiguration threshold L_{high}	62
3.10	IPC and bandwidth weight for various reconfiguration time interval .	63
3.11	IPC, network traffic and bandwidth weight tracing for selected benchmarks	65
4.1	Logical topology of a silicon-photonics crossbar NoC	73
4.2	Physical layout of the silicon-photonics crossbar NoC	74
4.3	Flowchart for runtime reconfiguration	76
4.4	The choice of thresholds T_{high} and T_{low}	78
4.5	L2 cache bank reconfiguration process	80
4.6	IPC, replacement rate and l2 bank count tracing for selected benchmarks	83
4.7	Impact of runtime reconfiguration on system performance and power	84
4.8	EDP improvement across vs. waveguide loss	85
4.9	IPC degradation vs. laser switch ON time	86
4.10	Reconfiguration overhead for various benchmarks	87
5.1	Overview of the 3D flip-chip manycore system and layouts for each layer	91
5.2	Cross-sectional view of our target 3D manycore system	92
5.3	P-I characteristics of a laser source at various temperatures	96
5.4	Wall-plug efficiency vs Input current at various temperatures	97
5.5	Wall-plug efficiency vs Laser Source Lengths at various temperature .	97
5.6	Laser source temperature vs. electrical input power	98
5.7	WPE vs optical output power for various sharing granularity	99
5.8	The laser source sharing methods	101
5.9	Effective WPE vs. sharing granularity of laser sources	103
5.10	Flowchart for deciding the sharing and placement of on-chip laser sources	104
5.11	Total laser power vs. waveguide loss for various sharing and placement	106

6-1	Average per core utilization of FPUs when running FMM application on a 64-core system	112
6-2	3D stacked manycore system	116
6-3	Breakdown of area and static power of each core	118
6-4	Abstract view of the main components of a processor core	119
6-5	Packets required to access EUCloud	121
6-6	The silicon-photonics between cores and EUCloud	123
6-7	NoC power for EUCloud	125
6-8	The execution stage using the pipelined 3-cycle FPa1u unit at EUCloud	127
6-9	Extensions to support bundles in EUCloudOpt	130
6-10	The workload allocation among EUs	131
6-11	Required number of FPUs for for each 64-thread application	135
6-12	Offered bandwidth for each 64-thread application	136
6-13	Performance comparison between EUCloud and EUCloudOpt	138
6-14	Performance for EUCloudOpt-based manycore system	140
6-15	Performance for EUCloudOpt-based improved manycore system	142
6-16	Performance comparison Cluster vs. Global EUCloud under aggregated IPC metric	143

List of Abbreviations

AGU	Address Generation Unit
ALU	Arithmetic Logic Unit
AMD	Advanced Micro Devices
AP	Access Point
AVX	Advanced Vector Extensions
BRU	Branch Unit
BTB	Branch Target Buffer
CAP	Core Access Point
CMOS	Complementary MetalOxideSemiconductor
CPU	Central Processing Unit
DCache	Data Cache
DC	Data Center
DDE	Data-traffic Dependent Energy
DWDM	Dense Wavelength-Division Multiplexing
EAP	EUCloud Access Point
EDP	Energy Delay Product
EEVDF	Earliest Eligible Virtual Deadline First
EUCloud	Execution Unit Cloud
EUCloudOpt	Optimized Execution Unit Cloud
EU	Execution Unit
EVC	Express Virtual Channels
EVC-T	Express Virtual Channels with Taps
FP	Floating Point
FPALU	Floating Point Arithmetic Logic Unit
FPGA	Field-Programmable Gate Array
FPmov	Floating Point Move
FPmuldiv	Floating Point Multiplication and Division
FPMult	Floating Point Multiplication
FPU	Floating Point Unit
GaAs	Gallium Arsenide
GOPS	Giga Operations per Second
HB	Home Bank
HPC	High Performance Computer

ICache	Instruction Cache
ID	Identification
IEEE	Institute of Electrical and Electronics Engineers
ILP	Instruction-Level Parallelism
InGaAs	Indium Gallium Arsenide
IntALU	Integer Arithmetic Logic Unit
IntMult	Integer Multiplication
IPC	Instruction per Cycle
ITRS	International Technology Roadmap for Semiconductors
LSTP	Low Static Power
MECS	Multidrop Express Channels
MC	Memory Controller
MIPS	Million Instructions per Second
MPSoC	Multiprocessor System-on-Chip
MWSR	Multi-Writer Single-Reader
NACK	Negative Acknowledgement
NAS	National Aeronautics and Space Administration
NoC	Networks-on-Chip
OoO	Out-of-Order
PIDRAM	Photonicallly Interconnected DRAM
PTM	Predictive Technology Model
RC	Reconfiguration Controller
RS	Reservation Station
SCC	Single Chip Cloud Computer
SIMD	Single Instruction Multiple Data
SiN	Silicon Nitride
SiO ₂	Silicon Dioxide
SOI	Silicon on Insulator
SRAM	Static Random-Access Memory
SSE	Streaming SIMD Extensions
SWMR	Single-Write Multiple-Read
TDM	Time-Division Multiplexing
TLP	Thread-Level Parallelism
TSV	Through Silicon Via
TT	Thermal Tuning Circuits
VC	Virtual Channel
VLSI	Very-Large-Scale-Integration
WDM	Wavelength-Division Multiplexing
WPE	Wall-Plug Efficiency

Chapter 1

Background and Introduction

1.1 Trends in VLSI Computing System

The general-purpose compute capacity of the world grew at an annual rate of 58% from 1986 to 2007 (Hilbert and Lopez, 2011). The very large scale integration (VLSI) community was able to use technology scaling to sustain this increase in compute capacity. Figure 1-1 shows the trends in transistors, performance, and power for general-purpose processors over the past three decades. As predicted by Moore's law (Moore, 1965), the number of transistors per unit area doubled approximately every two years. As the transistor number increased, processor performance was improved by designing more complicated core architecture and using higher core frequency. The MIPS R2000 had 110K transistors, ran at 16.7 MHz, had no on-chip caches, and used a very simple five-stage pipeline. It had a total compute capacity of 12 MIPS in 1985. Intel Pentium 4 processor released in 2000, had 42M transistors, ran at 2.0+ GHz, included large on-chip caches, and used a 20-stage pipeline with superscalar issue and a 126-entry reorder buffer for deep out-of-order execution. The compute capacity of the Pentium 4 Extreme Edition was 9,726 MIPS at 3.2 GHz in 2005.

Around 2005, the computing community hit the proverbial "power wall". Hence, to sustain the historic performance improvement implied by Moores law, processors were

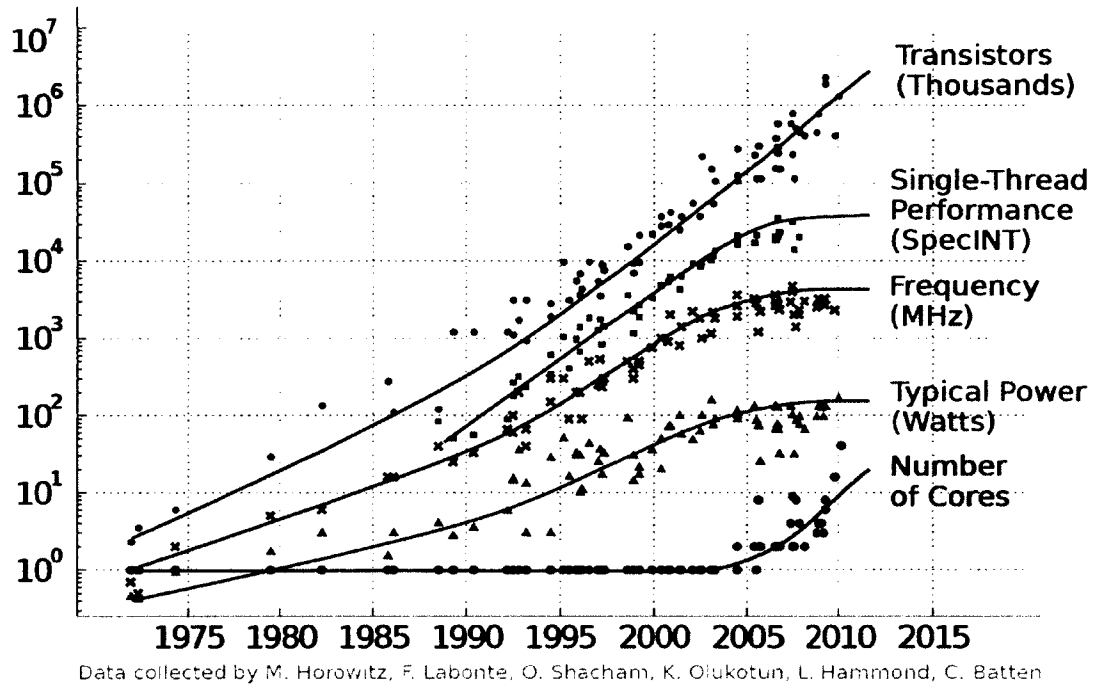


Figure 1-1: Trends in transistor count, performance, core count and power over the past decades

designed to have multiple cores on a single die. Today's systems have hundreds of cores on a single die. The TILE64 processor released in 2008 has 64 tile processors that are connected through 2D mesh network. The compute capacity of TILE64 processor was up to 384 aggregate GOPS at 750 MHz. The Intel Single-chip Cloud Computer (SCC) (Howard et al., 2010), released in 2009, enabled Tera-scale computing research. It integrated 48 Pentium class IA-32 cores that were connected through a 2D mesh network. The core count on the processor is expected to increase into the thousands in the near future.

1.2 Electrical NoC

These processors with large number of cores on a chip require the design of scalable energy-efficient network-on-chip (NoC) for on-chip communication. Most contempo-

rary multicore processors use electrical link technology for communication among the cores. The performance of these electrical NoC affects the latency of each individual cache/memory access and in turn affect the performance of the entire manycore system. The bus topology was widely used in multicore processors with less than ten cores. However, to provide the bandwidth required by a manycore system with tens to hundreds of cores, researchers have proposed various NoC topologies, ranging from low-radix high-diameter mesh network to high-radix low-diameter crossbar network (Dally and Towles, 2003). Several commercial systems such as Tileria (Bell et al., 2008) and Intel Single-Chip Cloud (SCC) (Howard et al., 2010) use low-radix high-diameter mesh network. This mesh network is easy to design in hardware using repeater-inserted electrical links. Several high-radix and low-diameter network topologies, such as flattened butterfly, clos, and MECS (Kim et al., 2007; Joshi et al., 2009; Grot et al., 2009) have been proposed for lower network latency distribution. These network topologies provide low network latency distributions and high network throughput by connecting distant routers with physical express channels. However, the energy overhead of physical express channels makes it difficult to justify their use for current and future power-limited systems. To improve the performance of low-radix and high-diameter network topologies, express virtual channels (EVC) and the corresponding flow control technique have been proposed in (Kumar et al., 2008). This technique enables the intermediate routers to forward the received packets immediately without buffering, arbitration, and crossbar switching. However, EVC is not efficient for transmitting broadcast packets due to the multiple transmissions of one broadcast packet on the same physical channel. To support effective transmission of broadcast packets, we extend the traditional EVC technique to express virtual channel with taps (EVC-T) that has multiple taps along the EVC. This approach transmits both broadcast packets and data packets with reduced traffic overhead and

network latencies.

In Addition, these packet switch-based NoC cannot maintain the packet order due to the variations in transmission distance and contentions in packet routing. However, maintaining the order of broadcast packets is critical for cache coherency. A number of techniques have been proposed for resolving the packet ordering issue in the NoC. Ordered broadcast trees and ring topologies address the cache coherence problem on packet switch-based NoCs by creating ordering points (Charlesworth, 2002; Marty and Hill, 2006). Although the ordering points method in these techniques is convenient and straightforward, the technique increases packet latency. A similar approach has been proposed in (Strauss et al., 2007), where a ring cache coherence protocol is used for ordering. In this case, in addition to the snoop request broadcast, the requester also initiates a response message that collects responses from all nodes as it travels around the ring. A global ordering of networks has been proposed using isotach-like networks in (Reynolds et al., 1997; Bilir et al., 1999; Williams et al., 2000). To maintain the orders of broadcast packets, some approaches use snoop ordering (Agarwal et al., 2009). This method avoids using the ordering points. However, the received broadcast packets have to wait for other packets with lower snoop orders, which reduces system performance. We propose a broadcasting technique with notification trees as the supporting networks for cache coherence. Our broadcasting technique allows caches to process received broadcast packets with much shorter waiting time and limited hardware overhead.

1.3 Silicon-photonic NoC

1.3.1 Laser Power Management

In future manycore systems that have hundreds to thousands of cores integrated on a single chip, the electrical links may not be able to provide the required bandwidth within reasonable power budgets. Silicon-photonic link technology has been extensively explored as a potential replacement to the electrical link technology in the design of NoC for manycore systems. Researchers have explored silicon-photonic implementations of the entire spectrum of network topologies. The large number of global buses needed for the high-radix low-diameter crossbar that provide non-blocking connectivity can be efficiently implemented using silicon-photonics technology (Kirman et al., 2006; Psota et al., 2010; Vantrease et al., 2008). The silicon-photonic implementation of low-radix high-diameter networks like mesh and torus lying at the other end of the network spectrum have also been investigated (A. Shacham, K. Bergman and L. P. Carloni, 2007; Kirman and Martínez, 2010; Cianchetti et al., 2009; Petracca et al., 2008). Silicon-photonic designs of intermediate network topologies like Clos and fat-tree that offer the same network guarantees like the global crossbar with potentially lower resource requirements have also been explored (Gu et al., 2009; Joshi et al., 2009; Pan et al., 2009).

A general consensus among the various efforts so far is that silicon-photonic networks provide a bandwidth density and data-dependent energy advantage for NoC communication. However, the fixed amount of power consumed in the laser sources that drive these networks negates these advantages. Figure 1-2 shows the power breakdown in the electrical and photonic local meshes global switches (lmg) inter-chip network (Batten et al., 2008), and electrical mesh, electrical concentrated mesh (cmesh), electrical Clos and photonic Clos intra-chip networks for the listed target

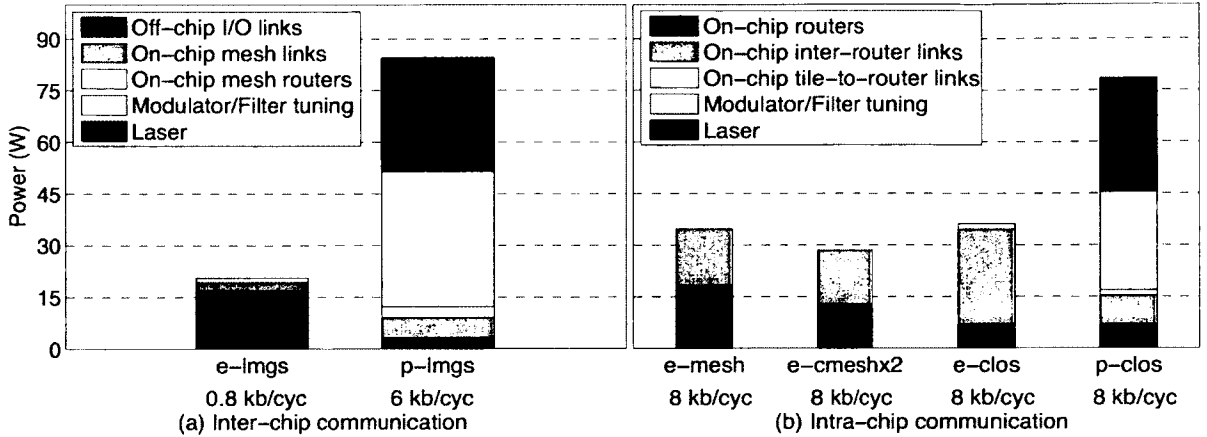


Figure 1-2: Power breakdown of manycore network – Power breakdown in the electrical and photonic local meshes global switches (lmgs) inter-chip network (Batten et al., 2008), and electrical mesh, electrical concentrated mesh (cmesh), electrical Clos and photonic Clos intra-chip networks for the listed target bandwidths. Laser power and modulator/filter tuning power more than offsets any power savings in the data-dependent power of the silicon-photonic link. Uniform random traffic pattern was used for evaluating both networks. For modulator/filter tuning, we assumed a tuning cost of $10 \mu\text{W}$ per K per ring and a tuning range of 20 K. For the laser source power, we assumed the laser source has an efficiency of 30% (Batten et al., 2008; Joshi et al., 2009).

bandwidths. As shown in Figure 1-2, a non-trivial amount of power is required in the laser source that is used in driving the silicon-photonic networks and in tuning of modulators and filters against thermal variations. In fact, the laser power and thermal tuning power can more than offset the bandwidth density and data-dependent energy advantages of the silicon-photonic links. To use silicon-photonic link technology in future manycore system, we need to reduce these two sources of power consumption. To enable the use of silicon-photonic NoC in future manycore systems, we need to develop techniques to proactively manage laser power.

At the device level, standard design-time solutions to reduce optical loss in silicon-photonic devices and in turn reduce the laser power range from exploring different

materials to process flows to device geometries. At the circuit level, we can explore the design of receivers that can operate with low-sensitivity photodetectors or use photonic devices that have lower losses but are more susceptible to noise (Bauters et al., 2011; Bauters et al., 2013), and use error detection/correction techniques to tackle any errors. At the architecture level, a nanophotonic crossbar architecture that uses optical channel sharing to manage static power dissipation is proposed in (Pan et al., 2010). Here a token-stream mechanism is used for channel arbitration and credit distribution, to enable efficient global sharing of crossbar channels. Similarly, a reconfigurable photonic network for board-to-board communication is proposed in (Kodi and Louri, 2011) for improving performance and reducing power. Here, depending on the network traffic, idle channels are reallocated to busy channels to improve performance, and bit rate and supply voltages of individual channels are regulated to manage power. In (Li et al., 2012b), the authors propose to divide the photonic NoC into subnets and also share photonic channels for sending arbitration and data packets to reduce laser power. In (Zhou and Kodi, 2013), the authors propose a prediction mechanism to dynamically scale the NoC bandwidth depending on the demands of the overlying application and in turn reduce optical power consumption.

For a multi-bus silicon-photonic NoC, we propose to use time-division multiplex to control the photonic bandwidth output from the laser source across all the channels based on weights that change at runtime to maximize the manycore system performance. We switch ON/OFF the laser source (i.e. reduce the net bandwidth of the network) to further reduce laser power. The ultimate goal is to maximize the overall execution efficiency of the manycore system. The decisions on the magnitude of change of the multiplexing weights and network bandwidth (through switching ON/OFF of laser sources) are made based on the average network packet latency for an application over fixed sampling intervals. Our technique ensures that the

application runs at peak performance while consuming minimum amount of laser power. We have also proposed a multi-bus NoC architecture that is well suited to the proposed weighted time-division multiplexing technique and have presented a head-to-head comparison of this multi-bus NoC architecture with conventional Clos and butterfly NoC architecture. A time-division multiplexed arbitration technique for silicon-photonic mesh NoC is proposed in (Hendry et al., 2011). In contrast to our runtime approach, here the time division-multiplexed photonic paths between the various pairs of network access points are established statically during design time and do not change at runtime. The key idea is to provide complete network connectivity, with each pair of access points getting fair access to large network bandwidth.

We also propose to manage the laser power consumed in the silicon-photonic NoC between the private L1 cache and distributed shared L2 cache using cache reconfiguration. In our approach, we determine the minimum number of L2 cache banks that are required to sustain the application performance. The extra L2 cache banks are deactivated and the silicon-photonic links associated with those L2 cache banks are also switched OFF to save laser power. The idea of reconfiguring the cache architecture to reduce cache energy and/or improve performance has been explored in the past (Sim et al., 2012; Qureshi et al., 2007; Wu et al., 2011). However, it has never been used to manage the power consumed in a silicon-photonic NoC. The main goal here is to minimize the EDP of the overall manycore system by leveraging the spatial and temporal variations in the behavior of the applications.

1.3.2 On-chip Laser Source

A widespread adoption of silicon-photonic link technology in designing the NoC of manycore systems is not yet possible as packaging a silicon-photonic manycore system where several off-chip laser sources drive the photonic NoC through optical fibers is

extremely challenging. Hence, on-chip laser sources are being considered as a potential alternative to these off-chip laser sources for driving the silicon-photonic NoC (Kurian et al., 2012; Heck and Bowers, 2014). These on-chip laser sources can be on the layer adjacent to the layer containing the photonic devices, making it easier to couple the laser source output to the photonic NoC. In addition to packaging challenges, the fixed power consumed in the laser sources and the power consumed in thermal management of the photonic links can be non-trivial and can negate the bandwidth advantages of silicon-photonic links. These on-chip laser sources can be switched ON/OFF relatively quickly compared to off-chip laser sources making them compatible with the runtime laser power management techniques (Chen and Joshi, 2013; Li et al., 2012a; Zhou and Kodi, 2013) that have been proposed. Similarly, given that laser sources are on the layer next to the photonic device layer it would be easier to match the temperatures of the laser source and the photonic devices, which would in turn make the thermal management techniques (Zhang et al., 2014; Nitta et al., 2011; Zheng et al., 2012; Li et al., 2012b) more effective.

Several automated tool flows have been developed to enable rapid exploration of the silicon-photonic link design space as well as the silicon-photonic NoC design space. In (Ding et al., 2009), the authors propose a linear programming technique to design the physical layout of the photonic devices on a separate photonic layer with the goal of minimizing the power consumed by an off-chip laser source. Similarly, in (Condrat et al., 2013), the authors propose a methodology to route the photonic waveguide such that the number of waveguide crossings is minimized. For a comprehensive evaluation of the photonic network design space, the authors in (Chan et al., 2011) propose a methodology and a tool that jointly explores the link-level and system-level designs of the network topologies. In (Hendry et al., 2011), the authors propose a toolflow for placement and routing of photonic devices to hierarchically design large

complex photonic networks. In (Boos et al., 2013), the authors have proposed a tool for placement and routing of optical NoC topologies with the goal of enabling a realistic analysis of the optical NoC design space. We explore the design space for sharing and placement of on-chip laser sources. Our analysis jointly considers NoC bandwidth constraints, thermal constraints and physical layout constraints to determine the optimal sharing as well as the placement of the on-chip laser sources with the goal of minimizing the laser power consumption. The approach that we have adopted can be easily integrated with the design automation tools described above to generate an optimal end-to-end design of the NoC.

1.3.3 The Use of Large Bandwidth

The idea of sharing computational resources such as floating point units has already been demonstrated as an efficient strategy to save power and chip area. Processor architectures that share floating point units have already been introduced by several commercial vendors including IBM (Meltzer, 1999), Texas Instrument (Kahle and Moore, 2000), Sun/Oracle (Leon et al., 2006) and AMD (McIntyre et al., 2012). These processor architecture designs are however only suitable for small core counts and do not readily scale to large core counts. Recently, (Kakoe et al., 2013) proposed a shared FPU for low-power embedded MPSoCs. They rely on a dedicated interconnection network to convey all traffic to access a shared set of FPUs, along with a FPU allocator that assigns processor's requests to available FPUs. This design can only support a small processor count as the overhead of a dedicated network scales non-linearly with core count. FPU Sharing has also been proposed in the context of FPGAs (Castells-Rufas et al., 2011) leading to significant area savings with minimal performance degradation for a modest scale of sharing.

Contemporary solutions cannot adapt well to hundreds of processor cores in future

manycore systems targeting multi-programmed workloads and/or server consolidation scenarios. The localized sharing of FPUs among a reduced fraction of processor cores can lead to significant performance losses when applications belonging to a multi-programmed workload need more FPUs than those available in a manycore partition, or over-provisioned partitions for applications that need less number of FPUs than the number of FPUs that are available in the partition. To maximize the savings in area and power, we propose a novel EUCloud-based manycore that implements efficient global sharing of computing resources without degrading application/workload performance, by leveraging low-latency and high-bandwidth density of silicon-photonics technology.

1.4 Contribution and Organization

In Chapter 2, we present the use of a contention-free notification tree per core as a supporting network for broadcasting to maintain the sequential consistency for snoop-based cache coherence. These notification trees ensure each core knows when to expect a broadcast packet and the exact timestamp when a broadcast packet was generated. This information guarantees that on average a core has to wait for less than a cycle before it can make the decision about processing a broadcast packet and all the broadcast packets are processed in the correct order. We also present the use of EVC-T flow control mechanism to transmit both broadcast and data packets in a snoop-based cache coherency protocol to achieve low network latency and save energy. For broadcasting, our EVC-T technique allows intermediate routers to receive and store broadcast packets while forwarding them to downstream routers simultaneously. For NAS parallel benchmarks (Bailey et al., 1994), EVC-T reduces the average packet latency (data and broadcast) by 24% and improves the system energy efficiency, reducing the energy delay product (EDP) by 13% on average.

In Chapter 3, we present a silicon-photonic multi-bus NoC architecture between private L1 caches and distributed L2 cache banks that uses weighted time-division multiplexing (W-TDM) to distribute the laser power across multiple buses based on the runtime variations in the bandwidth requirements within and across applications to maximize energy efficiency. The multi-bus NoC architecture also harnesses the opportunities to switch OFF laser sources at runtime, during low bandwidth requirements, to reduce laser power consumption. Using detailed system-level simulations, we evaluate the multi-bus NoC architecture and runtime laser power management technique on a 64-core system running NAS parallel benchmark suite. The silicon-photonic multi-bus NoC architecture provides more than $2\times$ better performance than silicon-photonic Clos and butterfly NoC architectures, while consuming the same laser power. Using runtime laser power management technique, the average laser power is reduced by more than 49% with minimal impact on the system performance.

In Chapter 4, we present a runtime management technique to reduce the laser power dissipation at by dynamically activating/deactivating L2 cache banks and switching ON/OFF the corresponding silicon-photonic links in the NoC. Since the total laser power dissipation depends on the number of on-chip silicon-photonic links, this method effectively throttles the total on-chip NoC bandwidth at runtime according to the memory access features of the applications running on the manycore system. Full-system simulation utilizing PARSEC and SPLASH parallel benchmarks (Bienia et al., 2008; Woo et al., 1995) reveal that our proposed technique achieves 23.8% savings in laser power and 5.52% lower energy-delay product (EDP) for the whole system at the cost of 0.65% loss in IPC on average.

In Chapter 5, we propose the sharing and placement methodology of on-chip laser sources that simplify packaging and lower photonic losses in the links, compared to off-chip laser sources. The electrical input power of these laser sources is dependent

on the laser source temperature and the optical output power. The laser source temperature depends on the power consumed by the laser as well as the cores in the laser's neighborhood, while the optical output power of the laser source depends on the network bandwidth requirements and physical layout of the network. We first explore the power, efficiency and temperature tradeoffs associated with on-chip laser source. After that, using a 3D manycore system, we explore the design space for sharing and placement of the laser source by jointly considering the network bandwidth requirements, thermal constraints and physical layout constraints to determine the optimal sharing and placement of the laser sources, which minimizes the laser power. As part of this exploration we consider three different topologies – 8-ary 3-stage Clos, 16-ary 3-stage Clos and 16 x 16 crossbar, two different physical layouts – U-shaped and W-shaped and three different sharing/placement strategies – locally-placed laser sources with no sharing, locally-placed laser sources with sharing and laser sources placed along the edge with sharing. Our analysis shows that depending on the network topology, physical layout and waveguide losses, the sharing granularity and placement decisions of the laser source changes.

Finally, in Chapter 6, we present a manycore architecture that uses silicon-photonics links that features high bandwidth density and low latency communication, to achieve efficient global sharing of computing resources located in a globally-shared Execution Unit Cloud (EUCloud). Manycore systems exploit massive thread-level parallelism sacrificing instruction-level parallelism. This translates into lower utilization of the core's execution units (EUs) that creates opportunities for sharing of EUs (rather than using the complete set of EUs per core), which in turn can reduce area and static power. EU sharing is implemented in today's systems, but these mechanisms are not devised to adapt well to manycore systems running multi-programmed workloads or server consolidation scenarios. We consider a 256-core processor manufac-

tured at 16 nm, an EUCloud composed of Floating Point Units (FPUs), and we use multi-programmed workloads with a representative set of benchmarks to evaluate the EUCloud architecture. Our analysis shows that FPUs have a maximum of 20% utilization rate for the most FPU-intensive workload but occupy around 22% of the entire core area and consume 16% of static power. In our proposed manycore system with EUCloud, to sustain application performance we need just 96 FPUs instead of the 256 FPUs in the nominal case, which results in 13.75% and 10% reductions in the total manycore area and power respectively. We harness the unused area and power budgets of each core to boost performance by using larger caches achieving 8.13% performance improvement for the nominal power budget. Alternately, we can increase the core complexity, thus leading to 29.4% higher performance for an 8% increase in power budget.

Chapter 2

Broadcast through Express Virtual Channel with Taps in Electrical NoC

1

2.1 Introduction

The general trend for NoC architectures is towards designing low-radix high-diameter network topologies (e.g., mesh) that have short router-to-router channels (Bell et al., 2008; Howard et al., 2010). These topologies are easier to design from the hardware perspective. However, mapping an application to these topologies is difficult due to the large variance in packet latencies. High-radix low-diameter topologies such as crossbar are more amenable to application mapping due to low network diameters, but are difficult to design from the hardware perspective because of the long wires.

Another issue with NoC-based manycore systems is maintaining cache coherency across multiple caches. Previously, snoop-based and directory-based cache coherency protocols have been investigated for manycore systems (Culler et al., 1998). Snoop-based cache architecture uses a broadcasting mechanism for cache coherency, and is commonly used for bus-based topology in systems with a small number of cores

¹This chapter was previously published. © 2011 IEEE. Reprinted with permission from Chao Chen, Jie Meng, Ayse K. Coskun and Ajay Joshi, “Express Virtual Channels with Taps (EVC-T): A Flow Control Technique for Network-on-Chip (NoC) in Manycore Systems,” 2011 IEEE 19th Annual Symposium on High Performance Interconnects (HOTI), August 2011 (Chen et al., 2011)

(e.g., fewer than 10 cores). This protocol, however, does not scale well for NoC-based manycore systems due to the packet latency distribution.

To harness the true potential of manycore systems we need to develop low-cost, high-performance and energy-efficient NoC architectures. This chapter makes two contributions towards achieving this goal. First, we propose a new broadcasting mechanism for snoop-based cache coherency protocol for manycore systems. Each core uses a dedicated notification tree to rapidly inform all other cores of an incoming broadcast message. In this way, other cores can make early decisions to wait for the packet or to proceed with execution in presence of simultaneously transmitted packets over the shared network. The proposed broadcast mechanism has higher performance in comparison to conventional snoop-based broadcast mechanisms for low network traffic. For high network traffic, the performance of the proposed approach is similar to the conventional approaches.

We also propose a novel network flow-control mechanism: Express Virtual Channels with Taps (EVC-T) (Chen et al., 2011). Our flow-control mechanism, when mapped to a physical concentrated mesh (cmesh) network, results in a logical topology with high radix and low diameter. As a result, this NoC is easy to design from the hardware perspective and easy to program. The logical topology is similar to multidrop express channel (MECS) (Grot et al., 2009). However, unlike MECS, the proposed logical topology supports both data and broadcast packets, and does not use physically separated express channels.

Our novel contributions in the area of broadcast in electrical NoC are as follows:

- To maintain the sequential consistency for snoop-based cache coherence, we propose using a contention-free notification tree per core as a supporting network for broadcasting. These notification trees ensure each core knows when to ex-

pect a broadcast packet and the exact timestamp when a broadcast packet was generated. This information guarantees that on average a core has to wait for less than a cycle before it can make the decision about processing a broadcast packet and all the broadcast packets are processed in the correct order.

- To achieve low network latency and save energy, we propose EVC-T flow control mechanism to transmit both broadcast and data packets in a snoop-based cache coherency protocol. For broadcasting, our EVC-T technique allows intermediate routers to receive and store broadcast packets while forwarding them to downstream routers simultaneously. For NAS parallel benchmarks (Bailey et al., 1994), EVC-T reduces the average packet latency (data and broadcast) by 24% and improves the system energy efficiency, reducing the energy delay product (EDP) by 13% on average.

In this chapter, Section 2.2 provides the details of our target system. Section 2.3 explains the use of notification trees as supporting networks for cache coherence. Section 2.4 describes the EVC-T flow control technique. Section 2.5 evaluates our techniques using synthetic traffic and the NAS parallel benchmarks. Section 2.6 summarizes our analysis.

2.2 Target System

We choose a 64-core processor as our target system that is manufactured using 22 nm technology process as a representative node for future manycore chips (Kuhn et al., 2010). Each core on the processor supports 2-way issue out-of-order execution, and has two integer ALUs, one integer multiplication unit, one floating-point ALU, and one floating-point multiplication unit. The core architecture is configured based on the cores used in Intel’s 48-core SCC (Howard et al., 2010). The micro-architectural

parameters are listed in Table 2.1. The cores operate at 1 GHz frequency and have a supply voltage of 0.9 V, while the on-chip network operates at 2 GHz.

Each core has 16 KB private L1 instruction cache and 16 KB private L1 data cache. We use a shared memory programming model and explore a distributed L2 cache architecture. The manycore system uses the snoop-based MESI protocol for maintaining the cache coherence.

Figure 2-1 shows the physical layout of our 64-core target system. There are 64 cores, 16 L2 cache banks (1 MB each), 16 memory controllers that are uniformly distributed across the chip. Four cores and one L2 bank share one router and communicate with other cores and L2 banks through the on-chip network. Each memory controller is associated with one L2 bank and there is a dedicated channel between them, which is not shown in Figure 2-1. Each router has four-cycle zero-load latency for the four pipelined routing stages: route computation, virtual channel allocation, switch allocation, and switch traversal (Krishna et al., 2009). After energy optimization by repeater insertion, each 5 mm channel between two neighboring routers has single-cycle latency.

Table 2.1: Micro-architectural parameters of the 64-core system

Micro-architecture Configuration	
Core Frequency	22nm, 1.0 GHz @ 0.9 V
Branch Predictor	Tournament predictor
Issue	2-way Out-of-order
Reorder Buffer	128 entries
Functional Units	2 IntALUs, 1 IntMult, 1 FPALU, 1 FPMult
Physical Regs	128 Int, 128 FP
Instruction Queue	64 entries
Private L1 I/D-Cache	16 KB each @ 2 ns
Distributed L2 Cache	4-way, 64B/block, 16 x 1 MB @ 6 ns
Cache Coherence	Snoop based MESI (Papamarcos and Patel, 1984)
NoC	mesh @ 2.0 GHz
Memory	16 MCs @ 100 ns

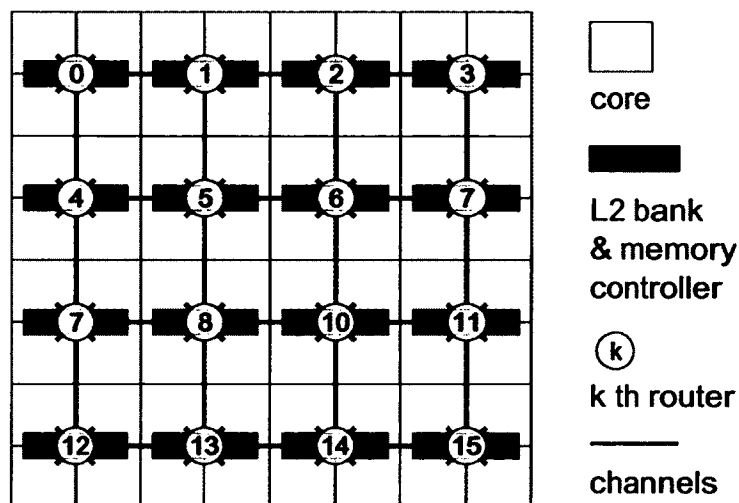


Figure 2-1: Physical layout of our 64-core system with electrical NoC. – 16 L2 cache banks (1 MB each) are uniformly distributed across the chip. Each L2 bank has one memory controller physically located next to it. Each router uses a concentration of 4 cores and one L2 cache bank. It has 13 input and 13 output ports (4 for inter-router interconnect, 8 for L1 I-cache and D-caches of 4 cores, and 1 for 1 L2 cache bank).

2.3 Notification Trees for Broadcasting Order

For network topologies such as cmesh, clos or crossbar, where the network enables parallel accesses, multiple sources can insert packets into the network at the same time. The latency of each packet varies based on the traffic workload as well as the physical location of its source L1 cache. Therefore, a destination L1 cache can potentially receive broadcast packets in a different order than the original order in which the broadcast packets were generated. Hence, a destination L1 cache needs to wait for all broadcast packets that are on the fly before processing the received broadcast packet. The worst-case waiting time can be determined using empirical methods. However, depending on the size of the manycore system, the waiting period could be considerably long, which has a negative impact on the system performance. In our 64-core target system with a cmesh network, the cores at the corners have

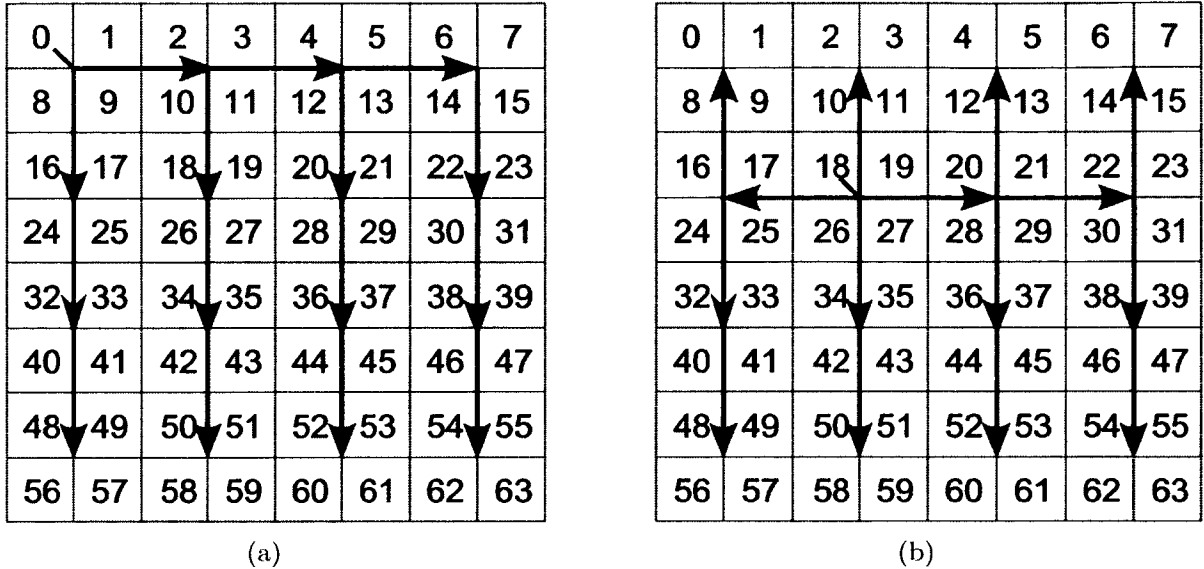


Figure 2-2: Example of Notification trees – (a) Notification tree from core 0. (b) Notification tree from core 18.

to wait for more than 34 cycles assuming the shared cmesh network has zero-load latency. As the network traffic becomes high, the waiting time increases because of network contention. When an L1 cache knows in advance how many cycles to wait for the broadcast packets that are already on-the-fly, we can avoid the wasted cycles at the cache.

We propose using notification trees along with the existing shared network to maintain the sequential consistency of broadcast packets that are transmitted on unordered interconnects. Each L1 cache in a core has a dedicated notification tree, connecting it to all other L1 caches. Figure 2-2 shows the notification trees from the L1 caches in core 0 and core 18. Each notification tree is pipelined and each pipeline segment uses energy-optimized repeater-inserted single-bit wire.

The notification tree for an L1 cache sends a notification pulse to all other destination L1 caches whenever it has a read/write cache miss, and a new broadcast packet requesting the missing cache line is generated. The actual broadcast packets

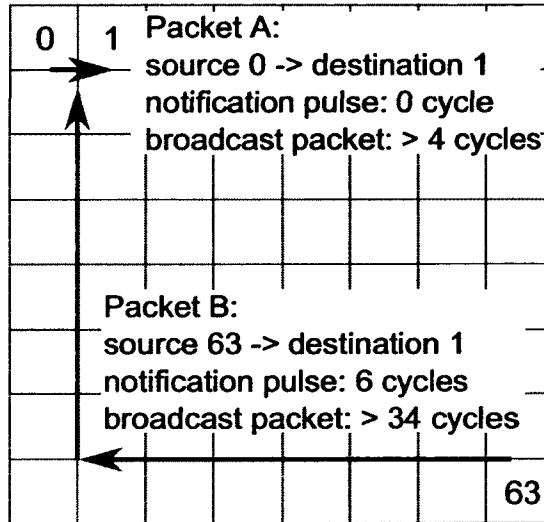


Figure 2.3: Example of broadcast packets – Here, broadcast packets are sent from core 0 and core 63 to ALL receivers across the chip. For clarity, only one receiver is shown.

are transmitted over the shared network. As the notification trees are contention free, the notification pulses reach destination L1 caches within fixed latencies. Thus, each destination L1 cache can determine the exact waiting time before other received broadcast packet get processed. The waiting time of any broadcast packet is calculated by subtracting the broadcast packet transmission time from the notification transmission time for the farthest network source. The maximum waiting time in a destination L1 cache depends on its physical location relative to other L1 caches.

Table 2.2: Timing analysis of the notification tree architecture – Notification arrival time corresponds to the latency of a notification pulse through the dedicated notification tree. Packet arrival time corresponds to the zero-load latency of a broadcast packet traveling through the shared cmesh network.

Packet ID	Source Core	Send Time	Dest. Core	Notification Arrival Time	Packet Arrival Time
A	0	T	1	T	T + 4
B	63	T - 1	1	T + 5	T + 33

Figure 2-3 shows an example for two broadcast packets: packet A and packet B from the L1 caches in core 0 and core 63, respectively. Packet A reaches the L1 cache of core 1 after traveling through one router, resulting in a zero-load latency of 4 cycles. Packet B reaches the L1 cache in core 1 after traveling through 7 routers and 6 channels, resulting in a zero-load latency of 34 cycles. On the other hand, the notification pulses for both packets reach their destinations much faster. The notification pulse for packet A reaches the L1 cache in core 1 right after packet A is generated and the notification pulse for packet B reaches the L1 cache in core 1 in 6 cycles after traveling through 6 wire segments on the notification tree.

In our analysis, packet A is generated at time ‘T’, while packet B is generated at time ‘T-1’. The notification pulses for packet A and B reach the L1 cache in core 1 at time ‘T’ and ‘T+5’, respectively. To maintain cache coherency, packet B has to be processed before packet A. After the L1 cache for core 1 receives the broadcast packet A at ‘T+4’, it monitors the notification trees for the remaining L1 caches for an additional cycle. The 1-cycle waiting time is calculated based on the fact that the core 63 is farthest away from core 1, and it takes 6 cycles for a notification pulse to be transmitted from the L1 cache in core 63 to the L1 cache in core 1. As the L1 cache in core 1 receives the notification for packet B at ‘T+5’, it can formulate the correct order for processing the packets. On average, the L1 cache in a core has to monitor the notification trees for less than a cycle before it can decide on the broadcast packet processing order. The exact time when a broadcast packet is processed depends on the network latency of the broadcast packet. Table 2.2 shows the timing analysis of these two broadcast packets at their sources (core 0 and core 63) and at the destination (core 1) as an example.

The overall hardware overhead for the proposed broadcast technique includes 64 notification trees, buffers in each core to store incoming notifications, and combinational

logic in each core to decide on the processing order. The area overhead of 64 notification trees is 31% with respect to the wiring area of the existing shared network and the power overhead is 0.27 W. The area and power overheads for buffers and combinational logic are negligible. The total hardware overhead can be reduced by $4\times$ using a shared architecture, where a group of 4 cores shares a notification tree.

2.4 EVC-T Flow Control Mechanism

In this section, we propose a novel flow control mechanism that, when used with a low-radix high-diameter physical network, provides a high-radix low-diameter logical topology. This reduces the network latency for broadcast and data packets, and therefore improves the manycore system performance. We introduce the traditional EVC flow control mechanism as the background at first, and then describe our proposed EVC-T flow control mechanism and its application in our target system.

2.4.1 Express Virtual Channels (EVC)

The EVC flow control mechanism proposed in (Kumar et al., 2008) enables packets to entirely bypass routers. Figure 2-4 shows the router architecture supporting EVCs. Each router receives four types of packets – a packet generated by a core connected to that router, a packet that will bypass the router, a packet that will change direction ($X \rightarrow Y$) in the router, and a packet that has one of the attached core as its destination. The EVC controllers at the input ports differentiate between the packets that will be buffered by the router (change direction, get ejected to, or are injected from an attached core) and those that will bypass the router. The packets that bypass the router get priority to access the downstream inter-router physical channel among all packets.

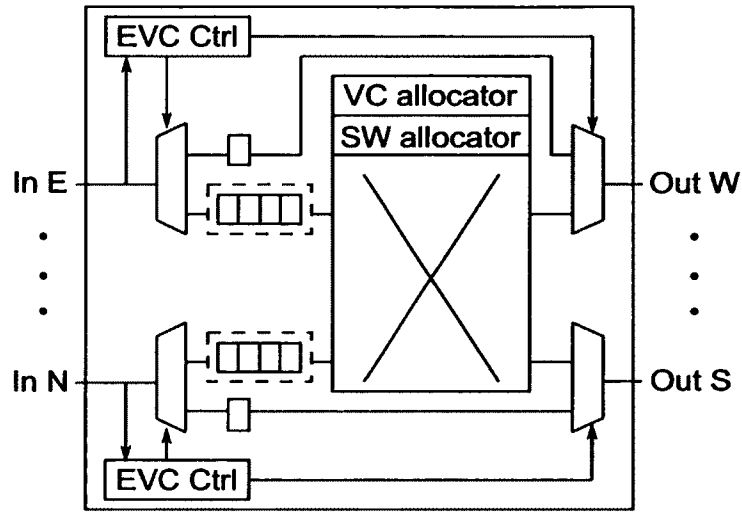


Figure 2-4: The router architecture for EVC (Kumar et al., 2008). A packet received at an intermediate router of an EVC is immediately forwarded to the subsequent physical channel of the same EVC. A packet received at a destination router of an EVC is buffered and switched to channels in another direction.

At low network traffic, when using EVCs, a data packet effectively passes through a series of inter-router channels as it bypasses all intermediate routers. As a result, it has lower latency compared to the equivalent networks with no EVCs. On the other hand, at high network traffic, the latency and saturation throughput are comparable to equivalent networks with no EVCs. Figure 2-5 shows an example network, where the network has the same physical layout as cmesh topology; i.e, it connects the neighboring routers with short physical channels. By connecting distant routers with EVCs, this network approaches the low zero-load latency of physical flattened butterfly topology while maintaining the low energy cost benefits of the traditional cmesh topology.

The limitation of EVC is that they cannot transmit broadcast packets as efficiently as data packets. On a traditional cmesh network, broadcast packets are transmitted through multiple hops and at each hop each broadcast packet is duplicated and transmitted in both X and dimensions. Here, a broadcast packet is transmitted only

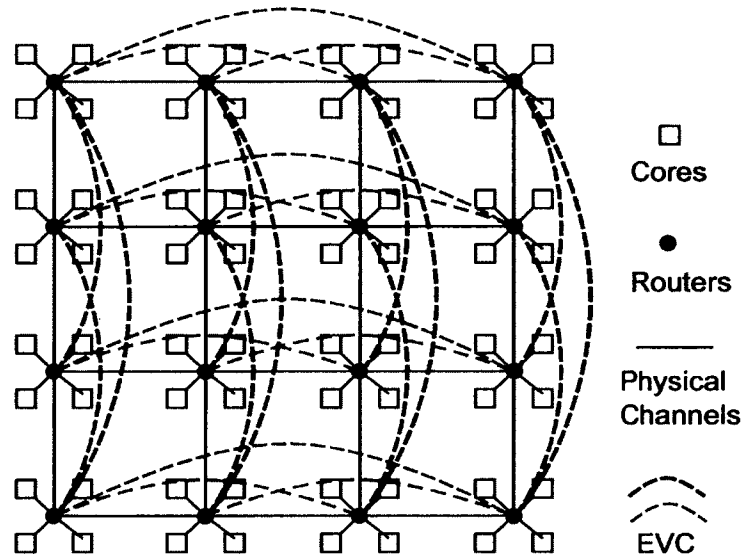


Figure 2-5: The C-EVC network: physical cmesh layout with EVC – EVCs use the existing physical channels to establish express paths between each pair of two distant routers in both X and Y dimensions. The resulting logical topology is effectively a flattened butterfly.

once through any physical channel. However, if the EVC flow control mechanism is used, a broadcast packet is transmitted through multiple EVCs from one source router to several destination routers, resulting in multiple transmissions through one physical channel shared by those EVCs. This will increase the network congestion, which increases the packet latencies and negatively affects the manycore system performance. We propose an upgraded EVC-T flow control mechanism that transmits both broadcast and data packets with low latency and power consumption on a single shared network.

2.4.2 Express Virtual Channels with Taps (EVC-T)

The EVC-T maintains the key feature of EVC: reducing packet latency through router bypassing. The difference is that EVC-T establishes an express virtual path from one source router to multiple receiver routers. For data transmission, only one target

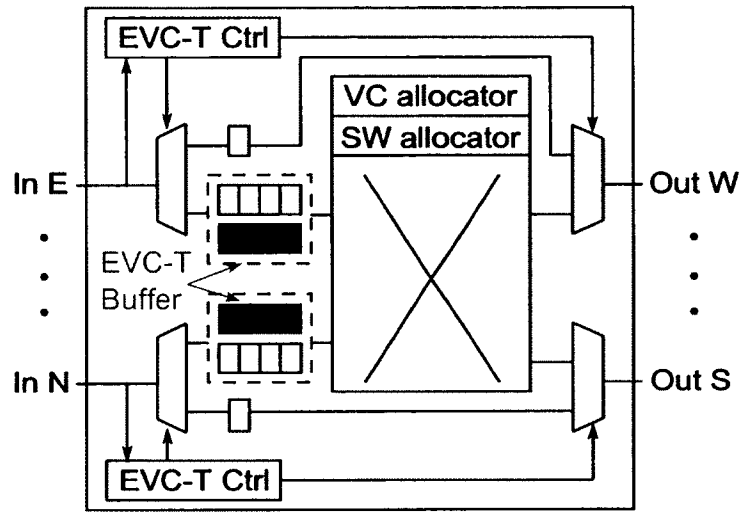


Figure 2-6: The router architecture supporting EVC-T – The received packets can be buffered in the router and forwarded to the subsequent channel simultaneously.

receiver router buffers the data packet, while other routers are bypassed. For broadcasting, all receiver routers simultaneously buffer and forward the broadcast packet. The source router uses one EVC-T to transmit a broadcast packet to all receiver routers in each direction (east, west, north, or south). This ensures that a broadcast packet is transmitted only once through any physical channel.

Figure 2-6 shows the router architecture supporting EVC-T. Each EVC-T has input buffers at all receiver routers, including the intermediate routers and the destination router. Thus, an incoming broadcast packet can be simultaneously forwarded to the subsequent channel in the current direction, while buffered and switched to channels in other directions. The EVC-T controllers make decisions for buffering and/or forwarding the incoming packets. Similar to EVC, the packets that are bypassing the router will have priority access to the subsequent physical channel in comparison to other packets that are switching directions in the router.

Figure 2-7 shows an example network using EVC-T. Here we use the same router ID as labeled in Figure 2-1. On the top of physical cmesh layout, each router is connected

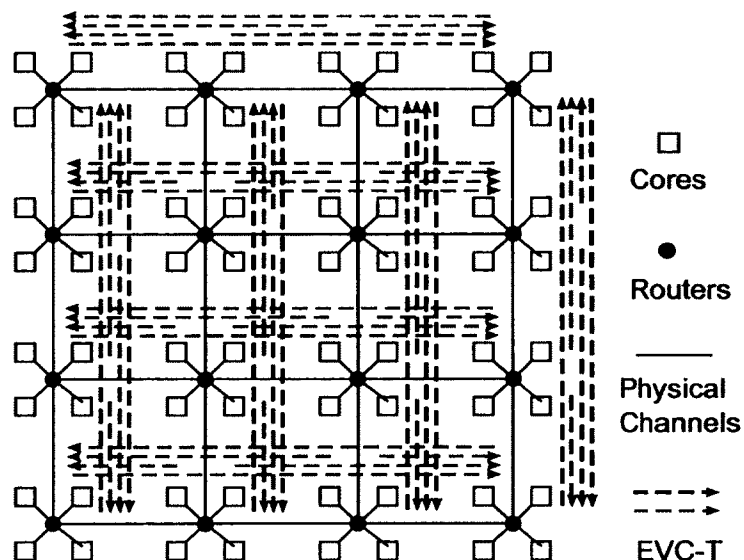


Figure 2-7: The C-EVC-T network: the physical cmesh layout with EVC-T – Single EVC-T connects one source router to all receiver routers in each direction (east, west, north, or south). The resulting logical topology is similar to MECS.

to multiple receiver routers through a single EVC-T in each direction (east, west, north, or south). For example, a broadcast packet injected at router 0 is transmitted to router 1, 2, and 3 through a single EVC-T to the east. Each of these three routers buffer and transmit the broadcast packet to three more routers through its EVC-T to the south. The broadcast packet reaches all network destinations within two network hops – one in each dimension. For another example, a data packet from router 0 to router 14 is transmitted through the same EVC-T to the east used by the above broadcast packet. Router 2 buffers and transmits the data packet to router 14 through its EVC-T to the south.

Unlike the data packets, the broadcast packets usually request the EVC-Ts for multiple output ports at routers. For example, a broadcast packet injected at router 0 would request an EVC-T to the south and an EVC-T to the east at the same time. The routing delay and energy costs will be reduced if the VC allocator grants both

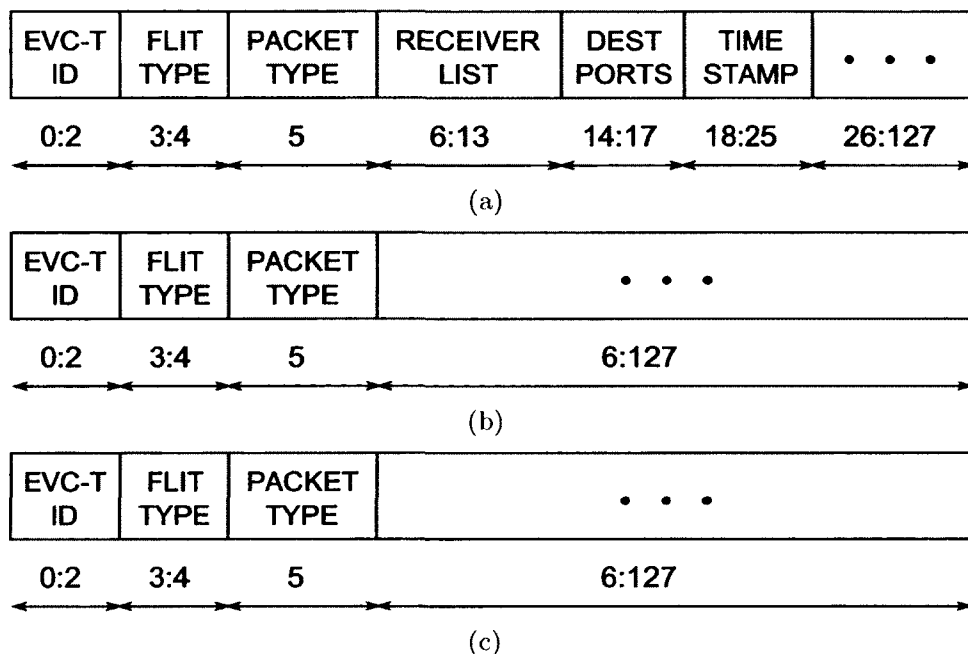


Figure 2-8: Flit organization of an EVC-T packet – (a) Head flit (b) Body flit (c) Tail flit

EVC-Ts and the crossbar transmits the broadcast packet to both output ports simultaneously. We design our VC allocator based on the ISLIP routing scheduling algorithm (McKeown, 1999). We change the grant phase of ISLIP algorithm to grant one EVC-T for each requested output port. The SW allocator controls the switch crossbar and grants the switching paths between crossbar input ports and output ports. We change the traditional SW allocator to turn on several switches along a crossbar input wire, which enables one-to-many transmission through these crossbar switches. With our upgraded VC router, the broadcast packets have the same zero-load routing delay as the data packets. In the shared network, broadcast and data packets have the same priorities; i.e., a broadcast packet has to wait if another packet is using the requested output port.

2.4.3 Flit Organization for EVC-T

Figure 2-8 shows the flit organization of an EVC-T packet. The EVC-T controller uses this information to decide between buffering and/or forwarding a packet. Because there are 6 EVC-Ts in each row or column, each EVC-T uses a unique 3-bit identification number (EVC-T ID). The EVC-T controller uses this identification number to store the packet in the appropriate input buffer of a router as each router maintains a dedicated input buffer for each incoming EVC-T. The 2-bit FLIT_TYPE identifies the type of the received flit: header flit (10), body flit (00), or tail flit (01). The EVC-T controllers make buffering and/or forwarding decisions based on the data in the header flit and keep the decisions for the following body flits and tail flits. The PACKET_TYPE identifies whether the received packet is a broadcast packet (1) or a data packet (0). The 8-bit RECEIVER_LIST (4-bits for each dimension) indicates the routers along the EVC-T path where the packet has to be buffered. For example, for a broadcast packet, RECEIVER_LIST has a value of 1111.1111. For a data packet, there is only one receiver router and two bits (one for each dimension) will be set in the RECEIVER_LIST field. For example, a data packet from router 0 to router 14 uses 0010.0001 to notify router 2 and router 14 to buffer the received data packet. The 4-bit DEST_PORTS indicates the ejection ports at the destination routers for data packets. In our target system, 9 ejection ports are used because 4 cores (4 L1 ICache caches and 4 L1 DCache) and 1 L2 bank share one router. For broadcast packets, the DEST_PORTS is not used because the receiver routers must forward broadcast packets to all ejection ports. The 10-bit TIME_STAMP is the packet generation time used for our proposed cache coherence technique in Section 2.3. The 10-bit length is designed to cover the time interval between notification arrival time and broadcast packet arrival time, which is variable due to the potential contentions in the shared network. When broadcast packets reach their destinations, they are matched with

corresponding notification pulses by comparing the `TIME_STAMP` with the stored notification pulse generation time. The remaining bits in the packet are used for data.

2.4.4 Credit Channels for EVC-T

For avoiding buffer overflow, a source router maintains the buffer status of all receiver routers of connected EVC-Ts. Each receiver router has a dedicated credit channel to the source router. For the C-EVC-T network in Figure 2-7, each pair of routers in the same row or column is connected with a pair of credit channels. The overhead of these credit channels are minimal because they are 1 to 2-bit wide. A source router grants an EVC-T after making sure that all target receiver routers have available credits. For example, a broadcast packet injected at router 0 is granted with an EVC-T to the east only when router 1, 2, 3 have available credits, and a data packet (from router 0 to router 14) is granted with the above EVC-T only when router 2 has at least one available credit. The source router deducts a credit for each target receiver router after sending every flit and increments a credit after receiving a new credit through a dedicated credit channel.

2.5 Evaluation

In this section, we evaluate our proposed EVC-T technique using both synthetic traffic running on BookSim simulator, and NAS parallel benchmarks running on an integrated Gem5-BookSim full-system simulator. We compare four networks as shown in Table 2.3. The first three networks have the same cmesh physical layout but different logical topologies. VC type is the flow control techniques used to build various logical topologies. VC count is the number of VC, EVC, or EVC-T channels

used between one source router and any of its destination routers. The C-VC network uses three VCs to connect neighboring routers and has a cmesh logical topology. The C-EVC network uses a dedicated EVC to connect each pair of routers in both X and Y dimensions, as shown in Figure 2-5, and has a flattened butterfly (flatfly) logical topology. The C-EVC-T uses a single EVC-T to connect one router to multiple routers in each direction (east, west, north, or south), as shown in Figure 2-7, resulting in a logical topology similar to the MECS network proposed in (Grot et al., 2009). The MECS network has a logical and physical MECS topology that uses a one-to-many communication model enabling a high degree of connectivity in a bandwidth-efficient manner. However, the logical MECS topology in the C-EVC-T network supports more efficient broadcast packet transmission and has less hardware overhead than the physical MECS network.

All four networks support both broadcast and data packets. The data packets in the C-EVC network and C-EVC-T reach the destination routers through virtual express paths, while the data packets in C-VC reach the destination routers through multiple hops. The broadcast packets in C-EVC-T reach multiple destination routers through a single virtual express path, while the broadcast packets in C-VC and C-EVC network reach destination routers through multiple hops. The C-EVC-T is the only network that uses virtual express paths to transmit both broadcast and data packets.

Table 2.3: Network architecture details

Network	Physical Layout	Logical Topology	VC Type	VC Count
C-VC	cmesh	cmesh	VC	3
C-EVC	cmesh	flatfly	EVC	1
C-EVC-T	cmesh	MECS	EVC-T	1
MECS	MECS	MECS	N.A	N.A.

2.5.1 Simulation Methodology

We use BookSim network-on-chip simulator to evaluate synthetic network traffic patterns. To evaluate the impact of the proposed EVC-T technique on the whole many-core system, we integrate BookSim into Gem5 full-system simulator. Gem5 is an event-based manycore simulator that uses Alpha instruction set architecture (ISA), while BookSim is a cycle-precise network simulator. BookSim is integrated as a sub-module of Gem5, and we add a network interface for handling the communication between the two simulators. The packets generated by Gem5 are converted to the BookSim format and injected into the network instantiated by BookSim. The injected packets are preserved in a flying packet list in the network interface for tracking. Gem5 schedules events for checking the network output ports at every cycle if there are outstanding packets in the flying packet list. When completed network packets are detected by Gem5 events, they are converted back to the Gem5 format by referencing the flying packet list and are processed at the destination caches.

Using our integrated Gem5-BookSim full-system simulator, we run 8 benchmarks from the NAS parallel benchmark suite (`mg`, `ep`, `is`, `cg`, `lu`, `sp`, `ua`, and `ft`) with class B problem set. We fast-forward 2 billion instructions to warm up the system for avoiding cold-start effects and to reach the parallel execution phase of these applications. We execute 1 billion instructions after the fast-forward phase using the detailed out-of-order CPUs in Gem5 for all benchmarks to quantify their performance. The performance statistics are also used as inputs for our power model.

We use application IPC, defined in Equation (2.1) (Meng et al., 2011), as the metric to evaluate the performance of the benchmarks. This metric considers the variations of the execution time among different threads. It accumulates all the instructions executed across all threads and then divides the total instruction count by the number

of cycles for the longest thread, as the longest thread determines the application finish time.

$$IPC_{app} = \frac{\sum_{i=1}^{num_core} Committed_instructions_{core[i]}}{\max_{1 \leq i \leq num_core} Number_of_cycles_{core[i]}} \quad (2.1)$$

To estimate power for the cores in our target system, we utilize McPAT 0.7 (Li et al., 2009) that computes the power costs based on the performance statistics collected by Gem5. We calibrate the McPAT outputs to match the published core power values of the Intel SCC (Howard et al., 2010) using the scaling method introduced in (Meng et al., 2011). The L2 cache power is computed using CACTI 5.3 (Thoziyoor et al., 2008). Since the current version of CACTI does not support 22 nm process technology, we calculate L2 cache power in 32 nm technology and calibrate it using the same scaling method used for calculating core power.

The power for the network is estimated using detailed circuit modeling. For the 64-core target system, we use energy-optimized repeater-inserted wires for inter-router channels. The power dissipated in the SRAM array and crossbar of the router is calculated using the methodology described in (Liang et al., 2007) and (Wang et al., 2003), respectively. We design the network channels and routers using PTM for 22 nm technology (Kuhn et al., 2010). The static power consumed by the network depends on the physical layout and the dynamic power is determined by the flow control mechanism and network traffic workloads.

We use EDP defined in equation (2.2) as a metric to evaluate system energy efficiency. *System_power* includes core power, cache power and NoC power. The running time is calculated by dividing the maximum number of execution cycles among all the cores by the system frequency.

$$\begin{aligned}
 EDP &= System_power \cdot Application_running_time^2 \\
 &= System_power \cdot \left(\frac{Longest_thread_execution_cycles}{System_frequency} \right)^2
 \end{aligned}
 \tag{2.2}$$

2.5.2 Evaluation Results with Synthetic Benchmarks

For synthetic network traffic, we assume uniform random (UR) traffic pattern, which is widely used for NoC evaluations. We consider UR selection of sources and destinations for data packets and UR selection of sources for broadcast packets. We assume the same number of broadcast and data packets are injected into the network. However, for real-world network traffic, a higher number of broadcast packets may exist in comparison to the number of data packets due to operations such as write back and global status update. For example, an L1 cache requests exclusive access of a cache line by sending a broadcast packet to all other L1 caches. The receivers remove the local cache lines without sending response packets. Such packets are ignored in the synthetic traffic simulation but are included in the Gem5-BookSim full-system simulation.

Figure 2.9 compares the latencies and power costs of the four networks listed in Table 2.3 for data traffic, broadcast traffic, and mixed traffic of broadcast and data packets. For data traffic, in Figure 2.9(a) and Figure 2.9(b), the C-EVC and C-EVC-T networks have lower latencies than the C-VC network because some data packets are bypassed at several intermediate routers. The C-EVC-T and MECS network have comparable low latencies when the network traffic is not high. The MECS network has higher saturation throughput than the C-EVC-T network but the physical express channels in the MECS network consumes lots of fixed power, which is inefficient for

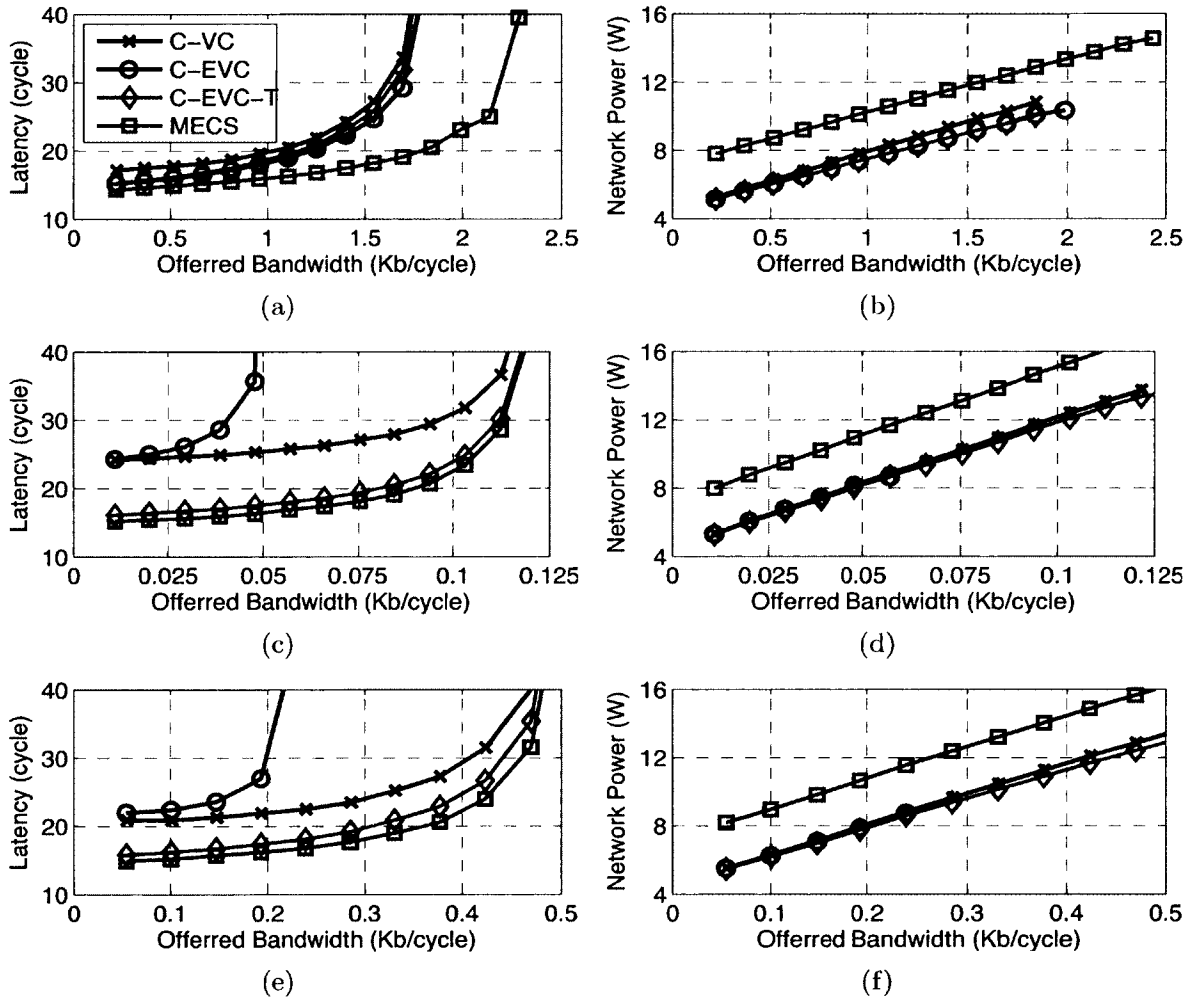


Figure 2-9: Network latency and power vs. offered bandwidth – We use uniform random traffic. (a) and (b) has data traffic; (c) and (d) has broadcast traffic; (e) and (f) has mixed traffic of data and broadcast traffic.

low network traffic. The C-VC, C-EVC and E-EVC-T networks have comparable power costs because they are using the same numbers of physical channels, router buffers and crossbar sizes. The slight power difference is due to their respective router control logic. When the network traffic increases, the C-EVC and C-EVC-T networks have lower power costs than the C-VC network because the dynamic power is not consumed in buffering and crossbar switching at the bypassed routers in the C-EVC and C-EVC-T networks.

For broadcast traffic, in Figure 2.9(c), the C-VC and C-EVC network have higher latencies than the C-EVC-T network because of the multiple hop transmission of broadcast packets in these two networks. Here, we assume that broadcast packets complete network transmission after reaching all destinations. In a traditional MECS channel, a data packet has one target receiver router and only the target receiver router drops the passing packet. For comparison of broadcast traffic in the C-EVC-T and MECS networks, we modify the packet format and router architecture of the MECS network to support broadcasting. A broadcast packet in MECS channel has multiple target receivers, and multiple target routers drop the passing broadcast packet. Figure 2.9(d) shows that the EVC-T and MECS networks have comparable latencies and saturation throughput for broadcast traffic. The C-EVC-T and MECS networks get saturated by high broadcast traffic because of increased contentions inside the routers. The higher bisection bandwidth of the MECS network does not increase its saturation throughput for broadcast traffic.

The mixed traffic of broadcast and data packets provides a realistic comparison of the four networks. In Figure 2.9(e), the latency is the average value for all broadcast and data packets. The C-EVC-T shows better performance than the C-VC and C-EVC networks because it is the only network that uses virtual express paths for both broadcast and data packets. The saturation throughput of the C-EVC-T network

is similar to the C-VC network. In Figure 2.9(f), the C-EVC-T network has the similar power costs to the C-VC network. In summary, the MECS network provides the best performance—low network latency and high saturation throughput because of its physical express channels and high bisection bandwidth. The C-VC network provides the best energy efficiency—low power cost, due to its short channels and simple control logic. The proposed C-EVC-T network can achieve the similar high performance to the MECS network while maintaining the similar low energy costs to the C-VC network.

2.5.3 Evaluation Results with NAS Benchmarks

We next evaluate our EVC-T technique by running the NAS parallel benchmark suite on our Gem5-BookSim simulator. In the full-system simulation, we compare the C-EVC-T and C-VC networks to demonstrate that EVC-T can help improve the system IPC and reduce the EDP of parallel applications. Figure 2-10 shows the full-system simulation results for NAS parallel benchmarks. Figure 2.10(a) shows the offered bandwidth for each benchmark. For our distributed L2 cache architecture, the offered bandwidth is calculated as the number of transmitted packet per cycle multiplied by the packet size. We assume that one data packet consists of four 128-bit flits and one broadcast packet consists of single 128-bit flit. The offered bandwidth shows that NAS benchmarks have various network demands, but none of them reaches the saturation region of the C-VC and C-EVC-T networks (see Figure 2.9(e)). For the offered bandwidth, the C-EVC-T network has lower average latency as shown in Figure 2.10(b). On average, the C-EVC-T network reduces the latency by 24%. If the C-EVC-T network runs benchmarks that have higher offered bandwidth, then the performance and power will be comparable to C-VC, which still outperforms than C-EVC.

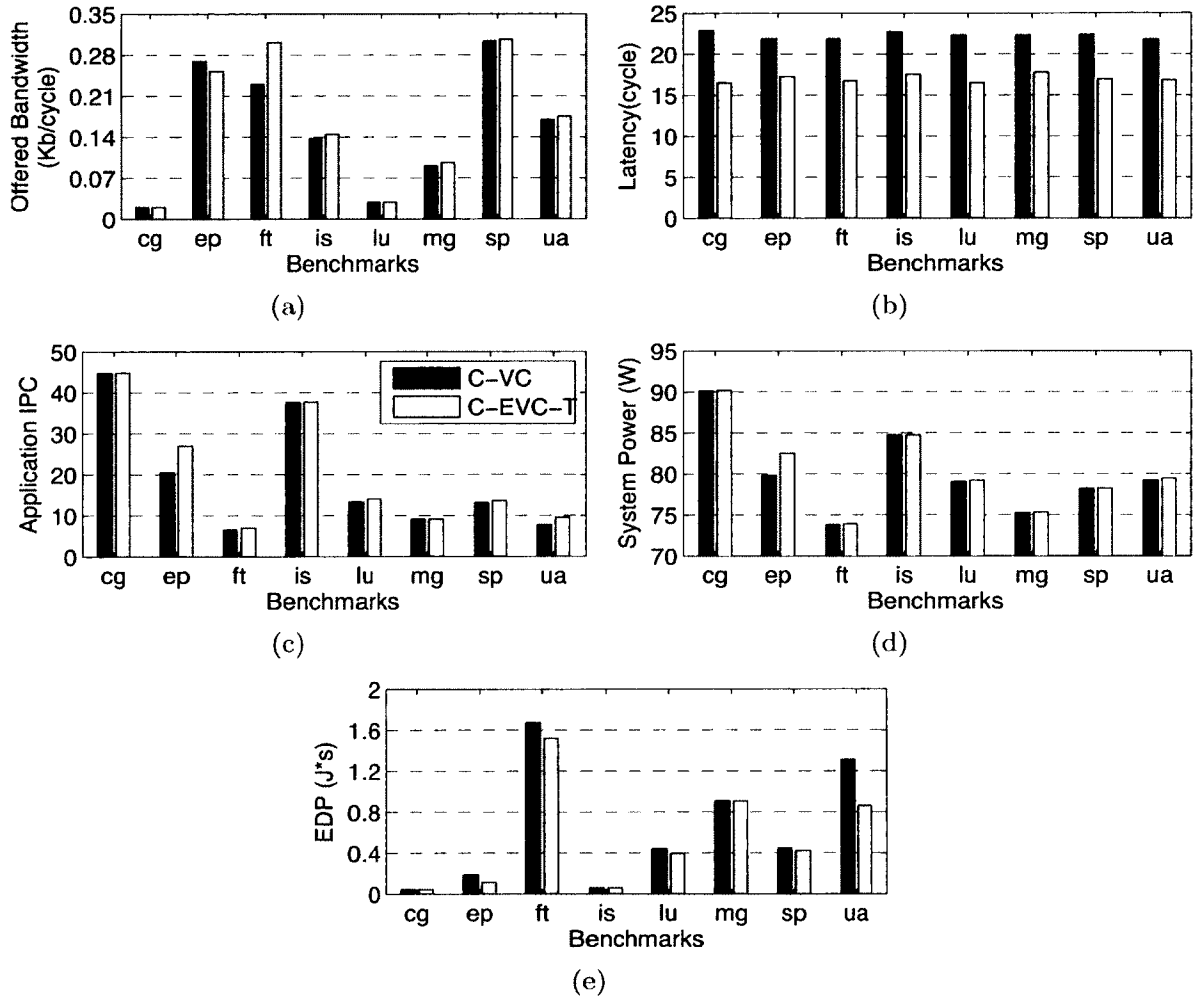


Figure 2.10: Comparison of C-VC and C-EVC-T networks – We run NAS benchmarks on the 64-core target system. (a) Offered bandwidth. (b) Average network latency. (c) Application IPC. (d) System power. (e) EDP. The detailed system configuration is given in Table 2.1. BookSim was integrated into Gem5 to enable a detailed cycle-accurate comparison.

Figure 2.10(c) shows the application IPC for each benchmark. The ‘ep’ and ‘ua’ benchmarks show an application IPC improvement of 31% and 24%, respectively, after using EVC-T. This improvement is a result of the reduced execution time of the longest thread (see Equation 2.1). The ‘ep’ benchmark is one of the highly parallel benchmarks in NAS benchmark suite and has the greatest improvement on application IPC. The application IPC of other benchmarks improve by up to 6%. Some benchmarks, such as ‘cg’ and ‘is’, have very high application IPCs, however, their low offered bandwidth indicates that they are relatively less dependent on network performance; i.e., they are not memory bound. Figure 2.10(d) shows the system power costs for each benchmark, which includes the power of networks and cores and caches. The ‘ep’ benchmark shows 3% power increase because of the improved IPC, while other benchmarks show less than 1% power increase because of the relatively non-uniform workload distribution. Figure 2.10(e) shows the EDP for each benchmark. The ‘ep’ and ‘ua’ benchmarks show 40% and 34% EDP reduction after using EVC-T. The average EDP reduction by EVC-T is 13%. The system simulation results show that our proposed EVC-T improves the system energy efficiency as well as the system performance.

2.6 Summary

We proposed a new flow control technique: express virtual channels with taps (EVC-T) for NoC architectures in manycore systems. When used with cmesh physical layout, EVC-T helps create a logical topology with low diameter. This provides a NoC architecture which is easy to design from the hardware perspective and easy to program from the software perspective. In addition, we have also proposed a contention-free tree architecture that supports broadcasting on unordered on-chip interconnects for snoop-based cache coherency protocols. The notification trees enable

a core to wait for less than one cycle after broadcast packet is received on average to make decisions on the correct processing order of broadcast packets.

We evaluated the EVC-T flow control technique and the new broadcast mechanism for snoop-based cache coherency protocols using BookSim network simulator that is integrated into M5 full-system simulator. We have explored both synthetic traffic and parallel benchmarks from the NAS suite. The synthetic benchmark analysis shows the potential of our proposed techniques where the average packet (data and broadcast) latency is reduced by 24%, while consuming the same amount of power as a conventional cmesh network. For NAS parallel benchmarks, our techniques increase the application IPC by 9% on average with negligible changes in power. The system energy efficiency (quantified by EDP) is increased by 13% on average.

Chapter 3

Bandwidth Reconfiguration for Managing Laser Power in Silicon-photonic NoC

1

3.1 Introduction

Future high performance computers (HPCs) and data centers (DCs) will use several manycore processors with each processor having dozens to hundreds of cores on a die. The performance of these manycore processors and in turn that of the HPCs and DCs will be driven by the energy-limited bandwidth of both processor-to-memory (inter-chip), and core-to-core/core-to-cache/cache-to-cache (intra-chip) communication networks. Hence, high bandwidth density and low power communication networks are needed to maximize the performance of these manycore processors. To this end, silicon-photonic links have been projected to supplant the electrical links, in both inter-chip and intra-chip communication networks in future manycore processors. Silicon-photonic link technology projections indicate an order of magnitude higher bandwidth density and several times lower energy cost compared to the projected electrical link technology (Batten et al., 2008; Joshi et al., 2009). This will significantly improve the throughput of both inter-chip and intra-chip communication

¹This chapter was previously published. © 2013 IEEE. Reprinted with permission from Chao Chen and Ajay Joshi, "Runtime Management of Laser Power in Silicon-Photonic Multibus NoC Architecture," IEEE Journal of Selected Topics in Quantum Electronics, March-April 2013 (Chen and Joshi, 2013)

networks, and also improve the energy efficiency of the overall manycore processor system.

Our work focus on intra-chip communication network designed using silicon-photonic link technology. The use of silicon-photonic link technology for intra-chip communication has been widely explored. Several different silicon-photonic intra-chip communication network architectures, referred to as network-on-chip (NoC) architectures, have been investigated (Joshi et al., 2009; Kirman and Martínez, 2010; Pan et al., 2009; A. Shacham, K. Bergman and L. P. Carloni, 2007; Kirman et al., 2006; Morris and Kodi, 2010; Psota et al., 2010; Vantrease et al., 2008; Cianchetti et al., 2009). Though the silicon-photonic NoC provides bandwidth density and data-dependent link energy advantages, a non-trivial amount of power is required in the laser source that is used for driving the silicon-photonic NoC (Joshi et al., 2009; Pan et al., 2010). In fact, the laser power could more than offset the bandwidth density and data-dependent energy advantages of the silicon-photonic links. To use silicon-photonic link technology for communication in future HPCs and DCs, it is imperative to explore techniques to reduce the power consumed in the laser sources.

Typically, the applications running on HPCs and DCs exhibit spatially variant and/or temporally variant behavior. The application characteristics including instructions committed per cycle, cache/memory accesses and generated NoC traffic could vary spatially across applications as well as temporally within an application. This provides us with an opportunity to proactively reconfigure the NoC architecture to minimize the power consumed in the laser source while maintaining application performance. In other words, we provide the minimum NoC bandwidth (which is directly proportional to laser power) required for an application to achieve the maximum possible performance (number of instructions committed per cycle) at any given point of time.

We propose a policy for runtime management of the power consumed by one or more

laser sources that drive the silicon-photonics NoC of manycore processor. We explore the application of our policy on a multi-bus NoC architecture that connects the private L1 caches of each core and the distributed L2 cache banks of the manycore processor. For laser power management, we adopt a weighted time-division multiplexing (TDM) mechanism, where depending on the spatial and temporal variations in the NoC bandwidth requirements of an application, we distribute the entire available NoC bandwidth by adjusting the bandwidth multiplexing weights associated with each bus to maximize application performance. In addition, we also switch ON/OFF one or more laser sources if there is a significant increase/decrease in the NoC bandwidth requirements of an application. Our policy uses the average packet latency to determine the minimum bandwidth required to keep the NoC out of saturation for an application, and then reconfigures the NoC architecture accordingly at runtime. The ultimate goal here is to sustain the application performance (instructions committed per cycle), while minimizing the power consumed in the laser.

The specific contributions towards development of laser power management techniques through weight TDM are as follows:

- We propose a token-based multi-bus NoC architecture implemented using silicon-photonics technology for a manycore processor. The weight TDM approach is used to multiplex the laser output power, i.e, network bandwidth, across the different buses in the network. This NoC architecture is specifically geared towards application-aware dynamic laser power management.
- We propose a policy for runtime laser power management. Based on the NoC bandwidth requirement of the running applications at each sampling interval, we adjust the bandwidth weights associated with each bus to redistribute the aggregate NoC bandwidth across all the buses at runtime. In addition, we switch ON/OFF one or more laser sources depending on the aggregate NoC bandwidth

requirements. The use of runtime laser power management technique further reduces average laser power by more than 49% with minimal impact on the system performance ($< 6\%$).

In this chapter, Section 3.2 describes our target system, which is followed by a discussion of its underlying silicon-photonics link technology in Section 3.3. A detailed description of the multi-bus NoC architecture and a brief overview of Clos and butterfly NoC architectures are presented in Section 3.4. Section 3.5 explains our policy for dynamic management of laser power, and the evaluation of our proposed policy using a full-system simulator is presented in Section 3.6. Section 3.7 gives an discussion about the application of our proposed policy to other NoC and manycore processor architectures. Section 3.8 summarizes our analysis.

3.2 Target System

We choose a 64-core processor that is manufactured using 22 nm CMOS technology (Kuhn et al., 2010) as our target system. Each core supports 2-way issue out-

Table 3.1: Micro-architectural parameters of the 64-core system

Micro-architecture Configuration	
Core Frequency	22 nm, 2.5 GHz @ 0.9 V
Branch Predictor	Tournament predictor
Issue	2-way Out-of-order
Reorder Buffer	128 entries
Functional Units	2 IntALUs, 1 IntMult, 1 FPALU, 1 FPMult
Physical Regs	128 Int, 128 FP
Instruction Queue	64 entries
Private L1 I/D-Cache	16 KB each @ 2 ns
Distributed L2 Cache	4-way, 64B/block, 8 x 2 MB @ 6 ns
Cache Coherence	Directory based MESI (Papamarcos and Patel, 1984)
NoC	silicon-photonics multi-bus
Memory	8 MCs + 8 PIDRAM @ 50 ns

of-order execution, and has two integer ALUs, one integer multiplication unit, one floating-point ALU, and one floating-point multiplication unit. The core architecture is configured based on the cores used in Intel’s 48-core SCC (Howard et al., 2010). The micro-architectural parameters are listed in Table 3.1. The cores operate at 2.5 GHz frequency and have a supply voltage of 0.9 V.

Each core has 16 KB L1 instruction cache and 16 KB L1 data cache. The system has a 16 MB distributed L2 cache (8 banks, 2 MB/bank) with each bank mapping to a unique set of memory addresses. The L2 caches are located at one edge of the chip. The manycore processor has 8 memory controllers, a memory controller per L2 cache bank, with each memory controller located next to the corresponding L2 cache bank. Cache lines are interleaved across eight banks to improve the parallel accessibility. We use MESI directory-based protocol (Papamarcos and Patel, 1984) for maintaining cache coherency between the L1 and L2 caches. The cache coherency directories are located next to the associated L2 cache banks. They maintain a copy of cache line status by monitoring the on-chip network transactions. For our analysis, we consider three different NoC architectures - multi-bus, Clos and butterfly, that provide connectivity between L1 and L2 caches. A detailed discussion for these NoC architectures is presented in Section 3.4.

There is a separate off-chip photonic network that connects memory controllers to PIDRAM chips (Beamer et al., 2010). We assume an average time of 50 ns for the communication from the memory controllers to PIDRAMs and back. We ignore the variations in queuing latencies at the input of the memory controllers because the high off-chip bandwidth using PIDRAM significantly reduces the number of outstanding memory requests in the queue.

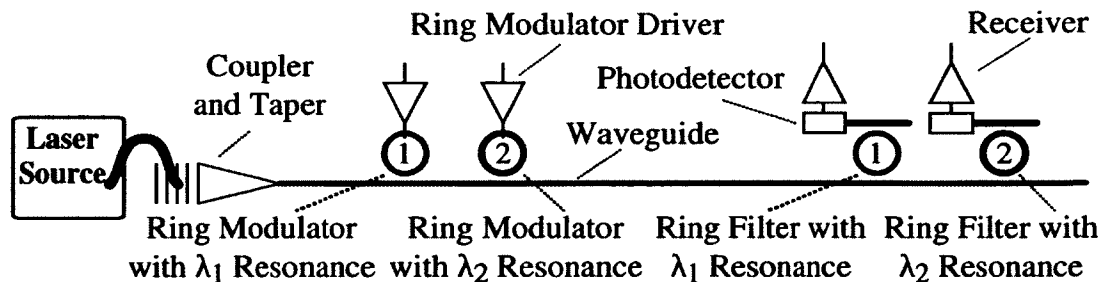


Figure 3-1: Photonic Link Components – Two point-to-point photonic links implemented with WDM.

3.3 Photonic Technology

Currently, there are several efforts in place in both academia and industry to integrate photonic devices with VLSI chips. In particular, in 2010, Intel demonstrated a 50 Gbps silicon-photonic link with integrated lasers using Hybrid Silicon Laser technology (Int, 2010). Meanwhile, IBM created a CMOS integrated nanophotonic technology that can integrate monolithically both the electrical and optical circuits on the same silicon chip on the front-end of the standard CMOS line (Green et al., 2010).

Our proposed technique for runtime management of laser power consumption in manycore processor is relatively agnostic of the exact underlying silicon-photonic technology. It is applicable to both, monolithic integration of photonic devices using bulk CMOS process (Orcutt et al., 2008; Holzwarth et al., 2008; Orcutt et al., 2011) or SOI process (Gunn, 2006; Q. Xu, B. Schmidt, S. Pradhan and M. Lipson, 2005; Assefa et al., 2010; Gnan et al., 2008; Chen et al., 2009; Sridaran and Bhave, 2010; T. Baehr-Jones, M. Hochberg, C. Walker and A. Scherer, 2004; Osgood et al., 2005), and 3D integration of photonic devices by depositing SiN (Barwicz et al., 2007; Hosseini et al., 2009; Gorin et al., 2008; Biberman et al., 2011) or polycrystalline silicon (Preston et al., 2007; Preston et al., 2009; Preston and Lipson, 2009) on top of the metal

stack, design approaches. Figure 3-1 shows a generic wavelength-division multiplexed (WDM) photonic link used for intra-chip communication. Light waves of wavelength λ_1 and λ_2 emitted by an off-chip laser source are coupled into the chip using vertical couplers. The vertical coupler guides the light waves into the waveguides. These light waves pass next to a series of ring modulators controlled by modulator drivers based on the data to be transmitted on the link. The modulators convert data from electrical medium to photonic medium. The modulated light waves travel along the waveguide and can pass through zero or more ring filters. At the receiver side, the ring filter “drops” the light wave having the filter’s resonant wavelength onto a photodetector. The resulting photodetector current is sensed by an electrical receiver. At this stage data is converted back into the electrical medium from the photonic medium.

For our analysis, we use the silicon-photonic link design described in (Joshi et al., 2009). We consider double-ring filters and a 4 THz free-spectral range, which enables up to 128λ modulated at 10 Gb/s on each waveguide (64λ in each direction, interleaved to alleviate filter roll-off requirements and crosstalk). A non-linearity limit of 30 mW at 1 dB loss is assumed for the waveguides. The waveguides are single mode and have a pitch of 4 μm to minimize the crosstalk between neighboring waveguides. We assume modulator ring and filter ring diameters of $\approx 10 \mu\text{m}$. The latency of a

Table 3.2: Energy and Power Projections for Photonic Devices

– Tx = Modulator driver circuits, Rx = Receiver circuits, TT = Thermal tuning circuits, fJ/bt = average energy per bit-time, DDE = Data-traffic dependent energy, FE = Fixed energy (clock, leakage) (Joshi et al., 2009).

Design	Tx (fJ/bt)		Rx (fJ/bt)		TT (fJ/bt/heater)
	DDE	FE	DDE	FE	
Aggressive	20	5	20	5	16
Conservative	80	10	40	20	32

global photonic link is assumed to be 3 cycles (1 cycle in flight and 1 cycle each for E/O and O/E conversion). We assume a 5 μm separation between the photonic and electrical devices to maintain signal integrity. We use the projected silicon-photonic link energy cost (Table 3.2) and projected silicon-photonic device losses (Table 3.3) for our analysis (Joshi et al., 2009).

The laser ON/OFF characteristics are an important consideration when we reconfigure the laser source number to save the laser power. When the laser supply current is turned on, both the carrier density and photon density in the laser active medium reach a steady-state condition rapidly, on the order of ns (Coldren et al., 2012). The laser dissipates power in the form of heat when turned ON, therefore some temperature stabilization time is required. However, considering all of these effects, semiconductor diode lasers have demonstrated stable pulses with pulse widths (time to switch ON and OFF the laser current) of 35 ns (Klamkin et al., 2010). This switch ON and OFF time is suitable for our approach as we reconfigure the L2 cache every 10 million instructions, and if even more precision were required to ensure power and lasing wavelength stability, a simple wavelength locking circuit could be employed (Sarlet et al., 1999). Here, we explore the application of our power management technique considering laser source switch ON/OFF times in the range of 0 – 100 ms.

Table 3.3: Optical Loss per Component (Joshi et al., 2009)

Photonic device	Optical Loss (dB)
Optical Fiber (per cm)	0.5e-5
Coupler	1
Splitter	0.2
Non-linearity (at 30 mW)	1
Modulator Insertion	0
Waveguide (per cm)	1~5
Waveguide crossing	0.05
Filter through	1e-4
Filter drop	1.5
Photodetector	0.1

3.4 Multi-bus NoC Architecture

In this section, we will provide a detailed description of our silicon-photonic multi-bus NoC architecture and an overview of silicon-photonic Clos and butterfly NoC architectures that are used as comparison points, for a manycore processor.

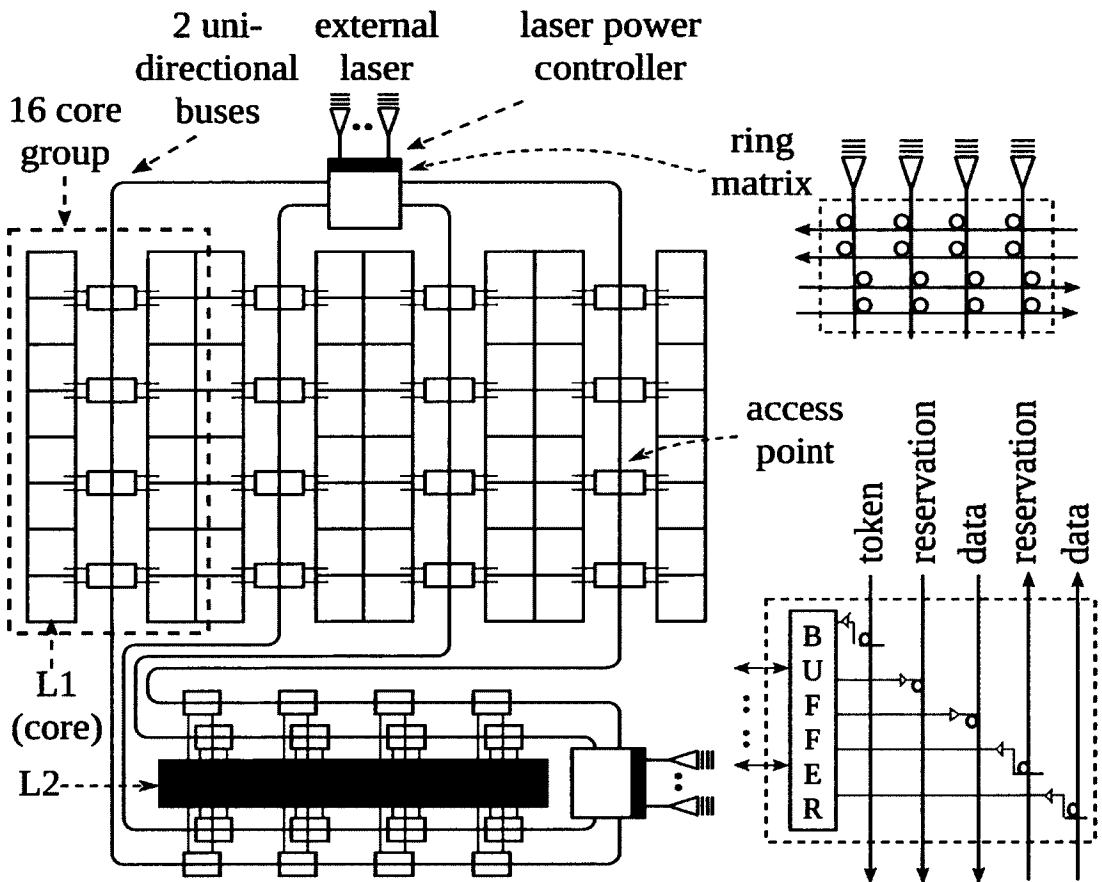


Figure 3-2: Physical layout of the silicon-photonic multi-bus NoC architecture – The 64 cores are divided into 4 groups and each group communicates with L2 banks through two uni-directional buses. Any L1-to-L1 communication is through the cache coherence directory located at the L2 banks. The two laser power controllers dynamically guide light waves into the appropriate buses using time-division multiplexing through associated the ring matrices.

Figure 3-2 shows the physical layout of the silicon-photonic multi-bus NoC architecture for our target 64-core system. We divide the 64 cores into 4 logical groups

with 16 cores in two adjacent columns allocated to each group. Each group has 2 uni-directional silicon-photonics buses, one each for L1-to-L2 and L2-to-L1 communication. Any L1-to-L1 communication is through the cache coherence directory located at the L2 banks. The light waves emitted by the off-chip laser sources are time-division multiplexed across all the buses using ring matrices controlled by two laser power controllers. Each uni-directional bus consists of three channels – token channel, reservation channel and data channel. On each bus, there are 4 L1 access points – one each for a set of 4 cores, and 4 L2 access points – one each for a set of 2 L2 banks. The concentration does not require extra local electrical links considering the physical locations of the L1 and L2 caches. Moreover, the concentration helps to maximize the utilization of the multi-bus access points. A simple round-robin arbitration is used within each access point. Fixed priority arbitration is used across access points based on their physical locations on the bus. The access point that is closest to the laser source has the highest priority to grab the token and use the data channel.

In each bus, we divide the NoC packet into 4-bit sets and each set is mapped onto a wavelength to match the processor frequency of 2.5 GHz and photonic link bandwidth of 10 Gb/s. If the data channel uses ω wavelengths, we need a total of $\omega + 2$ wavelengths in each uni-directional bus (1 for token channel, 1 for reservation channel and ω for data channel). In Figure 3-2, each ring shown in the ring matrix represents $\omega + 2$ rings (1 for token, 1 for reservation and ω for data). Thus, each ring matrix has $(\omega + 2) \times 16$ rings when using 4 laser sources and 4 bus channels. Each access point has $\omega + 2$ rings for transmitter (1 filter ring for token, 1 modulator ring for reservation and ω modulator ring for data) and $\omega + 1$ filter rings for receiver (1 for reservation and ω for data). Considering the non-linearity limit of 30 mW per waveguide and photonic device losses listed in Table 3.3, we need ω waveguides for each bus channel

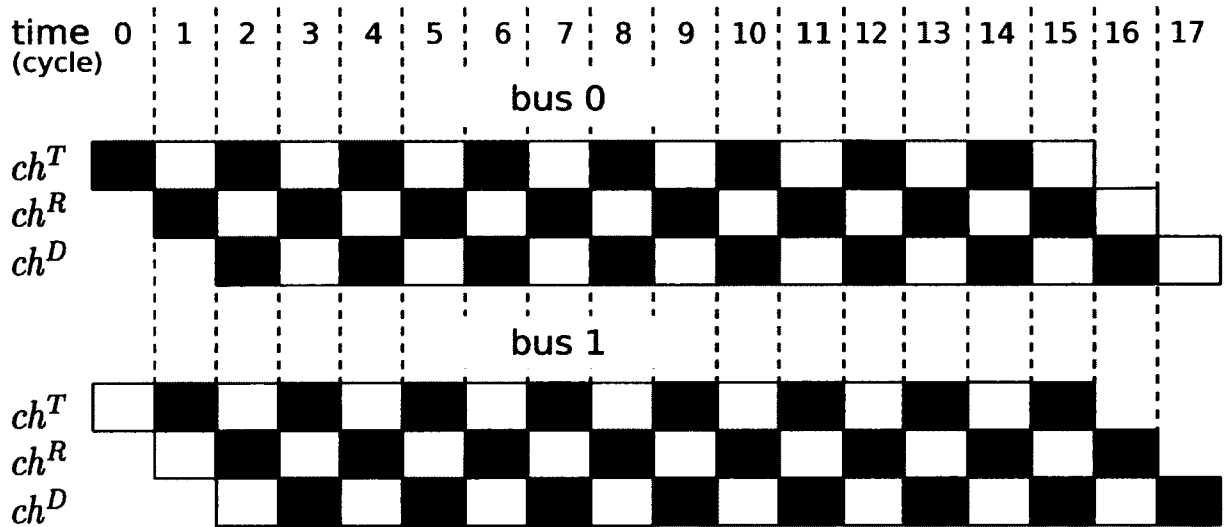


Figure 3-3: Timing diagram for token stream arbitration – Here ch^T = token channel, ch^R = reservation channel, ch^D = data channel, Tx = Token for time slot ‘x’, Rx = Reservation request for time slot ‘x’ and Dx = Data in time slot ‘x’. For L1-to-L2 communication, the L1 access point grabs a token and then reserves the destination L2 access point before modulating the data through data channel. The grabbing of token, transmission of reservation request and transmission of data packet is performed in consecutive cycles. In the example, the two buses have 50% utilization each.

with the conservative waveguide design (3 dB/cm loss) or $\omega/6$ waveguides for each bus channel with the aggressive waveguide design (1 dB/cm loss). Our analysis in Section 3.6 uses 32 wavelengths per channel ($\omega = 32$) that can meet the worst-case bandwidth demands of NAS benchmarks. In that case, the tuning power overhead of the two ring matrices is 0.5 W and the area occupied by all the silicon photonic devices is less than 1.54% of the total chip area of 400 mm².

During a L1 cache miss or while sending other cache coherency messages, the L1 access point uses a token stream protocol to arbitrate for the data channel access for L1-to-L2 communication. In this protocol, a token per TDM slot is issued by the laser source. Figure 3-3 shows the timing of token distribution, reservation request and data transmission of two L1-to-L2 buses. In this example, tokens, and effectively

the TDM slots, are multiplexed onto the two buses using time-division multiplexing. To access the data channel, a L1 access point needs to grab a token from the token channel two cycles prior to the actual slot of data transmission. We use single-round token channels in which the L1 access point near the laser source has the highest priority to obtain the photonic tokens. Fairness can be pursued by using alternate token-based arbitration protocols (Pan et al., 2010; Vantrease et al., 2009). After grabbing the token, the L1 access point sends a reservation request by setting one out of the 4 bits that can be mapped on the reservation channel wavelength. Here each bit corresponds to the destination L2 access point. Each L2 access point can only filter its associated bit on the reservation channel wavelength. It uses that bit to tune its ring filters that will filter the data received on the data channel in the following cycle. Only the L1 access point that has a token can use the associated reservation slot on the reservation channel. This notification over the reservation channel ensures that the destination L2 access point is ready to receive data from the L1 access point in the following cycle (2 cycles after token is grabbed). In this following cycle, the L1 access point transmits the data to the L2 access point over the data channel. Each L2 bank has a dedicated access point for each bus. The laser power controller can decide the rate at which each bus receives photonic tokens. In Figure 3-3, 16 photonic tokens are issued within a 16 cycle period, with a 50% of data channel utilization of each bus. We define the number of photonic tokens received by one bus in the 16 cycle period as bandwidth weight, which will be used in runtime laser power management (see in Section 3.5). In this example, the bandwidth weight for each bus is $1/2$. The same token-based weighted time-division multiplexed protocol is used for communication on the L2-to-L1 buses.

We compare our proposed multi-bus NoC architecture with silicon-photonic Clos and butterfly NoC architectures. Both these NoC architectures are well-suited to be

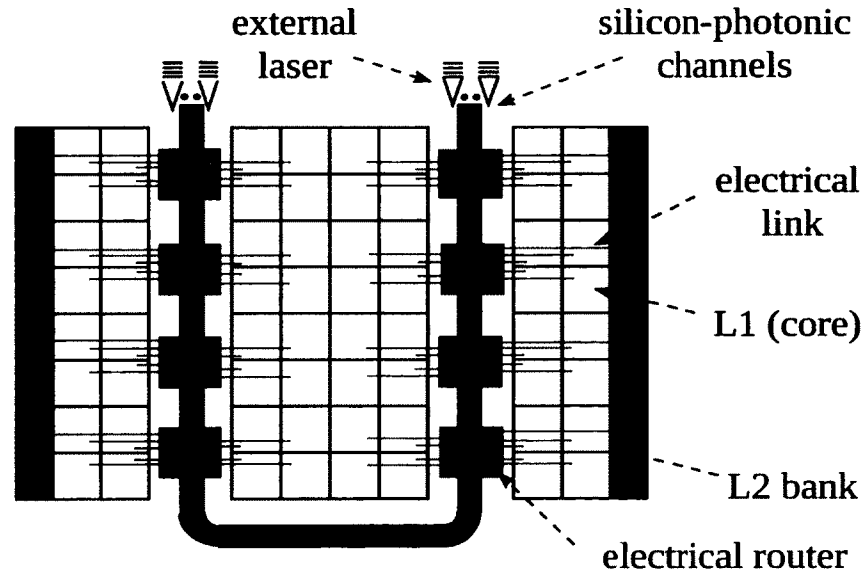


Figure 3-4: Physical layout of the silicon-photonic Clos and butterfly NoC architecture

designed using silicon-photonic link technology. They both use smaller routers, but long global channels. They have less hop counts than mesh and at the same time do not need global arbitration like the crossbar. In addition, Clos provides extensive path diversity that can be used to minimize congestion in the NoC. We did not choose the mesh and crossbar NoC architecture for comparison as it has been shown previously that mesh and crossbar topologies are not suitable for silicon-photonic link technology (Joshi et al., 2009). For a fair comparison, we designed the physical layouts and chose the NoC parameters of the three topologies such that the total laser power consumption of the three topologies is the same.

Figure 3-4 shows the physical layout for the Clos and butterfly NoC architectures. A concentration of 8 cores (L1 caches) and 1 L2 bank is used at each router, and electrical links provide local communication between L1/L2 caches and routers. Each ‘electrical router’ represents at least one injecting router and one ejecting router. In case of butterfly, the ‘silicon-photonic channels’ represents 64 dedicated channels fully connecting the 8 injecting routers to the 8 ejecting routers. In case of Clos, κ of 8

‘electrical router’ represents extra κ middle routers. The ‘silicon-photon channels’ represents 64 dedicated channels fully connecting the 8 injecting routers to the κ middle routers, and another 64 dedicated channels fully connecting the κ middle routers to the 8 ejecting routers. A detailed comparison of the three topologies is presented in Section 3.6.

3.5 Laser Power Management

In this section, we describe our proposed runtime network reconfiguration methodology – joint application of weighted TDM and switching ON/OFF laser sources, to reduce laser power while maintaining application performance.

3.5.1 Runtime Network Reconfiguration

For making network reconfiguration decisions on the L1-to-L2 network, each core group periodically sends its average packet latency calculated over a fixed time interval to the laser power controllers that are responsible for multiplexing the network bandwidth across the buses. This information about the average packet latency is used by the controller to decide on the new bandwidth weights (see more details in Section 3.5.2) for each bus. The range and granularity of bandwidth weights depend on number of buses in the multi-bus NoC architecture and the desired level of control. For our target 64-core system, we have 4 buses and bandwidth weights range from $1/16$ to $16/16$, with a step size of $1/16$. The precision of bandwidth weights could be increased for finer control. Depending on the bandwidth weights that are periodically calculated, the bandwidth is proportionally multiplexed across the buses at run time. However, a simple round-robin arbitration does not work for laser power allocation if the four L1-to-L2 buses in our target system have different bandwidth

weights. We use a proportional share resource algorithm similar to Earliest Eligible Virtual Deadline First (EEVDF) (Stoica et al., 1996) to assign the laser sources to buses according to their newly calculated bandwidth weights.

Using the new bandwidth weights that are required for each bus to sustain the performance of the various applications, the laser power controller determines the total L1-to-L2 bandwidth required across all buses. This total required bandwidth is in turn used to determine the total number of laser sources required in the system. In our case study, each laser source supplies the bandwidth equivalent to the baseline bandwidth of an individual bus channel. It uses 16 multiplexing slots corresponding to 16 consecutive clock cycles. At each time slot (or cycle), the light waves of the laser source are guided to the targeted L1-to-L2 bus. So if the bandwidth weight of a bus is 16/16, we need to dedicate an entire laser source to that bus. The total number of laser sources can be calculated using

$$\text{Laser Source Number} = \left\lceil \sum_{i=0}^{N-1} W_i \right\rceil \quad (3.1)$$

where, W_i is the bandwidth weight for the i^{th} bus and N is the number of L1-to-L2 buses. For example, if the total number of laser sources is computed to be 3.5, and if only three laser sources are currently in use, then the laser power controller decides to switch ON one more laser source. It should be noted that in this example half the bandwidth of the fourth laser source remains unutilized and gets wasted. We could uniformly distribute this unutilized bandwidth across all the buses to further improve the application performance. However, we use only the calculated bandwidth to ensure that the calculated bandwidth weights track the application bandwidth requirements. For example, if the laser power controller decides to reduce the band-

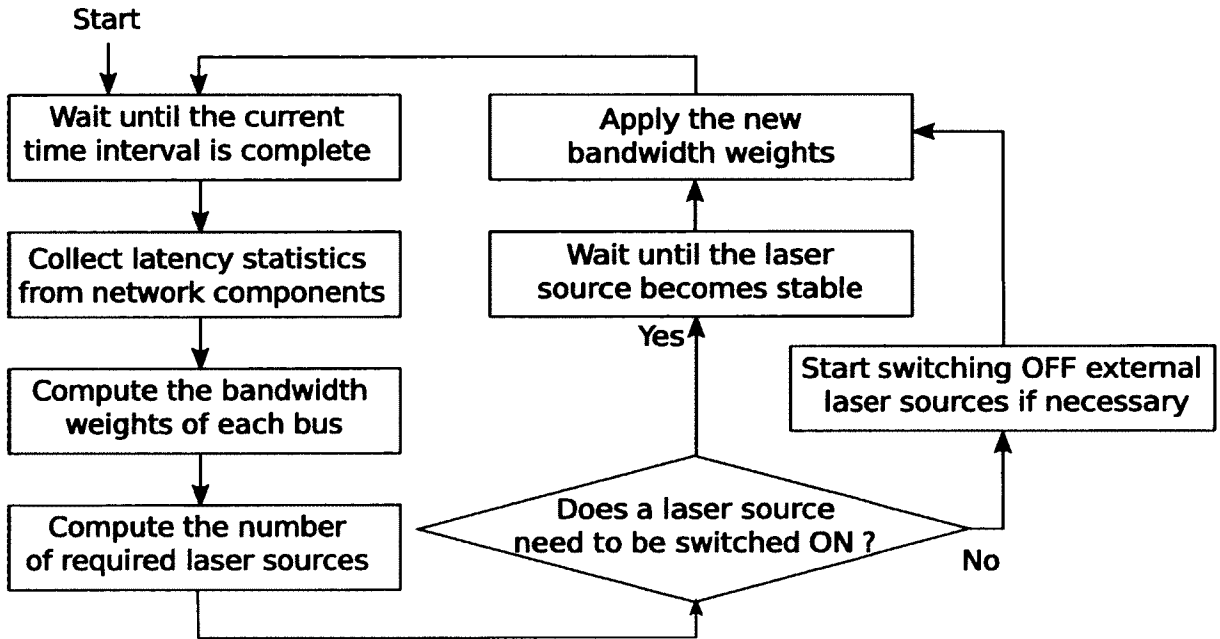


Figure 3-5: Flow chart for runtime laser power management using weighted TDM – The laser power are switch ON/OFF based bandwidth weight.

width weight of a bus from $\alpha/16$ to $(\alpha - 1)/16$, and the unutilized bandwidth is allocated to that bus resulting in an actual bandwidth weight of $\alpha/16$ for the new time interval, then the bandwidth weight will never go down to $(\alpha - 1)/16$. As a result, the available network bandwidth may not necessarily track the required network bandwidth and can potentially result in unnecessary waste of laser power.

The ring matrices at the laser power controllers contain ring filters for each bus. These filters are tuned and detuned depending on the bandwidth weights associated with each bus. If the bandwidth weight for each bus needs to be updated but no new laser sources need to be switched ON, the entire network reconfiguration process takes less than 20 core clock cycles. On the other hand, if a new laser source needs to be switched ON, then we need to wait for a longer time for the laser source to switch ON and stabilize thermally. While the new laser source is stabilizing, the system does not stop execution and the laser power controllers use the older configuration to

distribute the laser power across the bus channels. Figure 3-5 shows the flow chart for the various steps involved in the network reconfiguration for laser power management. It should be noted that both bandwidth weight/laser number calculation block and ring filter tuning block can be implemented purely in hardware.

3.5.2 Bandwidth Weight Calculation

For runtime management, the bandwidth weight of each bus is determined and updated after every time interval. The bandwidth weight for each bus is calculated using the average packet latency for that bus over the previous time interval. We use a dual-threshold (L_{low} and L_{high}) approach for choosing the bandwidth weights. Here, if the average packet latency on a bus is greater than the upper threshold (L_{high}), then the bandwidth weight of that bus is increased by $1/16$ to effectively increase the bus bandwidth and in turn reduce the average packet latency. The key idea here is to move the bus out of its saturation region by increasing the bus bandwidth and minimize the impact of the packet latency on the performance of the application running on the associated group of cores.

Similarly, if the average packet latency is smaller than the lower threshold (L_{low}), then the associated bandwidth weight is reduced by $1/16$ to decrease the bus bandwidth. This reduction in bandwidth saves laser power, with potentially minimal impact on

Table 3.4: L_{low} for all bandwidth weight at various L_{high} - X means it is not possible to further reduce bandwidth weight, otherwise the serialization latency would be more than L_{high} .

Weight	$\frac{2}{16}$	$\frac{3}{16}$	$\frac{4}{16}$	$\frac{5}{16}$	$\frac{6}{16}$	$\frac{7}{16}$	$\frac{8}{16}$	$\frac{9}{16}$	$\frac{10}{16}$	$\frac{11}{16}$	$\frac{12}{16}$	$\frac{13}{16}$	$\frac{14}{16}$	$\frac{15}{16}$	$\frac{16}{16}$
$L_{high} = 10$	X	X	X	X	X	X	X	X	X	X	X	9.4	9.4	9.3	9.5
$L_{high} = 15$	X	X	X	X	X	13.7	13.3	13.2	13.0	13.1	13.4	13.0	13.2	13.0	12.8
$L_{high} = 20$	X	X	X	X	16.5	16.7	16.5	16.4	16.3	16.4	15.8	16.1	16.0	15.6	15.8
$L_{high} = 30$	X	X	22.4	22.1	21.9	20.9	22.1	20.1	21.5	20.8	20.3	20.4	19.2	18.0	19.6
$L_{high} = 50$	X	29.3	29.4	29.1	28.7	27.8	27.4	26.4	29.1	27.3	25.9	25.4	27.8	28.4	24.7

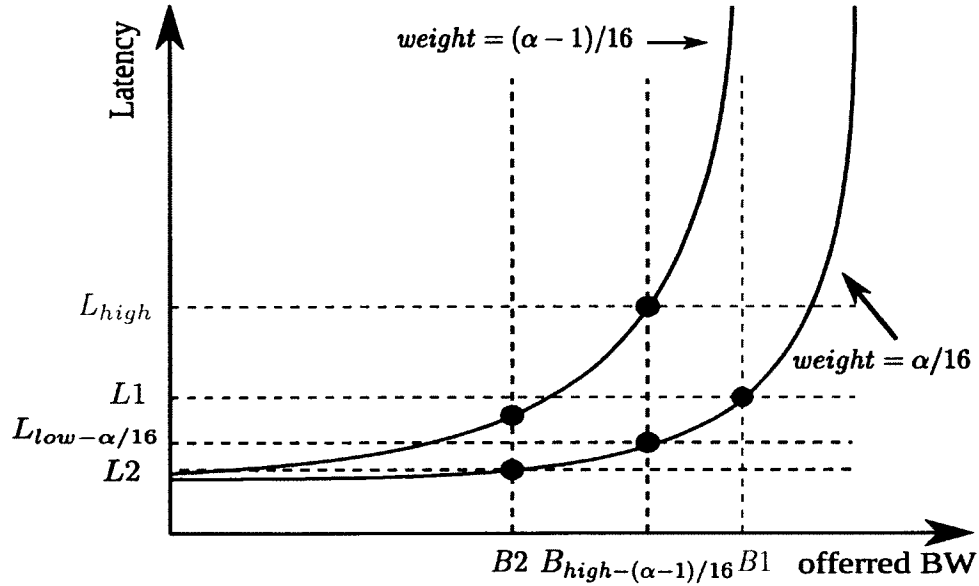


Figure 3-6: Methodology to determine threshold L_{low} for a bandwidth weight

the overall application performance. The value for L_{low} needs to be carefully chosen to ensure that after reducing the bandwidth weight, the bus does not become saturated in the next time interval. Figure 3-6 shows our strategy for determining L_{low} based on the latency-bandwidth plots of our multi-bus NoC architecture for a uniform random traffic pattern. From the latency-bandwidth plot for a bus with bandwidth weight of $(\alpha - 1)/16$, we can determine the bandwidth ($B_{high-(\alpha-1)/16}$) and the corresponding latency (L_{high}) beyond which the bus goes into saturation. By mapping this $B_{high-(\alpha-1)/16}$ onto the latency-bandwidth plot for bus with bandwidth weight of $\alpha/16$, we can determine the ideal lower threshold ($L_{low-\alpha/16}$) for bandwidth weight of $\alpha/16$. On a bus with bandwidth weight of $\alpha/16$ if the latency is less than $L_{low-\alpha/16}$, it would be safe to reduce bandwidth weight to $(\alpha - 1)/16$ since we can guarantee the bus would not saturate if the traffic pattern does not change. This approach can be used to determine the lower threshold for each bandwidth weight. In Figure 3-6, if the bus with bandwidth weight of $\alpha/16$ has an average packet latency of $L1$ ($> L_{low-\alpha/16}$), then the laser power controller should not reduce the bandwidth

weight to $(\alpha - 1)/16$, as the bus would get into the saturation region. If the bus with bandwidth weight of $\alpha/16$ has an average packet latency of $L_2 (< L_{low-\alpha/16})$, then the laser power controller can safely reduce the bandwidth weight to $(\alpha - 1)/16$ since the resulting increased latency would not exceed L_{high} .

We use the latency-bandwidth plot for random uniform traffic pattern to calculate the lower threshold L_{low} for runtime management. Table 3.4 shows the calculated L_{low} for each bandwidth weight while setting L_{high} as 10, 15, 20, 30 and 50 cycles. For our analysis in Section 3.6, the laser power controller can choose L_{low} by mapping the current bandwidth weight and predefined L_{high} on the Table 3.4 to find the corresponding L_{low} threshold. It should be noted that the latency threshold value is a function of the NoC architecture, and hence for each other NoC architecture, the latency thresholds need to separately determined using synthetic traffic patterns.

3.6 Evaluation

In this section, we first provide an overview of our evaluation platform followed by a detailed discussion of the reduction in laser power of a 64-core system through communication-driven runtime laser power management. We compare our multi-bus NoC architecture with Clos and butterfly NoC architectures, and then investigate the impact of different reconfiguration thresholds and reconfiguration time intervals on the overall system performance and laser power consumption when using the laser power management policy.

3.6.1 Simulation Methodology

Our 64-core target system has a directory-based cache coherency protocol. Since the Gem5 simulator uses broadcast-based cache coherency protocol, we modified the

simulator to trap all broadcast-based cache coherency operations and emulate the corresponding directory-based cache coherency operations. For example, if we trap a L1-to-L1 data response on a L2 cache miss in the broadcast-based cache coherency protocol, we translate it into a sequence of four consecutive network operations in directory-based cache coherency protocol. It starts with a request packet from the core with the L1 cache miss to the directory of the associated L2 bank, followed by a request packet from the directory to the core that has the missing cache line. After receiving the data response packet from the core having the missing cache line, the directory forwards the data to the original requester. This packet sequence is used to model the operations among the core with the L1 cache miss, the directory at the L2 bank and the L1 cache that has the missing cache line. Similarly, network operations corresponding to other directory-based operations such as upgrade request and data response from L2 are also trapped and emulated. Our emulation methodology approaches the real timing overhead of cache miss and the overall network traffic loads in a cache hierarchy with directory-based cache coherency protocol. In addition, our trap and emulation method is independent of the proposed multi-bus NoC architecture and laser power management technique as the variations in the network bandwidth requirements across benchmarks are maintained.

We integrated our multi-bus network model into Gem5 full-system simulator (Binkert et al., 2006), and we run NAS parallel benchmarks (*cg*, *ep*, *ft*, *is*, *lu*, *mg*, *sp* and *ua*) with class B problem sets (Bailey et al., 1994). We use a warm-up period of 2 billion instructions to get past the initialization phase and avoid cold-start effects. We execute each application for 100 ms with network reconfiguration at fixed time intervals (10 μ s, 100 μ s, 1 ms and 10 ms). We use application instruction committed per cycle (IPC) as the metric to prove that our runtime laser management has minimal impact on the system performance.

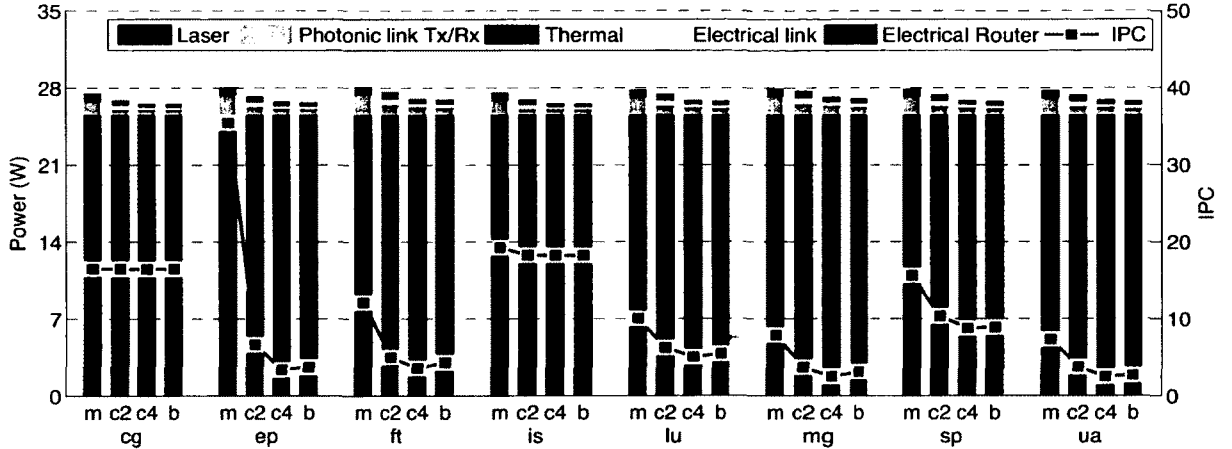


Figure 3-7: Power and IPC for various NoCs with the same laser power budget – Here ‘m’ = multi-bus, ‘c2’ = Clos with 2 middle routers, ‘c4’ = Clos with 4 middle routers and ‘b’ = butterfly. These four NoC architectures have 32λ , 8λ , 4λ and 4λ per channel, respectively to meet the same laser power budget. We assume the conservative design from Table 3.2 and Table 3.3.

3.6.2 Evaluation Results with NAS Benchmarks

Figure 3-7 shows a comparison of the power consumption and performance of various NoC architectures with the same laser power budget. We first determine the laser power consumption for a multi-bus having 32λ per channel and use that value as the laser power budget for other NoC architectures. The choice of 32λ is discussed later in this section. The channel width for other NoC architectures was determined using the same laser power budget of the entire network. For Clos we consider two architectures – one with 2 middle routers (8λ per channel) and one with 4 middle routers (4λ per channel). The butterfly has a channel width of 4λ . Both Clos and butterfly use a concentration of 9 (8 L1 caches and 1 L2 cache bank), which requires local electrical links for the communications between L1/L2 caches and network access points. For Figure 3-7, we assume the conservative transmitter/receiver circuits and thermal tuning circuits in Table 3.2 and the conservative waveguide design (3 dB/cm loss) in Table 3.3. The laser power consumption is more than 24 W with

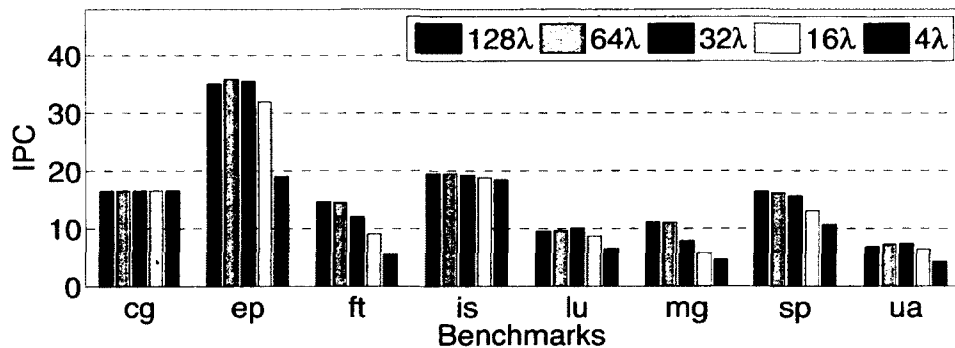


Figure 3-8: IPC for various baseline bandwidth – The ideal baseline bandwidth is between 32λ and 64λ . We choose 32λ conservatively to avoid the over-provisioning of the multi-bus NoC architecture.

these conservative assumptions. For the aggressive design (1 dB/cm loss), the laser power consumption is close to 5 W. In both cases, the the laser power consumption is dominant in the whole network and therefore the runtime laser power management is necessary. With the same laser power consumptions, the multi-bus topology achieves better performance than Clos and butterfly because it has the lowest serialization latencies due to the wider bus width. At the same time it has better bandwidth utilization than other NoC architectures as it shares bandwidth among multiple network access points, while both Clos and butterfly use dedicated channels between routers. Among the two Clos architectures, the architecture with 2 intermediate routers exhibits better performance than the architecture with 4 intermediate routers due to lower serialization latency. The butterfly and Clos with 4 intermediate routers exhibit similar performance due to same channel widths. The multi-bus achieves more than $2\times$ higher IPC on average than Clos and butterfly for NAS benchmarks.

To evaluate our laser power management policy using the multi-bus NoC architecture, we first determine its baseline bandwidth. Figure 3-8 shows the impact of choice of baseline bandwidth on the overall system performance in terms of IPC. The baseline bandwidth corresponds to the case where all the laser sources required for the

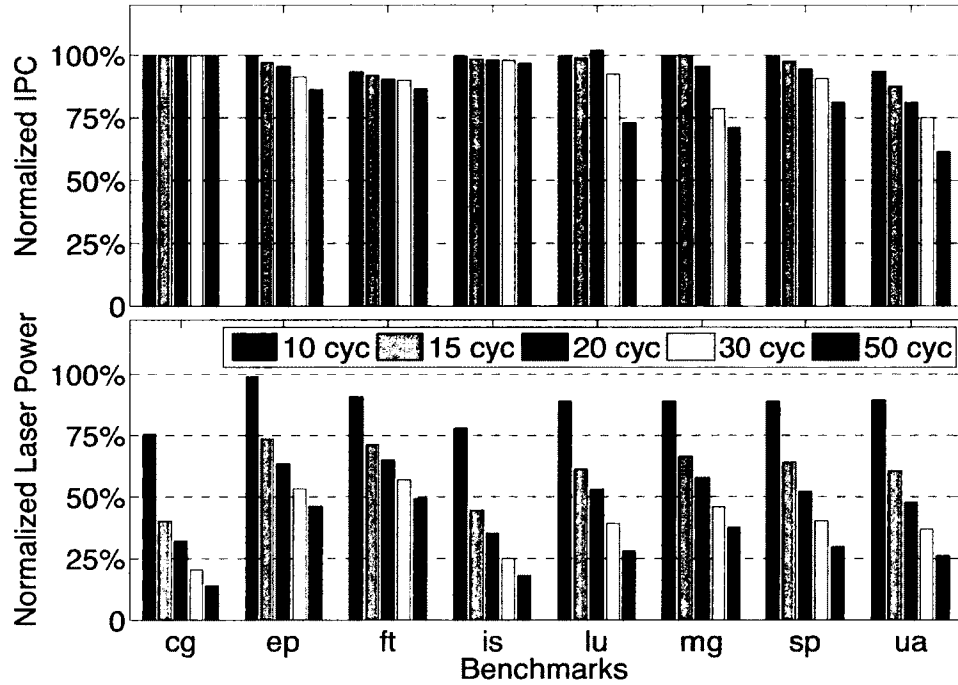


Figure 3-9: IPC and bandwidth weight for various reconfiguration threshold L_{high}

network are always ON i.e. it corresponds to the bandwidth weights of 16/16. The baseline bandwidth should be chosen such that it provides the minimum bandwidth that would meet the worst-case bandwidth demands of all applications, i.e. not limit the overall system performance. Figure 3-8 shows that **ft** and **mg** benchmarks require the highest NoC baseline bandwidth to sustain the system performance. Their performance decreases when the baseline bandwidth reduces below 64λ per channel, which indicate that the ideal baseline bandwidth could be between 64λ and 32λ per channel. We use a conservative baseline bandwidth of 32λ per channel for the evaluation of our runtime management technique.

Figure 3-9 shows the impact of reconfiguration threshold L_{high} and corresponding L_{low} on the system performance and laser power consumptions in the multi-bus NoC architecture. Here, the baseline bandwidth is 32λ per channel and the reconfiguration

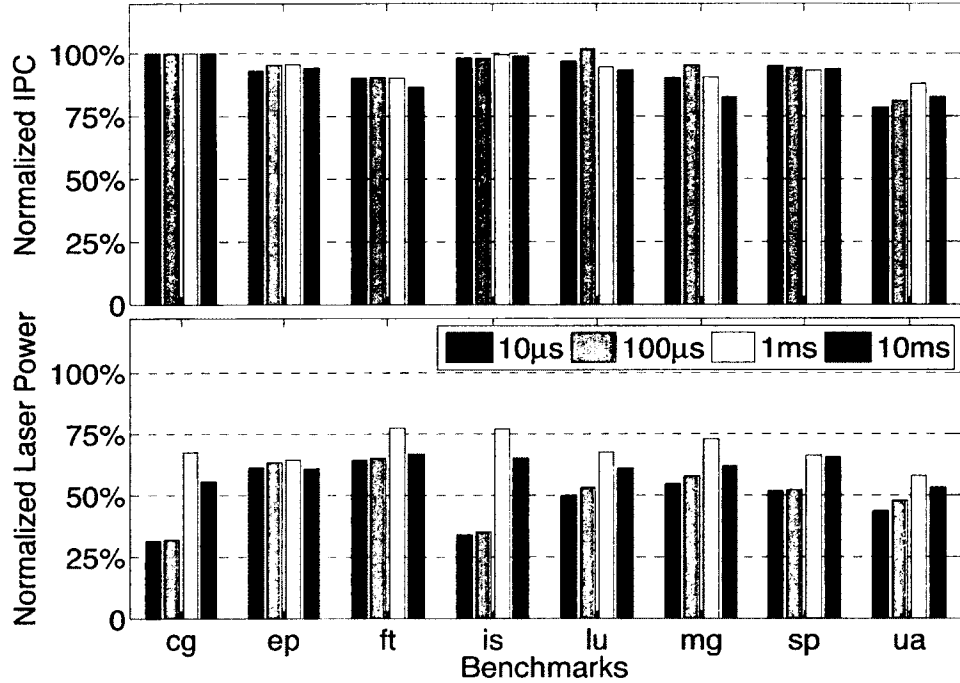
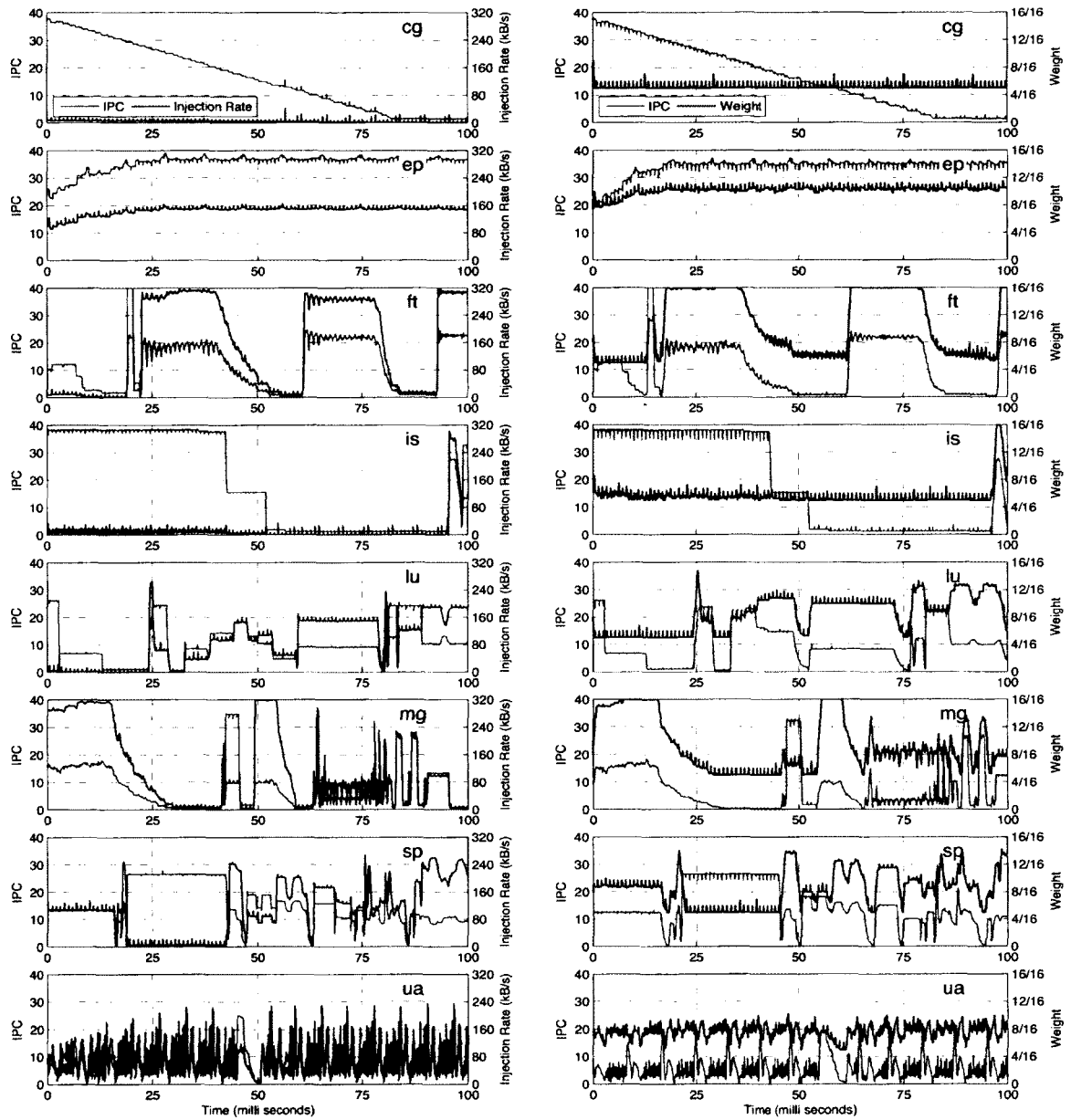


Figure 3-10: IPC and bandwidth weight for various reconfiguration time interval

time interval is $100 \mu\text{s}$. The performance is normalized to the performance of each benchmark running on the system without runtime management (as shown in Figure 3-7). The laser power is normalized to the baseline design that has laser sources switched ON all the time. The L_{high} and the corresponding L_{low} from Table 3.4 have an significant impact on the system performance and laser power consumption (proportional to the average bandwidth weight). We see that when the L_{high} increases, the performance (IPC) decreases and laser power consumption (proportional to the average bandwidth weight) decreases. This behavior is observed because a higher L_{high} makes it difficult for the controller to make the decision of increasing the bandwidth weight. A higher L_{low} (due to higher L_{high}) is preferable from the perspective of saving laser power consumption. However, a L_{high} higher than 20 cycles starts having a strong impact of the system performance, especially on `mg` benchmark. Thus, we choose to use $L_{high} = 20$ cycles for the NAS benchmarks.

Figure 3-10 shows the impact of the reconfiguration time interval on the system performance and laser power consumption. The baseline bandwidth is 32λ per channel with 16/16 bandwidth weight and reconfiguration threshold L_{high} is 20 cycles. The performance is normalized to the performance of each benchmark running on the system without runtime management (as shown in Figure 3-7). The laser power is normalized to the baseline design that has laser sources switched ON all the time. For the projected laser source stabilization values ranging from $1 \mu\text{s}$ to 10 ms, we consider a reconfiguration time intervals that are $10\times$ the stabilization times i.e. from $10 \mu\text{s}$ to 100 ms to minimize the effect of laser source stabilization. Figure 3-10 shows that with a shorter reconfiguration time interval ($10 \mu\text{s} \sim 100 \mu\text{s}$), the average bandwidth weight (i.e. laser power) is much lower than for longer reconfiguration time interval ($1 \text{ms} \sim 10 \text{ms}$). This is because a shorter reconfiguration time interval allows the laser power controller to quickly adjust the bandwidth weight based on the changes in the network traffic. As the reconfiguration time intervals increase, there is a slower response to changes in network traffic, which may result in the waste of laser power. This slower response to changes in network traffic can also impact system performance. For the 10 ms reconfiguration time interval, the average bandwidth weight does not match with the general trend of our simulation results as the initial bandwidth weight (during the start of the simulation) is chosen as 8/16, and our total simulation time of 100 ms does not provide the enough time to for warm up.

Figure 3-11 shows the performance, network traffic and bandwidth weight variations across time, with and without runtime management. The left column corresponds to the performance and network traffic variations in the multi-bus with static bandwidth allocation (16/16) i.e. baseline bandwidth. The temporal variations of network traffic for each individual benchmarks and the spatial variations across benchmarks provide opportunities for reducing the laser power consumption through runtime manage-



(a) Multi-bus without runtime reconfiguration

(b) Multi-bus with runtime reconfiguration

Figure 3-11: IPC, network traffic and bandwidth weight tracing for selected benchmarks – The left column shows the performance and network traffic with static bandwidth allocation (bandwidth weight = 16/16). The right column shows the performance and bandwidth weight while applying our runtime laser power management technique.

ment. The right column shows the performance and bandwidth weight variations after applying our runtime management technique. Here the baseline bandwidth is 32λ per channel, the reconfiguration threshold L_{high} is 20 cycles and the sampling interval is $100 \mu\text{s}$. The initial bandwidth weight for each bus is $8/16$. The runtime management technique allows the laser power controllers to dynamically allocate the bandwidth according to the temporal and spatial variations in the bandwidth demands. We see no significant change in the performance trace after applying the runtime management technique. The bandwidth weight tracks the variation of network traffic, and reduces the laser power consumption. The rate of bandwidth weight variation strongly depends on the variation of network traffic. For example, benchmarks like `mg`, `sp` and `ua` benchmarks show frequent bandwidth weight changes as the network traffic changes frequently. On the other hand, benchmarks like `ep`, `ft` and `lu` show much smooth changes in bandwidth weight since the network traffic remains constant for extended periods across the time. Benchmarks like `cg` and `is` do not show significant variations in bandwidth weight since the network traffic remains constant for most time.

The above analysis shows that the baseline bandwidth of 32λ per channel can meet the worst-case bandwidth demands of NAS parallel benchmarks. The application of our laser power management technique provides more than 49% savings in laser power on an average without significant impact on system performance (less than 6% on average). Our approach saves more than 12.4 W given the conservative waveguide design (3 dB/cm loss), and more than 2.4 W given the aggressive waveguide design (1 dB/cm loss).

3.7 Discussion

In this section we qualitatively discuss the limits and opportunities for application of our proposed multi-bus NoC architecture and the laser power management technique.

3.7.1 Large Core Counts

For our analysis we used a target system having 64 cores with 16 cores in adjacent columns sharing two uni-directional buses. Future manycore processors will have thousands of cores on a single die. Assuming corresponding progress in the area of parallel algorithms and programming, these large manycore processor systems could produce several times larger network traffic and require higher network bandwidth. Our multi-bus NoC architecture exhibits good scalability to larger core counts. We could either increase the number of buses, or increase the concentration at each bus access points and balance the energy consumed in the electrical and photonic links. For example, for a 1024-core processor we could use 32 uni-directional bus channels with 64 cores in adjacent core columns sharing two uni-directional bus channels. The baseline bandwidth of each bus channel will need to be increased to match the increased network traffic. If we increase the network concentration, a local electrical network is needed to provide communication between cores and bus access points. The use of local electrical network could help increase the utilization of each bus access points, and in turn improve the energy efficiency of the entire network.

3.7.2 Simultaneously Executing Applications

Most of the legacy applications do not scale well to large core counts. Hence, the OS will need to have the capability to execute multiple applications simultaneously

to support legacy applications. For example, if we have 4 applications and each application uses 16 threads with a thread per core, we could map each application onto a 16-core group. Our management policy can easily be applied to such situations. If these applications exhibit any variations in the network bandwidth demands, our policy can easily increase/decrease the network bandwidth of each individual bus based on the bandwidth demands of each application. This would help maximize the energy efficiency of the on-chip network as well as the manycore system as a whole.

3.7.3 Distributed L2 Cache vs. Private L2 Cache

The use distributed L2 cache vs private L2 cache for manycore architectures has been widely explored. Both approaches have their advantages and disadvantages. The distributed L2 cache enables good cache line sharing across the large number of cores, but at the same time it has higher L1 miss penalty and requires large amount of intra-chip communication for maintaining coherency. On the other hand, the private L2 cache architecture has low L2 hit latency, but the cache line sharing and synchronization is difficult. Moreover, the private L2 cache could have higher miss rates resulting in expensive off-chip memory accesses. Our multi-bus NoC architecture and laser management technique can be readily applied to both cache architectures. The multi-bus NoC architecture will provide L1 to L2 communication for distributed L2 cache architecture and L2 to memory controller communication for private L2 cache architecture. The key step is choosing the correct baseline design (bus bandwidth) for the multi-bus NoC such that the system can support the worst case network traffic. We could pursue an integrated solution where we jointly design the L2 cache architecture and silicon-photonics multi-bus NoC architecture to maximize the energy efficiency.

3.7.4 Alternate NoC Architectures

Our proposed laser power management policy is relatively agnostic of the underlying NoC architecture. It can be applied to NoC architectures like butterfly, Clos and crossbar. In these NoC architectures, we would expect to see laser power savings at a similar scale as the multi-bus NoC architecture. It should, however, be noted that multi-bus NoC architecture provides better performance than these NoC architectures at same laser power consumption, and hence would have better energy-efficiency. The laser power management policy could also be applied to silicon-photonics implementation of low-radix high-diameter NoC architectures like torus and mesh. The large channel count and distributed nature of these NoC architectures would however lead to significant overhead.

3.8 Summary

Silicon-photonics link technology is expected to replace electrical link technology in intra-chip and inter-chip communication networks in future manycore processors. However, the large laser power consumption in these silicon-photonics networks is limiting their widespread adoption. We proposed a multi-bus NoC architecture for a manycore processor and a runtime technique that dynamically manages the laser power of this multi-bus NoC depending on the communication bandwidth requirements of the various applications running on the manycore processor. For a silicon-photonics multi-bus NoC between the private L1 and distributed L2 caches, we propose a policy that uses weighted time-division multiplexing with token-stream flow control and switching ON/OFF laser sources as necessary, to maximize the total energy efficiency of the manycore processor. For a 64-core processor running the NAS parallel benchmark suite, we get an average of more than 49% reduction in laser power, with

a 6% reduction in application performance. The proposed technique can potentially pave the way for early adoption of silicon-photonics link technology in future manycore processors.

Chapter 4

Runtime Cache Reconfiguration for Managing Laser Power in Silicon-photonic NoC

4.1 Introduction

In addition to using weighted TDM, we also explored cache reconfiguration techniques for laser power management. The applications running on the manycore systems typically exhibit spatial variations and/or temporal variations in the requirements of NoC bandwidth, L1/L2 cache size and/or core count. This creates an opportunity for proactively reconfiguring the overall system at runtime to minimize laser power consumption. We propose to manage the laser power by proactively reconfiguring the size of shared L2 cache and, in turn, the NoC bandwidth between private L1 and shared L2 banks. We chose to use the photonic NoC between the L1 and L2 caches (private L1 and shared distributed L2 configuration) instead of between L2 and memory controllers because the smaller L1 caches have much higher miss rate than the larger L2 caches. This larger L1 miss rate creates the need for low-latency access to all the distributed L2 banks. This low-latency access can be provided by photonic link technology at lower energy. The L2 misses are addressed by the PIDRAM through the memory controller (sitting next to an L2 bank) with chip-to-chip photonic links between the memory controller and the PIDRAM.

The system periodically samples L2 replacement rate to make a decision on increasing/decreasing the number of active L2 banks while ensuring a minimal impact on the system performance. A reduction in the L2 bank count creates an opportunity to switch OFF the silicon-photonic links associated with the deactivated L2 banks and, in turn, save laser power. The key idea here is to provide the minimum L2 cache size and NoC bandwidth required for an application to sustain the maximum possible performance at any given point of time.

The specific contributions under laser power management through cache reconfiguration are as follows:

- We propose a runtime mechanism for managing the laser power consumed by a crossbar NoC between private L1 caches and distributed shared L2 banks. We use L2 replacement rate to determine the optimal L2 bank count and the number of silicon-photonic links at runtime. This runtime reconfiguration helps reduce the laser power and optimize the system energy efficiency.
- We present a methodology to deactivate L2 banks that involves flushing memory blocks from the deactivated L2 banks and the L1 caches. Similarly, we present the methodology to activate L2 banks that involves remapping the memory blocks across all active L2 banks. On average, our proposal saves laser power by 23.8% and system power by 6.39%, with IPC degradation of 0.65%, and EDP improvement by 5.52%.

In this chapter, Section 4.2 presents the detailed architecture of our target system. Section 4.3 provides a detailed discussion of our proposed runtime reconfiguration system followed by a detailed evaluation in Section 4.4. Section 4.5 summarizes our analysis.

4.2 Target System, Simulation tools

We choose a 64-core processor that is manufactured using 22 nm CMOS technology as our target system. The micro-architectural parameters are listed in Table 4.1. The core architecture is configured based on the cores used in Intel’s 48-core SCC (Howard et al., 2010). It has an L2 cache size of 4 MB. This size was determined by running all our target benchmarks and determining the minimum L2 cache size that sustains the maximum performance of all applications. Alternate set of applications could indicate the need for larger L2 cache sizes. However, our proposed technique of laser power management using L2 cache reconfiguration would still be applicable.

Figure 4-1 shows the logical topology of the silicon-photonic high-radix crossbar NoC architecture. To enable switching OFF/ON of silicon-photonic channels when L2 cache is reconfigured, each L2 bank uses dedicated silicon-photonic channels for communication to/from the L1 caches. The NoC employs the Multiple-Write-Single-Read (MWSR) mechanism for L1-to-L2 communication. A token based protocol is used to arbitrate between the L1 caches for getting access to L1-to-L2 communication

Table 4.1: Micro-architectural parameters of the 64-core system

Micro-architecture Configuration	
Core Frequency	22 nm, 1.25 GHz @ 0.9 V
Branch Predictor	Tournament predictor
Issue	2-way Out-of-order
Reorder Buffer	128 entries
Functional Units	2 IntALUs, 1 IntMult, 1 FPALU, 1 FPMult
Physical Regs	128 Int, 128 FP
Instruction Queue	64 entries
Private L1 I/D-Cache	16 KB each @ 2 ns
Distributed L2 Cache	4-way, 64B/block, 8 x 512 MB @ 6 ns
Cache Coherence	Directory based MESI (Papamarcos and Patel, 1984)
NoC	silicon-photonic crossbar
Memory	8 MCs + 8 PIDRAM @ 50 ns

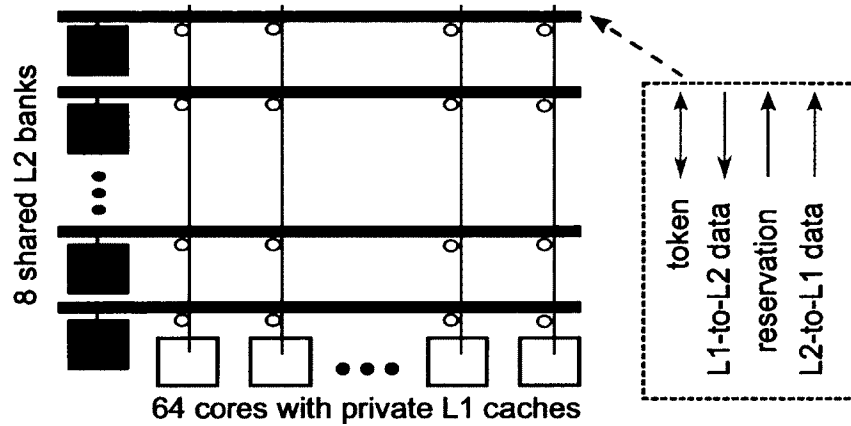


Figure 4-1: Logical topology of a silicon-photonic crossbar NoC – It connects 64 cores and 8 L2 banks.

channels (Vantrease et al., 2008). Tokens are assigned in a round-robin fashion for fairness. The network employs the Single-Write-Multiple-Read (SWMR) mechanism for L2-to-L1 communication. The reservation channel is used to enable the ring filters at targeted destination L1 caches before an L2 bank transmits packets onto the L2-to-L1 data channels (Pan et al., 2010). This approach of using MWSR mechanism for L1-to-L2 communication and SWMR for L2-to-L1 communication with all the communication associated with an L2 bank mapped onto a single dedicated photonic channel provides the opportunity to easily switch OFF/ON the silicon-photonic links associated with an L2 bank when it is deactivated/activated.

Figure 4-2 shows the physical layout of the silicon-photonic network between 64 private L1 caches and 8 L2 banks. Cache lines are interleaved across L2 banks to enable the parallel accessibility. We use a MESI directory-based protocol for maintaining cache coherency between L1 and L2 banks. The cache coherency directories are located next to the associated L2 banks. They maintain a copy of the cache line status by monitoring the on-chip network transactions associated with the corresponding L2 bank. Any core-to-core communications are transferred through the directory. The L2 banks and memory controllers are collocated on the edge of the processor;

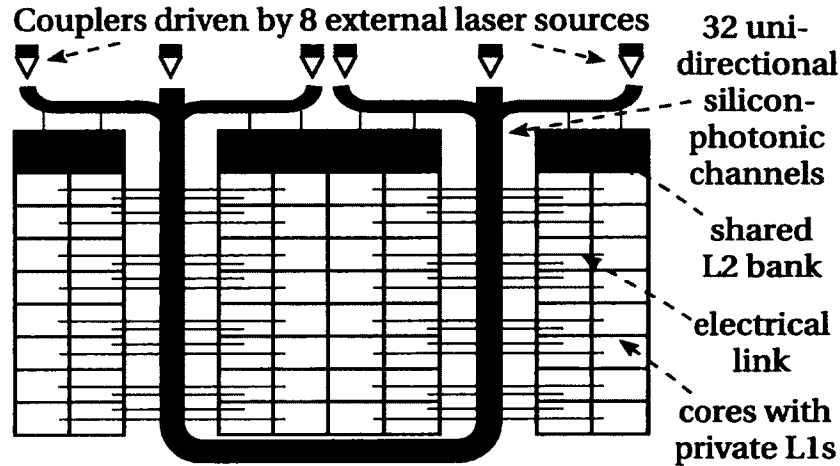


Figure 4-2: Physical layout of the silicon-photonic crossbar NoC – It connects private L1 caches of 64 cores and 8 L2 banks. The L2 banks are located at the ends of the silicon-photonic links for ease of arbitration.

therefore L2 banks can access the memory controllers (MCs) through local wiring upon L2 misses. There is a separate off-chip photonic network that connects MCs to PIDRAM chips (Beamer et al., 2010). We assume an average time of 50 ns for the communication from the MCs to PIDRAMs and back. We ignore the variations in queuing latencies at the inputs of MCs because the high off-chip bandwidth using PIDRAM significantly reduces the number of outstanding memory requests in the queue.

4.3 Runtime Reconfiguration

In this section, we describe our runtime L2 reconfiguration technique that can reduce laser power while sustaining performance. The key idea is to track the dynamic changes in the size of an application’s working set during different phases of application execution and reconfigure the L2 bank count at runtime. Essentially, if the entire working set can fit into a smaller L2 cache, we can save laser power by de-

activating some L2 banks and their associated silicon-photonics links. On the other hand, if the working set requires a larger L2 capacity, we need to activate more L2 banks and their corresponding silicon-photonics links to avoid excessive L2 misses and sustain application performance. It should be noted that we are not proposing a new L2 reconfiguration technique. Several techniques have already been explored for L2 reconfiguration, however the use of L2 reconfiguration to manage laser power in silicon-photonics NoC has never been explored before.

4.3.1 Reconfiguration Controller

We propose to use a reconfiguration controller (RC) located close to the L2 banks for tracking the variations in the working set size and making the decisions on the required changes in the L2 bank count. Figure 4-3 shows the flow chart of the reconfiguration process performed by the RC. After each sampling period, the RC collects cache statistics from all cores, and decides whether to increase/decrease the number of L2 banks and the number of active silicon-photonics links. In our system the RC controller increases/decreases the number of L2 banks in power of 2. Alternate granularities could be used, but that would unnecessarily complicate the reconfiguration process. The RC uses a single dedicated wavelength bus (10 Gbps) for communicating reconfiguration decision to L2 banks and laser sources, which corresponds to minimal power and area overhead. The total communication overhead for this reconfiguration process is ~ 100 cycles (for transmitting reconfiguration decision to all L1 caches, L2 cache banks, and off-chip laser sources) which is marginal compared to the large period (10 million instructions committed) over which we collect data to make reconfiguration decisions.

When the RC decides to activate one or more L2 banks, it needs to switch ON the laser sources that power the silicon-photonics links associated with those L2 banks.

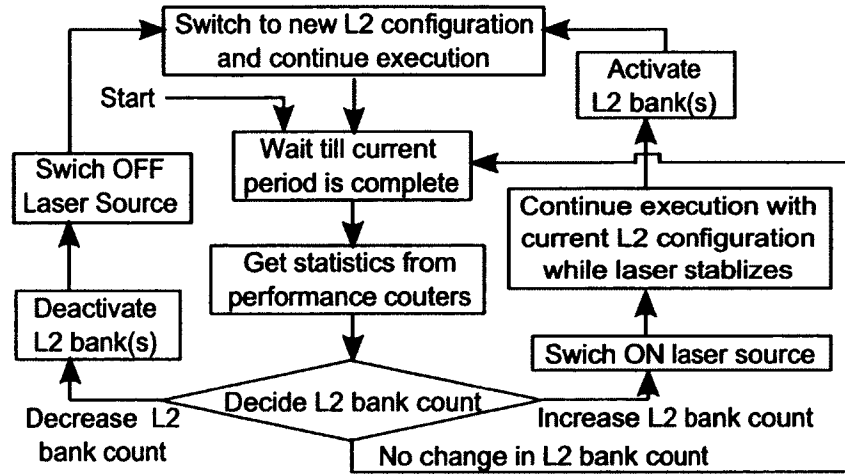


Figure 4.3: Flowchart for runtime reconfiguration – The steps for activating/deactivating L2 banks and the associated silicon-photon links at runtime.

Each laser source needs to stabilize before powering the silicon-photon links and as discussed in Section 3.3 we expect that the entire switch ON time of external laser sources (including the stabilization delay) is in a range of 10 ns – 100 ms. We discuss the impact of the stabilization delay on our proposed reconfiguration technique in Section 4.4. It should be noted that the application does not stop execution during the L2 reconfiguration step. It continues to execute using the older L2 cache configuration. Once the reconfiguration process is complete, it switches to the newer cache configuration. When the RC decides to deactivate some L2 banks, it needs to keep the external laser sources ON until those L2 banks flush their contents to the main memory and are completely deactivated. After deactivating those L2 banks, the application can continue the normal execution without waiting for the external laser sources to be completely switched OFF.

4.3.2 Reconfiguration Decision Process

To make reconfiguration decisions, ideally, the system should test all choices for L2 bank count and find the optimal L2 bank count that reduces power while keeping IPC degradation within an acceptable range. However, this method of choosing L2 bank count is not feasible at runtime. Moreover, IPC cannot be used for making the reconfiguration decision because the absolute value of IPC does not indicate the need for L2 reconfigurations. We propose to use L2 cache replacement rate ($\# \text{cache replacements} / \# \text{clock cycles}$ in the sampling period) as a metric to determine the need for increasing/decreasing L2 bank count. The absolute value of L2 replacement rate indicates if the current L2 cache size is sufficient to store the application's entire working set and if an increase/decrease in L2 bank count can improve/hurt the system performance.

The RC uses a dual-threshold approach to make reconfiguration decisions. It compares the L2 cache replacement rate with two thresholds: T_{high} and T_{low} . When the replacement rate is higher than T_{high} , it decides to increase the L2 bank count, and when the replacement rate is lower than T_{low} , it decides to decrease the L2 bank count. When the replacement rate is in between T_{low} and T_{high} , the RC makes a decision to maintain the current L2 bank count.

We evaluated the impact of various T_{high} and T_{low} values on system IPC and fluctuations in the L2 bank count, respectively (see Figure 4.4). When L2 replacement rate is larger than T_{high} , the system increases L2 bank count. A larger value for T_{high} provides larger savings in the laser power as the large T_{high} value reduces the probability of increasing L2 bank count and hence the probability of switching ON the associated silicon-photonics links. However, a downside to a large T_{high} value is that it can cause performance degradation as the system may not have sufficient number

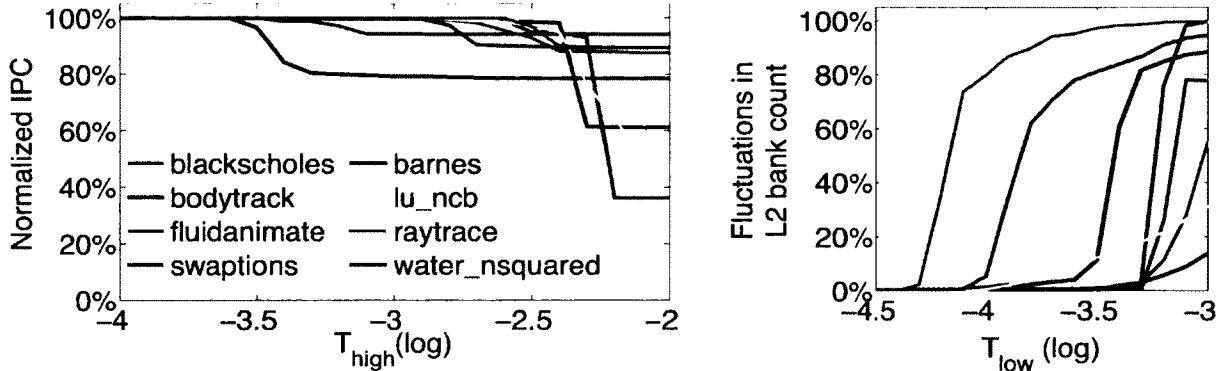


Figure 4.4: The choice of thresholds T_{high} and T_{low} – (a) IPC degradation when using different T_{high} values for each benchmark. (b) The number of fluctuations in L2 bank count for using various T_{low} for each benchmark when $T_{high} = 10^{-3}$.

of L2 banks to maintain the entire working set of the application on the processor chip. Our target benchmarks start showing degradation in performance for T_{high} values larger than $10^{-3.5}$ (see Figure 4.4(a),). We choose $T_{high}=10^{-3}$ where the average performance degradation is $<10\%$ across all benchmarks.

Figure 4.4(b) shows how we determine the T_{low} threshold. The system decreases L2 bank count, when L2 replacement rate becomes lower than T_{low} . A large value for T_{low} can maximize laser power savings as it would increase the probability of deactivating L2 banks and switching OFF the associated silicon-photonic links. However, a large T_{low} value could lead to an incorrect cache reconfiguration decision, which will need to be reversed after the very next sampling period. This can happen especially when values of T_{low} and T_{high} are very close to each other. Assume that the system increases the L2 bank count after observing a L2 replacement rate higher than T_{high} . There is a high probability that the system will need to decrease the L2 bank count in the next sampling period as the L2 replacement rate would drop below T_{low} . This decrease in the bank count would increase the value of the L2 replacement rate above T_{high} , which would again indicate the need for increasing the bank count. Essentially, we

might end up in a situation where we need to change the bank count after every sampling period. We need to choose the value of T_{low} such that it can avoid the need for reconfiguring the cache after every sampling period.

Moreover, our simulation based analysis revealed that each application has a different optimal T_{low} value (see in Figure A.4.4(b)). In this Figure, fluctuation is defined as the percentage of reconfiguration decisions that are reversed after the very next sampling period. We can choose a general T_{low} below $10^{-4.5}$ for all the benchmarks to minimize the fluctuations. However, such a low T_{low} would reduce the probability of the system reducing the L2 bank count to save laser power. We propose to use a learning process in the RC to find the best T_{low} for each benchmark. Hence, we choose $T_{low} = T_{high}$ when the application starts execution. Whenever RC observes a fluctuation, it decreases the value of T_{low} . Over time, the RC converges to the optimal T_{low} for an application. The optimal T_{low} values for each application could potentially be stored and can be reused when the application is executed again.

4.3.3 L2 Bank Activation/Deactivation Process

The manycore system needs to maintain cache coherence while activating/deactivating L2 banks. The reconfiguration process only affects the memory blocks that are associated with the L2 banks that are being activated/deactivated. Memory operations to other memory blocks can be performed during the reconfiguration, thereby minimizing any performance degradation in the system. We explain the reconfiguration process using a cache architecture in which the L1 and L2 caches are inclusive, and use a write-back writing policy and a directory-based coherence protocol.

Figure 4.5(a) shows an example in which there are 4 L1 caches ($L1\$-a$ to $L1\$-d$) and 2 active L2 banks ($L2\$-a$ to $L2\$-b$), and $L2\$-b$ needs to be deactivated. Since the

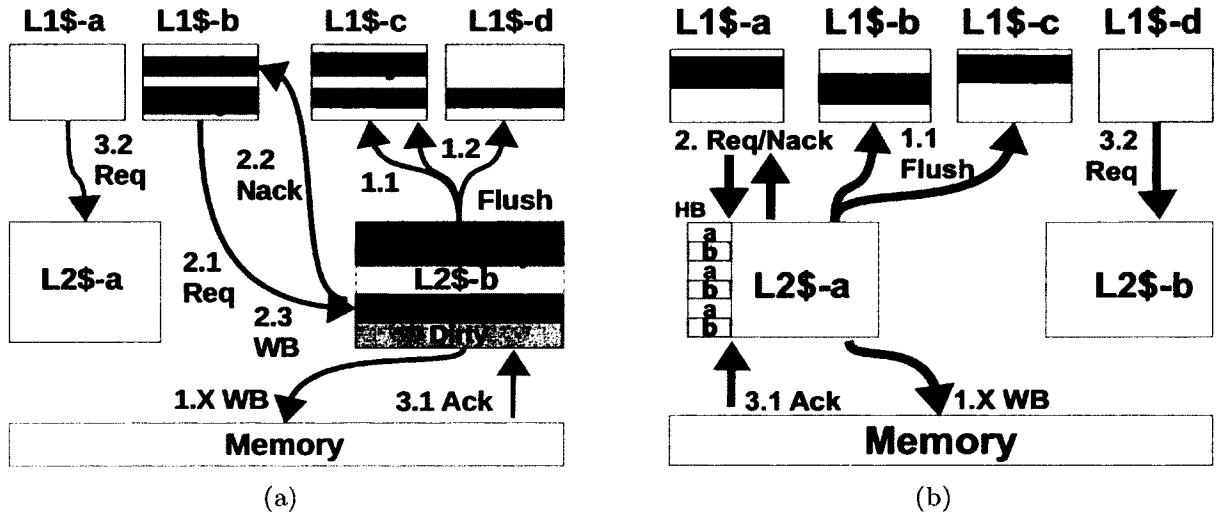


Figure 4-5: L2 cache bank reconfiguration process – (a) deactivate $L2\$-b$ (b) activate $L2\$-b$.

caches are inclusive and use write back policy, all entries of $L2\$-b$ must be checked to flush any corresponding dirty blocks in L1 caches. We can use the coherence-related invalidation messages from L2 banks to implement the flush operations. For every L2 cache block that is flushed, the L2 bank will either transmit a write back message to memory (1.X in Figure 4.5(a)) in case of dirty blocks (1.1 in Figure 4.5(a)), or it will simply remove the associated L2 bank entry in case of a clean block (1.2 in Figure 4.5(a)). Notice that, new requests (2.1 in Figure 4.5(a)) from L1 caches can be transmitted to the $L2\$-b$ during the deactivation process. In these cases, negative acknowledgments (2.2 Nack in Figure 4.5(a)) are sent to the requesting L1 caches. However, any requests for writing back of the dirty L1 cache blocks are accepted and forwarded to memory (2.3 in Figure 4.5(a)). Once all the entries in $L2\$-b$ along with all the corresponding L1 cache entries have been flushed and written to memory and the last acknowledgment is received from memory (see 3.1 Ack in Figure 4.5(a)), the L2 bank can be safely deactivated with its corresponding silicon-photonic link. Any requests to this deactivated L2 bank are redirected to one of the active L2 cache

banks (3.2 Req that is sent to $L2\$-a$ in Figure 4.5(a)). The RC system communicates information about the new home L2 bank for each memory address to all L1 caches so that memory requests to blocks belonging to the deactivated L2 bank can be redirected to the correct active L2 cache banks. This step is also required for the activation process described below.

Figure 4.5(b) also shows the example in which there are four L1 caches ($L1\$-a \sim L1\$-d$) and one active L2 cache bank ($L2\$-a$), and $L2\$-b$ bank is needs to be activated. We need to identify the L2 banks that have accepted the L2 cache requests (prior to reconfiguration) that would be served by $L2\$-b$ after reconfiguration. We need to flush the relevant cache blocks from these L2 banks back to memory. In addition, any dirty L1 cache blocks that would be served by $L2\$-b$ after reconfiguration also need to be flushed due to the inclusive property and write-back policy. The process of flushing L1 cache would be similar to that explained in the L2 bank deactivation process (1.1 Flush and 2. Req / Nack in Figure 4.5(a)). Here, the L2 banks that are currently active and whose mapping would change after reconfiguration would need to determine the entries that need to be flushed to memory. Typically, specific bits from a block memory address are used to determine the L2 bank. Each L2 bank uses an HB field to indicate which cache line remains in $L2\$-a$ and which cache line needs be flushed and remapped to $L2\$-b$ after reconfiguration.

4.4 Evaluation

In this section we quantify the power savings obtained by using our proposed cache reconfiguration technique.

4.4.1 Simulation Methodology

We use the Gem5 full-system simulator (Binkert et al., 2006) to simulate our proposed L2 cache reconfiguration technique on the 64-core target system described in Section 4.2. We enable the Ruby memory system for an accurate modeling of the shared multi-bank L2 cache hierarchy. We run PARSEC (Bienia et al., 2008) and SPLASH-2 (Woo et al., 1995) benchmarks in their parallel regions with the large input set, and check for the need for L2 cache reconfiguration after every 10 million instructions (committed by all 64 cores). Each benchmark executes a total of 4 billion instructions resulting in a 400 reconfiguration sampling period. After every sampling period we check the replacement rate using performance counters to analyze the need for L2 cache reconfiguration.

We use McPAT (Li et al., 2009) and Cacti 5.3 (Thoziyoor et al., 2008) to calculate the core power and cache power, respectively. The McPAT output is calibrated using the Intel SCC (Howard et al., 2010) published power values and is then scaled to 22 nm technology. We use the photonic technology described in Section 3.3 to calculate laser power, Tx/Rx power and thermal tuning power in the NoC. We calculate EDP (total system power * (application execution cycles/system frequency)²) of the entire system to evaluate the overall impact of our proposed technique on the balance of system performance and power.

4.4.2 Reconfiguration Opportunities

Figure 4.6(a) and Figure 4.6(b) show the IPC trace and corresponding L2 replacement rate trace, respectively, for a 64-core system with different number of L2 banks when running `blackscholes`, `bodytrack`, `fluidanimate`, `swaptions`, `barnes`, `lu_ncb`, `raytrace` and `water_nsquared` benchmarks. The replacement rate can be used as a metric for

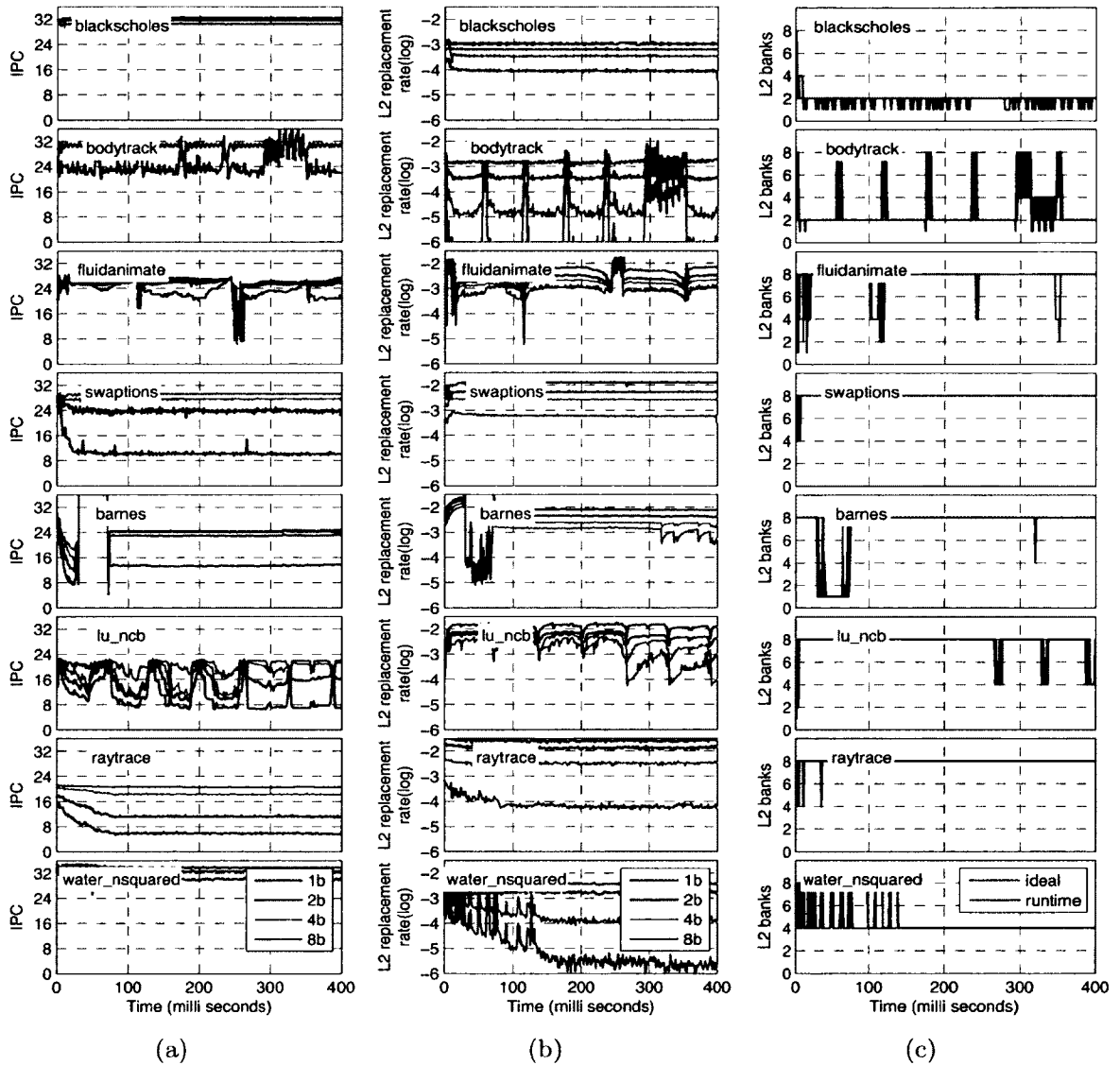


Figure 4-6: IPC, replacement rate and l2 bank count tracing for selected benchmarks - (a) The IPC of each benchmark with a fixed of L2 bank count: 1, 2, 4, 8 banks. (b) The measured replacement rate with a fixed L2 bank count. (c) The optimal L2 bank count determined offline and the L2 bank count chosen by the RC at runtime.

a fairly accurate prediction of whether we need to increase or decrease the number of banks. A high L2 replacement rate generally indicates the need for a larger cache capacity, while a low L2 replacement rate indicates that cache capacity is larger than what is required. Figure 4.6(c) compares the trace of the ideal L2 bank count determined using offline analysis and the trace of the L2 bank count chosen by the RC at runtime. The offline analysis chooses the minimal L2 bank count that keeps the L2 replacement rate lower than but as close as possible to T_{high} after each sampling period. RC determines the L2 bank count based on the L2 replacement rate during previous sampling period and compares it with two thresholds T_{high} and T_{low} to determine the L2 bank count for the next period. Figure 4.6(c) shows that the optimal bank count varies both across applications and within applications over time. Our proposed reconfiguration policy ensures that the L2 bank count tracks the changes in L2 replacement rate, and therefore harnesses any opportunity of saving laser by reducing the L2 bank count. The optimal L2 bank count for `blackscholes` is lower than the optimal L2 bank count for `barnes`. This indicates more savings in laser power when running `blackscholes`. In case of the `barnes` application, the optimal L2 bank count varies as the application goes through different execution phases, thus providing various levels of laser power savings.

4.4.3 Reconfiguration Benefits

Figure 4-7 shows the impact of reconfiguration on system performance and power consumption (using conservative silicon-photonics link design). Here, we compare the performance and power consumption of a 64-core system that uses a fixed number of ‘1’, ‘2’, ‘4’ and ‘8’ banks with a 64-core system that uses our proposed L2 cache reconfiguration policy. We did not consider cases with ‘3’, ‘5’, ‘6’ and ‘7’ active L2 banks as the L2 reconfiguration process becomes very complicated for these bank

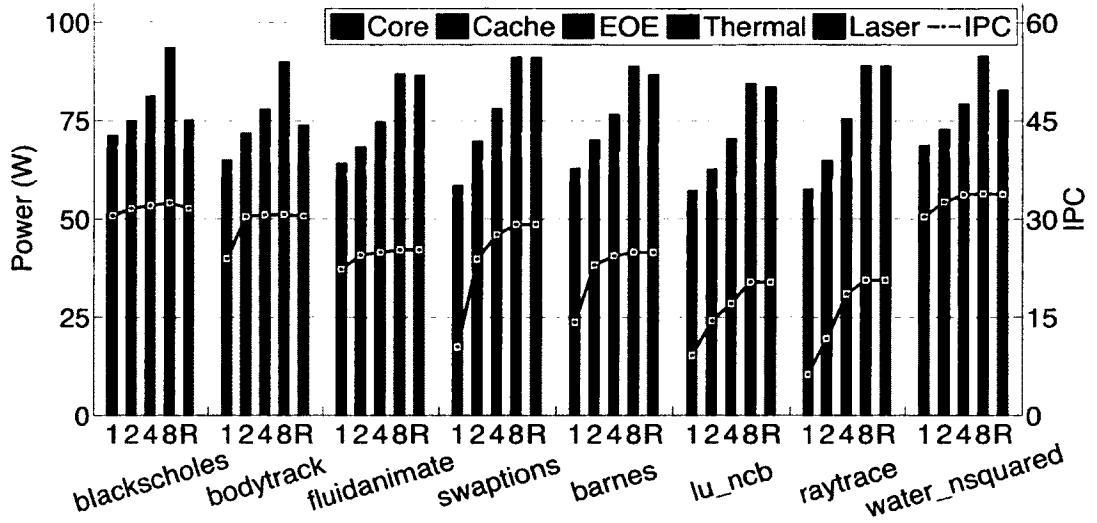


Figure 4-7: Impact of runtime reconfiguration on system performance and power – We compare a system with runtime reconfiguration to a system with fixed number of L2 banks – ‘1’, ‘2’, ‘4’, ‘8’ represents the simulations with fixed 1, 2, 4, 8, banks, respectively. And ‘R’ represents the simulations with run time reconfigurable L2 bank count.

counts leading to larger reconfiguration overhead. For this comparison we assume the waveguide loss is 3 dB/cm. By deactivating redundant L2 banks and their associated silicon-photonic links, the runtime reconfiguration saves laser power by 4.5 W (23.8%) compared to the case where all 8 L2 banks are ON all the time while having an IPC degradation of 0.65% on average.

Figure 4-8 shows EDP improvement after applying the runtime reconfiguration. The comparison baseline is the system running with all 8 L2 banks active all the time. The waveguide loss in the future silicon-photonic links is projected to be in a range of 1 – 3 dB/cm (see Table 3.3). If we assume a conservative waveguide loss of 3 dB/cm, the runtime reconfiguration reduces entire system power consumption by 6.39% and improves the entire system EDP by 5.52%. If we assume a more aggressive waveguide loss of 1 dB/cm, the reconfiguration saves system power by 2.25% and improves system EDP by 1.27%. As the waveguide loss scales down, the default laser power

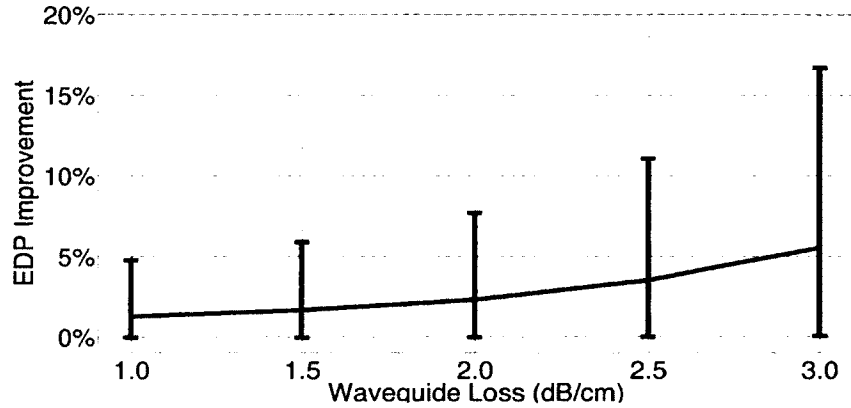


Figure 4-8: EDP improvement across vs. waveguide loss – We show the range of EDP improvement across all benchmarks and the average EDP improvement.

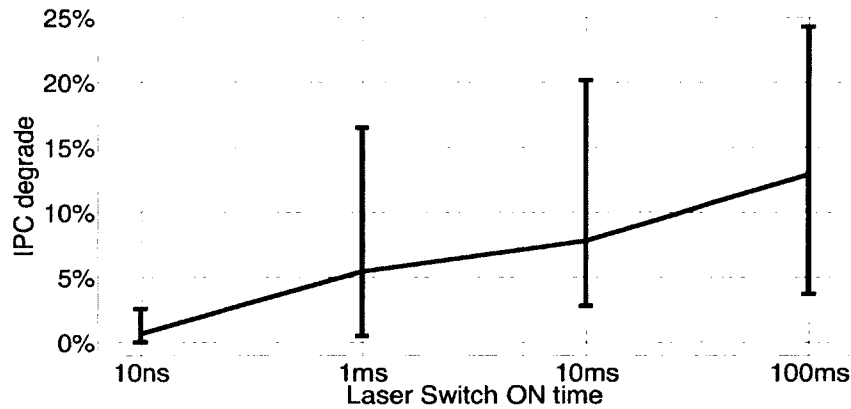


Figure 4-9: IPC degradation vs. laser switch ON time – We show the range of IPC degradation across all benchmarks and the average IPC degradation.

consumption decreases and becomes an insignificant portion of the entire system power consumption. Hence, the savings in laser power consumption become limited and reconfiguration overhead negates the laser power saving, causing a negative impact on system EDP for some benchmarks. Since the design of a silicon-photonics waveguide with less than 3 dB/cm losses remains challenging at the device level, and our proposed technique effectively helps the early adoption of silicon-photonics technology.

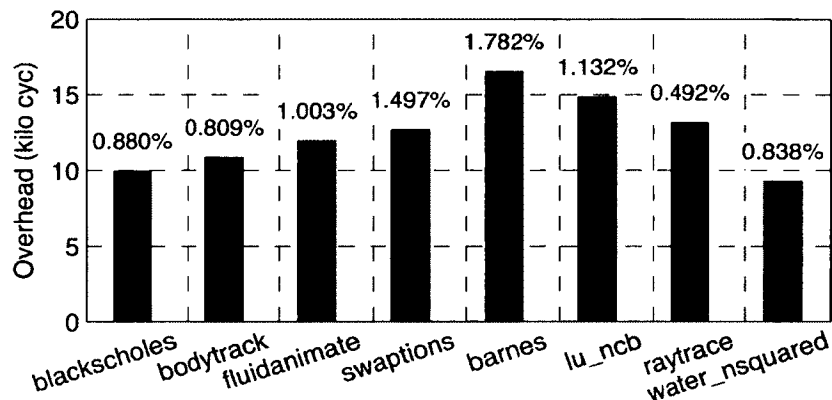


Figure 4-10: Reconfiguration overhead for various benchmarks

– The overhead includes the clock cycles spent in flushing memory blocks during reconfiguration and addressing extra L2 misses after reconfiguration (a partial cold start). The % value is calculated as $\#overhead\ cycles\ of\ one\ reconfiguration / execution\ time\ of\ one\ sampling\ interval * 100$.

As described in Section 3.3, the external laser sources need some time to stabilize after they are switched ON before they can drive the silicon-photonic links. Figure 4-9 shows the impact of stabilization delay on the IPC degradation. Here, we assume a range of 10 ns \sim 100 ms switch ON time, which is feasible using semiconductor diode laser sources as previously described. As the switch ON time increases, the system has to wait longer for its L2 banks to get activated and become accessible. Though the system continues to use the previous L2 configuration until the laser source stabilizes, the large laser stabilization times can lead to upto 12.91% average IPC degradation.

Figure 4-10 shows the overhead of the reconfiguration process. The reconfiguration process of L2 banks requires several cycles for flushing memory blocks from L2 banks and L1 cache (through L2) to the main memory, and fetching these memory blocks back to the appropriate active L2 banks. We measured the average reconfiguration overhead for each benchmark. At the maximum, the cycles spent in a reconfiguration is less than 18,000 cycles that accounts for less than 1.782% of the execution time of

one sampling interval (in the worst case). Our simulations also show that a maximum of 10 reconfigurations were required over the 400 sampling interval, therefore the timing overhead is less than 0.045% of the entire execution time of one benchmark. The average execution time of one benchmark is 12 ms, and after each reconfiguration we had an average of 2893 additional DRAM accesses for flushing and re-fetching 512-bit cache blocks. A DRAM access costs less than 10 pJ/bit (Vogelsang, 2010) that equates to 30 μ W/reconfiguration, which is negligible compared to the reduction of laser power by several watts through L2 reconfiguration. It should be noted that a benchmark does not stop execution while the system is being reconfigured. It continues to execute using the older L2 configuration. It switches to the new L2 configuration once the L2 banks and the associated silicon-photonics links are activated/deactivated.

4.5 Summary

The large laser power in silicon-photonics NoC is limiting its widespread adoption. We propose a runtime cache reconfiguration policy, where we activate/deactivate L2 banks depending on the spatial and temporal variations in the application behavior, and then switch ON/OFF the silicon-photonics links associated with these L2 banks to dynamically manage the laser power. The key idea is that for a given application at any given point of time, we operate the manycore system using the minimum number of L2 banks and silicon-photonics links required for maximizing energy efficiency. Our policy is scalable to large core counts and is applicable to alternate cache and NoC architectures. On a 64-core target system, our proposed technique reduces laser power and system power by 23.8% and 6.39%, respectively, and improves EDP by 5.52% on average. This potentially expedites the process of widespread adoption of silicon-photonics link technology in manycore processors.

Chapter 5

Sharing and Placement of On-chip Laser Sources for Managing Laser Power in Silicon-photonic NoC

5.1 Introduction

On-chip laser sources have the advantage of simpler packaging and easier management, and are being considered as a potential alternative to these off-chip laser sources for driving the silicon-photonic NoC (Kurian et al., 2012; Heck and Bowers, 2014). One key challenge associated with these on-chip laser sources is that the laser source efficiency is fairly small. Hence, in addition to device-level innovations, we need to determine the optimal laser source sharing and placement configurations that would maximize the laser source efficiency and minimize the electrical input power of the laser. The laser source efficiency depends on the optical output power that is output by the laser source. This optical output power depends on the physical layout of the silicon-photonic NoC and the bandwidth (i.e. number of required wavelengths) of the NoC channels. The laser source efficiency is low at small optical output power values. Hence, to operate the laser source at maximum efficiency we need to ensure the laser source outputs an optimal optical output power. This need for outputting optimal optical output power necessitates sharing of laser sources across waveguides. At the same time, the power density of each laser source defines the temperature in the

neighborhood of the laser source, which in turn defines the laser source efficiency. As the laser temperature increases, the efficiency of the laser decreases. Hence, the laser sources need to be strategically placed so that they operate at as minimum temperature as possible.

We explored the limits and opportunities for sharing and placement of on-chip laser sources by jointly considering the NoC bandwidth constraints driven by the applications running on the manycore system, thermal constraints driven by the power consumed by the cores and the laser source and physical layout constraints driven by the losses in the photonic devices to determine the optimal sharing as well as the placement of the on-chip laser sources with the goal of maximizing laser efficiency and minimizing the electrical input power consumption of the laser. Using a 256-core 3D-integrated system consisting of separate processor logic layer, photonic device layer and the laser source layer as our case study, for various NoC logical topologies and NoC physical layouts, we show that laser power consumption can be lowered by sharing of laser sources across the various silicon-photonic links and smartly placing these laser sources on the laser layer. It should be noted that our proposed approach is also valid for a system where all photonic devices are monolithically integrated with the CMOS devices with the laser sources on the adjacent layer.

In this chapter, Section 5.2 gives a description of the target system architecture that is used for our case study. In Section 5.3, we describe our methodology for determining the optimal design for the laser source, and then we evaluate the application of our proposed methodology across various logical topologies and physical layouts of the silicon-photonic NoC in Section 5.4. Section 5.5 summarizes our analysis.

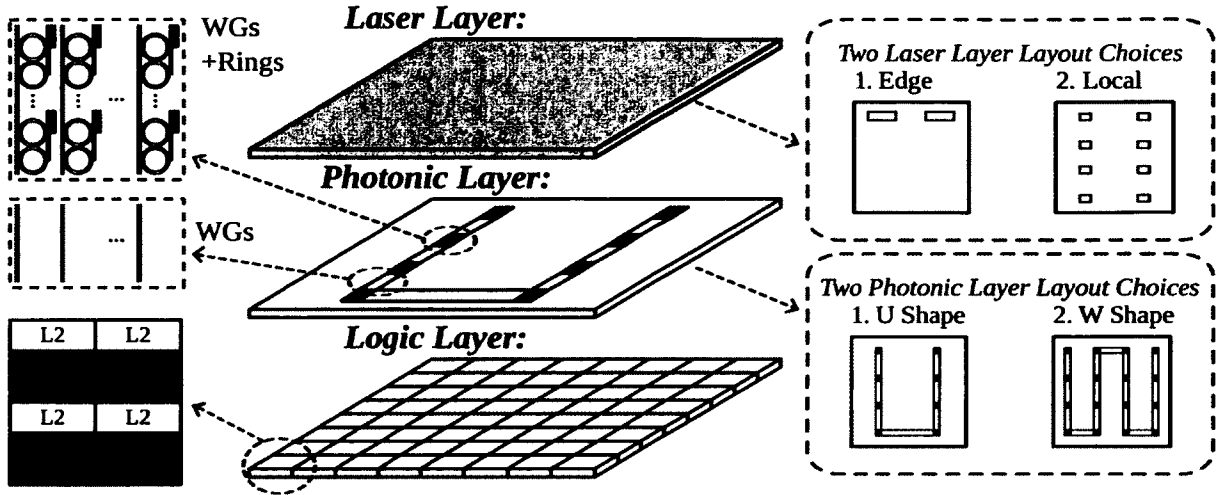


Figure 5-1: Overview of the 3D flip-chip manycore system and layouts for each layer – Output from the laser sources is routed into the waveguides through couplers. The ring modulator is driven through TSVs by the transmitter in the logic layer. Photodetector’s output is fed to the receiver on the logic layer through TSVs.

5.2 Target System

To explore the design space for sharing and placement of laser sources, we consider a 3D stacked flip-chip manycore system (see Figure 5-1 and Figure 5-2) with a logic layer containing 256 cores fabricated using standard bulk CMOS process, a photonic NoC layer next to the metal stack and a layer for the laser source. The logic layer and photonic layer are connected using through-silicon-vias (TSVs). The architecture of each core in the logic layer is similar to an IA-32 core used in the Intel Single-Chip Cloud Computer (Howard et al., 2010). We scale the core power and dimensions from 45 nm to 22 nm technology, resulting in a total chip area of 366.1 mm^2 (0.93 mm^2 per core, including L1, and 0.50 mm^2 for each 256KB private L2 cache). We choose the operating frequency as 800 MHz and the voltage as 0.65 V , and scale the per-core power based on the reported data of Intel 22nm Tri-Gate technology. The average per core power is 0.46 W , and the average per L2 cache power is 0.01

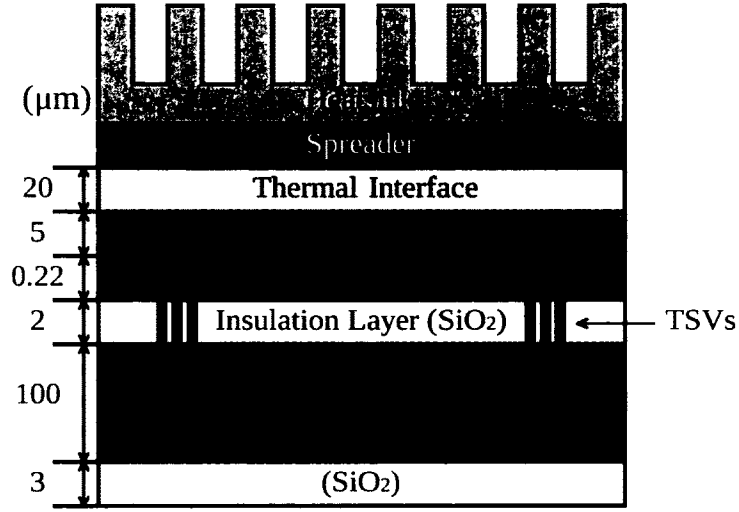


Figure 5-2: Cross-sectional view of our target 3D manycore system

W (based on ITRS-LSTP cache model in Cacti (Thoziyoor et al., 2008)). There are 16 memory controllers that are uniformly distributed along the two edges of the chip. The silicon-photonic NoC in our system is used for connecting the private L2 caches of the cores and the memory controllers. It should be noted that our proposed methodology for sharing and placement of laser sources is also applicable for photonic NoCs that provide connectivity between L1 cache and shared L2 cache.

In our target system, light waves emitted by one or more InGaAs/GaAs laser sources located on the laser layer are routed into waveguides located on the photonic device layer through a coupler. We placed the laser sources over the low-power L2 cache blocks to minimize the impact of the core power over the laser source temperature. The laser source is driven using TSVs by a driver located in the logic layer. Except for the photodetector which is made of *Ge*, all other photonic devices in the photonic device laser are made of mono *Si*. The cladding for the various devices in the photonic device layer is made of *SiO₂*. The light waves pass next to a ring modulator that is driven through the TSVs by the link transmitter circuit located on the logic layer. The

modulators convert data from electrical medium to photonic medium. The modulated light waves travel along the waveguide and can pass through zero or more ring filters. At the receiver side, the light waves are filtered by wavelength matching ring filters and these light waves are incident on a *Ge* photodetector. The current generated by the photodetector passes through the TSVs and is fed into the link receiver circuit located on the logic layer.

To explain the various tradeoffs associated with choosing the laser source configuration across various logical topologies, we compare an 8-ary 3-stage Clos topology, 16-ary 3-stage Clos topology and a 16 x 16 crossbar mapped to a U-shaped physical layout of the waveguides in the photonic layer of our target system. The choice of these topologies is driven by the fact that silicon-photonic links technology is most appropriate for high-radix low-diameter topologies like Clos and Crossbar. This is followed by a discussion of the tradeoffs associated with choosing the laser source configuration when both the 16 x 16 crossbar and the 16-ary 3-stage Clos are mapped to U-shaped and W-shaped layout. For our analysis, we use the projected photonic devices losses listed in Table 3.3. In general, it should be noted that our proposed methodology for choosing a laser source is applicable to any physical layout and any logical topology mapped to that layout.

5.3 Laser Sources

In this section we first describe the laser source model and the thermal modeling approach that we used to evaluate the laser source design space. We then provide a discussion of the power, efficiency and thermal tradeoffs associated with laser source designs, which is then followed by a description of our methodology to determine the sharing and placement of on-chip laser sources to minimize laser power consump-

tion.

5.3.1 Modeling

The laser wall-plug efficiency (η_{WPE}) is given by the optical output power (P_o) relative to the electrical input power (P_{IN}):

$$\eta_{WPE} = \frac{P_o}{P_{IN}}, \quad (5.1)$$

where P_o is equal to (Coldren et al., 2012)

$$P_o = \eta_i \eta_d \frac{hc}{\lambda q} (I - I_{th}), \quad (5.2)$$

where η_i and η_d are the laser internal efficiency and the differential quantum efficiency, respectively; h , c and q are the Plank's constant, speed of light and electron charge constant, respectively; λ is the operating laser wavelength; and I and I_{th} are the drive and the threshold currents, respectively.

The electrical input power of the laser is the product of the drive current and the total voltage across the laser's terminal and it can be calculated as follows

$$P_{IN} = I^2 R_s + IV_d, \quad (5.3)$$

where R_s is the laser series resistance and V_d represents the diode voltage.

One of the weakest points of semiconductor lasers is the strong dependence of the optical output power out P_o on temperature. Fortunately, simple empirical formulas match quite well with the measured characteristics of almost all types of lasers. These

empirical formulas are:

$$I_{th} = I_{th}^0 e^{T/T_0}, \quad (5.4a)$$

$$\eta_d = \eta_d^0 e^{-T/T_\eta}, \quad (5.4b)$$

where T_0 and T_η are called the characteristic temperatures of the threshold current and the differential quantum efficiency, respectively, while I_{th}^0 and η_d^0 are the threshold current and the differential quantum efficiency projected to a reference temperature. Additionally, the diode voltage V_d depends upon temperature through the Shockley diode equation:

$$V_d = \frac{k_B T}{q} \ln \left(\frac{I}{I_s} \right), \quad (5.5)$$

where k_B is the Boltzmann constant and I_s is the reverse bias saturation current. By substituting Eqs. (5.4) and (5.5) into Eqs. (5.2) and (5.3), one gets a simple laser temperature dependence model. Here, a strained 80 Angstrom InGaAs/GaAs with 5 quantum wells laser has been considered.

To evaluate the effect of temperature variations (due to variations in core power and laser power) on the laser efficiency, we use the 3D extension of HotSpot 5.02 (Meng et al., 2012) for our thermal simulations. We set the ambient temperature at 35°C and use the default package configurations in HotSpot. The cross-sectional view of the 3D system that we evaluated is shown in Figure 5.2. The 3D system has three main layers – laser layer, photonic layer and core layer. An insulation layer is inserted between photonic layer and core layer to reduce photonic losses in the devices in the silicon photonic layer. The laser layer contains single-band InGaAs/GaAs laser sources, and the photonic layer includes ring modulators/filters, waveguides and photodetectors. The light waves emitted by the laser sources enter into the waveguides on the photonic layer. These waves are first modulated on the transmitter

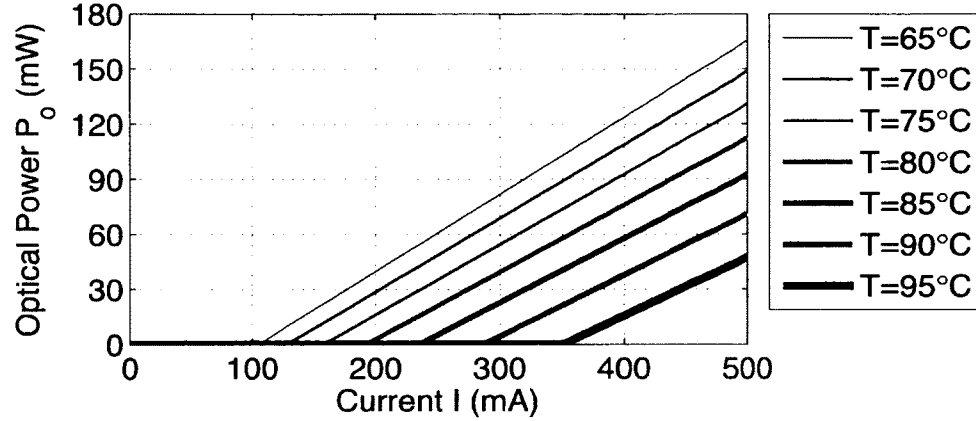


Figure 5-3: P-I characteristics of a laser source at various temperatures

side through TSVs and then filtered by one or more rings before being absorbed by the photodetector. The output of the photodetector is then transmitted to the receiver on the core layer through TSV connections. For thermal analysis, the laser sources are modeled individually on the laser layer because their power dissipation varies based on their locations. However, for modeling waveguide and ring on the photonic layer we aggregated them into larger-sized blocks in the floorplan as using a separate model for every waveguide and ring leads to large simulation time without any significant improvement in accuracy. Our aggregation methods provide desirable accuracy-simulation time tradeoffs in thermal simulation (Coskun et al., 2009). We compute the joint thermal resistivity for waveguide blocks and ring blocks using $R_{joint} = V_{total}/\Sigma(V_i/R_i)$, where R_i and V_i refer to the thermal resistivity and volume of material i in the blocks. The dimensions of our system are shown in Figure 5-1. All the thermal results we report in this work are from steady state analysis.

5.3.2 Optical Power, WPE and Temperature Tradeoffs

Figure 5-3 presents the optical output power of the laser source versus the input current (P-I characteristic) for various temperatures and shows that the threshold

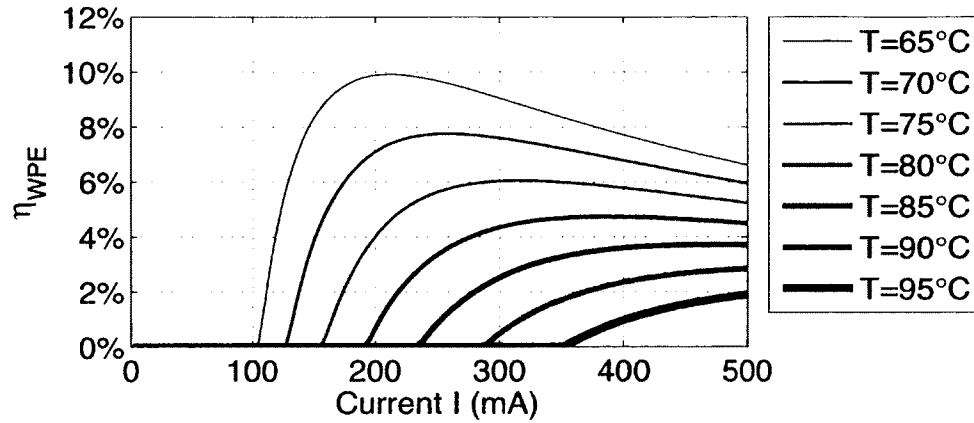


Figure 5-4: Wall-plug efficiency vs Input current at various temperatures

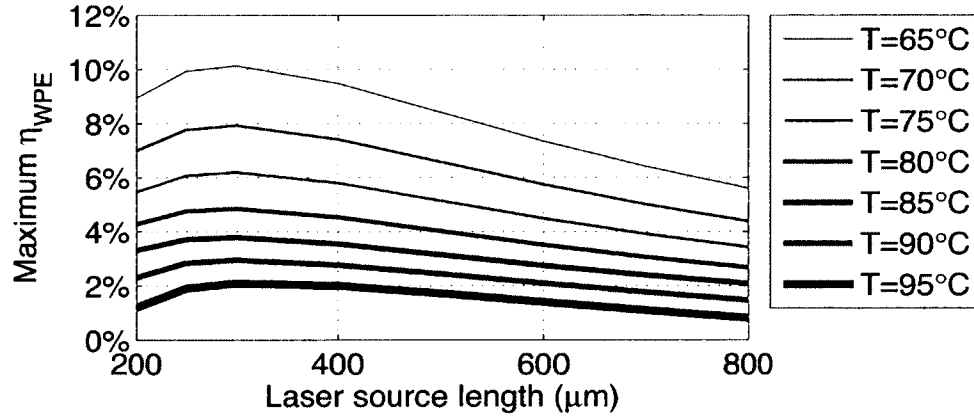


Figure 5-5: Wall-plug efficiency vs Laser Source Lengths at various temperature

current I_{th} increases with temperature while the laser optical output power goes down.

Figure 5-4 shows that the efficiency of the laser sources increases, and then decreases as the input current increases and the peak efficiency decreases with increase in temperature. Hence, we would like to operate the laser source at minimum temperature and ensure that the input current is optimal such that we maximize the laser efficiency.

Figure 5-5 shows the variation of laser source efficiency with laser source length (while

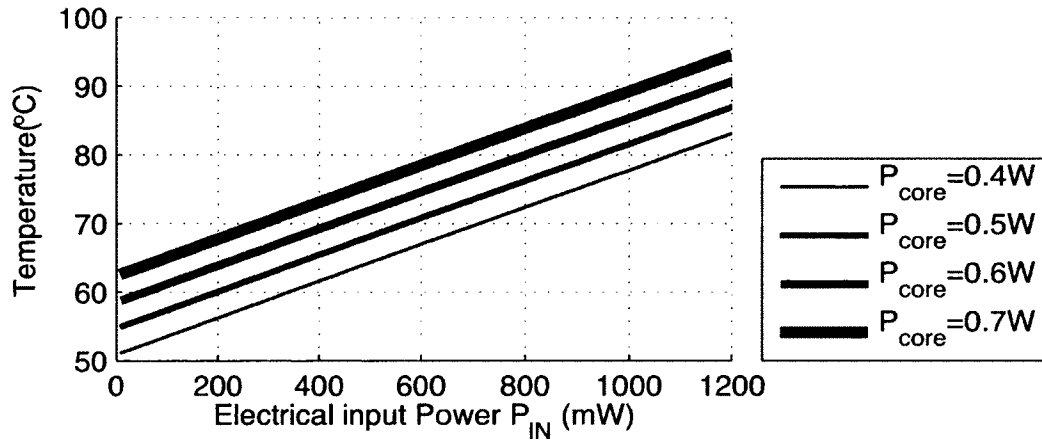
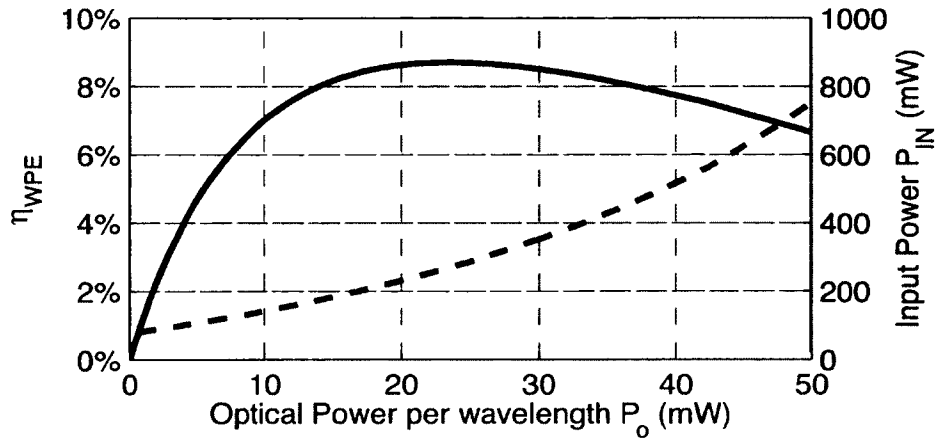


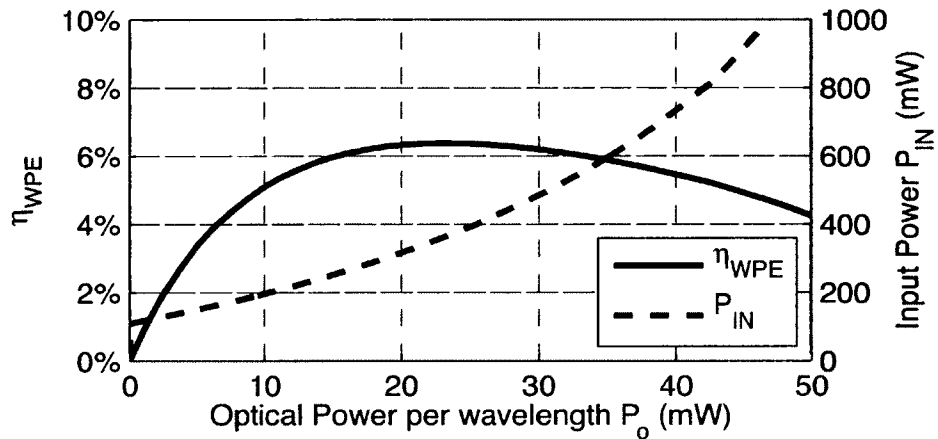
Figure 5-6: Laser source temperature vs. electrical input power – The dimension of the laser source is $300\mu\text{m} \times 50\mu\text{m}$

laser operates at the optimal input current) for various laser source temperature. Here, the laser source width is fixed at $50\mu\text{m}$ while the laser source length changes in a range of $200\mu\text{m}$ to $800\mu\text{m}$. Figure 5-5 shows that a laser source that is $300\mu\text{m}$ long has the highest efficiency at any given temperature. This behavior is typical from strained 80 Angstrom InGaAs/GaAs with 5 quantum wells lasers. Therefore for the remaining analysis, we use the $300\mu\text{m} \times 50\mu\text{m}$ laser source as our building block.

To determine how the electrical input power of the laser affects the temperature of the laser we ran HotSpot simulations for the 256-core target system where cores consumed 0.4 W, 0.5 W, 0.6 W and 0.7 W of power. We placed the laser sources over the low-power L2 cache blocks to minimize the impact of the core power over the laser source temperature. As shown in Figure 5-6, as the laser input power increases the temperature of the laser source increases, which in turn would adversely impact the laser source efficiency. The increase in the power consumption of the cores in the logical layer also increases the surrounding temperature of the laser source, which in turn would lower the laser source efficiency.



(a)



(b)

Figure 5.7: WPE vs optical output power for various sharing granularity – The layer operates at 0.4 W per core (a) and 0.7 W per core (b) respectively.

Based on above power-temperature-efficiency tradeoffs of the laser source Figure 5.7 shows the laser source efficiency and electrical input power of the laser varying with optical output power of the laser source used in two systems – one where each core is burning 0.4 W of power and the second where each core is burning 0.7 W of power. The optical output power that needs to be output by a laser source depends on the optical losses in a photonic links being driven by that laser source. For the case when

all cores are burning 0.4 W of power, Figure 5.7(a) shows that the optimal operation point is when the laser emits 22 mW of optical power per wavelength, where the laser source achieves the peak efficiency of 8.2% resulting in an electrical input power of the laser to be 268 mW. On the other hand, when each core's power consumption is 0.7 W, the optimal operation point is still around 22 mW, but the laser efficiency decreases to 6.2% due to higher temperature of the laser source resulting an electrical input power of the laser to be 355 mW. This shows that the electrical input power increases with increase in core power. Hence, one needs to develop runtime techniques that can maintain the peak temperature of the cores and lasers below a certain threshold to minimize laser power.

To operate at maximum efficiency a laser source needs to output a certain amount of optical power. Depending on the optical power required per λ , there may be a need to share laser source across two or more waveguides. Figure 5-8 shows the two methods for sharing the laser source across multiple waveguides. Figure 5.8(a) uses ring filters at the crossing of waveguides to filter and route each wavelength into the waveguide such that we can transmit a range of wavelengths in each each waveguide. If each waveguide crossing would cause 0.05 dB optical loss (Batten et al., 2008), then if 64 waveguides are sharing this laser source, this sharing approach would cause an overhead of 3.2 dB optical loss. Figure 5.8(b) shows an alternate method that first merges the light waves of multiple wavelengths from a set of laser sources (each emits one wavelength), and then split the lights into multiple waveguides. Assuming each split can cause a 0.2 dB optical loss, then if 64 waveguides are sharing these laser sources, this type of sharing would cause an overhead of 1.2 dB optical loss. Therefore, we choose the method shown in Figure 5.8(b).

Figure 5-9 shows a plot of variation in laser efficiency with optical output power per wavelength for different granularity of sharing of a single-band laser source across

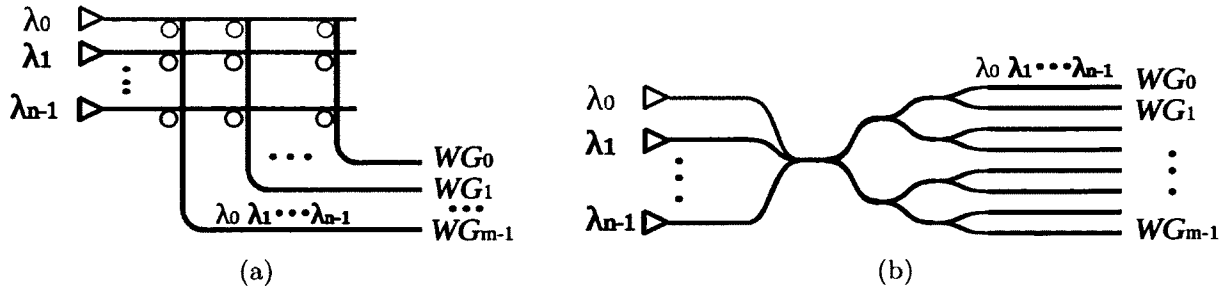


Figure 5-8: The laser source sharing methods – (a) The laser source sharing through ring filters at the crossings of waveguides (b) The laser source sharing through waveguide merger and splitters.

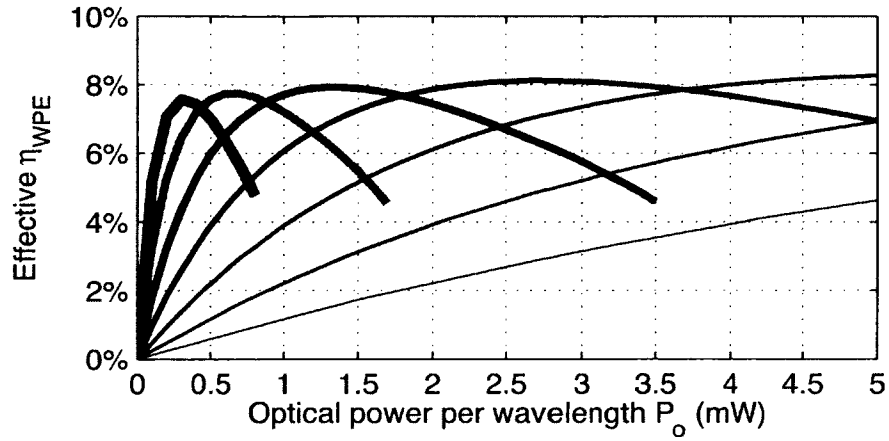
multiple waveguides. We show two cases – core power is 0.4 W and 0.7 W. For the 0.4 W case, if the total optical output power per wavelength is small, say 1 mW, then using a laser source per wavelength per waveguide results in a laser source efficiency of 1%. If we were to share the laser source across two waveguides, then the total optical output power that needs to be output by the laser source increases, which increases the efficiency to 2%. For this 1 mW optical output power case, it makes most sense to share the laser source across 16 waveguides as it provides the maximum efficiency. Sharing of this laser source across more than 16 waveguides increases the sharing cost and at the same time the laser source needs to emit even larger optical power, which decreases the laser source efficiency. Thus, optimal sharing of a laser source is critical to operating the laser source at its maximum efficiency and minimizing its electrical input power. A similar argument can be made for the case where the core power is 0.7 W. The only difference is that the laser source efficiency for the 0.7 W case would be lower as the higher core power results in higher laser source temperature, which lowers the laser source efficiency.

5.3.3 Broadband vs Single-band Laser Sources

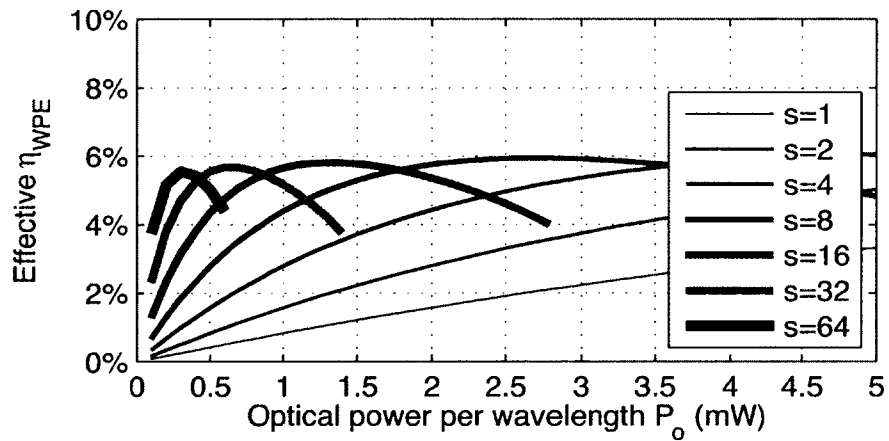
Broadband laser sources that can emit the entire range of wavelengths that are used by a waveguide could be used for our target. However, our analysis shows that using single-band laser sources with sharing results in lower electrical input power than using broadband laser source. For example, say we have 16 waveguides with 16λ per waveguide and each λ requires an optical power of 1 mW. We could use one broadband laser source (emitting those 16λ) per waveguide such that the total optical output power emitted by each broadband laser source is 16 mW (1 mW per λ). We could also use 16 single-band laser sources where the output of each laser source is shared across the 16 waveguides. Here the total optical output power emitted by each single-band laser source is 16 mW too. However, given that single-band laser source has a similar dimension as broadband laser source, the power density of using a broadband laser source is significantly higher than that of using 16 single-band laser sources. As a result, the single-band laser source will have much better efficiency than the broadband laser source. Moreover, because of the lower efficiency of the broadband laser source, it will need a large amount of electrical input power. This in turn will increase the laser source temperature, which will further lower the laser source efficiency. Hence, we propose to use single-band on-chip laser sources to power the silicon-photonics NoC.

5.3.4 Methodology to Determine Optimal Sharing and Placement

To determine the optimal placement of the laser source, we propose a cross-layer approach where we jointly consider the NoC bandwidth constraints driven by the applications running on the manycore system, thermal constraints driven by the power consumed by the cores and the laser source, physical layout constraints driven by the



(a)



(b)

Figure 5-9: Effective WPE vs. sharing granularity of laser sources – The layer operates at 0.4 W (a) and 0.7 W (b) per core, respectively. $s = x$ indicates that x waveguides are sharing the output of the laser source. Splitter loss is accounted.

losses in the photonic devices and laser source designs that are compatible with our proposed 3D system. Figure 5-10 shows a flowchart describing the algorithm for choosing the laser source and its placement in the 3D system.

The number of cores in the target system and the type of applications that are expected to run on the target system can be used to determine the amount of traffic that

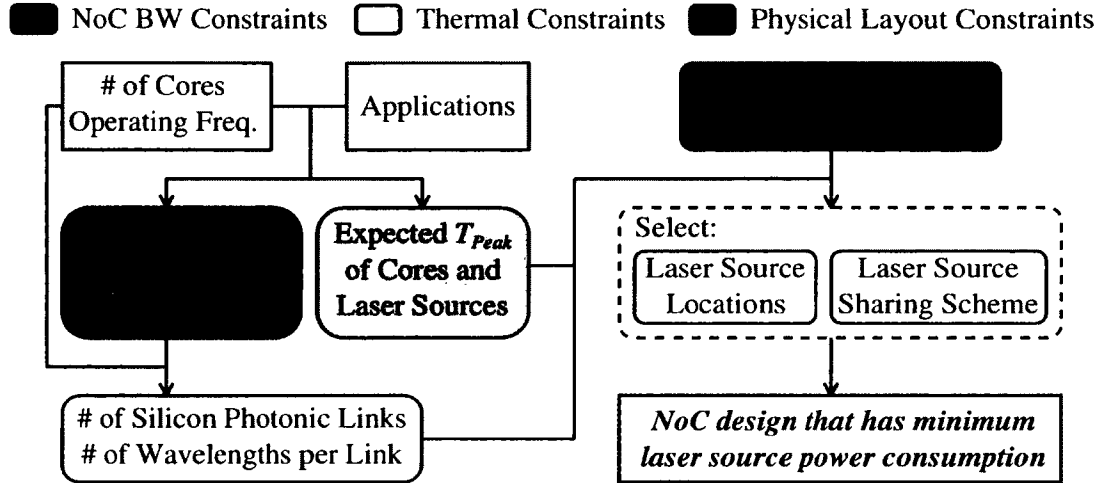


Figure 5-10: Flowchart for deciding the sharing and placement of on-chip laser sources

will be injected into the NoC. This in turn can be used to determine the optimal NoC topology at the logic level and the bandwidth of each channel in the NoC. Depending on the bandwidth per channel, operating frequency of the cores and bandwidth of each silicon-photonic link, we can determine the number of silicon-photonic links i.e. number of wavelengths that will be required for the target system. The chosen logical topology for an NoC can be mapped to several different layouts (Batten et al., 2012). Depending on the loss components (waveguide loss, through loss, crossing loss, etc) in a silicon-photonic link we identify various potential physical layouts of the NoC with three candidates for the location and sharing of the laser source – 1) all laser sources are placed locally next to the router with a laser source emitting one wavelength for one waveguide (no sharing), 2) all laser sources are placed locally next to the router with each laser source emitting one wavelength that is shared across two or more waveguides, and 3) all laser sources are placed along the edge with each laser source emitting one wavelength shared by two or more waveguides. For these candidate layouts and the target bandwidths per channel, depending on the thermal properties i.e. expected peak temperature, of the laser source at runtime and the available

laser source design we will decide the placement of the laser source and its sharing granularity such that the overall system will consume minimal laser power.

5.4 Case Studies

In this section, we present two case studies, where we show how the sharing and placement of laser sources changes with logical topologies and physical layouts. For this analysis we assumed all cores are consuming 0.7 W of power and we used results presented in Figure 5.7(b) and 5.9(b) to determine the laser source efficiency.

Figure 5.11(a) shows the total electrical input power of the laser sources for an 8-ary 3-stage Clos topology (Joshi et al., 2009) for different placements of the laser source. This 8-ary 3-stage Clos network uses 24 routers (8 routers in each stage), and each router in the 1st and 3rd stage is connected to 32 cores. There are 64 photonic channels connecting the 1st and 2nd stage of routers and another 64 photonic channels connecting the 2nd and 3rd stage of routers. The photonic channels are mapped to a U-shaped layout shown in Figure 5-1. For a waveguide loss of 2 dB/cm, we need to inject 0.15 mW optical output power per wavelength. if we use local laser sources that are not shared then the efficiency of this ‘local’ laser source for 0.15 mW optical output power is 0.12%. This results in a total electrical input power of the laser of

Table 5.1: Architectural-level parameters for 5 NoCs under consideration – U-shaped and W-shaped layouts are shown in Figure 5-1.

Logical topology	Physical layout	Dimension	Concentration	λ per channel	# of channels
Clos	U-shaped	8-ary	4	16	128
Clos	U-shaped	16-ary	1	4	512
Crossbar	U-shaped	16 x 16	4	64	32
Clos	W-shaped	16-ary	1	4	512
Crossbar	W-shaped	16 x 16	4	64	32

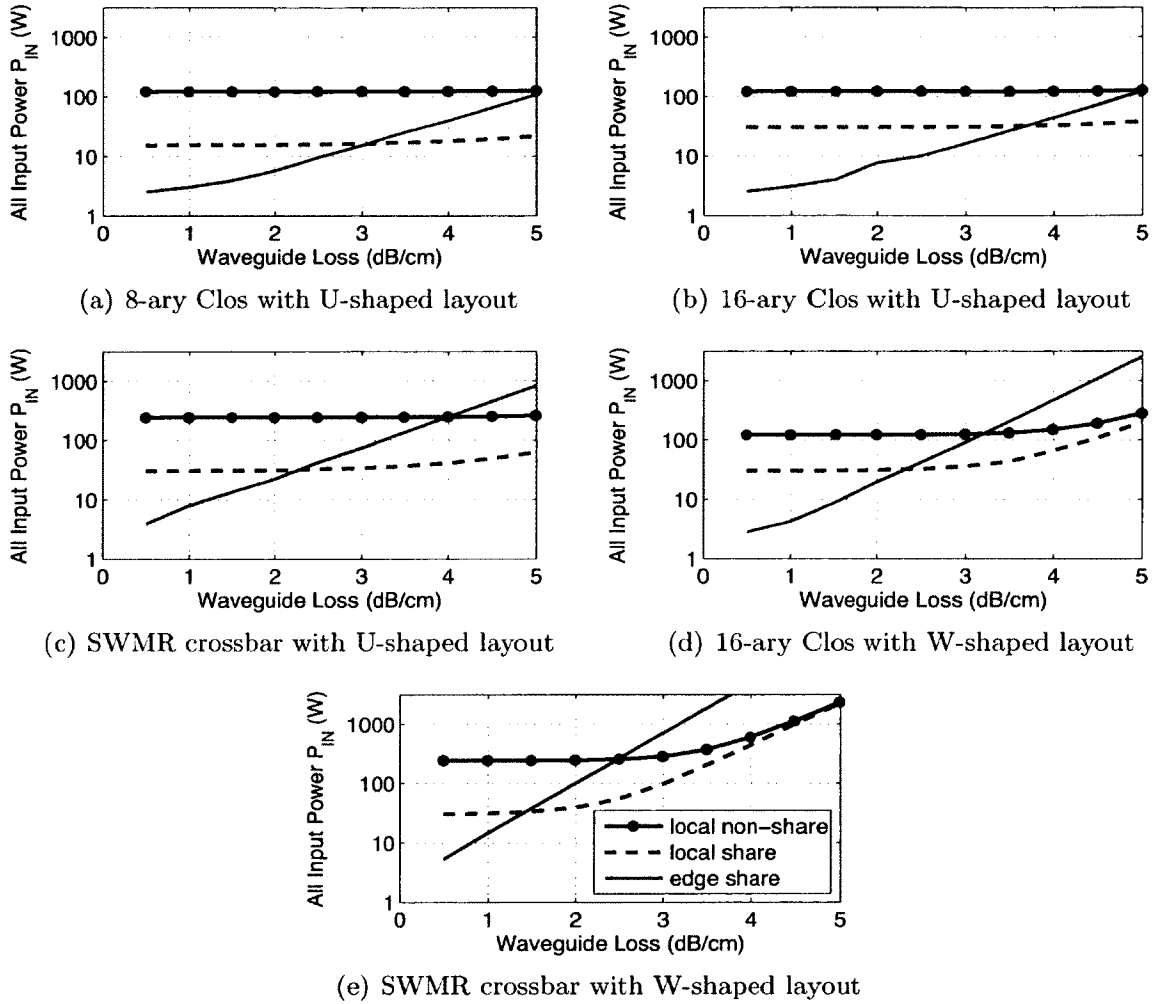


Figure 5-11: Total laser power vs. waveguide loss for various sharing and placement – (a), (b) and (c) compares various topologies with U-shaped layout. (b) and (d) compares various layouts for 16-ary Clos topology. (c) and (d) compares various layouts for SWMR crossbar topology. Here, the logical layer operates at the peak 0.7 W per core.

243 W (119 mW per laser source).

In the physical layout of this 8-ary 3-stage Clos topology, one router each from the 1st stage, 2nd stage and 3rd stage are placed next to each other. Hence, there is an opportunity for sharing of laser sources between the 16 photonic channels (8 from the 1st stage router and 8 from the 2nd stage router), whereby the optical output

power of a laser source is split and routed into the waveguides associated with these photonic channels. This sharing of laser sources increases the total optical output power of a laser source, which improves its efficiency. For this particular example, each one of the 16 photonic channels is mapped to a waveguide with 16λ per photonic channel i.e. 16λ per waveguide. If we share a laser source of each λ across these 16 waveguides, then the total optical output power of the laser source needs to be around 1.17 mW for each λ , which corresponds to a laser source efficiency of 0.96% and a total electrical input power of the laser of 15.48 W (121 mW per laser source).

For the example laser source, the maximum efficiency is achieved at an optical output power of 23 mW. If we want to use a laser source that outputs 23 mW of power, then we propose to place the laser source along the edge so that there are more opportunities for sharing. The 128 photonic channels that we have in the 8-ary 3-stage Clos topology correspond to 128 waveguides. For the case where we place the laser sources along the edge, the optical output power that needs to be injected in each waveguide will be 0.18 mW for each λ . The value is larger than 0.15 mW due to longer waveguide lengths. We can share 16 laser sources (1 for each λ) across these 128 waveguides so that we can operate them at 6.38% efficiency. This results in a total electrical input power of the laser to be 5.74 W (359 mW per laser source).

As the waveguide loss per cm increases, the total optical output power that needs to be injected into the waveguides increases. In case of local laser sources, this increase in optical output power improves the efficiency. As a result the total electrical input power for the case when using local laser sources that are not shared does not increase significantly as waveguide loss increases. In case of the local laser sources that are shared, the increase in the optical output power requirement per waveguide pushes the efficiency of the laser source towards its peak value. Hence, similar to local laser sources that are not shared, an increase in the waveguide loss results in a marginal

increase in the electrical input power of the laser sources.

The laser sources that are located along the edge have longer waveguides and experience a more pronounced effect when waveguide loss per cm increases. And they are already operating at their peak efficiency when the waveguide loss is 2 dB/cm. In case the waveguide loss per cm increases, it would increase the optical output power that we would need to inject in the waveguide, which would lower the laser source efficiency. Hence, we would need to reduce the number of waveguides that share a laser source i.e. increase the number of laser sources to reduce the total electrical input power of the lasers. For example, when the waveguide loss increases from 2 dB/cm to 2.5 dB/cm, we need to increase the number of laser sources per wavelength from 1 to 2 to maintain the total electrical input power of the lasers. There would however be a net increase in the laser power due to increase in the waveguide loss per cm. Overall, if the waveguide loss is small, we can place the laser sources along the edge and share them across several waveguides to minimize the total power consumed in the laser sources. On the other hand, if the waveguide loss is large, then using laser source that are local but shared across two or more waveguides would be beneficial.

For the same 256-core target system, we could use a 16-ary 3-stage Clos network if we want to reduce the contention among cores at the input of each router. This 16-ary 3-stage Clos topology has 48 routers (16 routers in each stage) with each router in the 1st and 3rd stage connected to 16 cores. This network will require a total of 512 channels. If we match bi-section bandwidth of this 16-ary 3-stage Clos with the 8-ary Clos, each channel will need 4 λ , and the system will have a total of 128 waveguides with 4 channels (4 λ for each channel) sharing one waveguide (16 λ in each waveguide). The total number of waveguides in 16-ary 3-stage and 8-ary 3-stage Clos are the same. Broadly, the trends for the electrical input power of the laser for this 16-ary 3-stage topology are similar to the trends for the 8-ary 3-stage topology.

One exception is that the electrical input power for the case where the laser sources are placed locally and shared is higher for the 16-ary 3-stage topology due to the decrease in the opportunities for sharing of laser sources. In addition, the choice of locally shared laser source over laser sources that are placed along the edge and shared occurs at a lower waveguide per cm loss in the 16-ary 3-stage topology compared to 8-ary 3-stage topology due to longer waveguides in the 16-ary 3-stage topology.

We could also use a 16 x 16 Single-Write-Multiple-Read (SWMR) crossbar topology having a concentration of 16 i.e. each input of the crossbar can be accessed by 16 cores (similar to 16-ary 3-stage Clos) and map it to the U-shaped physical layout. Here to match the bisection bandwidth of the crossbar with the Clos networks, we need 64λ for each channel (4 waveguides per channel). Figure 5.11(c) shows the total electrical input power for the laser. Compared to the Clos network, the SWMR channels are shared by more receivers, therefore the large number of rings along longer waveguides causes the higher laser power consumption than the Clos networks. For small waveguide loss per cm laser sources that are shared and placed along the edge are preferable. On the other hand, for larger waveguide loss per cm, the net optical output power for the laser sources placed along the edge significantly increases and so laser sources that are placed locally, but are shared are preferable.

If we were to use a W-shaped layout (see Figure 5.11(d)) to provide connectivity in the 16-ary 3-stage Clos then we would have higher waveguide losses, which in turn would increase the demand of optical output power. Hence, for all three placement options, compared to the 16-ary 3-stage topology mapped to a U-shaped topology, the electrical input power of the laser sources for 16-ary 3-stage Clos mapped to a W-shaped layout will be higher. We could also use a W-shaped layout (as shown in Figure 5-1) for the SWMR crossbar. W-shaped waveguides result in very long waveguide lengths which causes high waveguide losses. Figure 5.11(e) shows the total

electrical input power for the laser varying with waveguide loss per cm. Similar to the U-shaped layout for small waveguide loss per cm laser sources that are shared and placed along the edge are preferable, and for larger waveguide loss per cm laser sources that are placed locally, but are shared are preferable.

5.5 Summary

We explored limits and opportunities for sharing and placement of on-chip laser sources with the goal of minimizing the total laser power consumption. We first explored the power, efficiency and temperature tradeoffs associated with on-chip laser sources. We then used a cross-layer methodology where we jointly considered the NoC bandwidth constraints, the thermal constraints and the physical layout constraints to determine the optimal sharing of laser sources such that the optical output power of the on-chip laser source corresponds to its maximum efficiency and the optimal placement of the laser sources such that temperature of these laser sources is minimum and the optical output power is optimal so as to maximize the efficiency of the laser source. We explored the application of our methodology to three different logical topologies, two different physical layouts and three different sharing/placement strategies. Our analysis shows that the choice of laser source placement and sharing changes with the choice of logical topology and physical layout. For a matching bisection bandwidth, at low waveguide loss per cm, the 8-ary 3-stage Clos with locally placed and shared laser source consumes the least total laser power, while at large waveguide loss per cm, the 8-ary 3-stage Clos with shared laser source placed along the edge consumes the least total laser power.

Chapter 6

Sharing of Computing Resources through Large Bandwidth Provided by Silicon-photonic NoC

6.1 Introduction

The need for energy-efficient high-performance computing along with the rapid increase in application complexity have led to manycore architectures (Til, ; Howard et al., 2010) that exploit massive thread-level parallelism (TLP) using many lightweight (in terms of cache size, issue width, speculative execution, etc.) processor cores. At the same time, these lightweight cores need to support current instruction set architectures (ISA), which implies the need to provide every lightweight core with a variety of units for instruction execution. A typical set of core's execution units (EUs) includes arithmetic and logic units (ALU), a branch unit (BRU), an address generation unit (AGU), floating point units (FPU), and EUs that perform single instruction multiple data (SIMD) operations (e.g., Intel's SSE/AVX extensions (Intel Corp., 2013), or AMD's 3DNow! (Oberman et al., 1999)). These EUs, however, do not have 100% utilization as the lowering of core complexity reduces the number of opportunities for instruction-level parallelism (ILP). Figure 6-1 plots the total utilization of FPUs in a 64-core system with three FP subunits per core (see `FPalu`, `FPMulDiv` and `FPMov`) when running an `FMM`, a FPU-intensive application from SPLASH-2 bench-

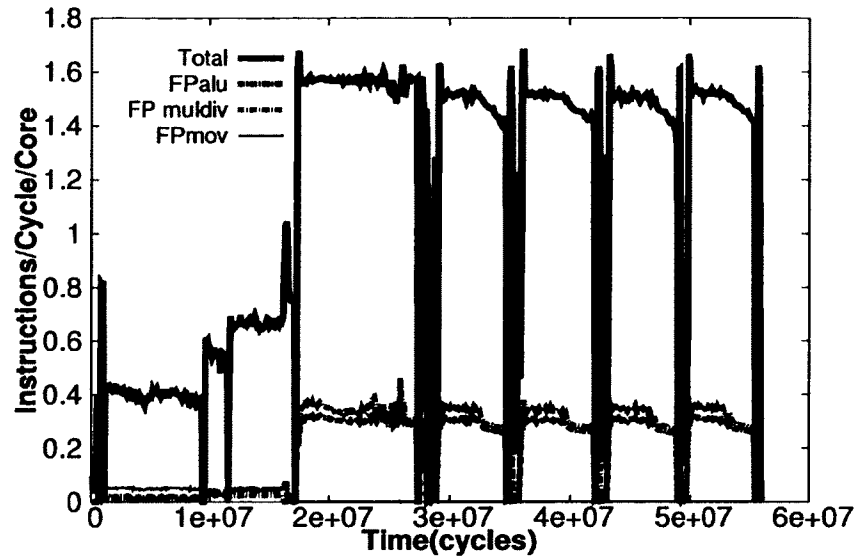


Figure 6-1: Average per core utilization of FPUs when running FMM application on a 64-core system

mark suite (Woo et al., 1995). Both FPUs are utilized an average of 20% of the time.

Mechanisms that leverage this low utilization and enable sharing of EUs like FPUs (that can account for 20% of the total core logic in simple cores (Jain et al., 2012; Gwenapp, 1995)) have been successfully applied to processors with up to a dozen cores to save both on-chip area and static power (Meltzer, 1999; Ahmad and Arslan, 2005; Kakoe et al., 2013; Castells-Rufas et al., 2011). A similar sharing of EUs (that have low utilization) could be implemented in current and future manycore systems to reduce on-chip power consumption and area. This reduction in area and power could be leveraged to improve the overall system performance, by optimizing core complexity, increasing cache sizes, etc.

To determine the opportunities for saving area and power through sharing of EUs in manycore systems, we executed a multi-programmed workload composed of four

different 64-thread applications – `canneal`, `radix`, `fft` and `blackscholes` from SPLASH-2 (Woo et al., 1995) and PARSEC (Bienia et al., 2008) benchmark suites, on our target 256-core system. The `canneal`, `radix`, `fft` and `blackscholes` benchmarks required 8, 12, 24 and 24 FPU, respectively for maximum performance. Hence, for this particular multi-programmed workload, we required at the most 68 globally shared FPU to achieve performance comparable to a 256-core system with 256 FPU. This provided us with area and power savings worth 188 FPU. We also considered the worst case, where we ran four instances of the `blackscholes` benchmark that required 24 FPU for maximum performance. Even for this case we required at the most 96 globally shared FPU to achieve performance comparable to a 256-core system with 256 FPU.

Contemporary techniques for sharing of EUs would need us to divide the manycore system into smaller partitions with localized sharing of EUs within each partition as contemporary strategies do not scale well with core count. This approach, though feasible, could limit the performance of applications that need more EUs than those available in the partition. Similarly, this partitioning approach could lead to the use of over-provisioned partitions for applications that need less number of EUs than the number of EUs that are available in the partition. As a result, we would not be able to maximize the savings in area and power.

A global sharing of EUs, whereby each application has access to the optimal number of EUs, could be used to maximize the area and power savings while sustaining application performance in manycore systems. However, implementing a global sharing of EUs in manycore systems with hundreds of cores is very challenging. First, every processor core must have a very low-latency access to the shared EUs. Second, the manycore system must have a low-overhead mechanism for managing contention among cores that simultaneously access the shared EUs. Implementing a NoC us-

ing electrical links that meets these requirements is not feasible because of either high energy cost (long electrical wires) or high latency (multi-hop short electrical wires). Silicon-photonic links have been proposed as a possible replacement to electrical links for on-chip communication (Joshi et al., 2009; Kirman and Martínez, 2010; Pan et al., 2009; A. Shacham, K. Bergman and L. P. Carloni, 2007; Morris and Kodi, 2010; Vantrease et al., 2008). The reason is that they can provide an order of magnitude higher bandwidth density, and low fixed latency and low energy consumption for global communication. These properties make silicon-photonic links ideal components to implement a NoC that enables efficient global sharing of EUs.

We propose a novel manycore system architecture that leverages silicon-photonic link technology to implement a global sharing of EUs for saving manycore area and power while sustaining application performance. In particular, we explore a 256-core system architecture, which utilizes a globally shared unit called Execution Unit Cloud (EUCloud) that implements the costly x87-compliant FP units (Intel Corp., 2013). Our EUCloud-based manycore system just integrates the required number of FPUs (96) to match performance as compared to a typical manycore architecture composed of the whole set of FPUs (256). The savings in terms of area and power due to global sharing of EUs are harnessed to increase core complexity or on-chip cache sizes, thereby enabling higher ILP per core. We show that for a given power and area budget, our EUCloud-based manycore architecture has better overall manycore performance.

To enable energy-efficient global sharing of execution units:

- We propose a novel EUCloud-based manycore system that implements efficient global sharing of computing resources (FPUs). Our proposal just needs a subset of all manycore’s FPUs to achieve maximum application performance, thus leading to important on-chip area and power savings, or to improve manycore

performance while staying within the area and power budget. We modify the core architecture to enable very fast remote FP execution at EUCloud.

- We advocate the use of silicon-photonic link technology in our EUCloud-based manycore processor architecture to achieve efficient global sharing of execution units. The silicon-photonic link technology is used to build a NoC architecture that provides energy-efficient and low-latency communication between both cores and EUCloud.
- We propose a distributed workload allocation mechanism for managing the EUCloud’s resources through the high speed tuning and detuning features of micro rings in the silicon-photonic links. The EU workloads are allocated fast and fairly among subsets of EUCloud’s resources in either a time-multiplexing fashion or a round-robin fashion.

In this chapter, Section 6.2 shows the details of our target system and our silicon-photonic link technology. In Section 6.3, we provide a detailed discussion of our proposed EUCloud architecture, the modified core architecture and the NoC architecture that supports this EUCloud-based manycore architecture. This is followed by evaluation of our proposed EUCloud-based manycore architecture in Section 6.4. Section 6.5 summarizes our analysis.

6.2 Target System

Our target baseline system is a 256-core architecture fabricated using 16 nm CMOS technology. To enable efficient communication between cores and memory controllers (MCs) we use a silicon-photonic multi-bus NoC (further details in Section 6.3.2). We assume a 3D stacked manycore architecture where the 256 cores and MCs are designed using traditional CMOS process on a ‘logic layer’ and the silicon-photonic

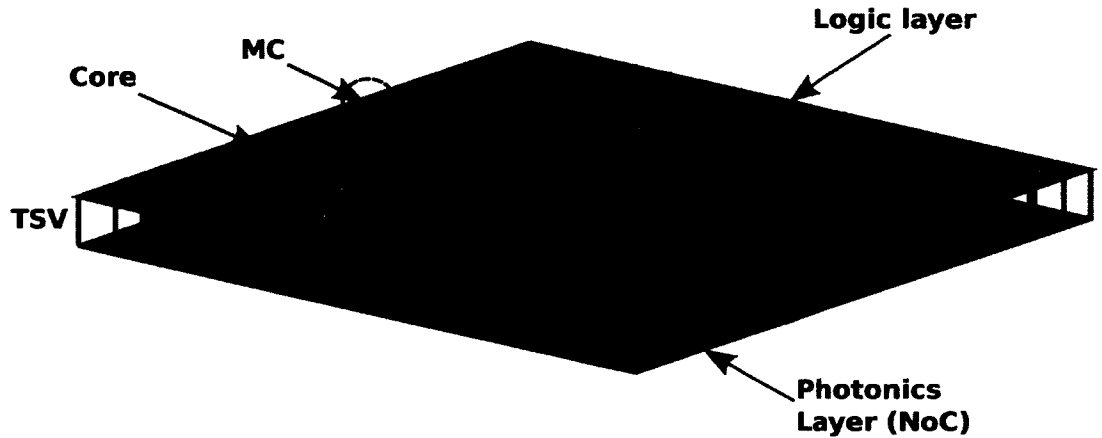


Figure 6-2: 3D stacked manycore system – cores and MCs are on logic layer, while NoC is on the silicon-photonic layer.

NoC is designed on a separate ‘photonic layer’. This 3D approach provides us with the flexibility to choose the materials and structures that minimize the losses in the photonic devices. A high-level representation of the 3D stacked manycore is illustrated in Figure 6-2. The logic and the photonic layers are connected using through-silicon-vias (TSVs). A separate off-chip photonic network is used for communication between the memory controllers and PIDRAM chips (Beamer et al., 2010).

6.2.1 Logic Layer

The logic layer contains the cores and MCs of the target manycore system. The main micro-architectural parameters of the components on the logic layer are shown in Table 6.1. Our cores implement a 2-way in-order issue, out-of-order execution superscalar pipeline that is based on the scaled up version of a Pentium Pro processor (Gwenapp, 1995) with a contemporary x87 FP execution environment based on Intel Nehalem (Kurd et al., 2008). The implementation of the FP units in our target system meets the IEEE Standard for Binary Floating-Point Arithmetic (IEEE-754) (IEE, 1985). Single and double precision FP registers are 32 bits and 64 bits in

size, respectively. Each core includes a set of EUs that consists of one AGU, one BRU, two IntALU, one Imuldiv, and two large FPUs: FPalu, for simple FP arithmetic and logic operations; and Fpmuldiv, that implements multiplication, division, sqrt and trigonometric operations. Moreover, data transfers of FP data between FP registers and to/from memory, and execution of FP instructions to control state and modes of x87 execution are performed by the FPmov unit. As shown in Table 6.1, the EUs are clustered around five execution ports for a maximum throughput of five different instructions per clock cycle if they go to different ports.

We use McPAT v0.8 tool (Li et al., 2009) to estimate power and area of the target manycore system assuming an operating voltage of 0.8 V and a frequency of 1 GHz. We scaled down technology from 22 nm (minimum supported by McPAT v0.8) to 16 nm. As a result, our manycore architecture has a total on-chip area of 349 mm² and meets a power budget of 180 Watts. Using McPAT tool, we show in Figure 6-3 a breakdown of the total area and static power per core in our target system to understand potential savings. As we can observe, the two large FPUs (FPalu and Fpmuldiv units) occupy around 22% of core area and dissipate 16% of static power, whereas

Table 6.1: Architectural parameters of the 256-core system

Micro-architecture Configuration	
Pipeline	2-way superscalar, OoO exec.
Technology	16 nm, 0.8 Volts, 1 GHz
Instruction Queue	64 entries
Reorder Buffer	128 entries
Reservation Stations	36 entries
Branch Predictor	256-entry BTB, 512-entry bimodal
Execution Units	Port0: BRU, IntALU Port1: IntALU, Fpmuldiv Port2: FPmov, FPalu Port3: AGU, Load Port4: AGU, Store
Private L1 I/D-Cache	4-way 32 KB @ 2 ns
Private Unified L2 Cache	8-way 256 KB @ 6 ns
Cache Coherence	Directory based MESI (Papamarcos and Patel, 1984)
Memory	16 MCs + 16 PIDRAM @ 50 ns

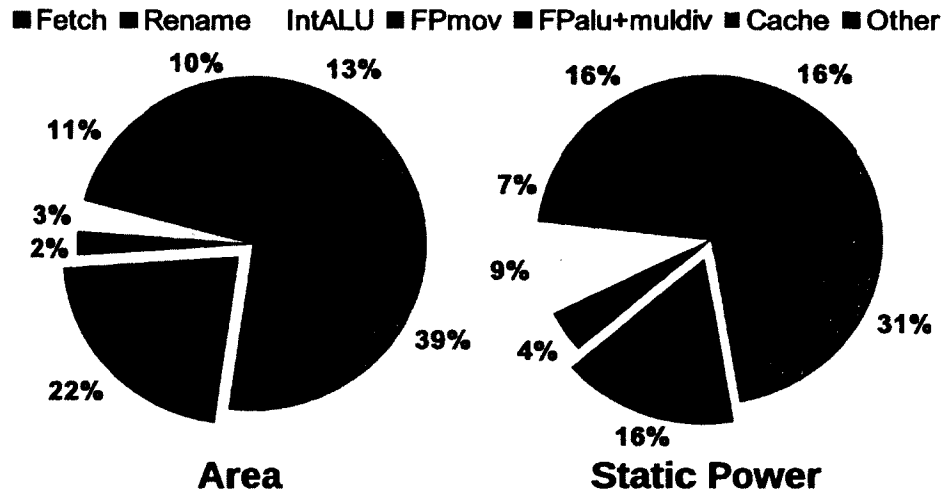


Figure 6-3: Breakdown of area and static power of each core

the FPmov unit occupies 2% of the total core area and dissipates 4% static power. It is worth noting that our proposed EUCloud-based manycore system explained in Section 6.3 integrates the two larger FPUs in the EUCloud for global sharing, whereas we implement a private FPmov unit per core due to its low cost and very low latency (1 or 2 clock cycles). Moreover, as FPmov is also used to modify core-dependent status referring to FPU execution modes and FP data movements that can involve access to main memory, an EUCloud composed of FPmov units would unnecessarily increase manycore complexity.

6.2.2 Photonic Layer

The photonic layer integrates all silicon-photonics devices required to implement the NoC of the manycore system. 3D integration provides us with the flexibility to choose the material for designing the photonic devices (SiN (Hosseini et al., 2009; Biberman et al., 2011) or polycrystalline silicon (Preston et al., 2009; Preston and Lipson, 2009)). The photonic layer and the logic layer are interfaced using Through-Silicon Vias (TSVs). In terms of the silicon-photonics link, we consider double-ring filters and

a 4 THz free-spectral range, which enables up to 128λ modulated at 10 Gb/s on each waveguide (64λ in each direction). The latency of a global photonic link is assumed to be 3 cycles (1 cycle in flight and 1 cycle each for E/O and O/E conversion that happens in TSV in the 3D stacked design). For our analysis, we use the projected silicon-photonic link energy cost in Table 3.2 and projected silicon-photonic device losses in Table 3.3. However, the waveguide loss is assumed to be 0.1~1 dB/cm since we are considering an aggressive design of silicon-photonic technology, in which the device level design has resolved the laser power issue.

6.3 The EUCloud-based Manycore System

The processor cores of our novel manycore architecture rely on the EUCloud for FP execution. The EUCloud contains the x87-compliant FPUs that are globally shared among all processor cores, and an arbitration mechanism is used to fairly process FP instruction execution requests from cores. After execution, the EUCloud sends the results back to the requesting cores. As explained in Section 6.2, the EUCloud maintains two types of FP units (`FPalu` and `FPmuldiv`).

6.3.1 Processor Core Design

Figure 6-4 illustrates all major components of the processor cores of our novel EUCloud-based manycore system. As we can observe, the core architecture resembles a typical OoO pipeline with a few modifications to the back-end (highlighted in dark gray color).

The modifications to the back-end of the pipeline are minimal since one just needs to decouple the part of reservation station (RS) that is devoted to FP instructions (see EUCloud RS in Figure 6-4) and adapt the issue logic to send the FP instruction

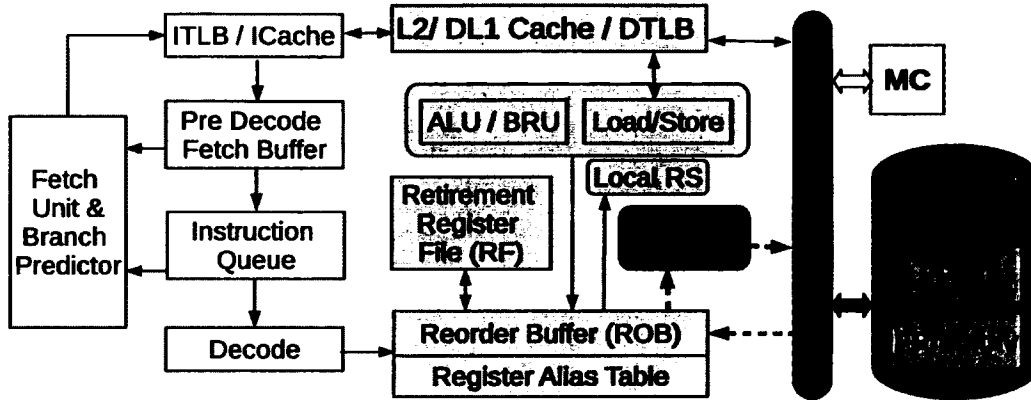


Figure 6-4: Abstract view of the main components of a processor core – Dark gray color represents the required modifications to access EUCloud.

to the EUCloud. The core’s network interface is in charge of encapsulating every FP instruction in a packet that is then sent through the silicon-photonic NoC to the EUCloud. Once the result is computed at EUCloud, the EUCloud returns the result to the requesting core, which in turn updates the ReOrder Buffer (ROB) and can enable new instructions to be issued for execution.

More specifically, when a new FP instruction is decoded, the allocation stage, which is responsible for register renaming and instruction dispatch is started. The renaming step, which removes all false dependencies (name dependencies) for the in-flight instructions, does not need any modification when using the EUCloud as we assume a renaming strategy through the ROB and use of a register alias table (RAT). The RAT maps every logical register to its architectural state (i.e., if the register has a speculative value at ROB, or a committed value at the register file). In this way, if an FP instruction has been sent to the EUCloud, the RAT indicates that its logical FP register that stores the result of the FP instruction is in a particular ROB entry. The dispatch stage reserves resource entries in both the ROB and two reservation stations (RSs): Local RS, for all instructions that will be executed locally; EUCloud

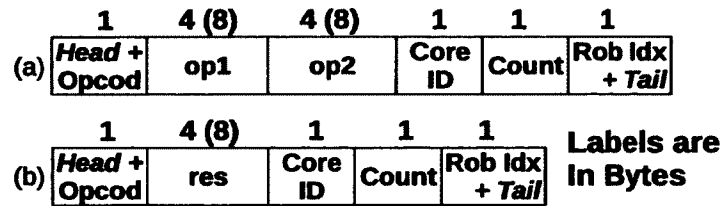


Figure 6-5: Packets required to access EUCloud – (a) Packet from core to EUCloud. (b) Packet from EUCloud to core. We consider both single-precision and double precision (in brackets) FPU instructions.

RS for the FP instructions. The pipeline is stalled at this stage if no entry is found at ROB or at EUCloud RS.

After the allocation stage, the issue logic, that checks availability of all source operands of all in-flight instruction sitting in the RS, begins its operation. As mentioned above, we allocate the reservation stations to the execution unit depending on whether it is placed locally (see *Local RS*) or in EUCloud (*EUCloud RS*). This issue logic also has access to the core’s network interface to send FP instructions to the EUCloud. To send a new FP instruction to the EUCloud, the network interface encapsulates the FP instruction in a packet which is 12 bytes in length for single precision operands or 20 bytes in length for double precision operands. As shown in Figure 6-5(a), this packet includes one byte for the head (1 bit) and the operation code (7 bits), two 4-byte source operands of the FP instruction (or two 8-byte operands in double precision). Assuming a ROB with 128 entries and a 256-core target system, two bytes are needed for the ROB entry id (ROB *idx* in the Figure) and the core identifier. The unused extra bit in the ROB *idx* field specifies the tail of the packet. In addition, we add one extra byte (*Count*) to solve any inconsistencies in the pipeline that could arise when a misspeculation (e.g., branch misprediction or an exception), requires all in-flight instructions to be flushed from the core pipeline while instructions generated by the core (earlier in time) are being executed in the EUCloud. The packet

stores the least significant byte of a core's performance counter that stores the current number of mispredictions. The EUCloud response packet includes this byte for the number of mispredictions (copied over from the number of mispredictions listed in the packet sent to the EUCloud – see **Count** in Figure 6-5(a)). If the number of mispredictions in the response packet from the EUCloud is different than the misprediction value stored in the 8-bit wrap-around counter in the processor core, then the response packet is ignored. A similar solution can be applied to solve the problem of context switches, where OS intervenes to switch current application (or thread) to another and there are in-flight FP instructions sent to EUCloud. In this case, before the context switch all in-flight instructions are flushed from the core pipeline and the above core's performance counter is incremented. Then, the processor core must simply ignore all packets received from EUCloud with different **Count** values to that held in the processor's counter.

In Figure 6-5(b) we illustrate the response packet from EUCloud. The first byte of this packet stores the head bit and other seven extra bits – we include the opcode of the instruction although it is not strictly necessary. In addition to the **Count** field, the response packet uses 4 bytes or 8 bytes to store the result of the remote FP execution of single precision or double precision operands, respectively. It also uses 1 byte each for **ROB index** and **Core id**. That results in a packet total size of 8 bytes (12 bytes in double precision). Once the result is obtained from the response packet, it is stored in the **ROB** and the wake-up logic is responsible for ensuring that dependent instructions stored at **EUCloud RS** are activated. The remaining commit stage of the pipeline, which checks if the older instructions in the **ROB** are completed, releases resources and updates the architectural stage of the pipeline (values in the register file), does not require any modifications.

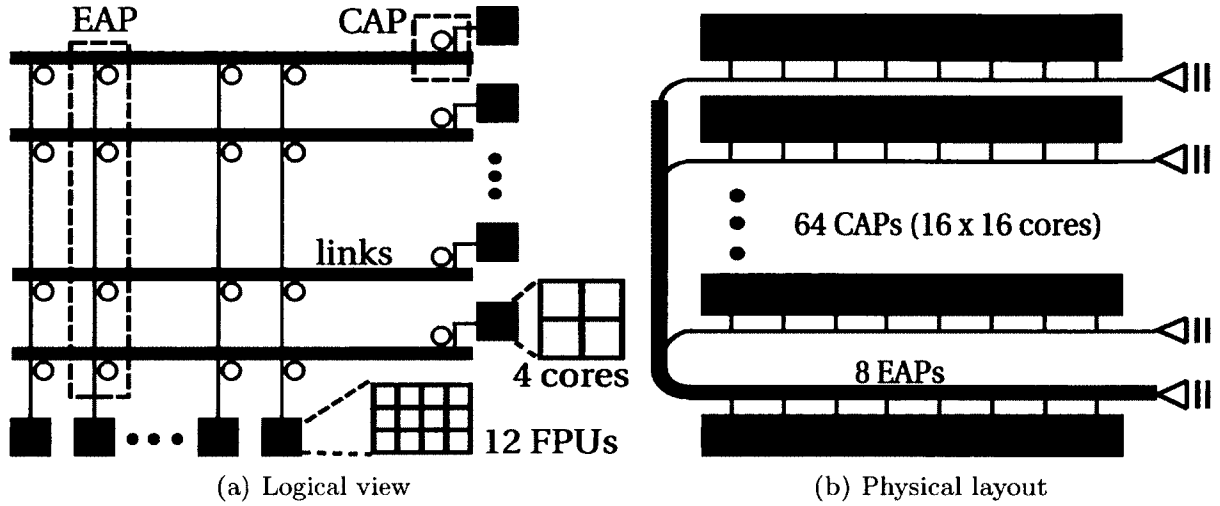


Figure 6-6: The silicon-photonic between cores and EUCloud – (a) logical view (b) physical view. The 256 cores and 96 FPUs access the NoC through 64 CAPs and 8 EAP, respectively.

6.3.2 Core-to-EUCloud NoC Architecture

In our 3D manycore target system explained in Section 6.2, the NoC is integrated in the silicon-photonic layer and interconnects cores and MCs. We utilize an efficient silicon-photonic multi-bus NoC (P-mem) that is made up of 8 buses where each bus is shared by a different group of 32 processor cores (one bus per every couple of rows of cores). Due to the very low cost of this NoC in terms of power and on-chip area discussed below, our proposed EUCloud-based manycore system implements another silicon-photonic multi-bus NoC for the communication between cores and EUCloud (P-EUCloud). We will focus on the description of this NoC as it is comparatively more complex and expensive than the P-mem NoC.

Figure 6.6(a) and Figure 6.6(b) show the logical view and physical layout of our P-EUCloud NoC. According to our analysis in Section 6.4.2, the 256-core system just requires 96 FPUs to sustain application performance. Therefore, the P-EUCloud must provide access between 256 cores and 96 FPUs at EUCloud. We optimize the use of

silicon-photonics resources by relying on concentration through different access points (APs). In particular, we need 64 APs for the 256 cores (CAP) and 8 APs for the FPU (EAP). Each CAP is used by 4 cores to access the silicon-photonics NoC, while each EAP serves as an AP to 12 FPUs.

The core-to-EUCloud packets have no predetermined destinations as any of the different EAPs can accept any request for FP execution from CAPs. In Section 6.3.5, we give a detailed description of our timing-based non-blocking technique for fairly distributing the core-to-EU packets among EAPs. The zero-load latency of core-to-EUCloud traffic is 3 cycles (1 for E-O conversion, 1 for transmission, 1 for O-E conversion). The EUCloud-to-core packets have predetermined destination and multiple EAPs may want to send EUCloud-to-core packets to a CAP at the same time. Hence, we use a token-based protocol, where an EAP can send a packet to a CAP by grabbing a token (Vantrease et al., 2008). The token-based protocol adds one extra cycle to the EUCloud-to-core packet transmission. Therefore the zero-load latency of EUCloud-to-core traffic is 4 cycles. As a result, each CAP has one dedicated one-to-many silicon-photonics channel for CAP-to-EAP communication and dedicated many-to-one silicon-photonics channel for EAP-to-CAP communication. In terms of the physical implementation, multiple silicon-photonics channels may be mapped to each silicon-photonics waveguide as shown in Figure 6.6(b).

The width of the channels in the P-EUCloud NoC is defined to be equal to the flit size. Each one of the 64 CAP-to-EUCloud channels has a width of 20 bytes, while each one of 64 EUCloud-to-CAP channels has width of 12 bytes. As each wavelength can be modulated at 10 Gb/s and the cores operate at 1 GHz, the CAP-to-EUCloud channel and the EUCloud-to-CAP channel have a bandwidth of 20 GB/s and 12 GB/s, respectively. So, the total bandwidth of the entire P-EUCloud NoC is 2.56 TB/s which represents eight times more bandwidth than needed by the P-mem NoC

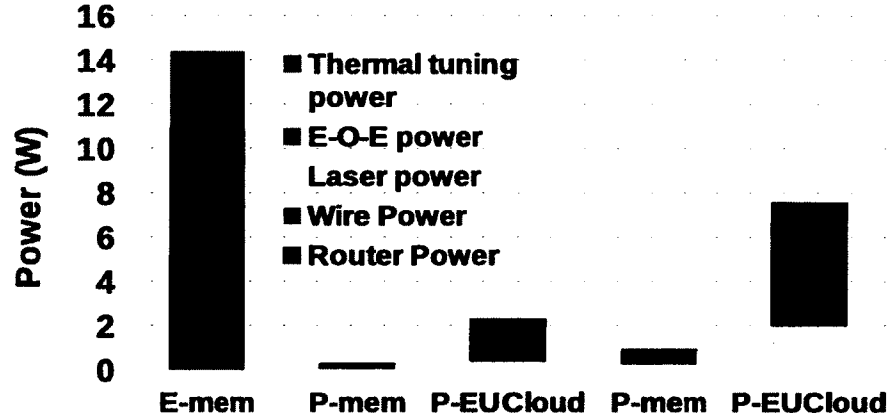


Figure 6-7: NoC power for EUCloud – ‘P-mem’ is the silicon-photonic NoC between cores and memory controllers, and ‘P-EUCloud’ is the silicon-photonic NoC between core and EUCloud. ‘E-mem’ is an electrical mesh NoC for between cores and memory controllers, shown here as a reference. For silicon-photonic NoCs, we consider both aggressive and conservative silicon-photonic link energy cost in Table 3.2 and device losses in Table 3.3.

according to our analysis in Section 6.4.2.

The power and area overhead of our proposed NoC for EUCloud depends on the projected silicon-photonic technology (see Section 6.2.2 for further details). In our 256-core target system, the 2.56 TB/s needs a total of 32 waveguides where each waveguide can carry 64λ . The NoC needs 18,432 TSVs (that require 3.6 mm^2) for connection between logic layer and photonic layer, and a total number of 18,432 rings (CAP+EAP), among which 11,008 rings are used as filters at receiver side and 7,424 rings are used as modulators at the sender side (CAP+EAP) of the silicon-photonic NoC. The NoC area is 6.4 mm^2 , which is just 1.83% of the 349 mm^2 available in the photonic layer. The E-O-E conversion power, laser power, thermal tuning power and area of the photonic devices for both aggressive and conservative photonic devices is shown in Figure 6-7. As we can see in the Figure, as expected due to the larger bandwidth, P-EUCloud is more expensive than the P-mem NoC. Moreover, it

is worth noting that even an optimized 4-way concentrated 2D-mesh electrical NoC for the memory-related traffic (**E-mem**) is more expensive than both aggressive and conservative designs for the P-EUCloud silicon-photonic NoCs.

6.3.3 Execution in EUCloud

According to previous section, the zero-load latencies of core-to-EUCloud and EUCloud-to-core traffics are 3 and 4 clock cycles respectively. In this section, we discuss how we design the EUCloud system to compensate for these extra clock cycles during FP execution such that we can still achieve very efficient remote execution of FP operations in EUCloud. To simplify the explanation, we assume an ideal implementation of the silicon-photonic NoC in which there are no extra clock cycles due to serialization or congestion when packets compete for network resources. We have however included the effect of serialization and congestion while evaluating our proposed EUCloud-based manycore architecture.

The impact on application performance due to remote FP execution in our proposed EUCloud architecture is directly related to the latency of the FPU in the EUCloud. The shorter the latency of an FP operation the higher the performance degradation when compared to local FP execution.

The performance of the EUCloud-based manycore system also depends on whether the FP unit that is shared at EUCloud is pipelined or not. In case the FP unit is pipelined, a new FP operation can theoretically be initiated at every clock cycle. To take advantage of pipelined FP units, the processor cores in our proposed manycore system are designed to send an FP instruction to the EUCloud as soon as the instruction's operands are available for execution (i.e., they are already stored at register file and/or in ROB just returned from the EUCloud). Similarly, both the NoC and the EUCloud

	1	2	3	4	5	6	7	8	9	10	11	12	13
	TSV	NoC	TSV	Ex0	Ex1	Ex2	tok	TSV	NoC	TSV			
		TSV	NoC	TSV	Ex0	Ex1	Ex2	tok	TSV	NoC	TSV		
			TSV	NoC	TSV	Ex0	Ex1	Ex2	tok	TSV	NoC	TSV	
				TSV	NoC	TSV	Ex0	Ex1	Ex2	tok	TSV	NoC	TSV

(a)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	TSV	NoC	TSV	Ex0	Ex1	Ex2	tok	TSV	NoC	TSV									
										TSV	NoC	TSV	Ex0	Ex1	Ex2	tok	TSV	NoC	TSV

(b)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	TSV	NoC	TSV	Ex0	Ex1	Ex2	tok	TSV	NoC	TSV									
		TSV	NoC	TSV			Ex0	Ex1	Ex2	tok	TSV	NoC	TSV						
			TSV	NoC	TSV					Ex0	Ex1	Ex2	tok	TSV	NoC	TSV			
				TSV	NoC	TSV							Ex0	Ex1	Ex2	tok	TSV	NoC	TSV

(c)

Figure 6-8: The execution stage using the pipelined 3-cycle FPalu unit at EUCloud – (a) No instruction dependencies. (b) Consecutive instruction dependencies. (c) Optimized implementation for consecutive instruction dependencies.

are designed to rapidly send and process the instruction, respectively.

Figure 6-8 depicts different scenarios of the pipelined execution for the FPalu unit assuming, for ease of explanation, that a processor core can send one instruction at a time to the EUCloud. Note that, our proposed core architecture can actually issue up to two different FP instructions to EUCloud per clock cycle through ports 1 and 2 (see Table 6.1). As we can observe in Figure 6.8(a), if there are no dependencies among consecutive FP instructions, a processor core can send instructions to the EUCloud consecutively. The more instructions we need to consecutively execute the lower performance degradation due to the remote execution at EUCloud. For instance, for 2 non-dependent consecutive instructions the performance overhead is

175% (4 cycles for local execution vs 11 cycles for EUCloud execution), but if there are 4 consecutive instructions it drops to 116% (6 cycles for local execution vs 13 cycles for EUCloud execution), etc. However, if there are dependencies among FP instructions then it could lead to significant performance degradation. For example, (as we illustrate in Figure 6.8(b)) instruction 2 depends on the result of instruction 1, and so instruction 2 cannot be sent to EUCloud until the core receives the result for instruction 1 from EUCloud, thereby the processor core serializes their execution leading to a very high performance degradation. The two dependent instructions require 19 cycles for execution compared to 6 cycles for local execution (slowdown of 216%). We propose an optimized version of EUCloud (described in Section 6.3.4) in which the execution of dependent FP operations is serialized at EUCloud rather than at processor core. This is shown in Figure 6.8(c), where for 2 dependent consecutive instructions the performance overhead drops to 116% (13 cycles vs. 6 cycles for local execution).

6.3.4 Execution in EUCloud using Instruction Bundling

To improve performance of our EUCloud system, we propose to serialize execution of dependent FP instructions remotely in the EUCloud. We refer to this modified EUCloud system as EUCloudOpt. To do that, a processor core simply groups together dependent instructions from EUCloud RS, and encapsulates them in the same NoC packet that is sent to the EUCloud for execution. We refer to this larger packet as a *bundle*. Figure 6-9 shows such a bundle containing three different FP instructions. As discussed in Section 6.3.2, for maximum efficiency, the flit size of our silicon-photonics NoC has been configured using the largest packet (20 bytes wide) supported by EUCloud. In this way, a bundle is fragmented into multiples flits that are sent in consecutive clock cycles to EUCloud – three flits are needed in the example of

Figure 6-9). It is worth noting that bundles can help lower the congestion of the NoC as a compacted representation of dependent instructions can be used. This is also illustrated in Figure 6-9, where the second and third instructions do not need to specify both operands like first instruction. The reason is that at least one of their two operands will be calculated by another instruction in the bundle. To determine the instruction that calculates the operand, we dedicate a new 1-byte field in every flit of the packet (see `Inst ID`) that stores its position in the bundle. So, we can reference up to 256 different instructions in a bundle. If the other operand must also be calculated, we use `op` field in the flit to specify the id of the instruction calculating that operand. In the head of the packet, we also include information to be able to reconstruct the bundle at EUCloud – e.g., the number of instructions stored in the bundle.

To support the execution of bundles, the EUCloudOpt requires – 1) semantic logic that can determine the semantic of the bundle (i.e., how many instructions are encapsulated, which operands are not available, etc.), 2) bypass logic that propagates the calculated results to the dependent instructions in the bundle and 3) reservation stations (RS) to store the dependent instructions and information about availability of operands. As our EUCloud supports two types of FPUs (`FPalu` and `FPmuldiv`), a bundle of dependent instructions can require both kind of FPUs – for example, an FP addition that depends on the result of an FP multiplication. The bypass logic therefore needs to be able to route data between the two types of FPUs.

Figure 6-9 shows a schematic view of the architecture for execution of bundles. To use this architecture, a processor core creates and sends a bundle in multiple flits over the silicon-photonics NoC towards EUCloudOpt. Each flit is converted to the electrical medium, and stored as input to the semantic logic that is in charge of identifying the type of bundle. If all instructions in the bundle can be executed in the same FPU

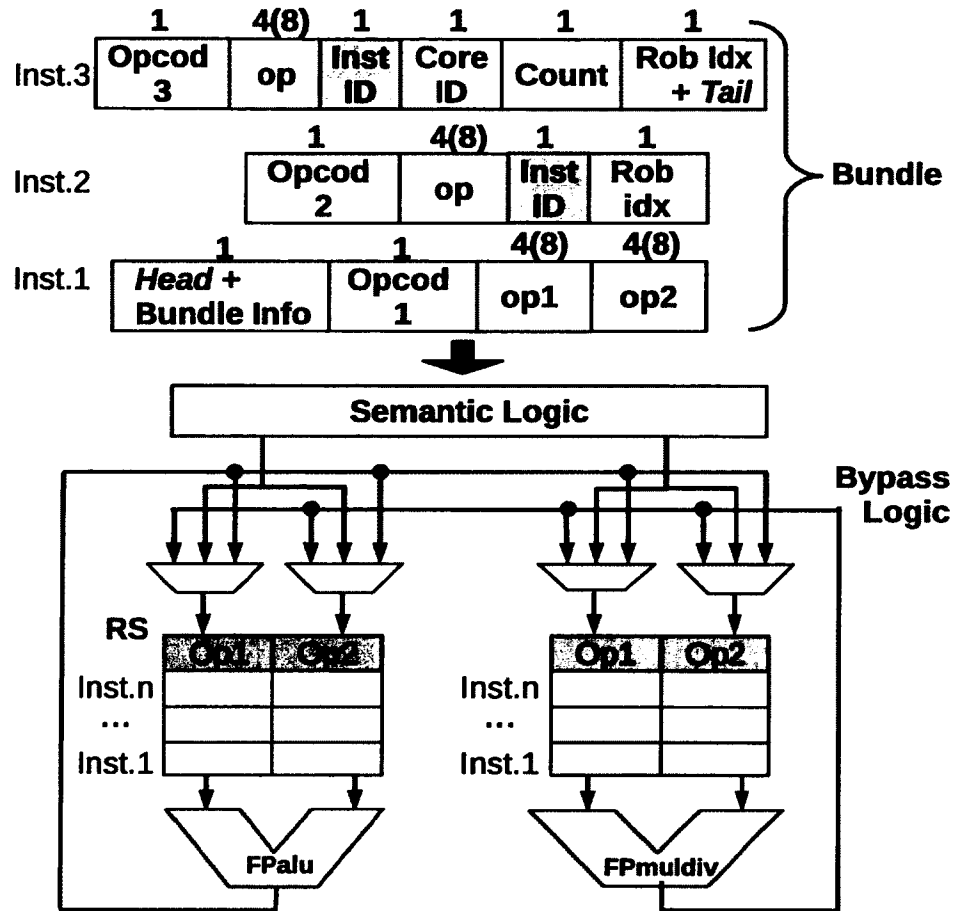


Figure 6-9: Extensions to support bundles in EUCloudOpt – Bundle of three FP instructions is shown. Labels are in bytes.

(the head of the packet indicates the number of FPUs needed), then the semantic logic extracts all the instructions in the bundle and stores them in the target RS for execution. The instructions are executed one after another and the bypass logic routes the results to the waiting FP instructions sitting at the particular RS. Similar to a core pipeline with RS, we allow OoO execution among the instructions in the bundle to avoid stalling the FP execution. After execution of all instructions in the bundle, all results are stored in different packets that are sent back to the requesting processor core as explained for EUCloud (see Figure 6-5(b)). In case the bundle contains FP instructions that require a pair of FPUs (e.g., FP addition that depends

on an FP multiplication), the semantic logic is in charge of distributing instructions between both RSs. Finally, the bypass logic is used for routing the new results from FPUs to the dependent instructions in either of the two RSs.

According to our study (see Section 6.4.3), we start seeing diminishing returns in system performance for bundle sizes beyond 8 instructions/bundle. The semantic logic will need a single cycle to manage a bundle of this size. Moreover, we determine that an RS structure should store 5 bundles (40 instructions). Assuming 18 Bytes per instruction required at every RS entry in worst-case scenario (double-precision operands) – opcode, op1, op2, and rob index fields of the packet sent to EUCloud (see Figure 6-5), each RS structure accounts for 720 Bytes. Since, according to our analysis of Section 6.4.2, the minimum number pairs of FPUs is 96 for EUCloudOpt the overhead of all RS structures for the entire EUCloud-based manycore is less than 70 KB.

6.3.5 Workload Allocation

We chose a physically distributed placement of EUCloud to ease the thermal dissipation in the EUCloud. However, this physically distributed placement needs additional mechanisms to distribute the workload among the EUs in the EUCloud, i.e, we need to fairly allocate packets from 256 cores (64 CAPs) among the 96 FPUs (8 EAPs). We propose two EUCloud workload allocation mechanisms – one for EUCloud, and another for EUCloudOpt.

We use a time-multiplexing based workload allocation for the EUCloud. Since the core-to-EUCloud packets have a fixed packet size of 20 bytes and the silicon-photonics channels from each CAP has 16λ , one core-to-EUCloud packet can be transmitted from one CAP to one EAP every cycle. Our protocol changes the destination EAP of

cycle	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C0	E0	E1		E3		E5	E6	E7	E0		E2	E3	E4	E5	E6	E7
C1	E1		E3	E4	E5		E7	E0	E1		E3	E4		E6	E7	E0
⋮																
C63	E7	E0		E2	E3	E4	E5	E6	E7		E1		E3	E4	E5	E6

(a)

cycle	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C0					E1	E1	E1	E1					E3			
C1	E1			E2	E2	E2	E2	E2				E3	E3	E3	E3	E3
⋮																
C63	E7	E7	E7	E7	E7	E7	E7	E7	E7						E0	E0

(b)

Figure 6-10: The workload allocation among EUs – (a) EUCloud, (b) EUCloudOpt. C0~C63 represent 64 CAPs and E0~E7 represent 8 EAPs. E_i in the row for C_j at time T means the i th EAP has been assigned as the receiver for packets from the j th CAP. A blank box in the row for C_j means EAPs are transferring the responsibility and the j th CAP needs stop sending packets for one cycle. The box in grey means the assigned EAP receives a packet or one flit of a packet at that cycle.

every CAP every cycle. Figure 6.10(a) shows an example of EU workload allocation using time-multiplexing based method. C0~C63 represent 64 CAPs and E0~E7 represent 8 EAPs. At cycle 0, E1 is responsible for receiving packets from C1, C9, ..., C49. At cycle 1, E1 takes over the responsibility from E0 and becomes responsible for receiving packets from C0, C8, ..., C48. In this way, at any time, each EAP is responsible for receiving packets from eight CAPs and the responsibility is shifted among the EAP over the time. The shifting of responsibilities ensures that there are no stalls if a CAP wants to send a packet to the EUCloud every cycle.

In the EUCloudOpt, the core-to-EUCloud packet size varies from 20 bytes to 97 bytes (an 8-instruction bundle assuming double-precision operands). That means each packet can take up to 5 cycles to complete the transmission through the silicon-

photonic link (five 20-byte flits). Therefore, it is no longer safe to shift the responsibility at every cycle, i.e. an EAP should not switch the responsibility until it finishes receiving the entire packets that has variable packet length. Figure 6.10(b) shows an example of workload allocation using a round-robin based workload allocation to maintain the fairness in the EUCloudOpt. The responsibility for a particular CAP is shifted to another EAP only when the currently assigned EAP receives one complete packet from that CAP. For example, the responsibility for C63 is shifted from E7 to E0 when E7 receives the entire packet from C63. After transmitting the packet, C63 needs to wait for one cycle as E7 needs one cycle to convert data from optical medium to electrical medium, check the completeness of received packet, and shift the responsibility to another EU.

The round-robin workload allocation approach ensures a fair allocation of EU workloads for packets with variable sizes. It needs some extra wires for shifting responsibilities among EAPs, but the overhead is minimal since these extra wires are only single bit wide and comparatively short. The EAP has the capability to switch the responsibility in advance if it determines that all its associated FPU's are busy and there are enough received packets that are waiting to be processed.

6.4 Evaluation

6.4.1 Simulation Methodology

To evaluate our proposed EUCloud-based manycore architecture, we used Sniper 5.0 (Carlson et al., 2011) full-system simulator. We updated the simulator to include a cycle-level core model, the EUCloud architecture and the EUCloudOpt architecture. We configured the Sniper simulator to simulate the following three different architectures of our 256-core target system – 1) **Baseline** architecture where each

core contains a dedicated FPU, 2) EUCloud architecture where cores do not have dedicated FPUs and send individual FPU instructions to the cloud, and 3) EUCloudOpt architecture where cores do not have dedicated FPUs and send bundles of FPU instructions to the cloud. We used multi-threaded applications from SPLASH-2 (Woo et al., 1995) and PARSEC (Bienia et al., 2008) with `sim_medium` inputs to design multi-programmed workloads, where each workload was composed of four instances of 64-thread benchmarks. We use their parallel phases as region of interest (ROI) to always evaluate the maximum aggregated demand of FPUs (there is no single-threaded sequential phase). Each 64-thread benchmark was mapped to a 64-core quadrant in the target system. Note that, the conclusions of our evaluation remains valid for smaller sizes of partitions and number of threads per application (e.g., 16 partitions and 32-thread applications) because we obtained similar ILP per core values as 32 and 16 thread-applications. The reason for this is that the chosen data inputs sets allow the benchmarks to scale well from 16 threads to 64 threads. Given that execution time of applications is different, the total execution time of our multi-programmed workload was set to be equal to the largest value out of the four execution times of the four applications. During the simulation period, the other three applications were restarted (one or more times) whenever they finished execution to ensure there is NoC/cache contention at all times by using the `--sim-end=last-restart` option implemented in Sniper. Since our EUCloud system meets the x87 execution environment, we activated the `-march=pentiumpro` flag when compiling our benchmarks with `gcc` compiler. This generates binary codes that solely rely on x87 for FP operations – the only x86 extensions supported by PentiumPro processor.

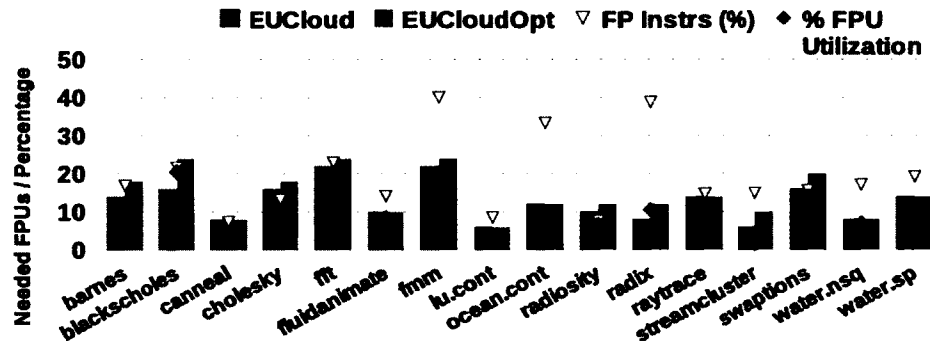


Figure 6-11: Required number of FPUs for for each 64-thread application – For each experiment, we utilize a single 64-core partition of the manycore system when applying global sharing through EUCloud and EUCloudOp

6.4.2 EUCloud and NoC Design

In this section we explain the process of determining the number of FPUs required in the EUCloud to sustain application performance and also validate choice of our NoC bandwidth. We simulated each 64-thread application in a 64-core partition of the EUCloud-based 256-core system for FPU counts ranging from 4 to 64 (the maximum number of FPUs in the nominal case for a 64-core partition). Our EUCloud integrates two types of FPUs (FPalu and FPMuldiv). We observed from our simulations that the demand for both kind of units is comparable across all applications. Hence, in the rest of this section, we use the generic FPU term to refer to a pair of FPalu and FPMuldiv units.

Figure 6-11 shows the minimum number of FPUs required in EUCloud and EUCloudOpt for each benchmark. It also shows the percentage of FP instructions and percentage of FPU utilization in the Baseline architecture. As we can see, the maximum percentage of FP instructions is 40% and the maximum utilization of core's FPUs is 20%. Similarly, EUCloud and EUCloudOpt requires a maximum of 22 and 24 FPUs, respectively, to achieve maximum application performance. This creates an

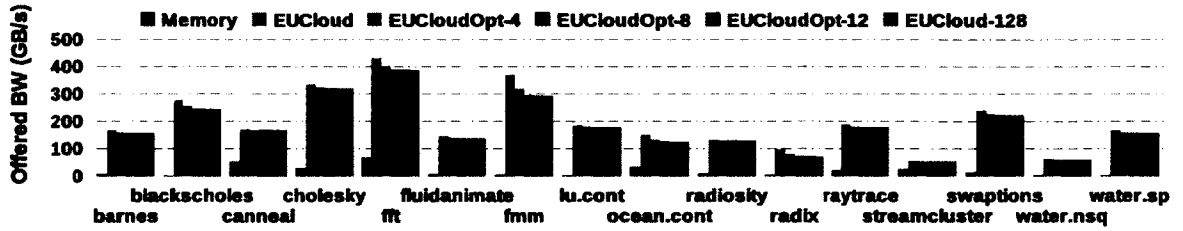


Figure 6-12: Offered bandwidth for each 64-thread application – For each experiment, we utilize a single 64-core partition of the manycore system. We distinguish between memory-related bandwidth (**Memory**) and bandwidth consumed when accessing EUCloud (**EUCloud**). We also show EUCloudOpt using different sizes of bundles (**EUCloudOpt-X**): 4, 8, 12 and 128 (ideal case).

opportunity to use less FPU units than the **Baseline** case with global FPU sharing. It is worth noting that EUCloudOpt requires more FPUs than EUCloud as the former executes FP instructions at a higher rate due to instruction bundling. As a result, a total number of 88 and 96 FPUs compared to the 256 FPUs in the **Baseline** system are actually needed in EUCloud and EUCloudOpt 256-core system, respectively.

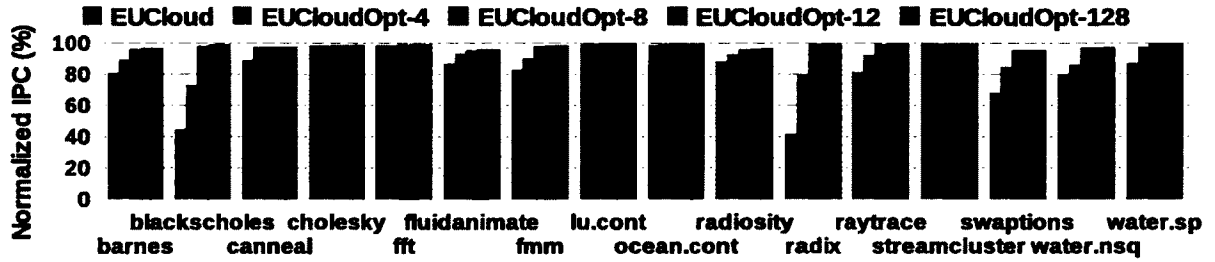
Figure 6-12 shows the bandwidth offered by each individual 64-thread application running on a 64-core quadrant of the EUCloud-based 256-core system. We split bandwidth results into memory-related bandwidth (see **Memory** bars), and EUCloud-related bandwidth (**EUCloud** bars). To validate the EUCloud-based manycore’s NoC using maximum offered bandwidth, we show in Figure 6-12 the traffic generated by EUCloud for different benchmarks. As we can see, **fft** is the application with the largest demand: 430 GB/s for EUCloud and 70 GB/s for memory. This translates into 1.72 TB/s and 0.28 TB/s as maximum bandwidth offered by a multi-programmed workload for EUCloud and memory, respectively. Note that, these bandwidths can be supported by the silicon-photonic **P-EUCloud** and **P-mem** NoC designs since they implement 2.56 TB/s and 0.32 TB/s, respectively (see Section 6.3.2). In general, the EUCloud accesses generate much more of traffic as compared to the traffic for

memory operations. This is expected since all FP instructions require NoC usage for remote execution at EUCloud, while not all memory-related instructions (e.g., load and stores) utilize the NoC due to hits at the cache hierarchy (two private levels of caches per core). We also include in the Figure the reduced bandwidth achieved by EUCloudOpt when bundles are used as they achieve a more compacted representation of dependent operands (see Section 6.3.4), thus reducing NoC traffic. We illustrate different configurations of bundles. As we can see, EUCloudOpt achieves an average of 9% reduction in NoC traffic across all benchmarks for the ideal configuration (EUCloud-128), and interestingly this outcome is comparable to that obtained by EUCloudOpt-8. So from the perspective of NoC traffic savings, the EUCloudOpt-8 is the best design.

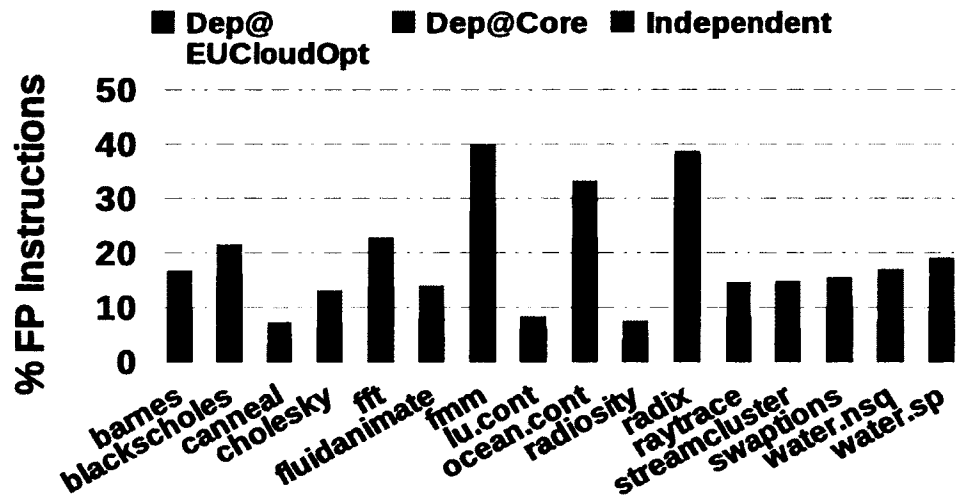
6.4.3 Single-Application Performance

In Figure 6-13, we compare the performance of EUCloud and EUCloudOpt with four different bundle sizes, with the Baseline system. To understand the performance trends, we studied each 64-thread application individually by simulating it in a single 64-core partition of the 256-core system. Compared to the EUCloud, EUCloudOpt can provide significant performance benefits: a maximum of 121% (radix) and 26.9% on average. We start seeing diminishing improvements in performance for bundles larger than 8 instructions for all benchmarks (see Figure 6.13(a)). Compared to the Baseline system, there is an average performance degradation of 1.71% (3.9% max) across all applications when we use EUCloudOpt-8.

A more in depth analysis of EUCloudOpt performance was carried out by analyzing the FP instructions in every 64-thread application. Figure 6.13(b) shows a breakdown of the different types of FP instructions that are executed. Dep@EUCloudOpt corresponds to instructions that are sent as part of a bundle to the EUCloudOpt.



(a)



(b)

Figure 6-13: Performance comparison between EUCloud and EUCloudOpt – (a) Normalized IPC against Baseline system. (b) Dependencies among FP instructions using EUCloudOpt-8. The values are normalized with respect to Baseline target system. For each experiment, we utilize a single 64-core partition of the manycore system. We show EUCloudOpt using different sizes of bundles (EUCloud-X): 4, 8, 12 and 128 (ideal case).

Independent corresponds to instructions that are not dependent on any other instructions and are executed either in the EUCloudOpt or in the core. Dep@Core instructions are executed by the cores' FPmov units and normally involve FP data movements using results from FP instructions executed at EUCloudOpt. This introduces stalls in the processor's pipeline as instructions wait for the results from EUCloud. Analyzing each bar in Figure 6.13(b) we can see that for applications containing a

small percentage of FP instructions (Radiosity and Canneal), we do not observe a large performance gap between EUCloud and EUCloudOpt. Applications (Blackscholes and Radix) with a large fraction of Dep@EUCloudOpt achieve higher performance improvement w.r.t EUCloud. Similarly, applications (Barnes and Swaptions) with larger fractions of Dep@Core instructions cannot fully leverage the EUCloudOpt.

6.4.4 Multi-programmed Workload Performance

Our analysis in Section 6.4.3 shows that EUCloudOpt-8 is the most efficient design for global sharing of FPUs for 64-thread applications running on a single partition of the 256-core target system. Here, we evaluate its performance by simulating the entire 256-core system using multi-programmed workloads composed of four 64-thread applications each. The applications in each workload are chosen depending on their FPU requirements (see Table 6.2).

We consider two cases – 1) Designs with suffix **x** – the number of FPUs in the EUCloudOpt is $4 \times 24 = 96$, where 24 corresponds to the largest number of FPUs by an application benchmark (out of the benchmarks that we simulated) to maintain performance, 2) Designs with suffix **a** – the number of FPUs in the EUCloudOpt is the

Table 6.2: Multi-programmed workloads composed of four 64-thread applications that are classified depending on demand of FPUs – L: low demand ([6,10] FPUs); M: medium demand ([11,17] FPUs); and H, Y and Z: high demand ([18,24] FPUs).

APP1	APP2	APP3	APP4	# FPUs
stream. (L)	fluid.(L)	wat.nsq (L)	lu (L)	34
fluid. (L)	wat.nsq (L)	ocean(M)	raytr. (M)	44
canneal (L)	radix (L)	fft (H)	black. (H)	68
radios. (M)	ocean (M)	raytr. (M)	wat.sp (M)	52
wat.sp (M)	radios. (M)	barnes (H)	black. (H)	62
swapt. (H)	barnes (H)	black. (H)	fmm (H)	80
black. (Y)	fft (Y)	fmm (Y)	black. (Y)	96
black. (Z)	black. (Z)	black. (Z)	black. (Z)	96

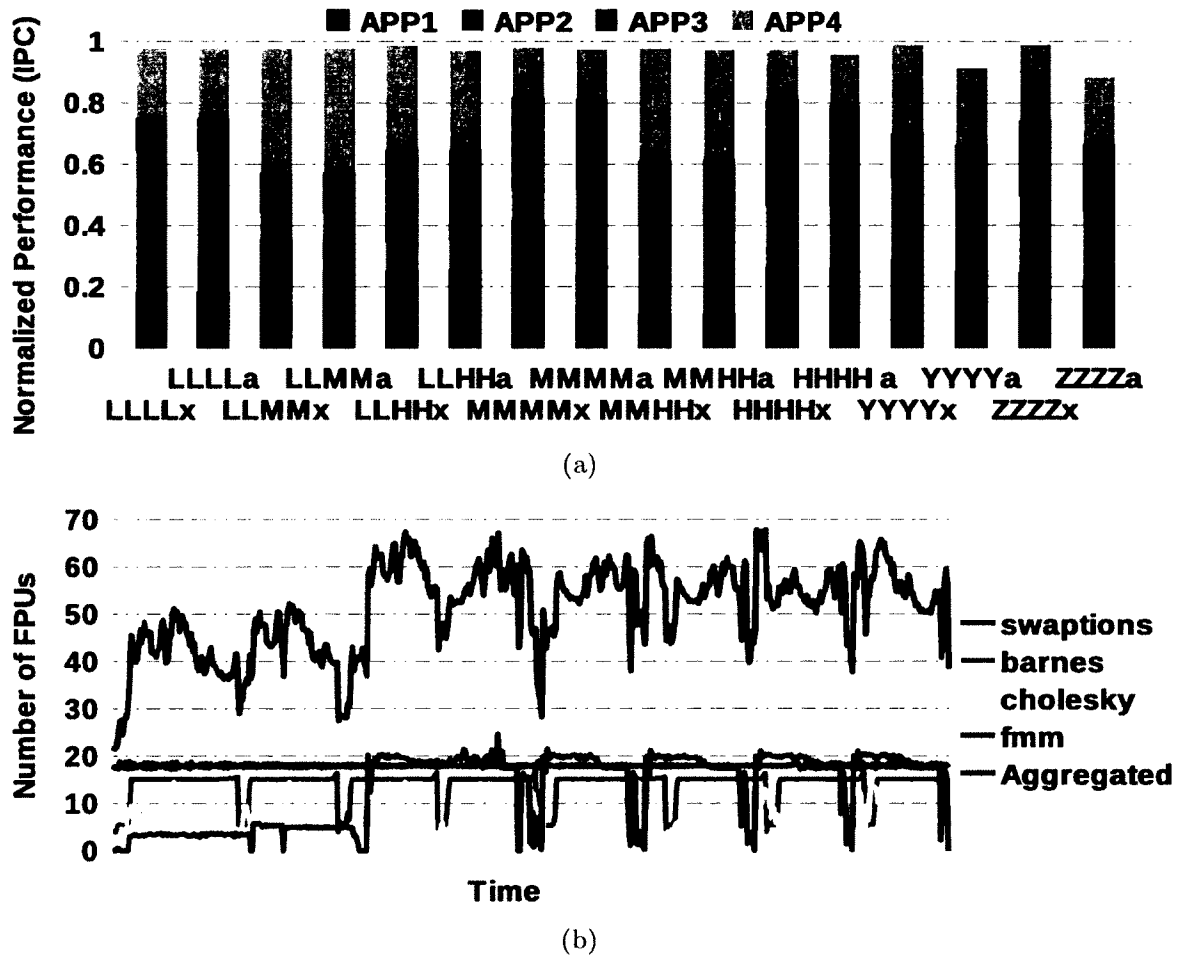


Figure 6.14: Performance for EUCloudOpt-based manycore system – (a) Normalized IPC against Baseline system (b) FPU utilization for HHHH multi-programmed workload. We use x and a suffixes when maximum (96) and average number (64) of FPU are used, respectively.

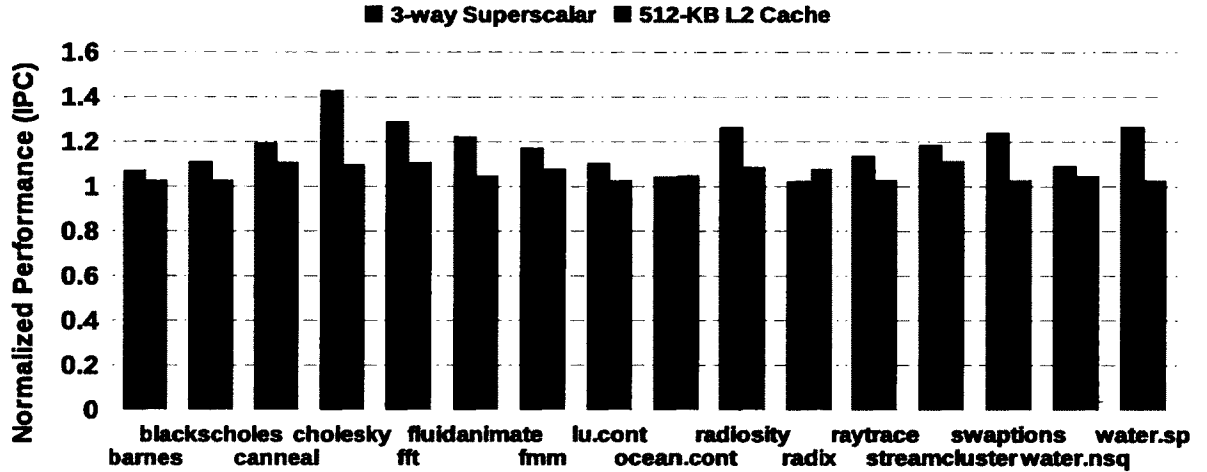
average number of FPU required by the various applications. This value is 56, which we round up to 64. As we can observe in Figure 6.14(a), compared to the Baseline system, the designs with 96 FPU have marginal performance degradation of 1.25% on average (2.3% max.). Similarly, compared to the Baseline system, the designs with 64 FPU the average performance degradation goes up to 1.81% (3.6% max.). Although the aggregated demand of FPU for multi-programmed workloads by pro-

jecting application’s FPUs demand is larger than the number of globally shared FPUs (64) in the cases LLHH (68) and HHHH (80), the reason of these small performance losses is that variations in the requirements of FPUs ensures that the aggregated demand rarely exceeds 64 FPUs – see Figure 6.14(b) for the worst-case HHHH workload. However, as shown in Table 6.2, for the YYYY and ZZZZ worst-case scenarios (96 FPUs) significant performance losses are reported (8.6% and 11.4%, respectively). In this way, 96 FPUs will be our design choice.

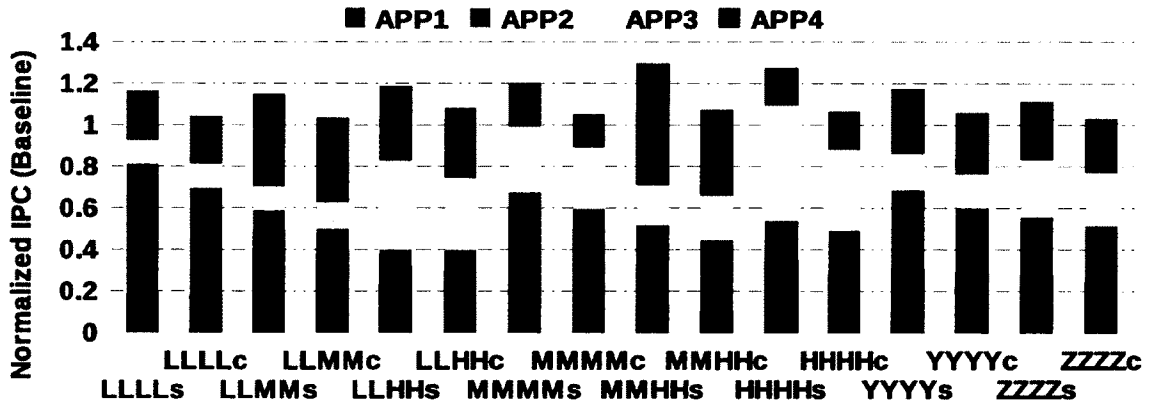
6.4.5 EUCloud-based Manycore Benefits

The two main benefits of using the EUCloudOpt approach are savings in area and power. Using McPAT tool (Li et al., 2009), the above reductions to 96 FPUs and 64 FPUs correspond to 13.75% and 16.50% of savings of total core area, respectively. Moreover, in terms of static power, we can reduce the total core static power by 10% and 12% when using 96 FPUs and 64 FPUs, respectively. Note that we cannot have any savings in dynamic power as EUCloud does not reduce the total number of FP instructions that are to be executed.

The savings in on-chip area and power can be harnessed to improve performance of the manycore system. We evaluate two variations of EUCloudOpt manycore systems. The first one includes larger L2 caches – 512 KB vs. 256 KB in the **Baseline** design, and the second one implements a more aggressive 3-way superscalar core vs. 2-way superscalar core in the **Baseline** design. Figure 6.15(a) shows the improved performance for each application running on a single 64-core partition of the 256-core system. As we can see, by increasing core complexity or using larger on-chip caches, we can achieve up to 42.9% or 11.5% performance improvements, respectively. As the increased performance comes from improved ILP of processor cores which in turn translates into higher FPU utilization, some applications require more FPUs to reach



(a) Improved performance for 64-thread applications



(b) Improved performance for multi-programmed workloads

Figure 6-15: Performance for EUCloudOpt-based improved manycore system – (a) Improved performance for 64-thread applications. (b) Improved performance for multi-programmed workloads. Bars are normalized against IPC achieved by the Baseline system. We use *s* and *c* suffixes for 3-way superscalar cores and larger-caches configurations, respectively.

maximum performance. For the modified EUCloudOpt manycore system, we need a total of 112 FPUs for the entire 256-core system. Using McPAT we estimated the total on-chip area and power dissipation of both updated designs for the logic layer of the 3D-stacked manycore to determine if they meet the power and area budgets (180 W and 349mm^2) of the Baseline system. The two designs require 94.9% and 96.3% of

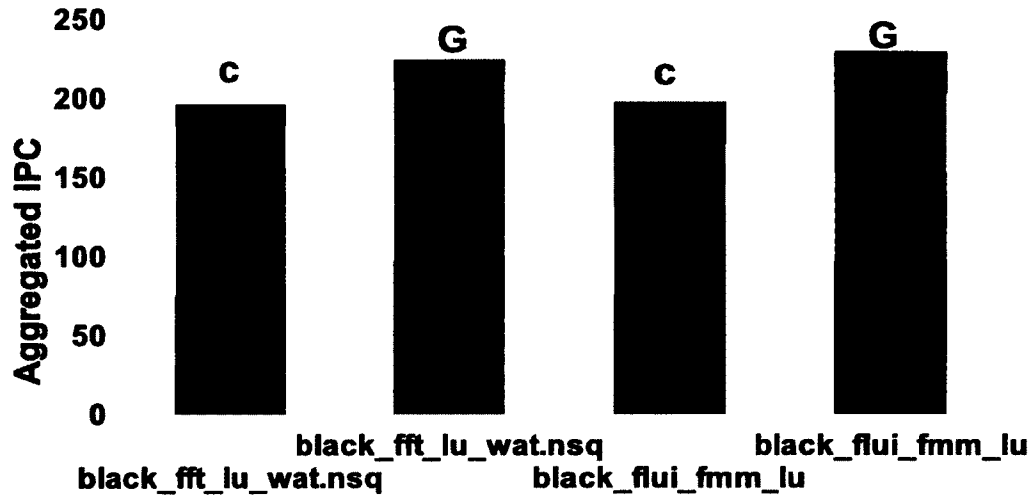


Figure 6-16: Performance comparison Cluster vs. Global EU-Cloud under aggregated IPC metric – The bottom fraction of the bars are for the first benchmark.

the area required for the **Baseline** design. With respect to power, the 3-way superscalar configuration consumes 8% more power than available in the budget, while the 512 KB design uses 99% of the power budget. Finally, Figure 6.15(b) shows the performance of the new settings when running the multi-programmed workloads utilized in Section 6.4.4. We get a 29.4% and 8.13% maximum performance improvements when using the complex-cores and larger-caches configurations, respectively.

6.4.6 Clustered vs. Global EUCloud

To understand the benefits of using global sharing, we consider two different layouts to access EUCloud’s resources: clustered and global. In a clustered layout, the target manycore system is split into different partitions (or clusters) and all the cores in a cluster utilize a subset of the available FPUs (i.e., there are sub-EUClouds; one per cluster). In the global case, all cores can access to all the FPUs so that there are no private subsets of FPUs to certain groups of cores.

Figure 6-16 shows a performance comparison between clustered and global EUCloud assuming the most optimized bEUCMA-8 system. A cluster is composed of a 64-core quadrant in the 256-core system, that results into four different clusters in the clustered EUCloud. For the study, we utilize multi-programmed workloads composed of four different 64-thread benchmarks – each 64-thread benchmark is mapped onto a different cluster. In the figure, over each bar we show C or G to indicate if the results refer to the clustered or global version, respectively. For each workload, we obtain the number of required groups of FPUs (a pair of `FPUalu` and `FPUmuldiv` units). In the clustered version, not all FPUs can be used but a fourth of this number because we evenly distribute them across the four clusters. As we can observe, global EUCloud achieves better performance (aggregated IPC) than the clustered counterpart (an average of 10% improvement and up to 16%). To understand the improvements, each bar in the figure is further decomposed into four different portions that represent the achieved IPC by each benchmark in the multi-programmed workload. As an example, the first workload comprises `blackscholes`, `fft`, `lu.cont` and `water_nsq`. They require a total of 62 pairs of FPUs (`FPUalu` and `FPUmuldiv`) and then, the clustered version assign 14 FPUs per cluster thereby reducing performance of `blackscholes` and `fft` as both require 24 groups of FPUs.

6.5 Summary

Manycore systems tradeoff ILP for massive TLP parallelism, in turn lowering the utilization of their core's execution units. This creates opportunities for sharing of execution units, which in turn can be leveraged to decrease area and power. We propose a novel manycore processor architecture that integrates an *Execution Unit Cloud* (EUCloud) containing an array of execution units (the costly x87-compliant floating point units – `FPUalu` and `FPUmuldiv`) that are globally shared by all the cores in the

manycore system. For an efficient implementation of our EUCloud-based manycore system, we leverage the low global communication latency and very high bandwidth density of silicon-photonics links to implement the communication infrastructure required to access EUCloud.

We evaluated our EUCloud-based manycore using a 3D-stacked 256-core manycore system. The most optimized version of our proposed architecture, EUCloudOpt, can achieve performance comparable (1.25% performance loss on average) to the nominal design, while using just 96 FPUs as compared to the 256 FPUs that are available in the nominal design. This leads to significant on-chip area and power savings in the logic layer (13.75% and 10%, respectively), which can be leveraged to improve manycore performance by increasing the core complexity or cache size. Compared to the nominal design, this increase in core complexity or cache size provides a 29.4% or 8.13% improvement in performance, with 8% increase or no increase in the power dissipation, respectively.

Chapter 7

Conclusion and Future Work

In this dissertation, we have proposed several techniques to improve the system performance and system energy efficiency of a manycore system with electrical NoC or silicon-photonic NoC. The EVC-T technique is helpful in broadcast based electrical NoC architectures that are currently the mainstream in manycore systems. The runtime laser power management techniques are helpful in reducing the laser power consumption and potentially expedites the process of widespread adoption of silicon-photonic link technology in manycore processors. The technique of placing and sharing on-chip laser sources is helpful in a manycore system that uses 3D integration of laser source, photonic devices and logic units. The EUCloud technology shows an example of using the large bandwidth provided by the silicon-photonic technique to improve processor performance by sharing the computational resources on the processor.

7.1 Conclusion

In Chapter 2, for electrical NoC, we proposed a new flow control technique: express virtual channels with taps (EVC-T) for NoC architectures in manycore systems. When used with cmesh physical layout, EVC-T helps create a logical topology with low diameter, that is easy to design from the hardware perspective and easy

to program from the software perspective. In addition, we have also proposed a contention-free tree architecture that supports broadcasting on unordered on-chip interconnects for snoop-based cache coherency protocols. The notification trees enable a core to wait for less than one cycle after broadcast packet is received on average to make decisions on the correct processing order of broadcast packets. The synthetic benchmark analysis shows our technique reduces the average packet (data and broadcast) latency by 24%, while consuming the same amount of power as a conventional cmesh network. For NAS parallel benchmarks, our techniques improve the IPC by 9% and EDP by 13% on average with negligible changes in power.

In Chapter 3, for silicon-photonic NoC in manycore systems, we proposed a runtime technique that dynamically manages the laser power of this multi-bus NoC depending on the communication bandwidth requirements of the various applications running on the manycore processor. For a silicon-photonic multi-bus NoC between the private L1 and distributed L2 caches, we proposed a policy that uses weighted time-division multiplexing with token-stream flow control and switching ON/OFF laser sources as necessary, to maximize the total energy efficiency of the manycore processor. For a 64-core processor running the NAS parallel benchmarks, we get an average of more than 49% reduction in laser power, with a 6% reduction in application performance.

In Chapter 4, we also proposed an alternate runtime laser power management technique, where we activate/deactivate L2 banks depending on the spatial and temporal variations in the application behavior, and then switch ON/OFF the silicon-photonic links associated with these L2 banks to dynamically manage the laser power. The key idea is that for a given application at any given point of time, we operate the manycore system using the minimum number of L2 banks and silicon-photonic links required for maximizing energy efficiency. Our policy is scalable to large core counts and is applicable to other cache and NoC architectures. On a 64-core target system,

our proposed technique reduces laser power and system power by 23.8% and 6.39%, respectively, and improves EDP by 5.52% on average.

In Chapter 5, for the silicon-photonics NoC, we also explored limits and opportunities for sharing/placement of on-chip laser sources with the goal of minimizing the laser power consumption. We first explored the power, efficiency and temperature trade-offs associated with on-chip laser sources. We then used a cross-layer methodology where we jointly considered the NoC bandwidth constraints, the thermal constraints and the physical layout constraints to determine the optimal sharing of laser sources such that the optical output power of the on-chip laser source corresponds to its maximum efficiency and the optimal placement of the laser sources such that their temperature is minimum and their optical output power is optimal so as to maximize the efficiency of the laser source. We applied our methodology to 3 logical topologies, 2 physical layouts and 3 sharing/placement strategies, and determined the optimal sharing/placement changes for a logical topology and physical layout. For a matching bisection bandwidth, at low waveguide loss per cm, the 8-ary 3-stage Clos with locally placed and shared laser source consumes the least laser power, while at large waveguide loss per cm, the 8-ary 3-stage Clos with shared laser source placed along the edge consumes the least laser power.

In Chapter 6, to justify the need for large NoC bandwidth, we explored a 256-core architecture that integrates an Execution Unit Cloud (EUCloud) containing an array of floating point units shared by cores through silicon-photonics links. The optimized EUCloudOpt can achieve performance comparable (1.25% performance loss on average) to the nominal design, while using just 96 FPUs as compared to the 256 FPUs that are available in the nominal design. This leads to significant on-chip area and power savings in the logic layer (13.75% and 10%, respectively), which can be leveraged to improve manycore performance by increasing the core complexity or cache

size. The increase in core complexity or cache size provides a 29.4% or 8.13% improvement in performance, with 8% increase or no increase in the power dissipation, respectively.

7.2 Future Work

7.2.1 Laser Power Management through Core Reconfigurations

The silicon-photonics NoC connects many on-chip components, including cores, caches, memory controllers and other peripherals. Therefore, we can save the laser power consumption by activating and deactivating these on-chip components based on the demands of the applications. In Chapter 4, we proposed an example of saving the laser power consumption by activating/deactivating the L2 cache banks at runtime and switching ON/OFF the associated silicon-photonics links. We can apply this methodology to other on-chip components. For example, a workload mapping technique can be explored to determine the optimal number of cores for each application by considering the trade-off between the core performance and power consumption in the silicon-photonics NoC. The silicon-photonics links associated with the unused cores could be switched OFF to save laser power.

7.2.2 Runtime Selection and Sharing of On-chip Laser Sources

The sharing and placement of on-chip laser sources can reduce the laser power consumption in silicon-photonics NoC. In Chapter 5, we made the static decision of sharing and placement based on the network topology, silicon-photonics link features and background temperature of the processors. This work can be extended to consider runtime variations in bandwidth requirements and chip temperature throughout the execution of the applications. Each application has multiple phases with varying

core power consumption and network bandwidth demands. For example, the system can choose locally placed laser sources when the application is running in the phase with low core power consumption, and choose remotely placed laser sources when the application is running in the phase with high core power consumption. The dynamical selection of the laser sources can help lower the laser source temperature and in turn, improve the laser source efficiency. In another scenarios, when the system turns IFF several silicon-photonics channels when the demand of application reduces, a remotely shared laser source can outperform a locally shared laser source, since the locally shared laser source may not achieve the optimal sharing degree for the reduced silicon-photonics channel count. In both scenarios, the system can achieve better laser power efficiency by dynamically choosing between remotely placed laser sources and locally placed laser sources.

7.2.3 Performance Improvement through Silicon-photonics NoCs

We can also explore the technique of improving the system performance by leveraging the high-bandwidth and low-latency communication provided by the silicon-photonics links. For example, the cache pre-fetching can reduce the cache miss rates and in turn, improve the system performance. However, the pre-fetching technique requires large bandwidth and is only used when there are extra network bandwidth available in the electrical NoC. The silicon-photonics NoC can provide large bandwidth for the pre-fetching activity, and therefore the system only need to consider the impact of pre-fetching on the performance of cache and memory. Similarly, the large bandwidth of silicon-photonics links can be used for the fast synchronization of threads running on remote cores. This would help different threads exchange information quickly and in turn, improve the performance of the entire system.

References

- Tilepro64 processor product brief.
- (1985). IEEE standard for binary floating-point arithmetic.
- (2010). The 50G silicon photonics link. White Paper, Intel Labs.
- A. Shacham, K. Bergman and L. P. Carloni (2007). On the design of a photonic network-on-chip. In *1st International Symposium on Networks-on-Chip, 2007. NOCS 2007*, pages 53–64.
- Agarwal, N., Peh, L.-S., and Jha, N. (2009). In-network snoop ordering (inso): Snoopy coherence on unordered interconnects. In *IEEE 15th International Symposium on High Performance Computer Architecture, 2009. HPCA 2009*, pages 67–78.
- Ahmad, B. and Arslan, T. (2005). Dynamically reconfigurable NoC for reconfigurable MPSoC. In *IEEE Custom Integrated Circuits Conference, 2005. CICC 2005*, pages 277–280.
- Assefa, S., Xia, F., Bedell, S. W., Zhang, Y., Topuria, T., Rice, P. M., and Vlasov, Y. A. (2010). CMOS-integrated high-speed MSM germanium waveguide photodetector. *Optics Express*, 18(5):4986–4999.
- Bailey, D., Barszcz, E., Barton, J., Browning, D., Carter, R., Dagum, L., Fatoohi, R., Fineberg, S., Frederickson, P., Lasinski, T., Schreiber, R., Simon, H., Venkatakrisnan, V., and Weeratunga, S. (1994). The NAS parallel benchmarks. Technical Report RNR-94-007.
- Barwicz, T., Byun, H., Gan, F., Holzwarth, C. W., Popovic, M. A., Rakich, P. T., Watts, M. R., Ippen, E. P., Kärtner, F. X., Smith, H. I., Orcutt, J. S., Ram, R. J., Stojanovic, V., Olubuyide, O. O., Hoyt, J. L., Spector, S., Geis, M., Grein, M., Lyszczarz, T., and Yoon, J. U. (2007). Silicon photonics for compact, energy-efficient interconnects (invited). *Journal of Optical Networking*, 6(1):63–73.
- Batten, C., Joshi, A., Orcutt, J., Khilo, A., Moss, B., Holzwarth, C., Popovic, M., Li, H., Smith, H. I., Hoyt, J., Kartner, F., Ram, R., Stojanovic, V., and Asanovic, K. (2008). Building manycore processor-to-DRAM networks with monolithic silicon photonics. In *16th IEEE Symposium on High Performance Interconnects, 2008. HOTI 2008*, pages 21–30.

- Batten, C., Joshi, A., Stojanovic, V., and Asanovic, K. (2012). Designing chip-level nanophotonic interconnection networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2(2):137–153.
- Bauters, J. F., Davenport, M. L., Heck, M. J. R., Doylend, J. K., Chen, A., Fang, A. W., and Bowers, J. E. (2013). Silicon on ultra-low-loss waveguide photonic integration platform. *Optics Express*, 21(1):544–555.
- Bauters, J. F., Heck, M. J. R., John, D., Dai, D., Tien, M.-C., Barton, J. S., Leinse, A., Heideman, R. G., Blumenthal, D. J., and Bowers, J. E. (2011). Ultra-low-loss high-aspect-ratio Si_3N_4 waveguides. *Optics Express*, 19(4):3163–3174.
- Beamer, S., Sun, C., Kwon, Y.-J., Joshi, A., Batten, C., Stojanović, V., and Asanović, K. (2010). Re-architecting DRAM memory systems with monolithically integrated silicon photonics. In *37th Annual International Symposium on Computer Architecture*, ISCA 2010, pages 129–140.
- Bell, S., Edwards, B., Amann, J., Conlin, R., Joyce, K., Leung, V., MacKay, J., Reif, M., Bao, L., Brown, J., Mattina, M., Miao, C.-C., Ramey, C., Wentzlaff, D., Anderson, W., Berger, E., Fairbanks, N., Khan, D., Montenegro, F., Stickney, J., and Zook, J. (2008). Tile64 - processor: A 64-Core SoC with mesh interconnect. In *2008 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 88–598.
- Biberman, A., Sherwood-Droz, N., Zhu, X., Preston, K., Hendry, G., Levy, J., Chan, J., Wang, H., Lipson, M., and Bergman, K. (2011). Photonic network-on-chip architecture using 3D integration. *Optoelectronic Integrated Circuits XIII, Proceedings of SPIE*, 7942(1):79420M.
- Bienia, C., Kumar, S., Singh, J. P., and Li, K. (2008). The PARSEC benchmark suite: Characterization and architectural implications. In *17th International Conference on Parallel architectures and Compilation Techniques*, PACT 2008, pages 72–81.
- Bilir, E., Dickson, R., Hu, Y., Plakal, M., Sorin, D., Hill, M., and Wood, D. (1999). Multicast snooping: A new coherence method using a multicast address network. In *26th International Symposium on Computer Architecture, 1999*, pages 294–304.
- Binkert, N., Dreslinski, R., Hsu, L., Lim, K., Saidi, A., and Reinhardt, S. (2006). The m5 Simulator: Modeling networked systems. *IEEE Micro*, 26(4):52–60.
- Boos, A., Ramini, L., Schlichtmann, U., and Bertozzi, D. (2013). Proton: An automatic place-and-route tool for optical networks-on-chip. In *IEEE/ACM International Conference on Computer-Aided Design, 2013. ICCAD 2013*, pages 138–145.
- Carlson, T., Heirman, W., and Eeckhout, L. (2011). Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *2011*

- International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–12.
- Castells-Rufas, D., Fernandez-Alonso, E., Carrabina, J., and Joven, J. (2011). Sharing fpus in many-soft-cores. In *International Conference on Field-Programmable Technology (FPT), 2011*, pages 1–6.
- Chan, J., Hendry, G., Bergman, K., and Carloni, L. (2011). Physical-Layer Modeling and System-Level Design of Chip-Scale Photonic Interconnection Networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(10):1507–1520.
- Charlesworth, A. (2002). The sun fireplane interconnect. *IEEE Micro*, 22(1):36–45.
- Chen, C. and Joshi, A. (2013). Runtime management of laser power in silicon-photonics multibus NoC architecture. *IEEE Journal of Selected Topics in Quantum Electronics*, 19(2):3700713–3700713.
- Chen, C., Meng, J., Coskun, A., and Joshi, A. (2011). Express virtual channels with taps (EVC-T): A flow control technique for network-on-chip (NoC) in manycore systems. In *2011 IEEE 19th Annual Symposium on High Performance Interconnects (HOTI)*, pages 1–10.
- Chen, L., Preston, K., Manipatruni, S., and Lipson, M. (2009). Integrated GHz silicon photonic interconnect with micrometer-scale modulators and detectors. *Optics Express*, 17(17):15248–15256.
- Cianchetti, M. J., Kerekes, J. C., and Albonesi, D. H. (2009). Phastlane: A rapid transit optical routing network. In *36th Annual International Symposium on Computer Architecture, ISCA 2009*, pages 441–450.
- Coldren, L. A., Corzine, S. W., and Mashanovitch, M. L. (2012). *Diode Laser and Photonic Integrated Circuits*. Wiley Series in Microwave and Optical Engineering, second edition.
- Condrat, C., Kalla, P., and Blair, S. (2013). Crossing-aware channel routing for photonic waveguides. In *2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 649–652.
- Coskun, A., Ayala, J., Atienza, D., and Rosing, T. (2009). Modeling and dynamic management of 3D multicore systems with liquid cooling. In *2009 17th IFIP International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 35–40.
- Culler, D., Singh, J., and Gupta, A. (1998). *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann, first edition. The Morgan Kaufmann Series in Computer Architecture and Design.

- Dally, W. and Towles, B. (2003). *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc.
- Ding, D., Zhang, Y., Huang, H., Chen, R., and Pan, D. (2009). O-router: An optical routing framework for low power on-chip silicon nano-photonics integration. In *46th ACM/IEEE Design Automation Conference, 2009. DAC 2009*, pages 264–269.
- Gnan, M., Thorns, S., Macintyre, D., De La Rue, R., and Sorel, M. (2008). Fabrication of low-loss photonic wires in silicon-on-insulator using hydrogen silsesquioxane electron-beam resist. *Electronics Letters*, 44(2):115–116.
- Gorin, A., Jaouad, A., Grondin, E., Aimez, V., and Charette, P. (2008). Fabrication of silicon nitride waveguides for visible-light using PECVD: A study of the effect of plasma frequency on optical properties. *Optics Express*, 16(18):13509–13516.
- Green, W., Assefa, S., Rylyakov, A., Schow, C., Horst, F., and Vlasov, Y. (2010). CMOS Integrated Silicon Nanophotonics: Enabling Technology for Exascale Computational Systems. Invited Talk, SEMICON Conference 2010.
- Grot, B., Hestness, J., Keckler, S., and Mutlu, O. (2009). Express cube topologies for on-chip interconnects. In *IEEE 15th International Symposium on High Performance Computer Architecture, 2009. HPCA 2009*, pages 163–174.
- Gu, H., Xu, J., and Zhang, W. (2009). A low-power fat tree-based optical network-on-chip for multiprocessor system-on-chip. In *Design, Automation and Test in Europe Conference and Exhibition, 2009. DATE 2009*, pages 3–8.
- Gunn, C. (2006). CMOS photonics for high-speed interconnects. *IEEE Micro*, 26(2):58–66.
- Gwenapp, L. (1995). Intel’s P6 Uses Decoupled Superscalar Design. *Microprocessor Report*, 9(2).
- Heck, M. and Bowers, J. (2014). Energy efficient and energy proportional optical interconnects for multi-core processors: Driving the need for on-chip sources. *IEEE Journal of Selected Topics in Quantum Electronics*, 20(4):1–12.
- Hendry, G., Chan, J., Carloni, L., and Bergman, K. (2011). Vandal: A tool for the design specification of nanophotonic networks. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2011*, pages 1–6.
- Hilbert, M. and Lpez, P. (2011). The Worlds Technological Capacity to Store, Communicate, and Compute Information. *Science*, 332(6025):60–65.
- Holzwarth, C., Orcutt, J., Li, H., Popovic, M., Stojanovic, V., Hoyt, J., Ram, R., and Smith, H. I. (2008). Localized substrate removal technique enabling strong-confinement microphotonics in bulk si CMOS processes. In *Conference on Lasers*

- and *Electro-Optics, 2008 and 2008 Conference on Quantum Electronics and Laser Science. CLEO/QELS 2008*, pages 1–2.
- Hosseini, E. S., Yegnanarayanan, S., Atabaki, A. H., Soltani, M., and Adibi, A. (2009). High quality planar silicon nitride microdisk resonators for integrated photonics in the visible wavelength range. *Optics Express*, 17(17):14543–14551.
- Howard, J., Dighe, S., Hoskote, Y., Vangal, S., Finan, D., Ruhl, G., Jenkins, D., Wilson, H., Borkar, N., Schrom, G., Paillet, F., Jain, S., Jacob, T., Yada, S., Marella, S., Salihundam, P., Erraguntla, V., Konow, M., Riepen, M., Droege, G., Lindemann, J., Gries, M., Apel, T., Henriss, K., Lund-Larsen, T., Steibl, S., Borkar, S., De, V., Van Der Wijngaart, R., and Mattson, T. (2010). A 48-Core ia-32 message-passing processor with DVFS in 45nm CMOS. In *2010 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 108–109.
- Intel Corp. (2013). *IntelTM 64 and IA-32 Architectures. Software Developer’s Manual*.
- Jain, S., Khare, S., Yada, S., Ambili, V., Salihundam, P., Ramani, S., Muthukumar, S., Srinivasan, M., Kumar, A., Gb, S., Ramanarayanan, R., Erraguntla, V., Howard, J., Vangal, S., Dighe, S., Ruhl, G., Aseron, P., Wilson, H., Borkar, N., De, V., and Borkar, S. (2012). A 280mv-to-1.2v wide-operating-range ia-32 processor in 32nm cmos. In *2012 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 66–68.
- Joshi, A., Batten, C., Kwon, Y.-J., Beamer, S., Shamim, I., Asanovic, K., and Stojanovic, V. (2009). Silicon-photonics networks for global on-chip communication. In *3rd ACM/IEEE International Symposium on Networks-on-Chip, 2009. NoCS 2009*, pages 124–133.
- Kahle, J. A. and Moore, C. R. (2000). Shared execution unit in a dual core processor. *US Patent 6725354 B1*.
- Kakoei, M. R., Loi, I., and Benini, L. (2013). A shared-fpu architecture for ultra-low power mpsoes. In *ACM International Conference on Computing Frontiers, CF 2013*, pages 3:1–3:8.
- Kim, J., Balfour, J., and Dally, W. (2007). Flattened butterfly topology for on-chip networks. In *40th Annual IEEE/ACM International Symposium on Microarchitecture, 2007. MICRO 2007*, pages 172–182.
- Kirman, N., Kirman, M., Dokania, R., Martinez, J., Apsel, A., Watkins, M., and Albonesi, D. (2006). Leveraging optical technology in future bus-based chip multiprocessors. In *39th Annual IEEE/ACM International Symposium on Microarchitecture, 2006. MICRO-39*, pages 492–503.
- Kirman, N. and Martínez, J. F. (2010). A power-efficient all-optical on-chip inter-

- connect using wavelength-based oblivious routing. In *15th edition of ASPLOS on Architectural support for programming languages and operating systems*, ASPLOS 2010, pages 15–28.
- Klamkin, J., Huang, R., Plant, J., Connors, M., Missaggia, L., Loh, W., Smith, G., Ray, K., O'Donnell, F., Donnelly, J., and Juodawlkis, P. (2010). Directly modulated narrowband slab-coupled optical waveguide laser. *Electronics Letters*, 46(7):522–523.
- Kodi, A. and Louri, A. (2011). Energy-efficient and bandwidth-reconfigurable photonic networks for high-performance computing (HPC) systems. *IEEE Journal of Selected Topics in Quantum Electronics*, 17(2):384–395.
- Krishna, T., Kumar, A., Peh, L.-S., Postman, J., Chiang, P., and Erez, M. (2009). Express virtual channels with capacitively driven global links. *IEEE Micro*, 29(4):48–61.
- Kuhn, K., Liu, M., and Kennel, H. (2010). Technology options for 22nm and beyond. In *2010 International Workshop on Junction Technology (IWJT)*, pages 1–6.
- Kumar, A., Peh, L.-S., and Jha, N. (2008). Token flow control. In *2008 41st IEEE/ACM International Symposium on Microarchitecture, 2008. MICRO-41*, pages 342–353.
- Kurd, N., Douglas, J., Mosalikanti, P., and Kumar, R. (2008). Next generation intel x00ae; micro-architecture (nehalem) clocking architecture. In *2008 IEEE Symposium on VLSI Circuits*, pages 62–63.
- Kurian, G., Sun, C., Chen, C.-H., Miller, J., Michel, J., Wei, L., Antoniadis, D., Peh, L.-S., Kimerling, L., Stojanovic, V., and Agarwal, A. (2012). Cross-layer energy and performance evaluation of a nanophotonic manycore processor system using real application workloads. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1117–1130.
- Leon, A., Langley, B., and Shin, J. (2006). The UltraSPARC T1 processor: CMT reliability. In *IEEE Custom Integrated Circuits Conference, 2006. CICC 2006*, pages 555–562.
- Li, C., Browning, M., Gratz, P. V., and Palermo, S. (2012a). Luminoc: A power-efficient, high-performance, photonic network-on-chip for future parallel architectures. In *21st International Conference on Parallel Architectures and Compilation Techniques*, PACT 2012, pages 421–422.
- Li, S., Ahn, J. H., Strong, R., Brockman, J., Tullsen, D., and Jouppi, N. (2009). Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *42nd Annual IEEE/ACM International Symposium on Microarchitecture, 2009. MICRO-42*, pages 469–480.

- Li, Z., Mohamed, M., Chen, X., Dudley, E., Meng, K., Shang, L., Mickelson, A., Joseph, R., Vachharajani, M., Schwartz, B., and Sun, Y. (2012b). Reliability modeling and management of nanophotonic on-chip networks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(1):98–111.
- Liang, X., Turgay, K., and Brooks, D. (2007). Architectural power models for sram and cam structures based on hybrid analytical/empirical techniques. In *IEEE/ACM International Conference on Computer-Aided Design, 2007. ICCAD 2007*, pages 824–830.
- Marty, M. and Hill, M. (2006). Coherence ordering for ring-based chip multiprocessors. In *39th Annual IEEE/ACM International Symposium on Microarchitecture, 2006. MICRO-39*, pages 309–320.
- McIntyre, H., Arekapudi, S., Busta, E., Fischer, T., Golden, M., Horiuchi, A., Meneghini, T., Naffziger, S., and Vinh, J. (2012). Design of the two-core x86-64 Amd “bulldozer” module in 32 nm SOI CMOS. *IEEE Solid-State Circuits*, 47(1):164–176.
- McKeown, N. (1999). The iSLIP scheduling algorithm for input-queued switches. *IEEE/ACM Transactions on Networking*, 7(2):188–201.
- Meltzer, D. (1999). Single chip multiprocessor with shared execution units. *US Patent 5987587 A*.
- Meng, J., Chen, C., Coskun, A. K., and Joshi, A. (2011). Run-time energy management of manycore systems through reconfigurable interconnects. In *21st edition of the Great lakes Symposium on VLSI, GLSVLSI 2011*, pages 43–48.
- Meng, J., Kawakami, K., and Coskun, A. (2012). Optimizing energy efficiency of 3-D multicore systems with stacked DRAM under power and thermal constraints. In *2012 49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 648–655.
- Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, 38(8):114–117.
- Morris, R. and Kodi, A. (2010). Power-efficient and high-performance multi-level hybrid nanophotonic interconnect for multicores. In *2010 4th ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, pages 207–214.
- Nitta, C., Farrens, M., and Akella, V. (2011). Addressing system-level trimming issues in on-chip nanophotonic networks. In *2011 IEEE 17th International Symposium on High Performance Computer Architecture (HPCA)*, pages 122–131.
- Oberman, S., Favor, G., and Weber, F. (1999). AMd 3Dnow! technology: architecture and implementations. *IEEE Micro*, 19(2):37–48.

- Orcutt, J., Khilo, A., Popovic, M., Holzwarth, C., Moss, B., Li, H., Dahlem, M., Bonifield, T., Kartner, F., Ippen, E., Hoyt, J., Ram, R., and Stojanovic, V. (2008). Demonstration of an electronic photonic integrated circuit in a commercial scaled bulk CMOS process. In *Conference on Lasers and Electro-Optics, 2008 and 2008 Conference on Quantum Electronics and Laser Science. CLEO/QELS 2008.*, pages 1–2.
- Orcutt, J. S., Khilo, A., Holzwarth, C. W., Popović, M. A., Li, H., Sun, J., Bonifield, T., Hollingsworth, R., Kärtner, F. X., Smith, H. I., Stojanović, V., and Ram, R. J. (2011). Nanophotonic integration in state-of-the-art CMOS foundries. *Optics Express*, 19(3):2335–2346.
- Osgood, Jr., R. M., Espinola, R. L., Dadap, J. I., McNab, S. J., and Vlasov, Y. A. (2005). Silicon nanowire active integrated optics. *Optoelectronic Integrated Circuits VII, Proceedings of SPIE*, 5729(1):110–117.
- Pan, Y., Kim, J., and Memik, G. (2010). Flexishare: Channel sharing for an energy-efficient nanophotonic crossbar. In *2010 IEEE 16th International Symposium on High Performance Computer Architecture (HPCA)*, pages 1–12.
- Pan, Y., Kumar, P., Kim, J., Memik, G., Zhang, Y., and Choudhary, A. (2009). Firefly: illuminating future network-on-chip with nanophotonics. In *36th Annual International Symposium on Computer Architecture, ISCA 2009*, pages 429–440.
- Papamarcos, M. S. and Patel, J. H. (1984). A low-overhead coherence solution for multiprocessors with private cache memories. In *11th Annual International Symposium on Computer Architecture, ISCA 1984*, pages 348–354.
- Petracca, M., Lee, B., Bergman, K., and Carloni, L. (2008). Design exploration of optical interconnection networks for chip multiprocessors. In *16th IEEE Symposium on High Performance Interconnects, 2008. HOTI 2008*, pages 31–40.
- Preston, K. and Lipson, M. (2009). Slot waveguides with polycrystalline silicon for electrical injection. *Optics Express*, 17(3):1527–1534.
- Preston, K., Manipatruni, S., Poitras, C., and Lipson, M. (2009). 2.5 Gbps electro-optic modulator in deposited silicon. In *Conference on Lasers and Electro-Optics, 2009 and 2009 Conference on Quantum electronics and Laser Science Conference. CLEO/QELS 2009*, pages 1–2.
- Preston, K., Schmidt, B., and Lipson, M. (2007). Polysilicon photonic resonators for large-scale 3D integration of optical networks. *Optics Express*, 15(25):17283–17290.
- Psota, J., Miller, J., Kurian, G., Hoffman, H., Beckmann, N., Eastep, J., and Agarwal, A. (2010). Atac: Improving performance and programmability with on-chip optical networks. In *2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3325–3328.

- Q. Xu, B. Schmidt, S. Pradhan and M. Lipson (2005). Micrometre-scale silicon electro-optic modulator. *Nature*, 435(7040):325–327.
- Qureshi, M. K., Jaleel, A., Patt, Y. N., Steely, S. C., and Emer, J. (2007). Adaptive insertion policies for high performance caching. In *34th Annual International Symposium on Computer Architecture*, ISCA 2007, pages 381–391.
- Reynolds, P.F., J., Williams, C., and Wagner, R.R., J. (1997). Isotach networks. *IEEE Transactions on Parallel and Distributed Systems*, 8(4):337–348.
- Sarlet, G., Morthier, G., and Baets, R. (1999). Wavelength and mode stabilization of widely tunable sg-dbr and ssg-dbr lasers. *IEEE Photonics Technology Letters*, 11(11):1351–1353.
- Sim, J., Lee, J., Qureshi, M., and Kim, H. (2012). Flexclusion: Balancing cache capacity and on-chip bandwidth via flexible exclusion. In *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, pages 321–332.
- Sridaran, S. and Bhave, S. A. (2010). Nanophotonic devices on thin buried oxide silicon-on-insulator substrates. *Optics Express*, 18(4):3850–3857.
- Stoica, I., Abdel-Wahab, H., Jeffay, K., Baruah, S., Gehrke, J., and Plaxton, C. (1996). A proportional share resource allocation algorithm for real-time, time-shared systems. In *Real-Time Systems Symposium, 1996., 17th IEEE*, pages 288–299.
- Strauss, K., Shen, X., and Torrellas, J. (2007). Uncorq: Unconstrained snoop request delivery in embedded-ring multiprocessors. In *40th Annual IEEE/ACM International Symposium on Microarchitecture, 2007. MICRO 2007*, pages 327–342.
- T. Baehr-Jones, M. Hochberg, C. Walker and A. Scherer (2004). High-Q ring resonators in thin silicon-on-insulator. *Applied Physics Letters*, 85(16):3346–3347.
- Thoziyoor, S., Muralimanohar, N., Ahn, J.-H., and Jouppi, N. P. (2008). CACTI 5.1. Technical Report HPL-2008-20, HP Labs.
- Vantrease, D., Binkert, N., Schreiber, R., and Lipasti, M. (2009). Light speed arbitration and flow control for nanophotonic interconnects. In *42nd Annual IEEE/ACM International Symposium on Microarchitecture, 2009. MICRO-42*, pages 304–315.
- Vantrease, D., Schreiber, R., Monchiero, M., McLaren, M., Jouppi, N., Fiorentino, M., Davis, A., Binkert, N., Beausoleil, R., and Ahn, J. (2008). Corona: System implications of emerging nanophotonic technology. In *35th International Symposium on Computer Architecture, 2008. ISCA 2008*, pages 153–164.
- Vogelsang, T. (2010). Understanding the Energy Consumption of Dynamic Random Access Memories. In *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 363–374.

- Wang, H., Peh, L.-S., and Malik, S. (2003). Power-driven design of router microarchitectures in on-chip networks. In *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36*, pages 105–116.
- Williams, C., Reynolds, P.F., J., and de Supinski, B. (2000). Delta coherence protocols. *IEEE Concurrency*, 8(3):23–29.
- Woo, S., Ohara, M., Torrie, E., Singh, J., and Gupta, A. (1995). The splash-2 programs: characterization and methodological considerations. In *22nd Annual International Symposium on Computer Architecture, 1995*, pages 24–36.
- Wu, C.-J., Jaleel, A., Hasenplaugh, W., Martonosi, M., Steely, Jr., S. C., and Emer, J. (2011). Ship: Signature-based hit predictor for high performance caching. In *44th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-44*, pages 430–441.
- Zhang, T., Abellan, J. L., Joshi, A., and Coskun, A. K. (2014). Thermal management of manycore systems with silicon-photonics networks. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*.
- Zheng, Y., Lisherness, P., Gao, M., Bovington, J., Yang, S., and Cheng, K.-T. (2012). Power-efficient calibration and reconfiguration for on-chip optical communication. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2012*, pages 1501–1506.
- Zhou, L. and Kodi, A. (2013). Probe: Prediction-based optical bandwidth scaling for energy-efficient NoCs. In *2013 Seventh IEEE/ACM International Symposium on Networks on Chip (NoCS)*, pages 1–8.

CURRICULUM VITAE

Chao Chen

Email: chen9810@bu.edu

Phone: +1(617)899-6852

EDUCATION

- **Doctor of Philosophy**, Computer Engineering
Boston University, Boston, MA, USA
Thesis Topic: Energy-efficient Electrical and Silicon-photonic Networks in Manycore Systems
Advisor: Prof. Ajay Joshi
September 2009 – May 2014
- **Master of Science**, Electronic and Electrical Engineering
Pohang University of Science and Technology, Pohang, Kyungbuk South Korea
Thesis Topic: BP based Parallel Algorithm and VLSI Architecture for Stereo Vision
Advisor: Prof. Hong Jeong
September 2009 – May 2014
- **Bachelor of Engineering**, Electronic Engineering
Shanghai Jiao Tong University, Shanghai, China
September 2001 – JUL 2005
- **Bachelor of Economics**, International Economics and Trade
Shanghai Jiao Tong University, Shanghai, China
September 2001 – JUL 2006

TECHNICAL SKILLS

- Simulators: Gem5, Booksim, McPAT, HotSpot, Graphite and Hornet
- Tested Benchmarks: Splash, PARSEC, NAS and EEMBC
- Design Tools: Virtuoso, Hspice, Scope, Spectre, FPGA, Modelsim, Ncverilog, Spyglass and Specman
- Programming Languages: Verilog, VHDL, System Verilog, System C, C/C++, Assembly, MATLAB, Python, Perl, Shell, Open CV and Java

WORK EXPERIENCE

- **Research Assistant**, Boston University, Boston, MA, USA
 - Proposed a 1000-core architecture that improves the core efficiency by sharing computational resources through high-bandwidth silicon-photonic network
 - Proposed a runtime reconfigurable cache architecture that saves the laser power and cache power by switching ON/OFF redundant cache banks and their associated silicon-photonic channels
 - Proposed a runtime laser power management mechanism that changes the bandwidth of silicon-photonic channels through time-division multiplexing to achieve power-performance targets
 - Proposed a network flow control mechanism (EVC-T) to transmit broadcast packets on the electrical mesh network and a supporting network to maintain the order of broadcast packets on unordered networks
 - Integrated a cycle-precise NoC simulator (BookSim) into an event-based manycore system simulator (Gem5) for analyzing system-level power-performance impacts of new network architectures

May 2010 – May 2014
- **Graduate Teaching Fellow**, Boston University, Boston, MA, USA
 - EC413 Computer Organization, Fall 2009, supervisor: Prof. Martin Herbordt
 - EC450 Microprocessors, Spring 2010, supervisor: Prof. Roscoe Giles

September 2009 – May 2010
- **SOC Engineer**, Core Logic Inc., Seoul, South Korea
 - Designed an AMBA 3.0 bus interface for SOC interconnects
 - Designed a display module to switch between preview and 1080p HD modes
 - Designed a test module to monitor the bandwidth and latency of AMBA bus
 - Integrated SOC peripherals including DSP, H264 Codec, DMAC and USB 2.0

October 2007 – September 2009
- **Research Assistant**, Pohang University of Science and Technology, Pohang, Kyungbuk, South Korea
 - Designed a FPGA system for stereo matching using Belief Propagation

May 2010 – May 2014

PUBLICATION

1. Chao Chen, Ajay Joshi and Erno Salminen, "Profiling EEMBC MultiBench Programs using Full-system Simulations," Proc. the 5th Workshop on SoCs, Heterogeneous Architectures and Workloads (SHAW-5), held in conjunction with the 20th International Symposium On High Performance Computer Architecture (HPCA-20) 2014.
2. Chao Chen and Ajay Joshi, "Runtime Management of Laser Power in Silicon-Photonic Multibus NoC Architecture," Selected Topics in Quantum Electronics, IEEE Journal of, vol.19, no.2, pp.338,350, March-April 2013
3. Chao Chen, Ajay Joshi, and Erno Salminen, "Profiling EEMBC MultiBench Programs in 64-core Machine," white paper, published by Open Core Protocol International Partnership (OCP-IP), June 2013.
4. Ajay Joshi, Chao Chen, Zafar Takhirov, and Bobak Nazer, "A Multi-layer Approach to Green Computing: Designing Energy-efficient Digital Circuits and Manycore Architectures," Proc. Workshop on Lighter-than-Green Dependable Multicore Architectures (LGDMA), Held in conjunction with International Green Computing Conference (IGCC) 2012
5. Chao Chen, Jie Meng, Ayse K. Coskun, and Ajay Joshi, "Express Virtual Channels with Taps (EVC-T): A Flow Control Technique for Network-on-Chip (NoC) in Manycore Systems," Proc. the 19th International Symposium on High-Performance Interconnects, HOTI 2011.
6. Jie Meng, Chao Chen, Ayse K. Coskun, and Ajay Joshi, "Run-time Energy Management of Manycore Systems through Reconfigurable Interconnects," Proc. the 21st edition of the Great Lakes Symposium on VLSI, GLSVLSI 2011.
7. Sungchan Park, Chao Chen, Hong Jeong, and Sang Hyun Han, "Real-time Stereo Matching Architecture Based on 2D MRF Model: A Memory-efficient Systolic Array," EURASIP Journal on Image and Video Processing 2011 2011:4.
8. Sungchan Park, Chao Chen, and Hong Jeong, "Stereo Matching using FBP: A Memory Efficient Parallel Architecture," Proc. International Conference on Image Processing, Computer Vision, and Pattern Recognition, IPCV 2008.
9. Sungchan Park, Chao Chen, and Hong Jeong, "VLSI Architecture for MRF based Stereo Matching," Proc. the 7th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS 2007.
10. Youngsu Kim, Sungchan Park, Chao Chen, and Hong Jeong, "Real-time Architecture of Stereo Vision for Robot Eye," Proc. the 8th International Conference on Single Processing, ICSP 2006.

AWARDS

- Silver prize, the 8th Korean Semiconductor Design Contest, Korean Intellectual Property Office, 2007
- Excellent Bachelor Design, Shanghai Jiao Tong University, 2005
- Exceptional Student Honor, Shanghai Jiao Tong University, 2003
- First prize, China Physics Olympiad, Chinese Physical Society, 2000
- Second prize, China Mathematics Olympiad, Chinese Mathematical Society, 2000
- Third prize, China Chemistry Olympiad, Chinese Chemical Society, 2000

SCHOLARSHIPS

- Research Assistant, Boston University, 2010 – 2014
- Graduate Teaching Fellowship, Boston University, 2009 – 2010
- Korean Government IT Scholarship for International Graduate Student, 2005 – 2007
- Peoples Scholarship, Shanghai Jiao Tong University, 2002 – 2005

LANGUAGE

- English
- Mandarin Chinese
- Shanghai Dialect, Chinese