Energy-Efficient and High-Performance Nanophotonic Interconnects for Shared Memory Multicores

A dissertation presented to
the faculty of
the Russ College of Engineering and Technology of Ohio University

In partial fulfillment
of the requirements for the degree
Doctor of Philosophy

Randy W. Morris, Jr.
June 2012
© 2012 Randy W. Morris, Jr. All Rights Reserved.
This dissertation titled
Energy-Efficient and High-Performance Nanophotonic Interconnects for Shared Memory Multicores

by
RANDY W. MORRIS, JR.

has been approved for
the School of Electrical Engineering and Computer Science
and the Russ College of Engineering and Technology by

_________________________________________________________________________

Avinash Karanth Kodi
Assistant Professor of Electrical Engineering and Computer Science

_________________________________________________________________________

Dennis Irwin
Dean, Russ College of Engineering and Technology
ABSTRACT

MORRIS, JR., RANDY W., Ph.D., June 2012, Electrical Engineering

Energy-Efficient and High-Performance Nanophotonic Interconnects for Shared Memory Multicores (154 pp.)

Director of Dissertation: Avinash Karanth Kodi

By 2018, both industry and academic leaders expect computer performance to reach the exascale level, allowing $10^{18}$ floating point calculations per second (FLOP). However, exascale computing can only exist if high-performance and energy-efficient chip multiprocessors (CMPs) can be realized. It is well-known that the power dissipation of metallic interconnects in future multicore architectures is projected to be a major bottleneck as we scale to the sub-nanometer regime. This has forced researchers to focus their attention on developing alternate power-efficient technology solutions for performance limitations of future multicore architectures. Nanophotonic interconnects are a disruptive technology solution that is capable of delivering the communication bandwidth at low power dissipation when the number of cores is scaled to large numbers. Furthermore, recent advances in complementary metal-oxide semiconductor (CMOS) compatible devices and circuits development have made nanophotonics a viable solution for on-chip applications. This dissertation proposes two nanophotonic architectures called 3D-NoC and PULSE. These architectures alleviate two major problems facing current multicores, which are to improve performance and programmability respectively. 3D-NoC combines the benefits of 3D integration with nanophotonics to construct a high-performance high-core (256 cores) CMP. In addition, to further maximize the performance of 3D-NoC, this dissertation proposes a reconfiguration algorithm whose purpose is to improve performance by adapting available network bandwidth to application demands. This is accomplished by monitoring the traffic load and applying a reconfiguration algorithm that works in the background without disrupting the on-going communication. PULSE is a tree-based broadcast network
which combines/splits optical signals using a combination of couplers and splitters in such a way that the same intensity light arrives at all the cores simultaneously ensuring the ordering required for snoopy protocols. In addition, this dissertation proposes a photonic cache filtering technique called multi-PULSE. This allows the broadcast network to rapidly morph into a multicast network by directing the address request to only those cores that actually share the block. Multi-PULSE allows for a reduction in cache access power as only the cores that have the cache block will receive the request. Moreover, 3D-NoC and PULSE are compared to other leading electrical and nanophotonic architectures using synthetic traffic, SPEC CPU2006, Splash-2 and PARSEC benchmarks. When PULSE is compared to other leading nanophotonic and electrical broadcast networks, simulation results show PULSE demonstrates a speed-up of 55% and power savings of 80% over other leading networks. On the other hand, when 3D-NoC is compared to other leading nanophotonic networks, simulation results indicate that 3D-NoC can further improve Splash-2, Parsec, and SPEC CPU2006 benchmarks by 10%-25%.

Approved: 

Avinash Karanth Kodi
Assistant Professor of Electrical Engineering and Computer Science
I would like to dedicate this Ph.D dissertation to my loving parents, Randy W. Morris, Sr.
and Cheryl L. Morris.
ACKNOWLEDGMENTS

I would first like to thank my parents who gave me the chance to go to college. If it was not for them then I would not be the first in my whole family with a PhD degree. Second, I would to thank my advisor, Dr. Avinash Kodi, for helping me through my PhD studies and putting up with my lack of writing skills. Next, I would like to thank Dr. Radcliff for helping my obtain a Stocker Fellowship for my masters studies. This started the ball rolling for wanting to get a PhD degree. Finally, I would like to thank my committee members Dr. Whaley, Dr. Kaya, Dr. Dill, Dr. Ingram, and Dr. Jung for making my PhD degree possible.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>3</td>
</tr>
<tr>
<td>Dedication</td>
<td>5</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>6</td>
</tr>
<tr>
<td>List of Tables</td>
<td>9</td>
</tr>
<tr>
<td>List of Figures</td>
<td>10</td>
</tr>
<tr>
<td>List of Acronyms</td>
<td>13</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>16</td>
</tr>
<tr>
<td>1.1 Network-on-Chip</td>
<td>18</td>
</tr>
<tr>
<td>1.2 Nanophotonics</td>
<td>20</td>
</tr>
<tr>
<td>1.3 Previous Research</td>
<td>23</td>
</tr>
<tr>
<td>1.4 Research Agenda</td>
<td>26</td>
</tr>
<tr>
<td>1.4.1 Improving Performance: 3D-NoC with Reconfiguration</td>
<td>26</td>
</tr>
<tr>
<td>1.4.2 Programmability: Energy-Efficient Snoopy Cache Coherent-Multiprocessor</td>
<td>27</td>
</tr>
<tr>
<td>1.5 Major Contributions of this Dissertation</td>
<td>28</td>
</tr>
<tr>
<td>2 Photonic 3D-NoC</td>
<td>30</td>
</tr>
<tr>
<td>2.1 3D CMOS Integration</td>
<td>30</td>
</tr>
<tr>
<td>2.1.1 Through Silicon Vias</td>
<td>31</td>
</tr>
<tr>
<td>2.1.2 Fabrication Issues</td>
<td>33</td>
</tr>
<tr>
<td>2.2 3D Nanophotonic Interconnect</td>
<td>34</td>
</tr>
<tr>
<td>2.3 3D-NoC Architecture</td>
<td>37</td>
</tr>
<tr>
<td>2.3.1 Network Topology</td>
<td>37</td>
</tr>
<tr>
<td>2.3.2 Communication</td>
<td>41</td>
</tr>
<tr>
<td>2.3.3 Router Architecture</td>
<td>44</td>
</tr>
<tr>
<td>2.3.4 Communication Pipeline</td>
<td>46</td>
</tr>
<tr>
<td>2.3.5 Optical Tokens</td>
<td>48</td>
</tr>
<tr>
<td>2.4 Reconfiguration</td>
<td>50</td>
</tr>
<tr>
<td>2.4.1 Reconfigurable 3D-NoC</td>
<td>51</td>
</tr>
</tbody>
</table>
3 Broadcast Network ........................................ 61
  3.1 Parallel Programming Models ......................... 61
  3.2 Cache Coherency ......................................... 62
    3.2.1 Snoopy Protocols .................................. 64
    3.2.2 Previous Research (Broadcast Networks) .......... 66
  3.3 PULSE .................................................... 68
    3.3.1 System Architecture ............................... 68
    3.3.2 Token Arbitration and Power Guiding .............. 71
    3.3.3 Multi-PULSE ...................................... 73
    3.3.4 Multi-PULSE Control Circuitry .................... 76
    3.3.5 Handling a L2 miss ................................ 79
    3.3.6 Example of Generating a Mask ..................... 79
    3.3.7 Mask Network .................................... 81
  3.4 Multi-PULSE Timing Diagram ............................. 83

4 Performance Results ........................................ 86
  4.1 3D-NoC Results .......................................... 86
    4.1.1 Simulation Setup .................................. 86
    4.1.2 Simulation Results ................................. 92
    4.1.3 Energy Comparison .................................. 100
    4.1.4 Area Analysis ...................................... 107
  4.2 PULSE Results ........................................... 110
    4.2.1 Simulation Setup .................................. 110
    4.2.2 Simulation Results .................................. 110
    4.2.3 Optical Losses ..................................... 115
    4.2.4 Energy Analysis .................................... 116
  4.3 Sensitivity Analysis ..................................... 121
    4.3.1 Bit Error Rate ..................................... 122
    4.3.2 Temperature Sensitivity ............................ 124
    4.3.3 Fabrication Sensitivity .............................. 130

5 Conclusions and Future Direction .......................... 132
  5.1 Results and their Implication .......................... 132
  5.2 Shortcomings of this Dissertation ..................... 133

6 Publications ................................................ 135
  6.1 Published Work ......................................... 135
  6.2 Papers Under Review .................................... 136
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Electrical and optical area overhead for select electrical and optical components</td>
<td>42</td>
</tr>
<tr>
<td>2.2 Pseudo Algorithm used by 3D-NoC</td>
<td>60</td>
</tr>
<tr>
<td>3.1 Multicast algorithm for generating a bit vector that is used to drive the multicast micro-ring resonators</td>
<td>80</td>
</tr>
<tr>
<td>4.1 Splash-2 suite application and kernel problem sizes used in simulation</td>
<td>90</td>
</tr>
<tr>
<td>4.2 Core and cache parameters used for Splash-2, PARSEC, and SPEC CPU2006 application on SIMICS using GEMS</td>
<td>92</td>
</tr>
<tr>
<td>4.3 Electrical and optical power losses</td>
<td>103</td>
</tr>
<tr>
<td>4.4 Electrical power dissipation for Nanophotonic interconnects</td>
<td>103</td>
</tr>
<tr>
<td>4.5 Electrical and optical area overhead for select electrical and optical components</td>
<td>109</td>
</tr>
<tr>
<td>4.6 Full-system configuration</td>
<td>111</td>
</tr>
<tr>
<td>4.7 Average number of sharers for select Splash-2 and PARSEC benchmarks</td>
<td>112</td>
</tr>
<tr>
<td>4.8 Percentage of multicast misses for select Splash-2 and PARSEC applications</td>
<td>115</td>
</tr>
<tr>
<td>4.9 Address network broadcast network delays (cycles)</td>
<td>117</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Decline in uniprocessor performance gains via ILP and higher clocking [1]</td>
<td>17</td>
</tr>
<tr>
<td>1.2</td>
<td>Example of an NoC (mesh topology).</td>
<td>19</td>
</tr>
<tr>
<td>1.3</td>
<td>(a) The tile power for Intel’s TeraFLOPS processor [2] and (b) the total power dissipation versus performance for Intel’s TeraFLOPS processor [2]</td>
<td>20</td>
</tr>
<tr>
<td>1.4</td>
<td>A typical nanophotonic interconnect, which consist of both electrical circuitry for modulation and detection and optical components for guiding optical data from source to destination nodes.</td>
<td>21</td>
</tr>
<tr>
<td>1.5</td>
<td>Micro-ring resonator operation (a) when $V_R = V_{OFF}$ and (b) when $V_R = V_{ON}$</td>
<td>22</td>
</tr>
<tr>
<td>2.1</td>
<td>3D CMOS fabrication: (a) face-to-face bonding and (b) face-to-back bonding [3].</td>
<td>32</td>
</tr>
<tr>
<td>2.2</td>
<td>3D CMOS fabrication showing TSVs connecting the different layers.</td>
<td>33</td>
</tr>
<tr>
<td>2.3</td>
<td>SIMOX fabrication process [4]: (a) silicon dioxide mask is place on top of the silicon-on-insulator wafer and oxygen ion are implanted into the wafer, (b) annealing embeds the silicon dioxide layer into the silicon wafer, (c) addition silicon is grown on top of wafer using epitaxy, and (d) the process repeats to create addition layers.</td>
<td>36</td>
</tr>
<tr>
<td>2.4</td>
<td>Illustration of an optical via built using micro-ring resonators (a) 'OFF' state, (b) switch optical signals to layer 1, (c) switch optical signals to layer 2 and (d) switch optical signals to layer 3.</td>
<td>37</td>
</tr>
<tr>
<td>2.5</td>
<td>Proposed 256-core 3D chip layout. (a) Electrical die consists of the core, caches, the memory controllers and TSVs to transmit signals between the two dies, and (b) 3D chip with four decomposed nanophotonic crossbars with an insert showing a tile with shared L2 cache and 4 cores.</td>
<td>38</td>
</tr>
<tr>
<td>2.6</td>
<td>The network layout for: (a) Corona[5] and (b) Firefly[6].</td>
<td>41</td>
</tr>
<tr>
<td>2.7</td>
<td>The layout of three different waveguides for Layer 0 of 3D-NoC.</td>
<td>43</td>
</tr>
<tr>
<td>2.8</td>
<td>The layout of three different waveguides for (a) Layer 1, (b) Layer2 and (c) Layer3 of 3D-NoC.</td>
<td>45</td>
</tr>
<tr>
<td>2.9</td>
<td>Router microarchitecture for a tile.</td>
<td>46</td>
</tr>
<tr>
<td>2.10</td>
<td>Router pipeline for data going from a source tile to a destination tile.</td>
<td>49</td>
</tr>
<tr>
<td>2.11</td>
<td>(a) Token layout for an optical communication channel and (b) token control block.</td>
<td>50</td>
</tr>
<tr>
<td>2.12</td>
<td>Group connections for Layer0, Layer 1 and Layer 2.</td>
<td>53</td>
</tr>
<tr>
<td>2.13</td>
<td>Illustration of reconfiguration where additional bandwidth is given to Group 0 to communicate with Group 3. (a) Inter-layer static and dynamic bandwidth re-allocation and (b) 3D switching of bandwidth.</td>
<td>54</td>
</tr>
</tbody>
</table>
Various configurations evaluated: (a) 3D-NOC-L1 (3D-NOC-Limited to 1 Layer), (b) 3D-NOC-LA (3D-NOC-Limited to adjacent layer), (c) 3D-NOC-L2 (3D-NOC-Limited to 2 Layers) and (d) 3D-NOC-L3 (3D-NOC-All 3 Layers) .................................................. 56

Illustration of reconfiguration hardware: (a) intra-group hardware where each tile has hardware counters to determine interconnect statistics and (b) inter-group RC hardware .................................................. 59

The memory hierarchy from the core to the interconnection network: (a) one core tile and (b) a four core tile .......................................................... 63

Cache coherence problem: (a) P1 reads memory location U from main memory and places the value of 5 in its local cache, (b) processor P3 also reads memory location U from main memory and places the value of 5 in its local cache, (c) processor P3 writes the value of 7 for memory location U into its cache, and (d) processor P1 and processor P2 read the memory location U and again read a value of 5 (should read a value of 7) [7] ................................. 65

A snooping cache-coherent multiprocessor [7] ........................................ 66

The address and data network nanophotonic interconnect layout for 16 cores .......................................................... 69

Design of the proposed token network showing the control center and related waveguides (return, inject and power) .................................................. 70

Control block operation: (a) initial optical token injection coming from the control block, (b) optical power supplied to power waveguide for column one, (c) after some time the optical token returns to the control center and shifts the registers, and (d) the control center injects the optical token into the next column of tiles .................................................. 72

Additional network components required for multicasting .......................................................... 75

Correctly predicted multicast example and (b) mispredicted multicast example .................................................. 77

(a) E/O circuitry required within the control block for multicasting and (b) Circuit required between the directories and control block for correcting wrong multicasts .................................................. 78

L2 cache miss flow-chart .......................................................... 81

Mask generation example, where L2 request for block address FFF1 generates a mask from addresses FFF0, FFF1 and FFF2 .................................................. 82

Cache sharing network used by the directories to send caching information to the tiles .................................................. 83

Timing diagram for (a) broadcast, (b) multicast, and (c) multicast with error .................................................. 85

Reconfiguration window $R_W$ size versus improvement in performance for blackscholes .................................................. 93

Simulation speed-up for 64-core using 64-wavelengths: (a) SPLASH-2, (b) PARSEC and (c) SPEC CPU2006 .................................................. 95
4.3 Simulation speed-up for 64-core using 8-wavelengths: (a) SPLASH-2, (b) PARSEC and (c) SPEC CPU2006. .......................................................... 97
4.4 Simulation speed-up for 64-core using 16-wavelengths: (a) SPLASH-2, (b) PARSEC and (c) SPEC CPU2006. .................................................. 98
4.5 Normalized saturation throughput for synthetic traffic patterns for 256 cores. .......................................................... 99
4.6 Workload completion traffic speed-up over mesh interconnect. .......................................................... 100
4.7 Average energy per-bit for electrical and nanophotonic interconnects: (a) uniform traffic, (b) complement traffic and (c) butterfly traffic. .......................................................... 105
4.8 (a) Increase in laser power for different values of receiver sensitivity and number of wavelengths, and (b) Increase in laser power for different values of waveguide loss and ring filtering loss. .......................................................... 106
4.9 Energy delay product for 64-core versions: (a) SPLASH-2 traffic traces, (b) PARSEC, and (c) SPEC CPU2006 traffic traces. .......................................................... 108
4.10 Speed-up running (a) Splash-2 and (b) PARSEC applications for 16 cores. .......................................................... 113
4.11 Speed-up running (a) Splash-2 and (b) PARSEC applications for 64 cores. .......................................................... 114
4.12 Energy per bit required to broadcast a bit of data. .......................................................... 118
4.13 Potential power savings for select (a) Splash-2 and (b) PARSEC applications (16-core). .......................................................... 119
4.14 Potential power savings for select (a) Splash-2 and (b) PARSEC applications (64-core). .......................................................... 120
4.15 Contour plot showing the increase in laser power when receiver sensitivity and total optical losses are varied. .......................................................... 121
4.16 Contour lines showing the variation of total laser power for different waveguide crossing losses. .......................................................... 122
4.17 OptiSystem setup for 3D-NoC. .......................................................... 124
4.18 Eye diagram for different bandwidths: (a) 5 gbps, (b) 10 gbps and (c) 12 gbps. The optical power vs time (d) at the photodetector for a 10 gbps signal. .......................................................... 125
4.19 Frequency spectrum diagrams: (a) the first 8 wavelengths coming from the laser and (b) the frequency spectrum at the photodetector after passing a micro-ring filter. .......................................................... 126
4.20 Shift in micro-ring resonator resonance wavelength due to an increase in micro-ring temperature (a). A 1 gbps voltage input signal (b) used to modulate a micro-ring resonator, and (c) the signal received at the photodiode assuming no shift in micro-ring temperature. Assuming a 5°C shift in micro-ring temperature, (d) the signal received at the photodiode. .......................................................... 127
4.21 Micro-ring resonator with ring heating element attached. .......................................................... 128
4.22 Micro-ring resonator heating power dissipation for different electrical networks. .......................................................... 129
4.23 Increase in laser power versus fabrication tolerance of micro-ring resonators: (a) 3D-NoCs and (b) PULSE. .......................................................... 131
**LIST OF ACRONYMS**

**BER** - Bit Error Rate

**BC** - Buffer Control

**BWD** - Buffer Write Destination

**BWS** - Buffer Write Source

**CMOS** - Complementary Metal-Oxide Semiconductor

**CMP** - Chip Multi-Processor

**DSM** - Distributed Shared Memory

**DOR** - Dimensional Order Routing

**DRAM** - Dynamic Random Access Memory

**EMI** - Electromagnetic Interference

**EO** - Electrical to Optical

**FFT** - Fast Fourier Transform

**FB** - Flattened-Butterfly

**HPC** - High Performance Computing

**IA** - Intel Architecture

**IB** - Input Buffers

**ILP** - Instruction Level Parallelism

**ITRS** - International Technology Roadmap for Semiconductors

**IC** - Intergraded Circuit

**KNC** - Knights Corner

**LLC** - Lowest Level Cache

**LU** - Lower Upper factorization

**MC** - Memory Controllers
MPI - Message Passing Interface
MSB - Most Significant Bit
MWSR - Multiple Write Single Read
MZ - Mach Zehnder Modulator
NIC - Network Interface Card
NOC - Network-on-Chip
NUCA - Non-Uniform Cache Access
OE - Optical to Electrical
OMPI - Open Message Passing Interface
OL - Optical Link Latency
QPI - Quick Path Interconnect
PARSEC - Princeton Application Repository for Shared-Memory Computers
RC - Routing Computation
RMS - Recognition, Mining, and Synthesis
$R_w$ - Reconfiguration Window
SA - Switch Allocation
SIMOX - Separation by IMplantation of Oxygen
SMP - Symmetric MultiProcessors
SNR - Signal to Noise Ratio
SWMR - Single Write Multiple Read
SPEC - Standard Performance Evaluation Corporation
SPLASH-2 - Stanford Parallel Applications for SHared memory
SPRT - Static Power Reduction Technique
SCC - Single-Chip Cloud Computer
**TI** - Token Inject

**TIA** - Transimpedance Amplifier

**TF** - Tera-Flop

**TR** - Token Release

**TSV** - Through Silicon Via

**VCSEL** - Vertical Cavity Surface Emitting Laser

**VSC** - Vector Sharing Cache

**WDM** - Wavelength Division Multiplexing
1 Introduction

Prior to year 2000, micro-processor performance had been increasing at a steady rate due to instruction level parallelism (ILP) and increase in clock frequencies [8–10]. However, due to diminishing returns from ILP and excessive power dissipation from higher clock frequencies, micro-processor performance has been increasing at a much slower rate after the year 2000 [8, 9]. Figure 1.1 shows the average latency (in pico-seconds) required to execute an instruction for different processors in successive years. The average latency to complete an instruction from 1980 to about 2000 had been decreasing at a rate of 52% per year, mainly from ILP and higher clock frequencies [1]. But, after the year 2000, the average latency has been decreasing at a slower rate of 19% per year. Figure 1.1 illustrates the gap in performance between a theoretical (Linear) and actual (Perf.) decrease in the average latency to complete an instruction. There is 1000× slowdown between the theoretical and actual latencies to complete an instruction in the current year (2012). Furthermore, it is expected that this slowdown will dramatically increase to 30000× by year 2020. In order to reduce this performance gap, researchers and engineers have shifted their attention to a multi-core paradigm to overcome the limitations of uniprocessors [8, 9, 11].

Multi-core processors allow for continuous increase in performance by distributing the application workload among the different processing cores. Moreover, with technology scaling, additional cores can be placed on-chip allowing for continual increase in computational power [8, 9, 11]. Presently, almost the entire computing industry is moving towards multicores; for example, Intel’s Core i7 has 6 cores with 12 threads [12], AMD’s Phantom II has 6 cores with 6 threads [13], Tilera has 64 and 100-core processors integrated on a chip [14], and so on. Clearly, multicores have proven to be a design paradigm that could alleviate the bottlenecks of uniprocessors while improving the performance of future multi-threaded applications.
Figure 1.1: Decline in uniprocessor performance gains via ILP and higher clocking [1].

With increasing core counts, the projected performance of multi-core processors is expected to increase substantially. Currently, Intel’s Knights Corner (KNC) has demonstrated one Tera FLOPS (TF) of computing power, which implies one trillion double precision floating point operations per second [15]. By the year 2018, the computing power of multi-core processors is expected to increase 10 fold to 10 TF of computing power [16]. As the number of cores increases (higher performance), the interconnection network will have a greater impact on the overall chip performance. Furthermore, with multi-core processors expected to be comprised of 100s and even 1000s of cores, simple topologies such as a shared bus will not be capable of providing the required network bandwidth. For example, the increase in throughput demand from Intel’s processors resulted in Intel shifting their interconnection network from a shared bus to a fully connected topology called Quick Path Interconnect (QPI). A fully connected topology, such as QPI, is capable of providing the required network performance for Intel’s current generation processors (scalable to 4 or 8 cores). However, as the number of cores increase, fully connected
network topologies are not feasible due to their poor scalability and fabrication complexity [11]. In order to meet the performance challenges required by future highly integrated multi-core processors, the interconnection network must adopt a more scalable and modular network-on-chip (NoC) architecture [11].

1.1 Network-on-Chip

NoCs for multi-core processors allow for the creation of high speed networks using routers and links that are able to scale to large cores [9–11, 17–21]. Figure 1.2 shows a mesh topology, where a core is connected to four neighboring cores [17, 18]. Each core has its own private cache memory (L1 and L2), where the L2 cache is connected to a router. The combination of a core and caches (L1 and L2) is called a tile. L1 cache is further divided into L1I (instruction) and L1D (data) cache that the processor loads or stores instructions or data to/from memory. The memory controllers on top of the chip are used as gateways to access the main memory. Inter-core traffic is injected into the network by the L2 cache. Depending on its destination, the request will traverse the mesh network in dimension-order routing (DOR) technique, where traffic will route in the x-direction first and then in the y-direction to the destination [17]. In addition, Figure 1.3 illustrates the interworking of a router which comprises of routing logic, buffers, and a crossbar.

Recent research has shown that the power consumption in NoCs is expected to become a major issue facing multi-core processor performance. Furthermore, electrical interconnects are plagued with problems such as, electromagnetic interference (EMI), crosstalk, and clock skew that worsen as transistors scale to the sub-nanometer regime [22–25]. For example, in the TeraFLOPS processor architecture, which is an 80-core processor fabricated by Intel for research purposes, 28% of the total on-chip power was dissipated by the NoC. Research shows optimal performance occurs when 10% of power is used for communication [2]. Figure 1.3(a) illustrates the breakdown of power consumption for
each tile and router used in the TeraFlops processor, where 28% of power dissipated by a tile is from routers and links. Moreover, Figure 1.3(b) illustrates a quadratic increase in TeraFLOPS power dissipation when compared to processor performance. For example, to achieve one TF of performance, the total power dissipation of the chip is 97 watts. If the performance was increased to 1.33 TF the total power dissipation would increase to 230 watts. There is a 2.3 fold increase in power for only a 1.33 increase in performance. Intel recently demonstrated an experimental Single-Chip Cloud Computer (SCC) which has 48 Intel Architecture (IA) cores that resembles a cloud of computers on a single chip [26]. SCC supports message passing [7] and incorporates advance power management techniques that allows the chip to run in a high performance mode (125 watts) or a low performance mode (25 Watts) to conserve energy. In order to overcome the high power
Figure 1.3: (a) The tile power for Intel’s TeraFLOPS processor [2] and (b) the total power dissipation versus performance for Intel’s TeraFLOPS processor [2].

The dissipation issue facing NoCs, engineers and researchers will be forced in the future to adopt an alternative disruptive technology over electrical interconnects [27].

1.2 Nanophotonics

One potential disruptive technology under serious consideration for on-chip communication is nanophotonics. Nanophotonics have several advantages over electronics such as (1) high bit rates (bandwidth), (2) performance that is not dependent on distance, (3) large bandwidth density, and (4) low power dissipation [5, 6, 28–34]. Moreover, recent advantages have been demonstrated in complementary metal-oxide semiconductor (CMOS) compatible nanophotonic devices with characteristics such as a small footprint (~ 12μm)), low energy (~ 3 fJ/bit) [35, 36] and high bandwidth (~ 12.5 Gbps/channel) [5, 22, 28, 29, 35, 37–48]. Furthermore, nanophotonics allow for multiple communication channels (wavelengths) to traverse down a single waveguide through wavelength division multiplexing (WDM), creating high bandwidth interconnects which can not be found in electrical interconnects.

A nanophotonic interconnect, shown in Figure 1.4, consists of (1) a laser for generating the optical signal required for data modulation, (2) an optical modulator used to modulate
Figure 1.4: A typical nanophotonic interconnect, which consists of both electrical circuitry for modulation and detection and optical components for guiding optical data from source to destination nodes.

data onto the optical signal, (3) an optical medium for guiding the light from source to destination, (4) a photodetector for converting the optical signal back to an electrical signal, and (5) electrical circuitry for recovering the electrical data.

Micro-ring resonators and Mach-Zehnder (MZ) modulators are two potential modulators used for on-chip applications. MZ modulators operate by diverting two beams of light through two different waveguides and then combines the two beams together [49]. As the beams are propagating, one of the optical beams traverses a waveguide whose refractive index changes on application of a bias voltage [49]. When the two optical beams combine together, they either intersect constructively or destructively with each other, causing an on or off modulation. On the other hand, micro-ring resonators are the most common devices used to encode data onto an optical signal and will only couple light into waveguide/fiber if the incoming light satisfies the relation $\lambda \times m = n_{\text{eff}} \times 2\pi R$, where $R$ is micro-ring resonator radius, $n_{\text{eff}}$ is the effective refractive index of the micro-ring resonator circular waveguide, $m$ an integer and $\lambda$ is the resonant wavelength [37, 50–52]. By changing $n_{\text{eff}}$, the resonant
wavelength of the micro-ring resonator can be changed, enabling it to function as an optical modulator (on/off keying). One reason why micro-ring resonators are preferred over MZ modulators is micro-ring resonators have smaller footprints (~12 μm), lower energy (~3 fJ/bit) [35, 36] and higher bandwidth (~12.5 Gbps/channel) over MZ modulators. Currently, micro-ring resonators have been demonstrated with extinction ratios greater than 9 dB, optical losses as low as -0.12 dB/cm, and modulator insertion loss of 1 dB, which are sufficient for the receiver design used in nanophotonic NoCs [25, 28, 35, 49–51, 53–58]. Figure 1.5 shows the two different operations of a micro-ring resonator. When a voltage is applied, free-carriers are injected into the micro-ring resonators’s waveguide through a process called free-carriers plasma dispersion [37]. These free-carriers change the effective index of the waveguide that causes the resonance wavelength to shift. In Figure 1.5(a), the voltage \( V_R \) is switched off resulting in a shift in the resonant frequency which allows the light to pass through. In Figure 1.5(b), the voltage \( V_R \) is switched on, resulting in the incoming light to be in resonance with the micro-ring resonator, which allows the light to be shifted to the drop port. By controlling the voltage \( V_R \), we can use micro-ring resonators as modulators, filters, etc.
1.3 Previous Research

As nanophotonics has several advantages for on-chip applications, there exists considerable previous works on nanophotonic research that have shown several network designs that can overcome the limited bandwidth and high power dissipation of electrical interconnects. A few of these networks will be explained in more detail: Corona [5], Firefly [6], MPNOC [59, 60], Circuit-Switch [29, 61, 62], ATAC [63], and FlexiShare [64].

(a) Corona

The Corona network is a 256-core optical bus network. Corona’s optical bus is constructed by using 64 multiple write single read (MWSR) nanophotonic channels, where many tiles can write onto the optical channel but only one tile can read the channel. In order to prevent two or more tiles from communicating at the same time, Corona uses optical tokens to only allow one tile to communicate at a time. An optical token is a burst of optical light that circulates through all the communicating tiles. When a tile needs to communicate with the destination tile, it will activate a micro-ring resonator and try to capture the circulating optical token. The issue with Corona is the high contention for optical tokens when two or more tiles require to communicate to the same destination.

(b) Firefly

The Firefly network is an electro-optical network, that uses cheaper electronics to route data to local tiles, and nanophotonics to route data to global tiles. Optical channels within Firefly are constructed using single write multiple read (SWMR) nanophotonic channels. In a SWMR nanophotonic channel, a single tile can write on the optical channel but multiple tiles can read the channel. To prevent tiles within a SWMR nanophotonic channel from receiving a signal that is not destined for them, Firefly implements a reservation system that activates micro-ring resonators to guide the optical signals to the correct destination.
tile. Firefly strikes a balance between cheaper electronics for local communication and nanophotonics for global communication. However, the issue with Firefly is the higher power dissipation required for data to traverse over the electrical network and the latency penalty due to the reservation system.

(c) MPNOC

The MPNOC network is a 3D nanophotonic NoC that uses multiple layers to create a crossbar with no optical waveguide crossover [59, 60]. MPNOC consists of 16 nanophotonic crossbars that are divided across four optical layers. MPNOC uses electrical through silicon vias (TSVs) to modulate optical signals on different nanophotonic layers. 3D-NoC, on the other hand uses optical vias instead of electrical TSVs to switch optical signals among different layers, thereby reduces the power dissipation. Optical vias also ease the fabrication of multiple optical layers as opposed to using electrical TSVs. 3D-NoC further improves the performance of decomposed crossbars by implementing a reconfiguration algorithm that works without disrupting the on-going communication.

(d) Circuit-Switch

The Circuit-Switch network uses a simple electrical network to setup and tear down a high speed optical circuit switched torus network. In the Circuit-Switch network, when a source tile needs to communicate with a destination tile, the source tile will send electrical data though a circuit setup network that activates the correct micro-ring resonators for guiding the optical data to the correct destination. After the destination tile receives the optical data, the destination tile sends electrical data in reverse (from destination to source) that tears down the optical network. The issue with this network is the increased delay for setting up the optical circuit network for small data packets and increased blocking delay due to contention for shared channels.
(e) **ATAC**

The ATAC network [63] is a 1024 core NoC that uses both an electrical mesh network to connect 16-core local groups together and a global optical network to connect 64, 16-core local groups. ATAC uses WDM for its optical network to create a fully connected arbitration-free network. During optical communication, a source tile communicates with a destination tile using a unique wavelength of light that is only used for this combination pair. Further increase in bandwidth can be implemented by adding more optical layers. The issue with the ATAC network is the high cost in terms of optical components, due to the complexity of a fully connected network (increase optical losses and large number of optical devices).

(f) **FlexiShare**

The FlexiShare network [64] is an optical crossbar network that combines the benefits of MWSR nanophotonic channels with SWMR nanophotonic channels. MWSR nanophotonic channels allow for multiple tiles to write data on a single communication channel but only one tile can read the data. SWMR nanophotonic channels allow one tile to write data and several tiles to read the data at once. By combining the benefits of MWSR and SWMR nanophotonic channels, a tile can use any nanophotonic channel to transmit data to any tile. To prevent two or more tiles from transmitting on the same nanophotonic channels, FlexiShare uses optical tokens similar to Corona. Once a tile captures an optical token, FlexiShare uses a technique similar to Firefly’s reservation system to prevent the incorrect tiles from receiving the data. The major advantage of FlexiShare is the ability to reduce the number of nanophotonic channels used in the network, as each nanophotonic channel is connected to all the tiles. The issue with Flexishare is the high number of micro-rings resonators required for each nanophotonic channel and the high optical losses due to long waveguides.
1.4 Research Agenda

This dissertation extends the design and implementation of nanophotonic interconnects within the framework of multicore architectures by tackling two critical problems of improving performance of the resource-constrained multicore networks and reducing the complexity of programming cache-coherent shared-memory multicores. Network performance is improved by combining the advantages of 3D integration that allows an increase in bandwidth density and applying reconfiguration techniques that dynamically re-allocate bandwidth without user interaction. On the other hand, reduction in parallel programming complexity is achieved by leveraging the advantages of nanophotonic interconnects to design high-bandwidth and low-complexity broadcast networks for cache coherent multicores.

1.4.1 Improving Performance: 3D-NoC with Reconfiguration

Current nanophotonic NoCs are 2D designs that are plagued with high optical losses due to waveguide crossings or long snake-like waveguides that coil around the chip to prevent waveguide crossings altogether. For example, a nanophotonic channel with 100 waveguide crossings will have a -5 dB loss if we assume a -0.05 dB loss per waveguide crossing [28, 30]. An alternative technology to overcome the high optical losses and the limited bandwidth of 2D designs is 3D complementary metal-oxide semiconductor (CMOS) integration. 3D CMOS integration has shown to be advantageous due to (1) shorter inter-layer channel, (2) reduced number of hops and (3) increased bandwidth density [3, 4, 65–72]. Moreover, recent advances in 3D CMOS technology have proposed the possibility of fabricating 3D nanophotonic devices using a process called SIMOX (Separation by Implantation of Oxygen) [4]. This fabrication process allows for the benefits of 3D CMOS technology to be expanded to nanophotonic interconnects.
High performance nanophotonic architectures are designed using static network topology, where channels (wavelengths/waveguides) are statically assigned. However, scientific and commercial applications have diverse traffic patterns making it impossible for one static topology to provide maximum performance for all applications. One technique to optimize the usage of network resources is to implement reconfiguration of the network resources to maximize performance. Reconfiguration takes place when bandwidth from underutilized channels are used to alleviate highly utilized bottleneck channels. In this research, I propose to combine the benefits of 3D CMOS nanophotonic channels along with reconfiguration to create a 3D-NoC that is capable of providing the required bandwidth for 256 cores within a reasonable power budget.

1.4.2 Programmability: Energy-Efficient Snoopy Cache Coherent-Mutlicore

Multicores improve performance by parallelizing applications and distributing the workloads among different cores. Parallel programming models such as shared-memory models or message-passing models are commonly used to maintain application correctness [7, 73–80]. These parallel programming models should only expose the programmer to a simple abstract understanding of the parallel hardware (cores and interconnect network), so that the programmer is relieved of actual data movement. Instead, most programming models directly expose multiple cores to the programmer, which complicates the programmability of the cores. One common parallel programming model used to maintain application correctness is snoopy protocols [7, 73].

Snoopy protocols makes it easy to program a multicore architecture as all cache/memory requests are writes (stores) and reads (loads) which is identical in implementation to a uniprocessor architecture. In snoopy protocols, all cache misses are broadcast on a naturally ordered network as a shared bus that results in data coherence and application correctness [7]. However, as the number of cores increases, the amount of data coherence
traffic will increase and put a tremendous pressure that will be beyond what a conventional electrical interconnects can handle. For example, Intel’s shared bus architecture provided 4.2 GB/s of bandwidth and was only capable of supporting up to four cores causing Intel to shift to a fully connected network (Quickpath Interconnect) [12].

Extending snooping cache coherence protocols to large scale multicore architectures should be straightforward; however the limited bandwidth and scalability of shared-buses are a major bottleneck. One potential technique for expanding the benefits of broadcast networks to high-core processors is to replace the limited bandwidth electrical channels with nanophotonics channels. These nanophotonic channels allow for the broadcast bandwidth to be scaled to support higher core counts within a reasonable power dissipation.

In this research, I propose to combine the benefits of high bandwidth nanophotonics with the ease of programming found in broadcast networks to create an optimized NoC that is capable of handling the cache coherent request traffic for high-core processor.

1.5 Major Contributions of this Dissertation

The following are the major contributions of this dissertation:

- In this dissertation, using the benefits nanophotonics and 3D CMOS integration, I propose a 256-core multi-layer nanophotonic interconnect called 3D-NoC that eliminates waveguide crossing, which reduces the power budget and improves the signal-to-noise ratio at the photodetector. 3D-NoC consists of 16 decomposed nanophotonic channel-based crossbars placed on four optical communication layers, thereby eliminating waveguide crossing and reducing the optical power losses.

- I maximize the available bandwidth by proposing a reconfiguration technique that, at run-time, reconfigures 3D-NoCs by monitoring the resource consumption and applying the reconfiguration algorithm without disrupting the on-going communication.
The proposed reconfiguration technique for 3D-NoC strikes a balance between hardware complexity and network performance.

- I propose a 16-core nanophotonic broadcast network called PULSE. PULSE is constructed similar to a tree network by combining and splitting optical signals, thereby ensuring the same intensity optical signal at all cores. I expand PULSE to 64 cores through core concentration, taking advantage of the benefits of nanophotonics and snoopy protocols for high-core count multi-core processors.

- I propose an adaptive network switch technique called Multi-PULSE. Multi-PULSE uses only micro-ring resonators that rapidly morphs the broadcast network into a multicast network allowing cache requests to be only sent to caches that are holding the cache block. This allows for a reduction in cache power as needless cache lookups will be eliminated. To ensure multicast correctness, I maintain a bit-vector sharing cache that checks the prediction, and if the multicast is incorrect, fixes the sharing vector by modifying the network on the fly.

- 3D-NoC and PULSE is evaluated in terms of performance, area overhead, and power dissipation to both electrical and nanophotonic networks using the SPLASH-2 [81], PARSEC [82] and SPEC CPU 2006 [83] benchmark application suites.

The rest of the dissertation is as follows: Chapter 2 discusses about 3D nanophotonics and reconfiguration and proposes the 3D-NoC network, Chapter 3 discusses programmability and proposes the PULSE nanophotonic broadcast network, Chapter 4 discusses performance results for both networks, and Chapter 5 concludes the dissertation.
This chapter discusses the motivation, design and implementation of nanophotonic 3D-NoC architecture for future multicores. 3D-NoC is a high speed NoC that combines the benefits of 3D integration and low power nanophotonics. Moreover, 3D-NoC incorporates a reconfiguration algorithm to further increase performance by shifting bandwidth from utilized channels to fully saturated channels.

### 2.1 3D CMOS Integration

Vertically stacking different CMOS layers is an emerging technology to overcome the limited bandwidth and high power dissipation found in 2D designs [3–5, 28, 65–71, 84]. 3D CMOS fabrication can be found in today’s processors and memory chips where different layers are used to connect different parts of a chip [85]. 3D CMOS integration provides benefits such as higher device density, shorter inter-layer interconnects, and greater routing interconnection options resulting in low latency [3, 16, 67]. Importantly, 3D integration has the ability to integrate different disparate technologies (electronics and nanophotonics) on a single chip that will permit the design of novel 3D architectures. 3D CMOS chips can be fabricated by vertically stacking two or more layers on top of each other through either a face-to-face bonding technique, which uses through silicon vias (TSVs), or a face-to-back (flip-chip or microbump) bonding technique. The back of a chip consists of the silicon substrate and includes all the transistors and logic circuitry. The face of a chip consists of all the metal layers (local, semi-global and global) that are used to connect the transistor circuitry together [3].

Figure 2.1(a) illustrates an example of face-to-face bonding, where the metal contacts of both the chips are stacked on top of each other [3]. In face-to-face bonding, the metal layers of two silicon wafers are bonded together, where the metal contacts of one layer line up with the metal contacts of the other layer. Next, the two metal layers are bonded
together and the silicon substrate of the second layer is thinned (polished) to a size of 10 to 20 μm. Lastly, through-silicon vias (TSVs) are etched into the silicon substrate to create outside contacts for the metal layers.

Figure 2.1(b) illustrates an example of face-to-back or micro-bump bonding, where the silicon substrate for layer 2 is located on top of the face for layer 1 [3]. In micro-bump bonding, the different CMOS layers are fabricated separately and are then bonded together using micro-bumps on each CMOS layer [3, 72]. These micro-bumps provide metal contacts or communication channels between the different layers. Micro-bumps allow for the creation of high density and high signal quality channels [72].

When the two bonding processes are compared to each other, face-to-face (TSVs) bonding has the advantage of increasing the interconnection density between each layer [3]. However, this bonding process has low alignment tolerance. Face-to-back or micro-bump/flip-chip bonding requires lower tolerance fabrication techniques but the interconnection bandwidth between the layers is limited as the interconnects require more area overhead. Micro-bump bonding pads are 100 μm × 200 μm dimensions vs about 10 μm for TSVs [3, 72].

2.1.1 Through Silicon Vias

Vertical interconnects or TSVs are responsible for connecting multiple layers together and are the most important element in 3D CMOS fabrication [3]. Currently, TSVs have been demonstrated with dimensions from 10 μm × 10 μm to 1 μm × 1 μm [3]. The dimension of a TSV is crucial for overall performance as the inter-layer bandwidth is dependent on the density of TSVs. In addition, the partitioning and layout of a 3D chip is greatly affected by the dimensions of TSVs, as different functioning units have different operating requirements. Overall, the performance characteristics (latency and power) of TSVs depends on the fabrication techniques along with the interconnect’s distance.
However, research has found that the performance of TSVs is similar to a short 2D interconnect (8 ps for a 10 μm TSV) [3]. Typical lengths of TSVs are 10 μm to 20 μm, which mostly depends on the size of a thinned silicon wafer. Figure 2.2 illustrates how TSVs are used to connect multiple layers together and their approximate dimensions. TSVs
are fabricated by etching vias through the silicon substrate and filling them with metallic materials (copper in most cases) using techniques such as evaporation, chemical vapor deposition, and epitaxy [3].

### 2.1.2 Fabrication Issues

A major issue facing 3D CMOS integration is dissipating the increased heat due to higher transistor densities. For example, given a 3D chip with N vertically stacked layers, the transistor density increases by a factor of N. Each additional layer added increases the thermal resistance, making it difficult for heat to be transferred to the heat sink. Typically, the heat sink is placed by the processor or logic circuitry as this layer dissipates the highest amount of heat. Research has shown that by carefully placing high power dissipation...
devices around low power dissipation circuitry, the overall power density can be reduced to a tolerable operating range [3]. Furthermore, highly active devices can be placed on the top layer, closer to the heat sink for quick dissipation of heat.

Another issue facing 3D CMOS integration is the low alignment tolerance in the fabrication process required to create a 3D chip. When two or more chips are stacked on top of each other, the overall performance (bandwidth) between the different layers will depend on how close the different interconnects are bonded together. If the distance between the interconnects in multiple layers increases, a sufficient connection may not exist between the different interconnects, rendering the chip non-functional. One technique to overcome this is to use highly accurate CMOS fabrication. However, this comes with an increase in fabrication cost.

2.2 3D Nanophotonic Interconnect

Recent advances in 3D CMOS integration have made it feasible to fabricate 3D nanophotonic devices on a silicon chip. Using the fabrication technique SIMOX (Separation by Implantation of Oxygen), Jalali’s group at UCLA has demonstrated the ability to fabricate vertically stacked, multiple-layer nanophotonic devices [4]. In this fabrication technique, they first implant oxygen ions into a silicon substrate at different depths. Afterwards, they use annealing (heating) to force the silicon atoms to bond with the buried oxygen ions to form a SiO$_2$ layer within the silicon. Using epitaxy, silicon is grown on top, allowing for the fabrication of a layer of nanophotonic devices. The SIMOX process is repeated, where oxygen ions are implanted into the silicon substrate followed by annealing and epitaxy, which are required for the fabrication of nanophotonic layers. The fabrication steps required in SIMOX are further illustrated in Figure 2.3. In Figure 2.3(a), a silicon dioxide mask is placed on top of the Silicon-on-Insulator wafer. Then oxygen ions are implanted into the wafer creating a silicon dioxide structure to form inside
the silicon. Then, the wafer is subjected to high temperature annealing causing the newly formed silicon dioxide layer to be buried into the silicon wafer, which is shown in Figure 2.3(b). Then additional silicon is grown on top of the wafer using epitaxy, which is shown in Figure 2.3(c). Lastly, the process is repeated to form additional layers, which is shown in Figure 2.3(d). A research team from Cornell University has shown another method for fabricating the micro-ring resonator in buried polycrystalline silicon [68]. Moreover, recent work on using silicon nitride has shown the possibility of designing multi-layer 3D integration of photonic layers. Biberman et. al. [71] evaluated a wide range of materials and fabrication techniques suitable for 3D nanophotonic interconnects using a combination of polysilicon for electro-optic devices and silicon nitride for passive optical devices.

Figure 2.4 illustrates an example of how 3D CMOS integration can be used to construct optical vias. The basic function of an optical via is to couple light from one silicon layer to another. In Figure 2.4(a), the incoming light is not in resonance with the first micro-ring resonator, allowing the light to pass straight through without coupling to the second layer. In Figure 2.4(b), the first micro-ring resonator is in resonance with the incoming light, allowing the light to be coupled into it. As light traverses the first micro-ring resonator, it will encounter the second micro-ring resonator, which is also in resonance with the incoming light and will be coupled to the second layer. Lastly, the light will encounter the second waveguide, allowing light to be switched from one layer to the other using just a micro-ring resonator. In this example, two micro-ring resonators are fabricated on top of each other and are used to create optical vias, which allows light to couple from one layer to another. To further illustrate optical vias, Figures 2.4(c,d) illustrate how micro-ring resonators can be used to construct optical vias to a higher number of layers. In Figure 2.4(c), four micro-ring resonators are used to construct an optical via for signals to traverse two layers. The first two micro-ring resonators are used to allow the optical signal to
Figure 2.3: SIMOX fabrication process [4]: (a) silicon dioxide mask is placed on top of the silicon-on-insulator wafer and oxygen ions are implanted into the wafer, (b) annealing embeds the silicon dioxide layer into the silicon wafer, (c) additional silicon is grown on top of the wafer using epitaxy, and (d) the process repeats to create additional layers.

Traverse to the second layer, and the other two micro-ring resonators are used to change the direction of the optical signals. In Figure 2.4(d), four micro-ring resonators are used to construct an optical via for a signal to traverse three layers. The first three micro-ring resonators are used for the optical signals to traverse to the third layer, and the other micro-ring resonator is used to change the direction of the optical signals.
Figure 2.4: Illustration of an optical via built using micro-ring resonators (a) 'OFF’ state, (b) switch optical signals to layer 1, (c) switch optical signals to layer 2 and (d) switch optical signals to layer 3.

2.3 3D-NoC Architecture

2.3.1 Network Topology

The 3D-NoC architecture consists of 256 cores that can be fabricated on a 400 mm² integrated circuit (IC) chip. As illustrated in Figure 2.5, 256 cores are placed in a grid fashion where four cores are concentrated together to create a tile [18]. From Figure 2.5(a), the bottom layer, called the electrical die, adjacent to the heat sink, contains the cores, caches and memory controllers. The left inset shown in Figure 2.5(b) illustrates a tile with four cores connected together using a shared-L2 cache. Core concentration is utilized to
Figure 2.5: Proposed 256-core 3D chip layout. (a) Electrical die consists of the core, caches, the memory controllers and TSVs to transmit signals between the two dies, and (b) 3D chip with four decomposed nanophotonic crossbars with an insert showing a tile with shared L2 cache and 4 cores.
reduce the number of required nanophotonic channels and to allow local communication to take place through cheaper electronics. Each tile has its own on-chip directory and a memory controller, where memory addresses are interleaved throughout the different directories. Directories are used to maintain cache coherence by holding a list of sharers for each cache block. Traffic injected into the network originates as a cache coherent request and traverses across the network to the L2 caches and directories. To utilize the advantage of a vertical implementation of signal routing, separate optical and electrical core/cache systems are used and are unified by a single set of connector vias. The upper die, called the optical die, consists of the electro-optic transceivers layer, which is driven by the cores through TSVs and four decomposed nanophotonic crossbar layers. The electro-optic layer consists of all the front-end system drivers and the back-end receiver circuitry for photonics. Using TSVs, each tile will modulate the optical signal from an external laser using micro-ring resonators and route the signal to the appropriate destination tile. Layers 0-3 contain optical signal routing elements composed of micro-ring resonators, bus waveguides and electrical contact for other layers, if necessary.

In the proposed 3D layout, tiles are divided into four groups (16 tiles per group) based on their physical location and are called Group 0, Group 1, Group 2, and Group 3. The motivation behind using 16 decomposed crossbars is to allow for the construction of an optical crossbar for 64 tiles without the need for long snake-like waveguides as is used in Corona [5]. By dividing the 16 decomposed crossbars among four different optical layers, 3D-NoC avoids waveguide crossing optical losses, which plague many 2D optical networks. Each nanophotonic crossbar is a $16 \times 16$ crossback in size and is used to connect tiles from two different groups together (inter-group). A nanophotonic crossbar consists of multiple write single read (MWSR) nanophotonic channels, as opposed to using single write multiple read (SWMR) channels which consumes higher optical power. A SWMR nanophotonic channel allows multiple tiles the ability to write on the channel but only one
tile can read the channel. This design reduces latency as no arbitration is required but comes at the cost of multiple tiles receiving the same optical signal which may not require the data. A technique to prevent multiple tiles from receiving a copy of the data was proposed in Firefly [6]. Firefly uses a reservation assistant sub-network that activates micro-ring resonators to divert light to tiles that only require the data; however, this technique comes at the cost of increased complexity and latency. The layout of waveguides used in 3D-NoC for each layer was determined to prevent waveguides from crossing each other with the minimum number of layers required. For example, if Group 0/Group 3 are connected together then only intra-group communication between Group 1 and Group 2 can take place without waveguide crossing. If different waveguide layouts were used then the number of layers required to prevent waveguide crossing would be higher than four.

To further illustrate the differences between 3D-NoC and other leading nanophotonic networks, Figure 2.6 illustrates the topology of Corona and Firefly. In Figure 2.6(a), the optical crossbar topology for a 64-core version of Corona is shown [5]. Each waveguide in Corona traverses around all the tiles in a snake fashion, where every tile can write onto a waveguide but only one tile can read a waveguide. 3D-NoC differs from Corona as 3D-NoC has four optical layers of 16×16 decomposed optical crossbars. In Figure 2.6(b), the optical topology for a 64-core version of Firefly is shown [6]. Firefly groups together local tiles using an electrical mesh topology and the same tiles located in different local groups are connected together using SWMR interconnects. 3D-NoC differs from Firefly in that 3D-NoC does not connect local tiles together with an electrical mesh network, instead 3D-NoC uses MWSR interconnects which is different from Firefly’s use of SWMR interconnects. 3D-NoC use of MWSR interconnects does not require reservation assist as in Firefly topology either. Table 2.1 shows the optical device requirements for 256-core version of Corona, Firefly and 3D-NoC. Firefly requires 114,688 photo-detectors, which is more than Corona and 3D-NoC. 3D-NoC requires 65,536 photodetectors which
is four times more photo-detectors than Corona because each tile can receive data from four tiles at the same time. In terms of micro-ring resonators, 3D-NoC requires slightly more micro-ring resonators than Corona. This is due to several nanophotonic channels requiring additional micro-ring resonators for inter-group communication. 3D-NoC is the most efficient network in terms of energy with an average energy per bit of 1.25 pJ/bit. Furthermore, 3D-NoC has the most bisection bandwidth of the three networks (65,536 GBps).

2.3.2 Communication

Each waveguide used within a nanophotonic crossbar has only one receiver which is defined as the receiver’s home channel. During communication, the source tile modulates optical data on the destination tile’s home channel (waveguides). An off-chip laser is used to create the required 64 wavelengths of light. Figure 2.7 shows the floor plan for the
Table 2.1: Electrical and optical area overhead for select electrical and optical components

<table>
<thead>
<tr>
<th>Component</th>
<th>Corona</th>
<th>Firefly</th>
<th>3D-NoC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network size</td>
<td>64</td>
<td>256</td>
<td>64</td>
</tr>
<tr>
<td>Network Radix (k)</td>
<td>4</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Concentration</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Network Diameter</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td># of Wavelengths per Channel</td>
<td>-</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>MRRs</td>
<td>65536</td>
<td>1048576</td>
<td>4wk^3</td>
</tr>
<tr>
<td>Photo-detectors</td>
<td>4wk^2</td>
<td>4096</td>
<td>16384</td>
</tr>
<tr>
<td>Electrical Links</td>
<td>-</td>
<td>-</td>
<td>16</td>
</tr>
<tr>
<td>Bisection Bandwidth (GBps)</td>
<td>4wk^2</td>
<td>4096</td>
<td>4wk^2</td>
</tr>
<tr>
<td>Bandwidth/channel (GBps)</td>
<td>-</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>Average Hops</td>
<td>1</td>
<td>1</td>
<td>1.75</td>
</tr>
<tr>
<td>Average Channel Energy (pJ/bit)</td>
<td>1.26</td>
<td>1.41</td>
<td>1.28</td>
</tr>
</tbody>
</table>
first optical layer. For clarity purposes only three waveguides are shown. A 32 waveguide bundle is used for communication between Groups 0 and 3, and two 16-waveguide bundles are used for communication within Groups 1 and 2. For inter-group communication between Group 0 and 3, the first 16-waveguide bundle is routed past Group 0 tiles so that any tile within Group 0 can transmit data to any destination tile in Group 3. Similarly, the next 16-waveguide bundle is routed past Group 3, so that any tile within Group 3 can communicate with the destination tiles located within Group 0. The bidirectional arrows illustrate that light travels in both directions and depends on which group is the source and which is the destination. The remaining two independent waveguide bundles (16 waveguides) are used for intra-group communication for Groups 1 and 2 respectively. Therefore, each waveguide requires a total of 64 waveguide bundles per layer.

As 3D-NoC will comprise of four optical layers, Figure 2.8 illustrates the communication combinations for the three additional layers. Figure 2.8(a) shows the group communications that will take place in the second layer of 3D-NoC. In the second layer, inter-group
communication will take place between Group 0/Group 2 and Group 1/Group 3. Figure 2.8(b) shows the group communications that will take place in the second layer of 3D-NoC. In the third layer, inter-group communication will take place between Group 1/Group 2 and inter-group communication will take place between Group 0/Group 3. Figure 2.8(c) shows the group communications that will take place in the second layer of 3D-NoC. In the fourth layer, inter-group communication will take place between Group 0/Group 1 and Group 2/Group 3.

2.3.3 Router Architecture

Figure 2.9 shows the router microarchitecture in 3D-NoC for tile 0. Any packet generated from the L2 cache is routed to the input demultiplexer with the header directed towards routing computation (RC). The two most significant bit (MSBs) are used to direct the packet to one of the four sets of input buffers (IB$_0$ - IB$_3$) corresponding to each optical layer (0-3). For the second set of demuxes, the packet will utilize an unique identifier, that corresponds to the core number. The unique identifier indicates the source of the packet to prevent any core from overwhelming the input buffers, thereby providing a fair share of buffer resources. Token (Request + Release) ensures that packets are transmitted from the input buffers (IBs) without collision and the micro-ring resonators are used to modulate the signal into the corresponding home channel. At the receiver, the reverse process takes place, where the packet from the optical layer is converted into electronics. According to the unique identifier, the packet will find one set of buffers available. Token Control is used to prevent buffer overflow at the home node by checking the number of empty buffer slots. If the number of empty buffer slots falls below a certain threshold, then the destination tile will capture the circulating token and will not re-inject the token until the number of free slots increases to the threshold. The packets will then contend to obtain the token to reach
Figure 2.8: The layout of three different waveguides for (a) Layer 1, (b) Layer2 and (c) Layer3 of 3D-NoC.

the L2 cache. It should be noted that the proposed unique identifier is similar to a virtual channel allocator, however it does not perform any allocation, as the decision to enter any buffer is determined by the core number (source or destination).
Figure 2.9: Router microarchitecture for a tile.

2.3.4 Communication Pipeline

Figure 2.10 illustrates the communication pipeline and latency a data packet will experience in 3D-NoC. The communication pipeline consists of both electrical and optical delays, where there is a total of seven different pipeline stages. The seven stages are as follows: routing computation (RC), buffer write source (BWS), electrical to optical (EO),
optical link traversal (OL), optical to electrical (OE), buffer write destination (OE), and switch allocation (SA).

(a) Routing Computation

In the first stage, packets go through routing computation (RC). In RC, data packets coming from the L2 cache are read and the destination of the packet is determined. After the destination is determined, the data packet is stored in buffers waiting for the next pipeline stage.

(b) Buffer Write Source

In BWS, the data packet is written into a free buffer slot calculated in the RC stage. In addition, the source tile will activate its micro-ring resonator and attempt to capture an optical token required to transmit on the optical channel.

(c) Electrical to Optical (E/O)

Once the tile captures an optical token the data goes through E/O. In E/O, electrical signals are converted into optical signals using micro-ring resonators. The electrical signals coming from the buffer slots in the BWS stage are fed into a buffer chain to quickly drive the micro-ring resonators. After this process, the micro-ring resonators modulate the incoming optical light. This results in an optical conversion of electrical data.

(d) Optical Link Traversal

After converting the electrical data into optical data, it is routed to the destination tile in the OL stage. The time (cycles) required for the optical signals to traverse the waveguide varies from one cycle to three cycles. This time varies between each source and destination pair, as the distance is different between each source/destination pair. The minimum number of cycles for an optical transmission is one cycle and the maximum is
three cycles. This is due to the fact that the distance between source and destination tiles vary from about 1.25 mm to about 50 mm.

(e) Optical to Electrical

In O/E, the optical signals are converted into electrical signals using photodetectors. During this stage, the incoming optical signals are filtered out using micro-ring resonators and sent to their corresponding photodetectors. The photodetectors convert the incoming optical signals into an electrical signal, where the electrical signal can be further processed.

(f) Buffer Write Destination

During the BWD stage, the data packet coming from the O/E stage is written into a buffer slot. The correct buffer slot is determined from the incoming packet’s destination waveguide. Each tile has four buffer slots, one for each of the four waveguides from which the tile reads.

(g) Switch Allocation

In the SA stage, the four buffer slots compete with each other for the right to access the shared-L2 cache. After a buffer slot is allocated permission, the data packet traverses to the shared-L2 cache and communication between the source and destination is completed.

2.3.5 Optical Tokens

3D-NoC prevents two or more tiles from communicating simultaneously by using optical tokens. An optical token is a short burst of light; once a tile captures an optical token, the tile can then transmit on the optical channel. After the tile completes communication with the destination tile, the optical token is injected back into the network for other tiles to use. Figure 2.11(a) shows the proposed token layout in 3D-NoC. The power waveguide is used by the home tile and other tiles for injecting the token into the
inject waveguide. The top left micro-ring resonator shown in the figure, labeled as BC (buffer control), is used to control the injection of tokens when the buffer slots are full. For example, when the home tile detects that the buffer slots are full, it will activate this micro-ring resonator, which will block the token from being injected. Once a free buffer slot becomes available, the home tile will re-inject the token by deactivating the BC micro-ring resonator. Figure 2.11(b) shows the proposed token control block. In the token control block, an optical token is injected when a free token is available (high TR signal) and the BC signal is low. If additional utilization of network resources is required, 3D-NoC can use optical token techniques such as Fexishare and Fair Token Slot [64, 86].

3D-NoC provides sufficient static bandwidth for inter-tile communication i.e. for a given pair of source-destination, the bandwidth is static and does not change. While this is beneficial for well-defined uniform traffic (every tile has equal probability for communication), for widely varying traffic seen in most multi-threaded application, the channel bandwidth may not be effectively used with the initially proposed static allocation.
The following section discusses a reconfiguration technique for 3D-NoC that further improves network resource utilization.

2.4 Reconfiguration

Communication patterns found in multi-threaded applications have a wide range of traffic patterns, making it impossible for a single topology to provide maximum performance for all applications. Reconfiguration allows network resources to be changed to meet the current needs of an application for maximum performance [87, 88]. Moreover, reconfiguration improves the power efficiency of the network as more resources are used and less static power is wasted for unused channels [89, 90]. The advantages of reconfiguration are as follows:

- The network topology found in 3D-NoC is static where nanophotonic channels can only send data from the same source tile to the same destination tile. Reconfiguration allows these source/destination pairs to be changed. This results in additional
bandwidth being available for highly utilized channels, which in turn increases the network performance.

- Nanophotonic devices such as lasers, continuously dissipate power, no matter if data is being transmitted on the network. During reconfiguration, more network bandwidth is being utilized which results in energy-proportional computation.

- Reconfigurable networks can be designed with limited network resources and can outperform nanophotonic networks where bandwidth is statically allocated.

Two major techniques exist for reconfiguration: software based reconfiguration and hardware based reconfiguration. In software based reconfiguration, compilers are used to estimate the network resource usage for each channel [90]. When the program is being executed, the network already has pre-determined the incoming traffic demand for each channel and can reconfigure accordingly. In hardware based reconfiguration, hardware counters monitor network resources, such as buffer and link utilization, and reconfiguration occurs when certain pre-determined thresholds are met. 3D-NoC uses hardware reconfiguration as this can be used by any program and can quickly adapt to fluctuations in application demands.

### 2.4.1 Reconfigurable 3D-NoC

Reconfiguration in 3D-NoC occurs by shifting unused nanophotonic channel bandwidth, using micro-ring resonators, from one layer to another. These micro-ring resonators are placed at points where the waveguide bundles from different layers are in close proximity to each other. In addition, the proposed reconfiguration uses hardware counters that monitor the traffic load and apply a reconfiguration algorithm that works in the background without disrupting the on-going communication.
(a) Reconfiguration Example

To illustrate with an example, Figure 2.12 shows the group connections, before reconfiguration, for Layer 0, Layer 1, and Layer 2. Consider a situation where tiles in Group 0 communicate only with tiles in Group 3. Without reconfigurations, tiles in Group 0 can only communicate with tiles in Group 3 through waveguides in Layer 2. Figure 2.13 shows the reconfiguration mechanism. The *static* allocation of channels for communication are in Layer 2, as shown in Figure 2.13(a). This is also shown in Figure 2.13(b) (top layer) where tile 0 (Group 0) communicates with tile 63 (Group 3). Suppose no tile within Group 1 (in Layer 1) communicates with Group 3. Then bandwidth can re-allocate from Group 1 to Group 0 to communicate with Group 3. To implement reconfiguration, however, two important requirements need to be satisfied: (1) There should be a source waveguide which should be freely available to start the communication on a source layer, and (2) there should be a destination waveguide which also should be freely available to receive the extra packets. As mentioned before, Groups 0 and 3 talk only to each other. I have the first set of waveguides on layer 0 (generally used to communicate within the group) available, therefore this satisfies the first condition. Here, I choose the waveguide which is used to communicate to destination tile 12 as the source waveguide to be reconfigured. As Group 1 does not communicate with Group 3, I can utilize the destination waveguide available in Layer 1 and satisfy the second condition. Figure 2.13(b) shows the waveguide selected in Layer 1 as the destination waveguide for tile 63 (shown as the dynamic allocation in Figure 2.13(a)). Therefore, during reconfiguration, tile 0 has doubled the bandwidth to communicate with tile 63 by communicating via layers 2 (static) and 1 (dynamic). Two different communications are disrupted when the reconfiguration occurs, namely, Group 0 in layer 0 can no longer communicate with itself (to destination tile 12) and Group 1 in layer 1 can no longer communicate with Group 3 (tile 63).
The objective of reconfiguration is to improve performance by re-allocating bandwidth from under-utilized to over-utilized links. The design space of reconfiguration is large, as there can be several combinations across multiple layers. Figure 2.14 shows four possible combinations that I will evaluate, as they cover most of the design space. A row-column matrix indicates the statically allocated communication. For example layer 0 - layer 0 shows three combinations: G0 < - > G0, G1 < - > G2 and G3 < - > G3. Group 0 communicates with itself, Groups 1 and 2 communicate with each other, and Group 3 communicates with itself. The square (red) boxes and the arrow show which layers can be used for reconfiguration. Figure 2.14(a) shows Layer 0 can reconfigure and take away bandwidth from Layer 1, similarly Layer 1 can reconfigure and can take away from bandwidth from Layer 0. Layer 2 can take bandwidth from layer 3 and vice versa. This approach is restricted to one additional layer that can be used for reconfiguration. I call this 3D-NoC-L1 (3D-NOC-Limited to 1 Layer). This restricted mechanism will reduce both...
Figure 2.13: Illustration of reconfiguration where additional bandwidth is given to Group 0 to communicate with Group 3. (a) Inter-layer static and dynamic bandwidth re-allocation and (b) 3D switching of bandwidth.

the power consumption and area overhead. Figure 2.14(b) shows reconfiguration across one or two layers; however both layers have to be adjacent. Layer 0 can only reconfigure
with Layer 1, where as Layer 1 can reconfigure with both Layer 0 and Layer 2 (adjacent). Adjacent layer reconfiguration is easier to implement, as the next layer, above or below, will be used. This improves on a single layer reconfiguration technique and I call this 3D-NOC-LA (3D-NOC-Limited to adjacent layer). Figure 2.14(c) shows reconfiguration across two layers, even if they are not adjacent, and I call this configuration 3D-NOC-L2 (3D-NOC-Limited to 2 Layers). This increases the power consumption as well as design fabrication, as more TSVs will be needed. One side-effect of this reconfiguration is that as more layers are involved, there are more channels lost due to reconfiguration. This is primarily due to the fact that as additional additional waveguides are consumed, thus restricting the number of layers that can be reconfigured. For adverse and embarrassingly parallel applications, this would be an interesting option, as more layers can be used for reconfiguration. Figure 2.14(d) shows the complete reconfiguration, as any layer can go to any other layer, and I call this configuration 3D-NOC-L3 (3D-NOC-All 3 Layers). This fully reconfigured design will require a large amount of area overhead and also incur higher complexity in terms of fabrication, as TSVs have to extend to all the layers. During reconfigurations, each group within a layer will have the ability to reconfigure with two other groups from the additional layer. The two other groups are determined from the source and destination of the reconfigurable group. For example, Group 0 bandwidth in layer 0 can be only reconfigured for increased bandwidth to only Group 0 and Group 1. This bandwidth can not be reconfigured with Group 2 and Group 3, as this is redundant and would not increase the total bandwidth for communication. The layout of the different layers in 3D-NoC was chosen arbitrarily, as each layout combination allows any Group to be reconfigured with each other group twice.
(b) Dynamic Reconfiguration Technique

The reconfiguration algorithm is designed with the following objectives: (a) The algorithm should not be overly sensitive to traffic fluctuations to prevent rapid changes in topology; (b) the algorithm should mostly work in the background and not hinder any on-going communication, and (c) the algorithm should ensure that no tile is starved from bandwidth due to reconfiguration, while ensuring minimum reconfiguration latency to take advantage of additional bandwidth. To implement such a reconfiguration, the reconfiguration hardware first take measurements that are available, such as link utilization ($\text{Link}_{\text{util}}$) and buffer utilization ($\text{Buffer}_{\text{util}}$), using hardware counters [87–89, 91]. This implies that each tile within a group will have four hardware counters (one for each of the
three groups) that will monitor traffic utilization and provide the link and buffer information to the Reconfiguration Controller (shown in 2.9). Both link and buffer utilization are used, as link utilization provides accurate information at low-medium network loads, while buffer utilization provides accurate information at high network loads [89, 91]. Link utilization represents the percentage of time a link is occupied for a given unit of time. Buffer utilization represents what percentage of buffer slots are full in a router interconnect. All these statistics are measured over a sampling time window called Reconfiguration window or phase, $R_W^t$, where $t$ represents the reconfiguration time. This sampling window impacts performance, as reconfiguring finally incurs a latency penalty, and reconfiguring coarsely may not adapt in time for traffic fluctuations. In our performance section, I evaluated a number of PARSEC applications to determine the optimum size for $R_W$. For calculation of $\text{Link}_{util}$ at configuration window $t$, I use the following equation [90]:

$$\text{Link}_{util}^t = \frac{\sum_{cycle=1}^{R_W} \text{Activity}(cycle)}{R_W}$$

(2.1)

where $\text{Activity}(cycle)$ is 1 if a flit is transmitted on the link or 0 if no flit is transmitted on the link for a given cycle. For calculation of $\text{Buffer}_{util}$ at configuration window $t$, I use the following equation [90]:

$$\text{Buffer}_{util}^t = \frac{\sum_{cycle=1}^{R_W} \text{Occupy}(cycle)/\text{Total buffers}}{R_W}$$

(2.2)

where $\text{Occupy}(cycle)$ is the number of buffers occupied at each cycle, and $\text{Total buffers}$ is the total number of buffers available for the given link. When traffic fluctuates dynamically due to short term bursty behavior, the buffers could fill up instantly. This can adversely impact the reconfiguration algorithm, as it tries to re-allocate the bandwidth faster, leading to fluctuating bandwidth allocation. To prevent temporal and spatial traffic fluctuations affecting performance, I take a weighted average of current network statistics ($\text{Link}_{util}$ and $\text{Buffer}_{util}$), so that the network will gradually re-allocate bandwidth. We calculate the
Buffer\textsubscript{util} using the following equation [90]:

\[
\text{Buffer}^t_w = \frac{\sum \text{Buffer}^t_{util} \times \text{weight} + \text{Buffer}^{t-1}_{util}}{\text{weight} + 1}
\] (2.3)

where \textit{weight} is a weighting factor and is set to three in our simulations [90].

After each \(R_W^t\), each tile will gather its link statistics (Link\textsubscript{util} and Buffer\textsubscript{util}) from the previous window \(R_W^{t-1}\) and send then to its local reconfiguration controller (RC) for analysis. I assume that Tile 0 of every group gathers the statistics from the remaining tiles, which can be a few bytes of information that is periodically transmitted. Next, when each RC\(_i\) (\(\forall i = 0, 1, 2, 3\)) has finished gathering link and buffer statistics from all its hardware controllers, each RC\(_i\) will evaluate the available bandwidth for each link depending on the Link\textsubscript{util} and Buffer\textsubscript{util} and will classify its available bandwidth into different thresholds \(\beta_{1-4}\) corresponding to 0%, 25%, 50% and 90%. We never allocate 100% of the bandwidth as the source group may have new packets to transmit to the destination tile before the next \(R_W\). RC\(_i\) will send link availability information to its neighbor RC\(_j\) (\(j \neq i\)). If RC\(_j\) needs the available bandwidth, it will notify the source and the destination RCs so that they can switch the micro-ring resonators and inform the tiles locally of the availability. Once the source/destination RCs have switched their reconfiguration micro-ring resonators, RC\(_i\) will notify RC\(_j\) that the bandwidth is available for use. On the other hand, if a node within RC\(_i\) that throttled its bandwidth requires it back due to increase in network demand, RC\(_i\) will notify that it requires the bandwidth back and afterwards will deactivate the corresponding micro-ring resonators. The above reconfiguration completes a three-way handshake where RC\(_i\) first notifies RC\(_j\), then RC\(_j\) notifies RC\(_i\) that RC\(_j\) will use the additional bandwidth, and finally RC\(_i\) notifies RC\(_j\) that the bandwidth can be used. Table 2.2 shows a pseudo-reconfiguration algorithm implemented in 3D-NOC. We assume Link\textsubscript{util} = 0.0 to indicate if the link is not being used, \(L_{min} = 0.10\) to indicate if the link is under-utilized, \(L_{min} = 0.25\) and \(B_{con} = 0.25\) to indicate if the link is normal-utilized and \(B_{con} = 0.5\) to indicate that
Figure 2.15: Illustration of reconfiguration hardware: (a) intra-group hardware where each tile has hardware counters to determine interconnect statistics and (b) inter-group RC hardware.

the link is over-utilized [91]. These values are chosen based on prior work using hardware counters as the link and buffer thresholds show the best performance [91].

(c) Reconfiguration Hardware

Figure 2.15 shows the hardware required for reconfiguration to take place. In Figure 2.15(a), the RC is shown for Group 0. The RC is responsible for monitoring incoming traffic and calculates reconfiguration opportunities from the three other groups. If the optical receiver and hardware counters for a tile detects that the incoming traffic for any of the four interconnects is below a threshold, the hardware detector will notify the local Group’s RC that the bandwidth is available for reconfiguration. After the RC collects all network statistics and determines which interconnects’ bandwidth are available for reconfiguration, the RC will pass this information onto the next group’s RC. Figure 2.15(b) illustrates how the different RCs pass information between each other.
Table 2.2: Pseudo Algorithm used by 3D-NoC

Step 1: Wait for Reconfiguration window, $R_w'$

Step 2: $RC_i$ sends a request packet to all local tiles requesting $Link_{Util}$ and $Buffer_{Util}$ for previous $R_w'^{-1}$

Step 3: Each hardware counter sends $Link_{Util}$ and $Buffer_{Util}$ statistics from the previous $R_w'^{-1}$ to $RC_i$

Step 4: $RC_i$ classifies the link statistic for each hardware counter as:

- If $Link_{util} = 0.0$
  - Not-Utilized: Use $\beta_4$
- If $Link_{util} \leq L_{min}$
  - Under-Utilized: Use $\beta_3$
- If $Link_{util} \geq L_{min}$ and $Buffer_{util} < B_{con}$
  - Normal-Utilized: Use $\beta_2$
- If $Buffer_{util} > B_{con}$
  - Over-Utilized: Use $\beta_1$

Step 5: Each $RC_i$ sends bandwidth available information to $RC_j$, ($i \neq j$).

Step 6: If $RC_j$ can use any of the free links then notify $RC_i$ of their use, else $RC_j$ will forward to next $RC_j$

Step 7a: $RC_i$ receives response back from $RC_j$ and activates corresponding microrings

Step 7b: $RC_j$ notifies the tiles of additional bandwidth and $RC_i$ notifies $RC_j$ that the additional bandwidth is now available

Step 8: Goto Step 1
3 Broadcast Network

This chapter discusses the motivation for broadcast networks and how nanophotonics can be applied to construct PULSE architecture. PULSE is a power efficient broadcast network that combines benefits of nanophotonics, for low power, and snoop protocols, for ease of programming. In addition, PULSE has the capacity to further reduce power dissipation by directing optical data to only the tiles that have a copy of the requested cache block (multicast).

3.1 Parallel Programming Models

As multicores operate on parallelized workloads, application correctness is maintained through the use of parallel programming models [7]. Two common parallel programming models used for maintaining application correctness are message passing and shared-memory [7]. In message passing, a parallel application is divided into a group of processes, where each process is executed on a separate processor with its own local memory [7, 92]. Application correctness is maintained through interprocessor communication of data messages. These data messages contain application data and synchronization messages. One common version of message passing is Message Passing Interface (MPI) or Open Message Passing Interface (OMPI) [7, 92, 93]. MPI and OMPI are a library of standard specifications that use message passing to maintain application correctness. Message passing is commonly found in High performance Computing (HPC) systems that are comprised of many different processing nodes [7]. Applications that use message passing can be easily optimized as the data communication infrastructure is known to the programmer. However, message passing can require a lot of overhead as the whole network architecture is exposed to the programmer [7].

In shared-memory, different application processes running on different multicores share a common address space. Application correctness is maintained through the use of locks
and semaphores [7]. Locks and semaphores allow for a range of address spaces to be locked, so only one processor can read and write to memory [7]. These locks or semaphores act as synchronization points between the different processors. Shared-memory parallel programming model can be found in symmetric multiprocessor (SMPs) machines and distributed shared memory (DSM) machines [7]. The advantage of shared-memory is that there is no concept of data ownership. This allows the programmer to focus on writing application code, and not on how the individual processes will communicate. Overall, this simplifies the development of applications when compared to message passing. However, the shared-memory programming model tends to lack in performance when compared to the message passing programming model [7]. This is because applications can not be fully optimized as the programmer is not exposed to the interworkings of data movement. For the rest of this dissertation, shared-memory will be used and discussed, as it is easier to be programmed and is commonly used for multicore architectures.

### 3.2 Cache Coherency

In shared-memory, cache is used as a local storage of data to offset the high latency penalty of accessing data from the main memory [7]. Figure 3.1 shows the cache hierarchy of a tile. Figure 3.1(a) illustrates a tile that consists of a single core, and Figure 3.1(b) illustrates a tile that consist of four cores. The closest level of cache to the core is the L1 cache [7]. The L1 cache is responsible for servicing load and store instructions from the core with data. The L1 cache is divided into a L1 Data (L1D) cache and L1 Instruction (L1I) cache. These two different caches are responsible for storing the data and instructions of the currently executing application in two separate locations. The next level is the L2 cache that is bigger and stores more data than L1 cache. The L2 cache is responsible for servicing data to L1 misses and servicing cache requests coming from the network [7].
Two different cache organizations exist for multicores: private cache and shared cache [7, 94–100]. In private cache, the lowest level caches (LLC) or L2 caches are private to the tile. This means each L2 cache has its own unique copy of data, and potentially the same copy of data can exist in the different L2 caches [7, 94]. The main advantage of using private-L2 cache is the increase in performance as each tile can have its own copy of data, which reduces latency penalty. However, private-L2 cache requires a high amount of area overhead as multiple copies of data have been replicated. In shared cache, the entire L2 cache is shared by all tiles and memory addresses are interleaved across the different L2 caches [7, 94, 95]. This allows for only one copy of data to exist within a L2 cache and creates a Non-Uniform Cache Access (NUCA) hierarchy [7, 94]. Using a NUCA cache design, cache requests that cannot be completed by the tile’s private-L1 cache will travel to a L2 cache that is responsible for the data. When the L2 cache responsible for the data, receives the cache request, it will forward the data to the requesting tile’s L1 cache. Shared-L2 cache designs have the advantage of reducing cache area overhead as only one copy of data will exist in the L2 cache [7, 94, 95]. However, shared-L2 designs have a potential
high latency penalty (decrease performance) as requests may have to travel further across the chip to be serviced compared to private-L2 designs.

In either private-L2 or shared-L2 designs, multiple copies of the same memory location will exist in different caches. These multiple copies all require the most up-to-date data. If the caches are not kept up-to-date, each core will read in a different value for the same memory location, causing the cache coherence problem as illustrated in Figure 3.2. In Figure 3.2(a), the first processor, P1, reads memory location U from main memory and places the value of 5 in its local cache. Then, in Figure 3.2(b), processor P3 also reads memory location U from main memory and places the value of 5 in its local cache. Then, in Figure 3.2(c), processor P3 writes the value of 7 for memory location U into its cache. Then processor P1 and processor P2 read the memory location U and again read a value of 5, shown in Figure 3.2(d). Reading a value of 5 is wrong as the most up to date value for U, which is located in processor P3’s cache is 7. This is the definition of the cache coherence problem, where two or more cache have different values for the same memory locations.

There are two major techniques or protocols used to maintain cache coherence: snoopy cache coherence protocol and directory cache coherent protocol. PULSE uses snoopy cache coherent protocols which will be discussed below [7].

### 3.2.1 Snoopy Protocols

Data coherence should be maintained to ensure that any load returns the last store to the memory location. In order to maintain cache coherence, a cache coherent protocol must ensure two important coherence properties: write propagation and write serialization. Write propagation requires all writes (store instruction) to become visible to all tiles, and write serialization requires all tiles to see the same order of writes (store instruction). Write propagation can be maintained by insuring all cache requests arrive at all tiles and write serialization can be maintained by insuring that all cache requests are seen by all caches.
Figure 3.2: Cache coherence problem: (a) P1 reads memory location U from main memory and places the value of 5 in its local cache, (b) processor P3 also reads memory location U from main memory and places the value of 5 in its local cache, (c) processor P3 writes the value of 7 for memory location U into its cache, and (d) processor P1 and processor P2 read the memory location U and again read a value of 5 (should read a value of 7) [7].

in the same order. The following discusses how snoopy protocols along with a naturally ordered bus network are used to maintain cache coherence.

Snoopy based protocols maintain cache coherence through the use of an interconnect with a natural ordering point [7, 73]. Shared-buses are commonly used for snoopy based protocols to maintain cache coherence, shown in Figure 3.3. When a memory transaction (load or store) takes place, the memory request will be placed on the bus and consequently will be seen by all nodes on the network. While the memory request is on the bus, all nodes
will read or snoop the bus, and depending on the memory transaction, will take appropriate action. By ensuring all writes are seen by all the tiles, in the same order, and at the same time, bus based networks ensure that the write propagation and write serialization cache coherence requirements are met. Figure 3.3 shows an example of how a memory request can be snooped (seen) by all caches and memory controllers at the same time. In the figure, P4 places a memory request on the bus, and once the memory request is on the bus it is available for all the other caches and memory controllers to read. The bus network creates a natural ordering of requests, and all memory requests are seen by all nodes on the bus at the same time, allowing for cache coherence to be maintained.

3.2.2 Previous Research (Broadcast Networks)

Several techniques have been proposed to overcome the limited bandwidth of ordered-networks such as split transaction buses [7]. In a split transaction bus, a cache request is split into an address request and a data response, where each transaction runs on
different bus networks. This allows multiple memory transactions to be serviced, resulting in an increase in the available network bandwidth. The problem with split transaction buses is that they are still limited in bandwidth and are only capable of supporting a few additional number of cores over the shared bus architecture. Sun Microsystem’s Fireplane interconnect was introduced to overcome the limited bandwidth of shared buses [101]. In Fireplane, an electrical tree topology is used to increase bandwidth and order cache requests. The problem with this network is the high power dissipation as data traverses across multiple electrical routers and channels. Moreover, several research works proposed techniques to use snoopy protocols on non-ordered networks [102–104]. In [102], cache requests are ordered by placing a time stamp on each cache request. As the cache requests traverse the network, they are arranged in order by the time stamp placed on them. In [103, 104], cache requests are injected into the network, where the network arranges cache requests in order. The problem with [102–104] is each technique requires more complex network topologies than is found in a shared-bus topology. Furthermore, both Intel Quick-path Interconnect [12] and AMD Hypertransport [13] implement high-speed serial point-to-point communication links between cores and I/O devices (as seen in Intels Core i7 and AMDs Phantom II processors respectively) that eliminates the high latency of bus networks. To further reduce power dissipation and reduce broadcast bandwidth techniques such as cache filtering can be used [75]. Cache filtering reduces the required bandwidth by filtering out memory requests that are not cached by the core, thereby directing the requests to only those caches that actually share the cache block [75].

There have been few significant research works that use nanophotonics for broadcast networks. In the Shared-Bus architecture [38], both electronics and nanophotonics are combined to create a hybrid 64-core opto-electronic broadcast network. The problem with this network is the higher power dissipation due to the electrical network. A WDM-based multicast nanophotonic network which uses different wavelengths to send the same copy
of the data was proposed in [105]. Each wavelength is used to communicate with a unique tile. By controlling which wavelengths the network modulates, data can be multicasted to tiles that are currently caching the requested data. While the electrical and nanophotonic solutions have yielded improvements in bandwidth and power, there currently has not been a solution that integrates the benefits of nanophotonics for broadcast networks using cache filtering techniques.

3.3 PULSE

Using the benefit of nanophotonics discussed above, the following section proposes the nanophotonic broadcast network PULSE.

3.3.1 System Architecture

Figure 3.4 illustrates the PULSE architecture, which consist of 16 tiles placed in grid fashion. In PULSE, a tile is comprised of a core, a private L1 instruction cache, a L1 data cache, a private L2 cache, and optical transmitters and receivers. Memory controllers (MCs) along with the sharing vector cache are placed adjacent to the bottom row of tiles because they are closer to the control center, allowing for quick communication between each other (as required in Multi- PULSE). When the optical signal arrives at the receivers that are in the bottom row of tiles, the electrical data after optical to electrical (O/E) conversion is sent to the L2 cache as well as the MC. We group together the bottom tiles and MCs to save on laser power as the optical signals do not require additional splitting. To expand PULSE to 64 cores, we concentrate four cores together where each core retains private L1 instruction and data cache and shared L2 cache. We minimize the increase in the optical transceivers by core concentration and intra-tile communication occurs with shared L2 cache without any additional crossbar network (as in private L2). The MCs are scaled to 16 for 64-core version, however these are again connected to the bottom row of tiles and placed closer to the control center. In our proposed snoopy system, we enforce single-
writer multiple-reader (SWMR) that ensures that only one tile is writing to the tree at any given clock. External light first enters into the control center, where the control center routes the optical signal to one of the columns of tiles. Within the column, only one tile can modulate the optical signal. Arbitration among the tiles within a column is accomplished using optical tokens. After the optical signals are modulated, the address request travels to the root of the optical tree where optical waveguides coming from the other three columns are combined together to form the root of the tree. When the modulated optical signal reaches the root, they traverse through a series of optical spitters and waveguides that allow for the same intensity of light to arrive at all the tiles simultaneously.

Figure 3.4: The address and data network nanophotonic interconnect layout for 16 cores.
Using 16 wavelengths, PULSE is constructed with a total of 9 waveguides, 5 for the address network and 4 for the data network. This results in 80 bits for the address network and 64 bits for the data network that can be placed on the optical network in a single clock cycle. Moreover, using separate address and data networks allow for split-transactions (simultaneous different address requests and data responses) of different cache requests to take place. The proposed architecture ensures that the rules of coherence are maintained i.e. each write request is made visible to all processors in program order.
3.3.2 Token Arbitration and Power Guiding

An arbitration technique for network resources is required to prevent two or more tiles from broadcasting data on the address and data networks at the same time. In PULSE, I use optical tokens to prevent multiple tiles from broadcasting data at the same time [5]. There are two optical tokens circulating simultaneously: one for the address network and the other for the data network. In addition, I restrict the circulation of tokens to an activate column of tiles for optical power savings. Figure 3.5 illustrates the optical token waveguide layout. In the figure, the optical token system consists of three different waveguides: power, inject and return. The power waveguide is responsible for providing optical power to the tiles so they can re-inject the token. The inject waveguide is where the optical tokens circulate, and the return waveguide is used for the token to return to the control center. As stated above, there are two optical tokens circulating around the network simultaneously, where each of the optical tokens represents the right to communicate on either the data or address networks. As these optical tokens circulate, they are not required to circulate in tandem around the same column together, as this allows for increased utilization of each broadcast network because all tiles in the selected column may not require both tokens. In order to allow fair sharing of network resources (prevent starvation), it is possible for the same group of tiles to keep re-capturing the same optical token. The control center will shut down optical power to the current active column and inject the optical token into the next column. When the optical token is first injected into the active column, the control center will start timing how long it takes for the optical token to return. After so many clock cycles have passed, and if the optical token has not returned, the control center will shut down optical power for the token injector waveguide so no tile can re-inject the optical token. Once sufficient time has passed for the most current broadcast to finish, the control center will inject the optical token into the next active column of tiles and supply optical
Figure 3.6: Control block operation: (a) initial optical token injection coming from the control block, (b) optical power supplied to power waveguide for column one, (c) after some time the optical token returns to the control center and shifts the registers, and (d) the control center injects the optical token into the next column of tiles.

power to them. If additional utilization of network resources is required, PULSE can use techniques such as Fexishare and Fair Token Slot [64, 86].

Figures 3.4 and 3.5 show the proposed control center at the bottom. The control center is used to guide optical power to the correct column of cores. Figure 3.6 illustrates the details of how the control block operates, which consists of two photodetectors and two four-bit shift registers. Within the control block, these two photodetectors are used to detect the return of each optical token (address and data network), and the two four bit shift registers are used to activate the correct set of micro-ring resonators for the current
active column of tiles. Figure 3.6(a) demonstrates the initial injection of an address network optical token into the first column of tiles. The correct combination of active micro-ring resonators for sending an optical token to the first column is determined by which bit in the four bit shift register is active, and, in this case, it is the first bit. After a clock cycle has passed, the inject waveguide is powered down and optical power is returned to the power waveguide, which is illustrated in Figure 3.6(b). The optical power is returned to the power waveguide, as the optical power is used by the tiles to re-inject the optical token after communication has completed (see Figure 3.5). Once the optical token arrives at the control block, the shift register shifts the high bit to the next register, which activates the next column of tiles, as illustrated in Figure 3.6(c). Finally, the control block injects the optical token into the second column of tiles, as illustrated in column 3.6(d). The timer block shown in the figure keeps track of how long a token has been circulating in a column and is used to prevent resource starvation. By limiting the direction of the external laser to only the active column, the optical power can be reduce by 75%.

### 3.3.3 Multi-PULSE

On a L2 read/write miss, traditional snoopy protocols broadcast the miss request to all L2 caches and the memory controller. However, sending miss requests to all tiles implies that both the power to broadcast the request and cache energy at every L2 access are consumed on each miss. Cache filtering has the ability to reduce the bandwidth and power by filtering requests and directing the request to only the subset of sharers. There have been several techniques proposed to identify the subset of sharers ranging from using dynamically adaptive filters [106], destination-based sharers [107] and multicast snooping [108]. In this work, we adopt the filtering technique similar to multicast snooping [108]. However, we extend multicast snooping in the following way: We implement bit-vector sharing caches (VSC) embedded with the MCs. VSCs retain the sharing information such
that the bit-vector can implement two critical functions of (a) correctness - checking the sharing information on a prediction and (b) modification - updating the correct information by changing the multicast network by triggering additional ring resonators to be switched on. The VSCs require four additional waveguides, one for each VSC and utilize wavelength routing technique. Each VSC is a direct mapped cache that has the same number of entries as a L2 cache. Each entry is 31 bits where 10 bits is used for the tag and 21 bits is used for the sharing information. The 21 bits is further broken down into 16 bits for maintaining sharing information, 4 bits to indicate the owner (tile or MC), and 1 bit for valid bit. The VSC for each tile requires less than 250 KB storage which is significantly less than the 4 MB of storage required for the L2 cache. VSC will increase the area overhead and power dissipation by 5% than the baseline due to prediction checking and correction.

In order for the address and data networks to multicast to a limited number of tiles, we place two micro-ring resonators at each waveguide splitter, which are used to guide the optical signals. We extend the token waveguide system by adding two more waveguides and electrical circuitry that are responsible for controlling the multicast micro-rings. The two additional waveguides (one for the address and one for the data network) are used to communicate with the control center on which tiles send the request. Since each waveguide has 16 wavelengths, each wavelength represents the right to transmit to a select tile. Figure 3.7 illustrates the network modification required for Multi-PULSE. In the figure, different optical splitting points are labeled from 0 to 14, where 0 is the root optical splitter and 7-14 are leaf splitters. At these points, two micro-ring resonators are used to control the optical signals and two different configurations are shown in the figure. The top splitter (point 0) shows where neither micro-rings are switched on; this splits the optical beam into both the waveguides. The bottom splitter (point 6) shows the design where the left micro-ring is activated; this switches the optical beam into only left half of the branch. Therefore, if neither ring-resonators are activated, the signal splits into both waveguides; activating
Figure 3.7: Additional network components required for multicasting.

Figure 3.8(a) illustrates the steps in case of correct prediction. When a multicast takes place, the sending tiles first tries to capture an optical token (Step 1). After capturing the optical token, the tile checks its sharing cache to see if it can generate a mask (Step 2; example mask involves tiles 0, 2, 3, 12, 14, 15). If there is a valid entry, a mask will be generated in a manner similar to [108], where the next lower and higher address entries are combined together to generate a valid mask. After the mask has been generated, the tile will send the multicast information to the control center, along a separate waveguide to notify which micro-ring resonators should be activated (Step 3). The sending tile places the data onto the address or data network, where the optical signals will be guided to the...
correct destinations (Step 4). When the data arrives at the tiles and VSC, the multicast data is checked against the correct list of sharers and the cache block owner is notified to verify that the multicast is correct (Step 5). If the multicast is correct, the VSC notifies the sending tile to release the token so that the process starts over. (Step 6 and Step 7).

Figure 3.8(b) illustrates the step in case of a misprediction. When a multicast takes place, the sending tiles first tries to capture an optical token (Step 1). After capturing the optical token, the tile checks its sharing cache to see if it can generate a mask (Step 2, example mask involves tiles 0,10,12,15). Steps 3, 4, and 5 are repeated. In this case, as the multicast is incorrect, updated mask information is sent to the control center. When the control center receives the corrected list of the sharers, the addition micro-ring resonators will be activated/de-activate to correct the multicast request (Step 6, tiles 2, 3 and 14 are included in the mask). Once the multicast has been corrected (Step 7), the home directory notifies the sending tile to release the token (Step 8 and Step 9).

3.3.4 Multi-PULSE Control Circuitry

Figure 3.9 shows the electrical circuit required for multicasting. Figure 3.9(a) shows the multicast information which arrives through one of the waveguides. As there is one-to-one correspondence between the wavelength and tile number, active wavelength signals are filtered, which is then used to drive the appropriate micro-ring resonators. Figure 3.9(a) shows only the drivers at the root splitter and the electrical circuitry. In the figure, the \( L \) indicates the driver for left micro-ring resonator and \( R \) represents the right micro-ring resonator. If neither micro-ring resonator is turned on, then the signal will be split into both channels. Figure 3.9(b) illustrates the electrical circuitry that the directories use to correct an incorrect multicast. The correct multicast inputs coming from the directories are ORed, with inputs from Figure 3.9(a) to fix the multicast.
Figure 3.8: (a) Correctly predicted multicast example and (b) mispredicted multicast example.
Figure 3.9: (a) E/O circuitry required within the control block for multicasting and (b) Circuit required between the directories and control block for correcting wrong multicasts.

For a clearer understanding, Table 3.1 shows the algorithm for generating a bit sequence for driving the multicast micro-ring resonators. From the table, I only show how the bit sequence is generated for the root of the tree (labeled 0 in Figure 5), a subroot (labeled 1 in Figure 3), a column subroot (labeled 3 in Figure 5), and a leaf (label 7 in Figure 5). We need a pair of drive bits, one for the left micro-ring resonator and the other for the right micro-ring resonator. For example, if a multicast would send data to only Tile 0, Tile 1, Tile 4, Tile 5, Tile 8, Tile 9, Tile 12, and/or Tile 13, then the left drive bit would be turned on. On the other hand, if a multicast would send data to only Tile 2, Tile 3, Tile 6, Tile 7, Tile 10, Tile 11, Tile 14, and/or Tile 15, then the right drive bit would be turned on. However, if the multicast is required to send data to both the right half and left half of tiles, then both micro-ring resonators will be turned off. This logic works the same for the subroot (label...
1). If a multicast would send data to only Tile 0, Tile 4, Tile 8, and Tile 12, then the left drive bit would be turned on, and the right drive bit would be on if the multicast would send data to only Tile 1, Tile 5, Tile 9, and Tile 13. This same logic works for all the other optical splitters, except it will be a different combination of tiles.

3.3.5 Handling a L2 miss

Figure 3.10 illustrates a flow chart to decide whether to broadcast/multicast a cache request. When a L2 miss occurs, the direct map shared cache is checked to see if a sharing mask can be generated. If a mask can be generated, a list of tiles to multicast is generated, and a multicast will take place. If a mask can not be generated (cache block not shared or shared information is not available), a broadcast takes place and all tiles receive the L2 cache request. After broadcasting the L2 request, the token is released and the L2 request is completed. When the multicast L2 request completes, the directory will check to see if the correct tiles have received the L2 request. If the correct tiles receive the L2 request, the token is released and the L2 request is completed. On the other hand, if the multicast L2 request did not have all the sharers, the directory will notify the control center and the control center will fix the multicast. Finally, the token will be released and the multicast L2 request is completed.

3.3.6 Example of Generating a Mask

In [108], each node has a direct map cache that holds the current list of sharers and the owner of the cache block. When the node makes a cache request, it generates a multicast mask by including the nodes that share the block, as well as adjacent cache blocks. The reason for including the current sharers as well as adjacent blocks is that the sharing pattern may change, and the mask for the sharing may not be current or valid. Moreover, including adjacent sharers has been shown to improve the performance of the mask prediction. Figure
Table 3.1: Multicast algorithm for generating a bit vector that is used to drive the multicast micro-ring resonators

| Step 1: | Tile multicast (B₀, B₁, B₂, ..., B₁₄, B₁₅) |
| Step 2: | Control center receives bit sequence |
| Step 3: | Micro-ring resonator drive bits generated |
| | R₀ₑ = 1 if only send to (0,1,4,5,8,9,12,13) |
| | R₀ₑ = 1 if only send to (2,3,6,7,10,11,14,15) |
| | R₁ₑ = 1 if only send to (0,4,8,12) |
| | R₁ₑ = 1 if only send to (1,5,9,13) |
| | R₃ₑ = 1 if only send to (0,4) |
| | R₃ₑ = 1 if only send to (8,12) |
| | R₇ₑ = 1 if only send to (0) |
| | R₇ₑ = 1 if only send to (4) |

| Step 4: | Drive micro-ring resonators |
| Step 5: | Multicast L2 request |
| Step 6: | L2 Request arrive at Directory |
| Step 7(a): | If L2 request correct release token |
| Step 7(b): | Directory fix multicast then release token |

3.11 shows how a multicast mask is generated. In the figure, the L2 cache request is for the block address FFF1. When the cache request accesses its shared cache, it finds a valid mask for address FFF1, and also a valid mask for addresses FFF0 and FFF2. These three
masks are combined together to form a final multicast mask, shown in the bottom of Figure 3.11. In this example, the L2 request will be sent to Tile 0, Tile 2, Tile 3, Tile 9, Tile 10, Tile 12, and Tile 15.

3.3.7 Mask Network

Cache sharing information in PULSE is sent by a directory to the tile that broadcasts/multicasts the cache request. Figure 3.12 illustrates the additional optical network required for the directories to send cache sharing information to the requesting tiles. From the figure, the cache sharing network uses wavelength division routing where a single wavelength of light is used to send information to a select tile. In PULSE, each directory only
Figure 3.11: Mask generation example, where L2 request for block address FFF1 generates a mask from addresses FFF0, FFF1 and FFF2.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Share Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFF0</td>
<td>(9, 12)</td>
</tr>
<tr>
<td>FFF1</td>
<td>(0, 2, 3, 12)</td>
</tr>
<tr>
<td>FFF2</td>
<td>(2, 3, 15)</td>
</tr>
</tbody>
</table>

Broadcast Mask
\(\Sigma(0, 2, 3, 9, 10, 12, 15)\)

requires a single waveguide to send information to the 16 tiles, as each waveguide has a total of 16-wavelengths. In Figure 10, four tiles and a MC are shown for clarity purposes. When a cache request arrives at the MC, which in this case is located at Tile 0, the MC directory’s cache sharing information is looked up, and cache sharing information is sent to requesting tile. In this example, three different wavelengths, \(\lambda_0^0\), \(\lambda_1^0\), and \(\lambda_2^0\) are used, where the superscript indicates from which MC the information is coming, and the subscript indicates which tile will receive the information. Routing the correct wavelength of light to the tiles is accomplished by placing micro-ring resonators at waveguide intersections. These micro-ring resonators guide the correct wavelength of light to a corresponding tile. From the figure, a micro-ring resonator is required to switch \(\lambda_1^0\) to Tile 1 and a micro-ring resonator is required to switch \(\lambda_2^0\) to Tile 2.
Figure 3.12: Cache sharing network used by the directories to send caching information to the tiles.

### 3.4 Multi-PULSE Timing Diagram

Figure 3.13 shows a detail timing diagrams for broadcasting and multicasting cache request in PULSE. In Figure 3.13(a), the broadcasting of a cache request is shown. After the tile acquires the optical token, it takes two cycles for the data to arrive at all the tiles and the home MC. Once the cache request arrives, each cache and MC does a look up to see if it is the owner of the cache block, and if it is then send the cache block to the requesting tile. Figure 3.13(b) illustrates the timing diagram for multicasting in PULSE when the multicast mask is correct. In a multicast cache request there is an addition cycle required for the micro-rings to be activated to guide the light to the correct tiles in the mask. In addition, timing is shown for the multicast control logic, where two cycles are required for multicast information to arrive at the control center and to adjust the micro-rings and two additional cycles to do a directory lookup and notify the tile to release the token. Figure 3.13(c) illustrates the timing of a multicast that does not have the correct sharers in the mask. When the cache request arrives at the home directory it detects an
error in the multicast and sends corrected electrical signals to the micro-ring resonators. This adds two additional cycles, one for generating the correct electrical signals and the other cycle for the correct optical signals to arrive at the tiles.
<table>
<thead>
<tr>
<th>Time (cycle)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
</table>

**Address**
- Req.
- Token
- Grant
- Token
- Addr
- Bus
- Cache
- access
- Cache
- read

**Data**
- Req.
- Token
- Grant
- Token
- Addr
- Bus
- Cache
- access
- Cache
- read
- D0
- D1
- D2
- D3

Figure 3.13: Timing diagram for (a) broadcast, (b) multicast, and (c) multicast with error.
4 PERFORMANCE RESULTS

In this chapter, the performance, power-efficiency and area overhead of 3D-NoC and PULSE will be compared to other leading electrical and optical networks.

4.1 3D-NoC Results

4.1.1 Simulation Setup

The cycle arcuate simulator models used simulates in detail the router pipeline, arbitration, switching and flow control. An aggressive single cycle electrical router is applied in each tile and the flit transversal time is one cycle from the local core to electrical router [109]. As the delay of Optical/Electrical (O/E) and Electrical/Optical (E/O) conversion can be reduced to less than 100 ps, the total optical transmissions latency is determined by physical location of source/destination pair (1 - 3 cycles) and two additional clock cycles for the conversion delay. In addition, a latency of 1 to 3 cycles was assumed for a tile to capture an optical token. Each input buffer holds 16 flits with each flit consisting of 128 bits. The packet size is 4 flits which will be sufficient to fit a complete cache line of 64 bytes. 3D-NoC uses a supply voltage $V_{dd}$ of 1.0 V and a router clock frequency of 5 Ghz [5, 6].

3D-NOC architecture is compared to three other crossbar-like nanophotonic interconnects, Corona [5], Firefly [6] and MPNOC [59] and two electrical interconnects (mesh and Flattened Butterfly (FB)) [19]. We implement all architectures such that four cores (one tile) are connected to a single router. Token slot is used for both 3D-NOC and Corona to pipeline the arbitration process to increase the efficiency. Multiple requests can be sent from the four local cores to optical channels to increase the arbitration efficiency. We use Fly_Src routing algorithm [6] for Firefly architectures, where intra-group communication via electrical mesh is implemented first and then inter-group via nanophotonic interconnects. For a fair comparison, each communication channel in either electrical or optical
network is 640 Gbps with 64 wavelengths. In addition, nanophotonic networks are evaluated by reducing the channel bandwidth to 16/8 wavelengths and communication bandwidth is limited to 160/80 Gbps.

(a) Open-loop Measurements

For open-loop measurement, the packet injection rate is varied from 0.1 to 0.9 of the network capacity, and packets are injected according to the Bernoulli process based on the given network load [17, 110]. We consider both uniform as well as permutation traffic such as bit-complement (bitcomp), bit-reversal (bitrev), transpose, butterfly, neighbor and perfect shuffle synthetic traffic patterns [17]. Synthetic traffic patterns represent well known network traffic patterns found in shared-memory applications such as SPLASH-2. The following briefly explains different synthetic traffic patterns.

Uniform: In uniform traffic, the destination of each packet is randomly chosen with each node having equal probability of being the destination node of the packet. This traffic pattern distributes the network load evenly across the network allowing for a thorough evaluation of the network’s throughput and average packet latency.

Bit-complement: In Bit-complement traffic, the destination of a packet is the complement of the packet’s source node. If the source node of a packet is given by \(x_n x_{n-1} \ldots x_1 x_0\) then the destination is \(\overline{x_n x_{n-1} \ldots x_1 x_0}\). This traffic pattern is good for stressing different ranges of the network as the average hop count (number of router a packet traverses) of a packet is higher than in uniform traffic.

Matrix-transpose: In matrix-transpose traffic, the destination of a packet is the bit reversal to the different halves of the source bit sequence. If the source node of a packet is given by \(x_n x_{n-1} \ldots x_1 x_0\) then the destination is \(x_{n/2-1} x_{n/2-2} \ldots x_{n} x_{n-1}\).
**Butterfly:** In butterfly traffic, the destination of a packet is the switch of the most significant bit with the least significant bit. If the source node of a packet is given by $x_nx_{n-1}...x_1x_0$ then the destination is $x_0x_{n-1}...x_1x_n$.

**Neighbor:** In neighbor traffic, the destination of a packet is one of the neighboring nodes relative the source node. If the source node of a packet is given by $x_nx_{n-1}...x_1x_0$ then the destination is $x_0x_{n-1}...x_1x_\bar{n}$.

**Perfect Shuffle:** In perfect shuffle traffic, the destination of a packet is the bit shift with a carrier of the source node. If the source node of a packet is given by $x_nx_{n-1}...x_1x_0$ then the destination is $x_{n-1}x_{n-2}...x_0x_n$.

**(b) Closed-loop Measurements**

For closed-loop measurement, traffic traces were collected from real applications using the full execution-driven simulator SIMICS from WindRiver, with the memory package GEMS enabled [111]. We evaluate the performance of 64-core versions of the networks on Stanford ParalleL Applications for SHared memory (Splash-2) [81], Princeton Application Repository for Shared-Memory Computers (PARSEC) [82, 112] and Standard Performance Evaluation Corporation (SPEC) CPU2006 workloads and 256-core version on synthetic and workload completion traffic (a mixture of synthetic traces). Table 4.2 shows the core and cache parameters used for Splash-2, PARSEC and SPEC2006 workloads. A 2 cycle latency is assumed to access the L1 cache (64 KB, 4-way), and a 4 cycle latency is assumed to access the L2 cache (4MB, 16-way). Each cache line size is 64 bytes and there a a 160 cycle latency to access main memory. In addition, there are 16 memory controllers used to access main memory and each processor can issue two threads. For Splash-2 traffic, Table 4.1 shows the applications and their input parameters used. In each simulation, six PARSEC applications with medium inputs (blackscholes, facesim, fluidanimate, freqmin, and streamcluster ) and two workloads from SPEC CPU2006 (bzip and hmmer) are used.
The SPLASH-2 applications are a collection of benchmark programs used to facilitate the study of shared-memory multi-processors. Each application represents a different aspect of high performance computing that stress different parts of network and processor. Applications ranges for solving a fourier transform to simulating how ocean currents move. The following gives a brief explanation of each SPLASH-2 applications used in closed-loop simulations [81].

**FFT:** The Fast Fourier Transform (FFT) application solves a 1-Dimensional array using the six step radix-$\sqrt{n}$ kernel algorithm. In addition, the six step radix-$\sqrt{n}$ algorithm is used for solve an FFT as inter-processor communication is minimized.

**LU:** The LU application factors a dense matrix into its upper and lower matrix components. The application solves this by dividing the dense matrix into smaller $N\times N$ arrays with $B\times B$ blocks is distributing the smaller array to different processors.

**Radix:** The radix application is an integer sort kernel. The kernel algorithm performs one step at a time in iterative step on each key. During each step, a key is assigned and a histogram is generated base of the data collected.

**Raytrace:** The Raytrace application creates a 3D scenery by tracing rays from different objects to be rendered. Each ray is trace from every pixel to the rendering object to create high detail image.

**Ocean:** The ocean application solves the movements of large ocean currents by focusing of the eddy and boundary current of a ocean wave.

**Water:** The water application solves the forces and moments of a system of water molecules over time.
Table 4.1: Splash-2 suite application and kernel problem sizes used in simulation.

<table>
<thead>
<tr>
<th>Applications</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>16 K particles</td>
</tr>
<tr>
<td>LU</td>
<td>512 × 512 particles</td>
</tr>
<tr>
<td>Ocean</td>
<td>258 × 258</td>
</tr>
<tr>
<td>Radix</td>
<td>1 M integers</td>
</tr>
<tr>
<td>Water</td>
<td>512 Molecules</td>
</tr>
</tbody>
</table>

PARSEc is a suite of multithreaded programs that emphasize on emerging workloads and future shared-memory applications for the study of CMPs [82, 112]. PARSEC differs from other benchmark applications as PARSEC focuses on a range of emerging multithreaded applications instead of high-performance computing applications. PARSEC application range from face tracking, data mining, recognition to Stock Market price prediction. In addition, PARSEC applications have been design and optimized for execution on CMPs. Standard Performance Evaluation Corporation (SPEC) CPU 2006 is industry-standardized range of applications that focus on stressing the CPU and the system memory subsystem with intensive workloads [83]. SPEC CPU 2006 provides a wide range of computer-intense applications that focus on workloads from real user applications. In addition, SPEC CPU 2006 applications stress the whole computer systems from the CPU/memory to I/O operations. The following gives a brief explanation of each PARSEC and SPEC CPU 2006 applications used in closed-loop simulations.

**Blackscholes:** Blackscholes is an Intel RMS (recognition, mining and synthesis) application that uses the Blackscholes partial differential equation to calculate the portfolio price in the stock market.
**Facesim:** Facesim application is an Intel RMS application that simulates the underlying physics of realistic facial movements.

**Fluidanimate:** Fluidanimate is an Intel RMS application that simulates the movements and interaction of incompressible fluids.

**Freqmin:** Freqmin is an Intel RMS benchmark used in data mining that utilizes the FP-growth (Frequent Pattern-growth) technique.

**Bzip:** Bzip is a SPEC CPU 2006 application that is used in compression and decompression of data.

**Hmmer:** Hmmer is a SPEC CPU 2006 application that is used in the search of gene sequences.

**(c) Rₖ Sizing**

Several benchmarks of PARSEC and Splash-2 were ran to determine the optimum size of Rₖ by varying the simulation cycles. While initially the performance improved with increasing window size as more statistics is available to make better decision, at very large window size, the performance diminishes as the algorithm cannot react well. Therefore, Rₖ is kept at 1300 cycles as this provides the best performance. In addition, a 100 cycle latency is assumed for the reconfiguration to take place after each Rₖ (three-way handshake delay) with only the links which are already experiencing light load are subject to these delays. Figure 4.1.1 illustrates the increase in performance for three different applications. In Figure 4.1.1(a), the speed for blackscholes is shown using different Rₖ values. In Figure 4.1.1(b), the speed for facesim is shown using different Rₖ values. In Figure 4.1.1(c), the speed for hmmer is shown using different Rₖ values.
Table 4.2: Core and cache parameters used for Splash-2, PARSEC, and SPEC CPU2006 application on SIMICS using GEMS.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1/L2 coherence</td>
<td>MOESI</td>
</tr>
<tr>
<td>L2 cache size/accoc</td>
<td>4MB/16-way</td>
</tr>
<tr>
<td>L2 cache line size</td>
<td>64</td>
</tr>
<tr>
<td>L2 access latency(cycles)</td>
<td>4</td>
</tr>
<tr>
<td>L1 cache/accoc</td>
<td>64KB/4-way</td>
</tr>
<tr>
<td>L1 cache line size</td>
<td>64</td>
</tr>
<tr>
<td>L1 access latency(cycles)</td>
<td>2</td>
</tr>
<tr>
<td>Core Frequency(GHz)</td>
<td>5</td>
</tr>
<tr>
<td>Threads(core)</td>
<td>2</td>
</tr>
<tr>
<td>Issue policy</td>
<td>In-order</td>
</tr>
<tr>
<td>Memory Size(GB)</td>
<td>4</td>
</tr>
<tr>
<td>Memory Controllers</td>
<td>16</td>
</tr>
<tr>
<td>Memory latency(cycle)</td>
<td>160</td>
</tr>
<tr>
<td>Directory latency(cycle)</td>
<td>80</td>
</tr>
</tbody>
</table>

4.1.2 Simulation Results

(a) Splash-2, PARSEC and SPEC CPU2006: 64 Cores

Figure 4.2(a) shows the speed-up for the Splash-2 [81] applications using 64 wavelengths, Figure 4.2(b) shows the speed-up for the PARSEC [82, 112] and Figure 4.2(c) shows the speed-up for SPEC CPU2006 [83] applications using 64 wavelengths. The different configuration of 3D-NOC-L1 have an average speed up of about 3.0 for Splash-2 application benchmark and 2.5 for PARSEC and Spec CPU2006 application benchmarks.
Figure 4.1: Reconfiguration window $R_W$ size versus improvement in performance for blackscholes.
over the mesh network for 64 wavelengths. In the water application, 3D-NOC has the highest speed-up with a factor of over 3 relative to mesh. This is a result of 3D-NOC’s decomposed crossbars allowing for fast arbitration of network resources (less contention) and water’s traffic pattern ability to take full advantage of the additional bandwidth provide during reconfiguration. When 3D-NOC is compared to FB, 3D-NOC has a 25% - 30% improvement. As FB is a two-hop network and most traffic under Splash-2 suite are two hops, the intermediate router reduces the throughput of the network. When 3D-NOC is compared to Firefly, 3D-NOC outperforms Firefly by about as much as 38% which is a result of Firefly routing its traffic through both an electrical and optical network. As for Corona, 3D-NoC has a speed-up range as low as 3% for the bzip application and as high as 22% for the facesim application. When the different configurations of 3D-NOC are compared to each other, it is no surprise there is an increase in performance. For the LU, water, streamcluster and facesim benchmarks, 3D-NOC-L2 and 3D-NOC-L3 show over a 10% increase in performance when compared to 3D-NOC-L1. These traffic patterns benefit from the additional bandwidth available via reconfiguration when compared to other benchmarks. On the other hand, FFT, radix, and hmmer benchmarks only show a very small increase in performance over 3D-NOC-L1. This is mainly due to the fact the additional reconfigurations options provided by 3D-NoC-LA, 3D-NOC-L2, and 3D-NOC-L3 can not be used by these benchmarks. From the figure, the average speed provided by 3D-NoC-LA, 3D-NOC-L2, and 3D-NOC-L3 over 3D-NOC-L1 ranges from about 1% to as high as 10%.

To further illustrate the potential increase in performance from reconfiguration, Figure 4.3(a), Figure 4.3(b), and Figure 4.3(b) show the results in a resource constrained environment with only using 8-wavelengths. Further, Figure 4.4(a), Figure 4.4(b) and Figure 4.4(c) show the results in a resource constrained environment with with only using 16-wavelengths. Using 8-wavelengths, 3D-NoC shows the high speed-up factor over
Figure 4.2: Simulation speed-up for 64-core using 64-wavelengths: (a) SPLASH-2, (b) PARSEC and (c) SPEC CPU2006.

MPNOC. For example, the LU application shows a speed-up of 40% for 3D-NOC-L3 when compared to MPNOC. This percentage increase in performance is about 5× more when compared to the 64-wavelengths results (only 6% increase over MPNOC) and is a result of network resources being further constrained and any additional bandwidth will greatly
speed up the application. 3D-NoC also shows high speed-up factor over MPNOC for water and ocean applications with 3D-NoC-L3 having a 33% increase in performance for the water application and a 30% increase in performance for the ocean application. Using 16-wavelengths, there is an increase speed-up over 64-wavelengths but the speed-up factor is not as high. For the LU application, 3D-NoC-L3 only shows a 25% increase in performance over MPNOC as compared to 40% increase in performance when 8 wavelengths are used. For the water application, 3D-NoC-L3 only shows a 17% increase in performance over MPNOC as compared to 33% increase in performance when 8 wavelengths are used.

(b) Synthetic Traffic: 256 Cores

The throughput for all synthetic traffic traces for 256-core implementations are shown in Figure 4.5 and is normalized to mesh network (for Uniform, the mesh has a throughput of 624 GBytes per sec). 3D-NOC-L1 has about a 2.5 × increase in throughput over Corona for uniform traffic due to the decomposition of the nanophotonic crossbar. The decomposed crossbars allow for a reduction in contention for optical tokens as now a single token is shared between 16 tiles instead of 64 tiles as in Corona. Firefly slightly outperforms 3D-NOC-L1 for uniform traffic due to the contention found in the decomposed nanophotonic crossbars. Moreover, Firefly uses a SWMR approach for communication which does not require optical arbitration. From the figure, 3D-NOC-L1 slightly outperforms Corona for bit-reversal and complement traffic traces. This is due to lower contention for optical tokens in the decomposed crossbars. 3D-NOC-L1 significantly outperforms mesh for the bit-reversal, matrix-transpose and complement traffic patterns. In these traffic patterns, packets need to traversal across multiple mesh routers which in turn increases the packet latency and thereby reduces the throughput. When 3D-NOC-L1 is compared to Firefly, 3D-NOC-L1 outperforms Firefly by 2.5 ×. In Firefly, most traffic patterns will require packets to traverse across multiple electrical routers and then traversal across an optical link resulting
in a reduction in the number of packets that can be injected into the network as compared to 3D-NOC-L1. 3D-NOC-L1 is able to out perform 3D-NOC-L1 for complement, matrix-transpose and perfect shuffle traffic traces. These permutation traffic traces exhibit adversial patterns which will benefit 3D-NOC-L1. In complement traffic, 3D-NOC-L1 has about
Figure 4.4: Simulation speed-up for 64-core using 16-wavelengths: (a) SPLASH-2, (b) PARSEC and (c) SPEC CPU2006.

a 55% increase in performance when compared to 3D-NOC-L1. Complement traffic pattern show cases the best performance as a single source tile will communicate with a single destination tile, thereby providing opportunities to improve performance via reconfiguration. When 3D-NOC-L1 is compared to 3D-NoC-LA, 3D-NoC-L2, and 3D-
NOC-L3, these other configurations of 3D-NOC do not show any additional speed up over 3D-NoC-L1 except for the complement traffic pattern.

(c) Workload Completion

For a further evaluation of network perform, we subject each network to a work load completion traffic. During work load completion traffic, each core injects 250 packets into the network where select regions of cores are subjected to either uniform, bit-reversal, complement or matrix-transpose synthetic traffic. Afterwards, the time it takes to complete the workload traffic is used as a benchmark for evaluation the performance of the select networks. Figure 4.6 shows speed-up of 3D-NoC-L1 and competitive networks normalized
to the mesh. From the figure, 3D-NoC-L1 is the best performing network. 3D-NoC-L1 is able to outperform 3D-NoC-L1 because the work load completion traffic is comprised of traffic traces that include complement and matrix-transpose where the reconfiguration algorithm plays a role in improving performance. 3D-NoC-L1 was only shown in the figure as the other version of 3D-NoC do not shows much increase in performance over 3D-NoC-L1.

### 4.1.3 Energy Comparison

The energy consumption of a nanophotonic interconnect can be divided into two parts, electrical energy and optical energy. Optical energy consists of the off-chip laser energy and on-chip micro-ring resonators heating energy. In what follows, I first discuss the electrical energy and then optical energy consumption.
(a) Electrical Energy Model

Electrical energy dissipated includes the energy of the link, router and back-end circuit for optical transmitter and receiver. The electrical power dissipation of a router is given:

$$E_{\text{router}} = E_{\text{cl}} + E_{\text{buf}} + E_{\text{cross}} + E_{\text{sw,arbiter}} + E_{\text{va,arbiter}}$$

(4.1)

where $E_{\text{cl}}$ is the energy dissipated in the clock circuitry, $E_{\text{buf}}$ is the energy dissipated in the VC buffers, $E_{\text{cross}}$ is the energy dissipated in a crossbar, $E_{\text{sw,arbiter}}$ is the energy dissipated in the switch arbitration unit and $E_{\text{va,arbiter}}$ is the energy dissipated in the virtual channel arbiter. We use ORION 2.0 [113, 114] to obtain the energy dissipation values for an electrical link and router and modified their parameters for 22nm technology according to ITRS. We assume all electrical links are optimized for delay and the injection rate to be 0.1. Moreover, I include the energy dissipated in both planar and vertical links (communicating only with layer one). The length of electrical links in Firefly and mesh are 20 mm/8 = 2.5 mm and 20 mm/16 = 1.25 mm respectively. The energy for planar link is conservatively obtained as 150 fJ/bit for Firefly and 75 fJ/bit for mesh under low swing voltage level [114]. It should be mentioned that the energy per bit per distance is the same in Firefly and mesh network. A mesh link dissipates half the energy as Firefly link because a mesh link is half the distance of a Firefly link. For a 10-layer chip, the vertical via is determined as ~100-200μm [44], which is significantly less than planar links. As a result, the power consumption of vertical links is very small and are not included in electrical link power model. For the electrical router power, we calculate the energy dissipated, per hop, in a 8 × 8 router to be 300 fJ/bit [114]. A 5 × 5 router with the same buffer size is 220 fJ/bit [114]. For the 8 × 8 router, the clock contributes 47 fJ/bit, the buffers contributes 70 fJ/bit, the crossbar contributes 178 fJ/bit, switch arbiter contributes 2.5 fJ/bit and the VA arbiter contributes 2.5 pJ/bit. As for the 5 × 5 router, the clock contributes 34 fJ/bit, the buffers contributes 0.051 pJ/bit, the crossbar contributes 135 fJ/bit, switch arbiter contributes 1.8
fJ/bit and the VA arbiter contributes 1.8 fJ/bit. For each optical transmitted bit, I need to provide electrical back end circuit for transmitter end and receiver end. We assume the O/E and E/O converter energy is 100fJ/b, as predicted in [115].

(b) Optical Energy and Loss Model

The optical power budget is the result of the laser power and the power dissipated for the micro-ring resonators. The equation for laser power is given by

$$P_{\text{laser}} = P_{\text{rx}} + C_{\text{loss}} + M_s$$

(4.2)

where $P_{\text{laser}}$ is the laser power requirement, $P_{\text{rx}}$ is the receiver sensitivity, $C_{\text{loss}}$ is the channel losses and $M_s$ is the system margin. In order to perform an accurate comparison with the other two optical architectures, I use the same optical device parameters and loss values provided in [28], as listed in Table 4.3.

Based on the energy model discussed in the previous section, I calculate the energy parameters of all four architectures as shown in Table 4.4. We test uniform traffic, complement traffic and butterfly traffic with 0.1 injection rate and obtain energy per-bit comparison shown in Figure 4.7. Figure 4.7(a) shows the energy per bit for uniform traffic. MPNOC is the most energy efficient network followed by Corona and 3D-NOC-L1 being the next most energy efficient network. In general, 3D-NOC-L1 saves 23.1%, 36.1% energy per bit compared to Firefly and mesh respectively. Figure 4.7(b) shows the energy per bit for complement traffic. For complement traffic, the average energy per bit increased for mesh, FB and firefly. This increase in power is contributed to the higher hop count of complement traffic over uniform traffic. Optical networks such as 3D-NOC show a similar energy per bit when compared to uniforms because all most all traffic will traverse across a single nanophotonic interconnect no matter what the traffic pattern used. Figure 4.7(c) shows the average energy per bit for butterfly traffic. In butterfly traffic,
Table 4.3: Electrical and optical power losses.

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laser efficiency</td>
<td>5</td>
<td>dB</td>
</tr>
<tr>
<td>Coupler</td>
<td>1.0</td>
<td>dB</td>
</tr>
<tr>
<td>Waveguide</td>
<td>1.0</td>
<td>dB/cm</td>
</tr>
<tr>
<td>Splitter</td>
<td>0.2</td>
<td>dB</td>
</tr>
<tr>
<td>Non-Linearity</td>
<td>1</td>
<td>dB</td>
</tr>
<tr>
<td>Ring insertion</td>
<td>1e-2 - 1e-4</td>
<td>dB</td>
</tr>
<tr>
<td>Ring Drop</td>
<td>1.0</td>
<td>dB</td>
</tr>
<tr>
<td>Waveguide Crossings</td>
<td>0.5</td>
<td>dB</td>
</tr>
<tr>
<td>Photo Detector</td>
<td>0.1</td>
<td>dB</td>
</tr>
<tr>
<td>Ring Heating</td>
<td>26</td>
<td>μW/ring</td>
</tr>
<tr>
<td>Ring Modulating</td>
<td>500</td>
<td>μW/ring</td>
</tr>
<tr>
<td>Receiver Sensitivity</td>
<td>-26</td>
<td>dBm</td>
</tr>
</tbody>
</table>

Table 4.4: Electrical power dissipation for Nanophotonic interconnects.

<table>
<thead>
<tr>
<th></th>
<th>Corona</th>
<th>Firefly</th>
<th>3D-NOC</th>
<th>Mesh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link(electric)</td>
<td>-</td>
<td>150fJ/b</td>
<td>-</td>
<td>75fJ/b</td>
</tr>
<tr>
<td>Router</td>
<td>220fJ/b</td>
<td>300fJ/b</td>
<td>220fJ/b</td>
<td>220fJ/b</td>
</tr>
<tr>
<td>O/E, E/O</td>
<td>100fJ/b</td>
<td>100fJ/b</td>
<td>100fJ/b</td>
<td>-</td>
</tr>
<tr>
<td>Optical loss</td>
<td>-25.2dB</td>
<td>-17.6 dB</td>
<td>-16dB</td>
<td>-</td>
</tr>
<tr>
<td>Power(λ)</td>
<td>0.81mW</td>
<td>0.14mW</td>
<td>0.10mW</td>
<td>-</td>
</tr>
<tr>
<td>Laser power</td>
<td>13.6W</td>
<td>2.4W</td>
<td>6.1W</td>
<td>-</td>
</tr>
<tr>
<td>Ring heating</td>
<td>26W</td>
<td>6.5W</td>
<td>27.5W</td>
<td>-</td>
</tr>
</tbody>
</table>
the average energy per bit for mesh and FB is lower than 3D-NOC-L1. This is due to most traffic in butterfly being neighbor traffic, while high energy is dissipated to traversal nanophotonic interconnect than using shorted electrical interconnects. It should be noted that when the network injection rate increases, 3D-NOC-L1 becomes much more energy efficient than other three architectures. 3D-NOC-L2 and 3D-NOC-LA has a slight increase in power dissipation over 3D-NOC-L1 due to the additional micro-ring resonators required for reconfiguration.

(c) Laser Power Variations

The optical losses shown in Table 4.4 are mostly conservative estimates that may not reflex the actual optical losses of future nanophotonic devices. In order to compensate for this potential variation in optical losses from the ones shown in Table 4.3, I provide Figure 4.8 that illustrate how varying optical losses affect the overall laser power. In Figure 4.8(a), the total laser power is a function of both receiver sensitivity and the number of available wavelengths. From the figure, it is easy to see that the laser power increases dramatically as the receiver sensitivity decreases. For example, a 6 dBm decease in receiver sensitivity (-20 dBm) would result in a 4x increase in total laser, resulting in an average energy per bit that is close to mesh. Furthermore, the laser power is doubled when the number of wavelengths is reduced by 50 percent, but on the other hand the laser power can be cut in half if the number of wavelengths available is 128.

As receiver sensitivity and number of wavelengths are not the only optical parameters affecting the total laser power, Figure 4.8(b) illustrates the affects ring filter loss and waveguide loss have on the total laser power. From the figure, the increase in total laser power from waveguide losses has a greater impact than ring filter losses. This is due to the optical signal traversing several centimeters before arriving at the photodetector. For example, a 0.5 dB increase in waveguide losses (1.8 dB/cm) would more than double the
Figure 4.7: Average energy per-bit for electrical and nanophotonic interconnects: (a) uniform traffic, (b) complement traffic and (c) butterfly traffic.
Figure 4.8: (a) Increase in laser power for different values of receiver sensitivity and number of wavelengths, and (b) Increase in laser power for different values of waveguide loss and ring filtering loss.
total laser power. Both Figure 4.8(a) and Figure 4.8(b) illustrate the impact on laser power when four optical parameters are changed. These select four parameters were selected from Table 3 because I believe they will have the greatest impact on the total laser power.

(d) 3D-NOC Energy-Delay Product

In this dissertation, I propose different configuration of 3D-NOC that have different amounts of reconfiguration (increase bandwidth performance). In addition from the performance figures, these different configurations of 3D-NOC dissipate different amounts of energy per bit. As such, the increase in performance due to more reconfiguration options may come at the price of higher energy dissipation. Figure 4.9 illustrate that energy-delay product for each network using the Splash-2, PARSEC, and SPEC CPU 2006 benchmarks. From the figure, it is easy to see that 3D-NOC-L1 and 3D-NOC-LA have a least energy-delay product. This is due to fact that the slight increase in energy per bit over MPNOC and 3D-NOC-L1 is outset to by increase in performance than the other networks. On the other hand, 3D-NOC-L2 and 3D-NOC-L3 have the highest energy-delay product of the all optics networks. This is obvious as these two networks have the high energy per bit and there is only a slight increase in performance when compared to 3D-NOC-L1, 3D-NOC-LA and Corona. Mesh has the highest energy-delay product as it is the worst network in terms of performance and has the highest energy per bit. When FB and Firefly are compared, their energy-delay product have similar values. This is a result of the decrease in benchmark speed up for Firefly is compensated by a proportionally decrease in the energy per bit for Firefly

4.1.4 Area Analysis

In this subsection, we analytically compare the optical and electrical area overhead of 3D-NoC to Firefly [6] and Corona [5] nanophotonic interconnects. For the optical area
overhead, I considered the area required for all waveguides, micro-ring resonators and photodetectors. For the electrical layer, we considered the area required for all routers, electrical links and electrical receiver circuitry. Table 4.5 shows the area overhead of
Table 4.5: Electrical and optical area overhead for select electrical and optical components

<table>
<thead>
<tr>
<th>Component</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrical Link</td>
<td>0.0085 (mm$^2$)</td>
</tr>
<tr>
<td>Router (8 × 8)</td>
<td>0.128 (mm$^2$)</td>
</tr>
<tr>
<td>Photodetector receiver circuitry</td>
<td>0.02625 (mm$^2$)</td>
</tr>
<tr>
<td>Microring resonator</td>
<td>100($\mu$m$^2$)</td>
</tr>
<tr>
<td>Photodetector</td>
<td>100($\mu$m$^2$)</td>
</tr>
<tr>
<td>Waveguide</td>
<td>5.5 $\mu$m</td>
</tr>
</tbody>
</table>

Both optical and electrical components used in the area calculation. From Table 4.5, each router and electrical link values were obtained from Orion 2.0 by directing scaling 32 nm technology values to 22 nm technology.

Both Corona and Firefly require 10% more optical area than 3D-NoC. This may be counter-intuitive, but 3D-NoC uses decomposed crossbars that permit waveguides in 3D-NoC to be shorter than the long serpentine waveguides used in both Corona and Firefly. In terms of electrical layer area overhead, 3D-NoC consumes 4X more electrical area than Corona. As each tile is connected to four optical layers to facilitate inter-group communication, each tile in turn should have the ability to receive four signals instead of one as in Corona. However, when 3D-NoC is compared to Firefly in terms of electrical area overhead, Firefly consumes about 75% more area. In Firefly, the electrical network can simultaneously receive from seven sets of optical receivers at once due to SWMR organization. 3D-NoC combines both MWSR (Corona) and SWMR (Firefly) communication channels, thereby increasing the communication channels to each tile while reducing the optical area overhead.
4.2 PULSE Results

4.2.1 Simulation Setup

Pulse is simulated using the full system simulator, called SIMICS [116] with the GEMS memory system [111] for select SPLASH-2 and PARSEC benchmarks. PULSE is compared to an electrical bus network (EB), an electrical tree network (ET), and the electro-optics Shared-Bus network (SB) with a similar cache coherent design. Table 4.6 shows the full-system configuration used in Simics simulator. For all networks, a clock frequency of 2.5 GHz and equal link bandwidth for all networks. Table 4.7 shows the number of cycles required for the cache request to traverse each network. The three different values for PULSE shows the latency for a broadcast, multicast (when correct) and multicast (when incorrect). The number of cycles required includes the E/O conversions, optical signal propagation latency and O/E conversions at the receiver. For all networks, we simulate them using the FFT, LU, Radix, Raytrace, and Ocean for the SPLASH-2 applications and Blackschole, Ferret, Frequency Mine (FREQ) and Fluid Animate (FLUID) for PARSEC applications.

4.2.2 Simulation Results

(a) 16-core

Figure 4.10 shows the speed-up of Splash-2 and PARSEC applications for 16-core multicores. Both PULSE and multi-PULSE (M-PULSE) are the best performing networks when compared to both electrical as well as nanophotonic networks. For each application, the time spent servicing cache hits are the highest percentage of the execution time. Network access along with cache misses consume the second and third highest percentage of execution time. VSC access consumes the least execution time for each application as it is a direct-mapped cache. PULSE shows a speed-up of around 2.5× when compared to
Table 4.6: Full-system configuration.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores</td>
<td>SPARC ISA, 2.5 GHz clock, 2-way in-order, 2 integer ALU, 2 FPU ALU, 1 FL mult/div,</td>
</tr>
<tr>
<td>Branch Predictor</td>
<td>Bimodal (512) + Gshare (1024)</td>
</tr>
<tr>
<td>BTB</td>
<td>4K entries; 16-way</td>
</tr>
<tr>
<td>L1 cache</td>
<td>32KB instruction/32 KB data, 4-way associative, 64B lines, 3 cycle access time</td>
</tr>
<tr>
<td>L2 cache</td>
<td>4MB, 64B lines, 16-way associative 6-cycle/bank access time</td>
</tr>
<tr>
<td>Memory</td>
<td>160 cycle access time, 4 on-chip memory controllers</td>
</tr>
<tr>
<td>SPLASH-2</td>
<td>FFT, LU, Radix, Raytrace</td>
</tr>
<tr>
<td>PARSEC</td>
<td>Blackscholes, Ferret, Fluidanimate</td>
</tr>
<tr>
<td></td>
<td>Freqmine, X26, Swaption</td>
</tr>
</tbody>
</table>

the electrical tree network and a speed-up of around 1.5× when compared to the electrical-bus and Shared-Bus for the Raytrace and Radix applications. These applications show a higher speed-up because they have large L2 misses. The high-speed nanophotonic network PULSE allows for cache requests to be quickly completed when compared to electrical networks. For the FFT application, the multicast version of PULSE marginally under-performs the Shared- Bus network due to higher multicast miss-predictions. For PARSEC applications, PULSE shows about a 1.66× speedup over the electrical tree network and about a 1.2× speedup over Shared-bus for the Ferret application. These performance
Table 4.7: Address network broadcast network delays (cycles).

<table>
<thead>
<tr>
<th>Network</th>
<th>Delay (cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrical Tree</td>
<td>11</td>
</tr>
<tr>
<td>Electrical Bus</td>
<td>8</td>
</tr>
<tr>
<td>Shared-Bus</td>
<td>5</td>
</tr>
<tr>
<td>PULSE (broadcast)</td>
<td>2</td>
</tr>
<tr>
<td>PULSE (multicast)</td>
<td>3</td>
</tr>
<tr>
<td>PULSE (multicast error)</td>
<td>5</td>
</tr>
</tbody>
</table>

Improvements clearly indicate the advantages of using nanophotonics for snoopy cache coherent protocols.

(b) 64-core

Figure 4.11 shows the execution time of the select Splash-2 and PARSEC applications for 64-cores. Both PULSE and Multi-PULSE are able to outperform the electrical tree by about 50% and electrical bus by about 20%. PULSE demonstrates the highest decrease in execution time by about 80% for the radix application. As was the case for the 16-core version of PULSE, the raytrace application can further reduce the time cache requests are on the network. Freq. and X26 demonstrate the least decrease in execution time as these applications show a small percentage of application time is spend in the network. When PULSE is compared to multi-PULSE, multi-PULSE only shows a small increase in execution time over PULSE. For the swaption application, multi-PULSE shows about 5% increase in execution time over PULSE as this application has a higher miss rate. For LU and Blackschole application shared-bus outperforms
Figure 4.10: Speed-up running (a) Splash-2 and (b) PARSEC applications for 16 cores.

(c) Multicast Performance

Table 4.8 shows the average number of sharers and Table 4.9 shows the percentage of cache misses for select Splash-2 and PARSEC application. Radix and FFT have the highest number of sharers, with each cache request for FFT having about 7.3 sharers and each cache request for Radix having about 8.5 sharers for 16-core PULSE. On the other hand, BlackScholes and Ocean have the lowest number of sharers, with each cache request for BlackScholes having 3 sharers and each cache request for Ocean having 3.4 sharers for 16-core PULSE. Raytrace and Freq. have the highest percentage of the misses, where 19% of multicast cache requests do not have the correct number of sharers for Ocean application and 17% of multicast cache request do not have the correct number of sharers. For both of
these applications, the number of tiles sharing a cache block and/or the owner of the cache block quickly changes making the cache sharing information not up to date. This can be seen in Figure 4.10 where this is significantly more decrease in performance for multi-
PULSE over PULSE. One the other hand, BlackScholes and Ferret have least number of sharers, where only 1% of multicast request do not have the correct number of sharers for BlackScholes and Ferret applications.
Table 4.8: Average number of sharers for select Splash-2 and PARSEC benchmarks.

<table>
<thead>
<tr>
<th>Application</th>
<th># Shares (16-cores)</th>
<th># Shares (64-cores)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>7.3</td>
<td>4.48</td>
</tr>
<tr>
<td>LU</td>
<td>6.5</td>
<td>2.97</td>
</tr>
<tr>
<td>Radix</td>
<td>8.5</td>
<td>4.8</td>
</tr>
<tr>
<td>Raytrace</td>
<td>4</td>
<td>2.16</td>
</tr>
<tr>
<td>BlackScholes</td>
<td>3</td>
<td>2.7</td>
</tr>
<tr>
<td>Ferret</td>
<td>5.4</td>
<td>5.2</td>
</tr>
<tr>
<td>Freq.</td>
<td>4.3</td>
<td>3.8</td>
</tr>
<tr>
<td>Fluid</td>
<td>5.5</td>
<td>4.6</td>
</tr>
<tr>
<td>Swaption</td>
<td>3.36</td>
<td>3.35</td>
</tr>
<tr>
<td>X26</td>
<td>1.92</td>
<td>1.9</td>
</tr>
</tbody>
</table>

4.2.3 Optical Losses

PULSE is constructed with nanophotonic components and as such the optical power dissipation of PULSE needs to be taken into account. Table 4.10 shows the optical losses and parameters for select optical devices used to construct PULSE. The maximum optical power loss is given by $8 \times L_S + 6.5 \times L_W + L_C + L_N + 3 \times L_I + L_F + 2 \times L_B + 60 \times L_{WC}$. This gives a maximum optical loss of approximately $-43.5$ $dB$ or $223.8$ $mW$ per wavelength for a total electrical laser power $12$ $W$. This is well within the power budget found in today CMPs. From the above calculated optical power loss, waveguide crossing constitutes a significant portion of the overall optical power. One technique to overcome this is to use multiple optical layers as this will significantly reduce waveguide crossings.
Table 4.9: Percentage of multicast misses for select Splash-2 and PARSEC applications.

<table>
<thead>
<tr>
<th>Application</th>
<th>Cache Misses (16-cores)</th>
<th>Cache Misses (64-cores)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>2.3%</td>
<td>4.5%</td>
</tr>
<tr>
<td>LU</td>
<td>7%</td>
<td>4.2%</td>
</tr>
<tr>
<td>Radix</td>
<td>4.2%</td>
<td>5%</td>
</tr>
<tr>
<td>Raytrace</td>
<td>17%</td>
<td>17%</td>
</tr>
<tr>
<td>BlackScholes</td>
<td>1%</td>
<td>1%</td>
</tr>
<tr>
<td>Ferret</td>
<td>1%</td>
<td>1.8%</td>
</tr>
<tr>
<td>Freq.</td>
<td>19%</td>
<td>3.6%</td>
</tr>
<tr>
<td>Fluid</td>
<td>2.1%</td>
<td>2%</td>
</tr>
<tr>
<td>Swpation</td>
<td>12%</td>
<td>14%</td>
</tr>
<tr>
<td>X26</td>
<td>1%</td>
<td>3.2%</td>
</tr>
</tbody>
</table>

4.2.4 Energy Analysis

(a) Energy Per Bit

Figure 4.12 shows the power per bit for PULSE and the other three networks required to broadcast a bit of data. The power dissipation for electrical network components (routers and electrical links) was calculated for 32 nm technology using ORION 2.0 [114] and were modified to 22nm technology according to ITRS scaling rules. In addition, the optical power dissipation for different versions of PULSE and Shared-Bus. From the figure, PULSE is the most energy efficient network with 9.1 pJ/bit required to broadcast a bit of data followed by Shared-Bus 11.6 pJ/bit to broadcast a bit of data. The worst efficient network is the electrical tree network which requires 13.4 pJ/bit to broadcast a bit of data.
Table 4.10: Address network broadcast network delays (cycles).

<table>
<thead>
<tr>
<th>Device</th>
<th>Loss (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coupler (Lc)</td>
<td>1</td>
</tr>
<tr>
<td>Non-linearity (Lc)</td>
<td>1</td>
</tr>
<tr>
<td>Modulator Insertion (Lm)</td>
<td>1</td>
</tr>
<tr>
<td>Waveguide (per cm)(lw)</td>
<td>1.3</td>
</tr>
<tr>
<td>Filter drop (Lf)</td>
<td>1</td>
</tr>
<tr>
<td>Bending (Lt)</td>
<td>1</td>
</tr>
<tr>
<td>Waveguide Cross (Lwc)</td>
<td>0.05</td>
</tr>
<tr>
<td>Receiver Sensitivity (Lrs)</td>
<td>-20dBm</td>
</tr>
<tr>
<td>Splitter (Ls)</td>
<td>3</td>
</tr>
<tr>
<td>Ring Heating</td>
<td>26μW/ring</td>
</tr>
<tr>
<td>Ring Modulating</td>
<td>500μW/ring</td>
</tr>
</tbody>
</table>

(b) 16-core

Figure 4.13 illustrates the power savings for each network when running Splash-2 and PARSEC applications for 16-cores. Both versions of PULSE are the most energy efficient networks with about 25% power saving over the electrical tree network and about a 75% power saving over the electrical tree networks. The energy savings in multi-PULSE is mostly due to the filtering of the cache requests, as this results in accessing L2 caches that actually share the blocks. Even when accounting for mispredictions, the penalty from the network is minimal. In all results, the network energy for both PULSE and multi-PULSE are almost the same. It is the energy in the cache access that we prevent through predictions.
Figure 4.12: Energy per bit required to broadcast a bit of data.

(c) 64-core

Figure 4.14 illustrates the power savings for each network when running Splash-2 and PARSEC applications for 64-core CMPs. As was the case for 16-core CMPs, both versions of PULSE are the most energy efficient networks with about a 50% power savings over the electrical tree network and about an 85% power saving over the electrical bus network. It should be noted that we can further save the network power if the external laser can be tuned to the network load such that we can throttle the power to maintain the minimum BER.

(d) Laser Power Variation

The optical losses shown in Table 4.10 are mostly conservative estimates that may not reflect the actual optical losses of future nanophotonic devices. In order to compensate for this potential variation in optical losses, we evaluate optical loss variation with laser power requirement in Figure 4.15. The total laser power is a function of both receiver sensitivity and the number of available wavelengths. From the figure, it is easy to see that
the laser power increases dramatically as the receiver sensitivity decreases. For example, a 6 dBm decrease in receiver sensitivity (-17 dBm) would result in a 2× increase in total laser, resulting in an average energy per bit of about 17.4 pJ/bit which is more than any other network. On the other hand, an increase in optical losses such as higher waveguide loss per cm would result in an substantially increase in optical losses. For example a 2.5dB increase in optical losses would result in an energy per bit of 12.4 which is higher than an electrical bus.

To further shows the impact of increase optical losses to total laser power of PULSE. Two potential optical loss parameters, waveguide losses and waveguide crossing efficiency,
Figure 4.14: Potential power savings for select (a) Splash-2 and (b) PARSEC applications (64-core).

can substantially affect the overall laser power dissipation. Waveguide loss is the loss in optical signal as it traverse the a waveguide. Waveguide crossing efficiency is the efficiency of optical signal as it traverses across a waveguide intersection. In order to compensate for this potential variation in optical losses, Figure 4.16 illustrates how varying the waveguide crossing efficiency for a range of different waveguide losses (-0.5dB - -2.5 dB cm) will effect to total laser total of PROPEL. The waveguide crossing efficiency ranges for 100% (0 dB) efficient to 97% (-0.13 dB) efficient. The laser power increases significantly as the waveguide crossing efficiency decreases. For example, a 3% dBm decease in crossing
4.3 Sensitivity Analysis

In this section, a sensitivity analysis on optical devices in terms of bit error rate (BER) and temperature. This gives an insight into the performance characteristics of future CMOS compatible nanophotonic devices.
Figure 4.16: Contour lines showing the variation of total laser power for different waveguide crossing losses.

4.3.1 Bit Error Rate

An optical parameter that effects the overall optical power of the network is bit error rate (BER). BER is the percentage of information that is received in error. The Signal-Noise-Ratio (SNR) for a given BER is calculated by

\[
BER = \frac{1}{2} - \frac{1}{2} \text{erf}(0.354 \sqrt{SNR})
\]  

(4.3)

and the minimum power for a given SNR is

\[
SNR = \frac{P_0 \cdot \eta}{NEP \cdot \sqrt{f}}
\]  

(4.4)
where \( \eta \) is the quantum efficiency of the detector, NEP is the Noise-Equivalent-Power, and \( f \) is the transmission frequency [117].

Using OptiSystem [118] and OptiSPICE [119] optical simulation packages, the BER for 3D-NoC is demonstrate for different optical data bit rates. Figure 4.17 illustrates the layout of optical components in OptiSystem used to simulate 3D-NoC optical properties. We use a 1550 nm laser that outputs 8 wavelengths of light that are 200 GHz apart and couple the light using an optical fiber into a silicon chip. A psudeo-random number generator and a non-return to zero voltage source is used to drive the optical modulators. For optical modulation, we use micro-ring resonators with an extinction ratio of -9 dBm and modulation bandwidth above 10 GHz [50]. To simulate the optical losses found in 3D-NoC, we use a series of optical attenuators that simulate the maximum total optical loss (-16 dB) found in the network. After the light passes through the silicon chip, we filter out a single wavelength to a PIN photodiode (0.56 A/W responsibility) for BER analysis [53]. In addition, we place BER, optical spectrum and optical power analyzers throughout the network to evaluate the optical characteristics of 3D-NoC.

From OptiSystem, the BER was calculated for a select range of frequencies (5 GHz, 10 GHz, and 12 GHz) [118]. Figure 4.18(a) shows the eye opening diagram for a 5 GHz optical signal, which has a BER that is well above \( 10^{15} \). This significant decrease in BER at 5 GHz come about as the photodetector has more time to absorb the optical energy over a 10 GHz signal. Figure 4.18(b) shows the eye diagram for 10 GHz, which corresponds to the proposed optical bandwidth for 3D-NoC. From Figure 4.18(b), there is a clear eye opening and the incoming signal has a BER above \( 10^{12} \). The use of 10 GHz for an optical signal provides adequate bandwidth and is within the high BER constraints found in NoCs. If the frequency is increased to 12 GHz, the eye diagram still shows an opening, where Figure 4.18(b) illustrates the eye diagram. The optical parameter used to construct 3D-NoC as not
only capable of operating at 10 GHz, but capable of operating at high frequencies. Figure 4.18(d) shows the optical power at the photodetector vs times for the 10 GHz signal. The optical signal power clearing swings from about 200 µw to 20 µw giving the average power at the photodetector to be about 100 µw.

4.3.2 Temperature Sensitivity

In this subsection, more detailed information is given about ring heating and how to reduce the power dissipation of ring heating. During large temperature swings, the effective index of a micro-ring resonator changes causing a shift in the resonance frequency for the micro-ring resonator. Unless a technique is used to return the micro-ring resonators into resonance, optical networks such as 3D-NoC will not be capable of performing at peak bandwidth. For a better understand, Figure 4.19 shows an optical frequency spectrum how micro-ring resonators are used to filter different frequencies of light. Figure 4.19(a) shows the first eight wavelengths and there frequency spectrum used in 3D-NoC.
Figure 4.18: Eye diagram for different bandwidths: (a) 5 gbps, (b) 10 gbps and (c) 12 gbps. The optical power vs time (d) at the photodetector for a 10 gbps signal.

From the figure, each wavelength is evenly spacing out with 200 GHz separating each other [28]. As an optical traverses the networks, it will eventually encounter a micro-ring resonator that is in resonance with the incoming light and will filter it out to a photodetector. Figure 14.19(b) shows the optical spectrum at a photodetector after a micro-ring resonator has filtered out a select frequency.

(a) Wavelength Shift

As mentioned above, temperature variation at a micro-ring resonator can affect its performance characteristic. Figure 4.20(a) demonstrated how the resonance wavelength
of a micro-ring resonator shifts with an increase in ring temperature. From the figure, four different resonances wavelength are shown for four different temperature variations: 0°C, 5°C, 10°C, and 20°C. The 0°C plot shows the normal resonances wavelength of the micro ring resonator, which is about 1558 nm. Using a refractive index change of $\Delta n=0.00018$ [120] and a temperature change of 5°C, the resonance wavelength of the resonator will shift to about 1558.47 nm. As for a temperature change of 10°C and 20°C, the new resonance wavelength of the micro-ring resonator will be about 1558.94 nm and 1559.88 nm respectfully. This temperature sensitivity of micro-ring resonators can be detrimental on the overall networks performance. Figure 4.20(b-d) illustrates how a 5°C increase in temperature can result in a optical signal not capable of being read at the photodiode. Figure 4.20(b) shows the 1 gpbs input signal (voltage) to the micro-ring resonator and Figure 4.20(c) shows the optical signal at the photodetector. From these two figures, it is easy to see the signal at the photodiode correctly resembles the electrical signal going to the micro-ring resonator. Figure 4.20(d) shows the same signal at the photodiode if there is a 5°C change in temperature. From the figure, it is hard to make out if the received signal is a binary one or a binary zero.
(b) Ring Heating

Recent works have shown that temperature variations can be mitigated either by adjusting the bias current (upto 15K) \cite{121} or by using a thermal slotted waveguide \cite{122}. In \cite{123}, they propose a ring heating heating technique where resistive heater are used to bias the micro-ring resonator to a known resonance operating temperature. In ring heating, resistive heaters are placed adjacent to micro-ring resonators and are used for thermal control. Resistive heaters operate by having a controlled amount of current pass...
through the a resistive element causing the micro-ring ring resonator to heat up to a known operating temperature. Figure 4.21 illustrates a micro-ring resonator with a ring resistive heater attached. The resistive heater is place close to the active region of the micro-ring resonator allowing the addition heat to transfer quickly to the micro-ring resonator.

A major drawback of ring heating is the increase power dissipation required to operate the resister heating elements. In [28], they proposed the ring heating energy per bit for a single micro-ring resonator to be around 25 fJ/bit. Using 25 fJ/bit and a 10 gbps optical link, the total power dissipation for ring heat is 0.25 mW. For proposed optical networks that use many micro-ring resonators, the total power dissipation to heat micro-ring resonators can be a substantial amount of the total network power. Figure 4.22 (resistive bar) illustrates the total ring heating power for 3D-NoC and other optical networks using resistive heating elements. The following section discuses a technique to reduce the ring heating power for micro-ring resonators.
One technique to reduce the ring heating power dissipation is to use backed reshuffling circuitry along with reduce ring heating, which is proposed in [123]. In this technique, each micro-ring resonator is tuned to the closest incoming light and if the micro-ring resonators tunes to a different wavelength than the original, back end electrical circuitry will reshuffle the bits into correct order. The reduction in ring heating power arises from the fact that less current will be needed to drive the resistive heaters. In [123], they demonstrated a 10× to 5× decrease in power dissipation over full ring heating. The large range arises from different possible heating currents needed to shift the micro-ring resonator to the closest resonance frequency. Figure 4.22 (bit shuffle bars) illustrates the potential decrease in ring heating power dissipation using the bit reshuffle technique. Two different bar graphs are shown. The higher bars represent the least potential power savings using the bit reshuffle technique for different networks. The lower bars represent the maximum potential power
savings using the bit reshuffle technique for different networks. For 3D-NoC, the ring heating power dissipation decreased to as low as 2.75 watts for 3D-NoC-L1/3D-NoC-LA and 3.66 watts for 3D-NoC-L2/3D-NoC-L3.

4.3.3 Fabrication Sensitivity

The fabrication tolerance of micro-ring resonators can greatly affect the overall performance of nanophotonic NoCs. To illustrate the impact of different fabrication tolerances, Figure 4.23 shows the increase in laser power for 3D-NoC and PULSE as the fabrication tolerance of micro-ring resonators decreases. In Figure 4.23(a), the increase in laser power for 3D-NoC is shown. The number of micro-ring resonators an optical signal will traverse through in 3D-NoC-L1 is four and in 3D-NoC-L3 is six. For a 0.5 dB increase in optical losses while traversing a micro-ring resonator, the laser power in 3D-NoC-L3 increases from 6.1 Watts to about 12 watts whereas in 3D-NoC-L1, laser power increases to about 8.5 watts. It should be mentioned that 3D-NoC-LA and 3D-NoC-L2 are not illustrated in Figure 4.23 as they would have similar power dissipation identical to 3D-NoC-L1 and 3D-NoC-L3.

For PULSE and Multi-PULSE, Figure 4.23(b) illustrates the increase in laser power as fabrication tolerance is varied for each micro-ring resonator. The number of micro-ring resonators an optical signal traverses in PULSE is four, and the number of micro-ring resonators an optical signal traverses in Multi-PULSE is eight. Laser power in Multi-PULSE increases faster than PULSE, which is due to the higher number of micro-ring resonators a signal travels through as the data is multicast. For a 0.5 dB increase in optical losses while traversing a micro-ring resonator, the laser power in multi-PULSE increases from 12 watts to about 30 watts whereas in PULSE, laser power increases to about 19 Watts.
Figure 4.23: Increase in laser power versus fabrication tolerance of micro-ring resonators: (a) 3D-NoCs and (b) PULSE.
5 CONCLUSIONS AND FUTURE DIRECTION

5.1 Results and their Implication

With the multicore era upon us and researchers predicting processors with 100s and 1000s of cores, high performance and power centric network-on-chips (NoC) will be required to meet the increasing inter-core communication bandwidth demands. Moreover, traditional metallic interconnects are not capable of providing the required network performance due to metallic interconnects being plagued with high power dissipation and limited bandwidth. One potential technology to overcome the limitation of metallic interconnects is nanophotonics. Nanophotonics have benefits such as bit rates independent of distance and higher bandwidth due to multiplexing of wavelengths on the same waveguide/fiber (wavelength division multiplexing (WDM)), which allow for the implementation of low power and high performance interconnects. Using nanophotonic interconnects, this dissertation overcomes the shortcomings of metallic interconnects by proposing two high performance and low power nanophotonic interconnects called 3D-NoC and PULSE.

3D-NoC is a nanophotonic NoC proposed for high performance networks. 3D-NoC is designed using 16 decomposed optical crossbars placed across four different optical layers. The use of 3D decomposed optical crossbars allow for the reduction in optical power as the average waveguide length is decreased when compared to other leading optical crossbar designs. To further improve 3D-NoC, this dissertation proposed a reconfiguration technique for 3D-NoC that allows different groups to use bandwidth from under-utilized interconnects. When 3D-NoC is compared to other leading nanophotonic networks, 3D-NoC shows a 40% improvement in terms of performance, a 23% reduction in power dissipation, and a 10% reduction in optical area overhead.
PULSE is a high performance and low power nanophotonic broadcast NoC for snoopy protocols. PULSE is constructed similar to a tree network by combining and splitting signals, thereby ensuring the same intensity signal at all cores. Moreover, PULSE uses an optical power guiding system that routes optical power to only those cores that will transmit an address request allowing for a reduction in optical power dissipation of about 75%. With most cache blocks only located in a small group of caches, this dissertation proposed an adaptive network switch technique, that rapidly morphs the broadcast network into a multicast network. This allows cache requests to be only sent to caches that are holding the cache block and results in a reduction in cache power as needless cache lookups will be eliminated. When PULSE is compared to other leading broadcast networks, PULSE shows a 55% increase in terms of performance and a 80% reduction in power dissipation.

5.2 Shortcomings of this Dissertation

This dissertation proposes two nanophotonic NoCs that overcome the limited bandwidth and high power found in metallic interconnects. However, this dissertation does not explore the full potential uses of nanophotonics for multicore processors. For example, the use of nanophotonics for memory access was not discussed. If a cache request requires access to main memory, the request will be sent to the memory modules using traditional metallic interconnects that have high latency. By using nanophotonic interconnects from the on-chip memory controllers to the memory modules, cache requests will be completed faster allowing for further speed-up of applications. Furthermore, nanophotonics can be incorporated within memory modules for quicker access to data banks within dynamic random access memory (DRAM).

Another shortcoming of this dissertation is the lack of nanophotonic technology being applied to improve the performance of cache coherent protocols. In this dissertation, each network uses well established directory (3D-NoC) and snoopy protocols (PULSE)
to maintain cache coherence. These cache protocols were developed and optimized using electrical interconnects. However, nanophotonics provides unique opportunities that are not found in electronics such as WDM. It is possible to apply these unique properties to new cache coherent protocols to improve the sharing of data among the different cores.

Engineers and researchers predict that multicore processors will exist with 100s and 1000s of cores. This dissertation proposes two nanophotonic NoCs: 3D-NoC and PULSE. However, PULSE was proposed for 16-core and 64-core multi-core processors and 3D-NoC was proposed for 256-core. As transistors continue to scale into the sub-nanometer regime, transistor counts will be high enough for 1000s of cores to exist on a single chip. Future work should undertake a feasibility study into the possibility of scaling 3D-NoC to 1024-core architecture.

In 3D-NoC, reconfiguration technique can be further improved to include partial bandwidth reconfiguration. When reconfiguration occurs, 3D-NoC should be able to reconfigure an interconnects bandwidth based on its bandwidth demand. For example, if an interconnect is partially being utilized then part of an interconnects bandwidth can be utilized in reconfiguration. Currently, the bandwidth of an interconnect is available for reconfiguration only if the interconnect has zero utilization.
6 Publications

6.1 Published Work


6.2 Papers Under Review


REFERENCES


multilayer deposited silicon materials for high-performance chip multiprocessors,”


