The Design and Use of High-Speed Transmission Line Links for Global On-Chip Communication

by

Aaron Carpenter

Submitted in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Supervised by

Professor Michael Huang

Department of Electrical and Computer Engineering
Arts, Sciences and Engineering
Edmund A. Hajim School of Engineering and Applied Sciences

University of Rochester
Rochester, New York

2012
To my wife, mother, father, and sister
for everything.
Curriculum Vitae

Aaron Carpenter was born in Schenectady, New York on July 7th, 1983. He attended the University of Rochester, where he graduated with a Bachelor of Science degree in Electrical and Computer Engineering in 2005, followed by a Master’s degree in 2006, also in Electrical and Computer Engineering (focusing on computer architecture and VLSI design). He then joined the PhD program at the University of Rochester in the Fall of 2006 where he continued his graduate studies and research in the area of computer architecture, under the supervision of Professor Michael Huang. During his time in the graduate program, he held a position both as a teaching assistant (from Fall 2005 through Spring 2007) and as a research assistant (from the Fall of 2007 to Fall of 2011). He also spent 8 months (January through August) in 2008 completing internship with Intel in the Graphics Architecture Group. During his PhD, Aaron has contributed to 5 original publications in peer-reviewed journals and conferences. Since January 2012, Aaron has been working as an Assistant Professor at Binghamton University (SUNY) in the Electrical and Computer Engineering Department.

List of Publications and Articles Accepted for Publication:


Acknowledgments

I would like to thank the many people who have helped me reach this point and make my thesis possible. First, I joined the PhD and got to this point in my graduate work, in no small part, because of my advisor, Professor Michael Huang. Throughout my time as a graduate student, he has offered advice, insight, and expertise, and has been instrumental to any success I have had through my graduate career.

I would also like to thank my thesis committee members, Professor Hui Wu, Professor Eby Friedman, and Professor Sandhya Dwarkadas, as well as my many teachers at the University of Rochester, both graduate and undergraduate.

I also must thank the many students with whom I have worked, learned, and interacted over the years. In particular, I have to mention Wasiur Rashid, Alok Garg, Regana Noor, Jing Xue, Raj Parihar, Övünç Kocabas, Xin Li, Lin Zhang, Jianyun Hu, Berkehan Ciftcioglu, and Ioannis Savidis.

I wish to thank my parents, Ken and Ellen, and my sister, Sarah, for all the love and support they have provided me, not only during my time as a graduate student, but for my entire life. Without them, I would not be where I am today, and words cannot accurately express my gratitude to them.

And of course, I want to thank my wife, Angela. She has been my best friend for over a decade, supporting me in every way possible. She has stood by me through the good and bad times and it is impossible to overstate how much her unending support and love have meant to me.
Abstract

As transistors approach the limits of traditional scaling, computer architects can no longer rely on the increase in density and core frequency to improve the overall system speed. Additionally, attempts to improve performance often result in disproportionately increased power and energy consumption. However, the increased performance and maximum frequency of the transistor allows us to build high-speed circuits specifically for on-chip communication. By incorporating the improving and emerging high-speed circuit technologies into the microprocessor design, it is possible to decrease the power and energy consumption, while simultaneously increasing system performance.

This thesis focuses on exploiting and analyzing the architectural opportunities provided by incorporating high-speed communication circuits, specifically on-chip transmission lines and simple high-speed transceivers. In broad terms, the transmission lines are used for a globally shared-medium on-chip interconnect, providing a low-latency, low-energy, packet-relay-free point-to-point link. Even a simple interconnect design can provide more than sufficient performance for small- to medium-scale chip multiprocessors. Additionally, with simple optimizations exploiting benefits of a TLL shared-medium bus, it is possible to mitigate scalability limitations, and provide performance and energy benefits for larger-scale systems. For example, an atomic, low-latency bus provides opportunities to change the cache coherence substrate and optimize Boolean data communication. This thesis will present and evaluate a number of these optimizations, and provide a final recommended design, showing performance and energy benefits with larger scale systems.
Table of Contents

Curriculum Vitae iii

Acknowledgments v

Abstract vi

List of Tables xii

List of Figures xiv

Foreword 1

1 Introduction 2

2 Interconnection Networks 7

2.1 Common Terminology & Basic Interconnect Components 8

2.2 Packet-Switched Networks-On-Chip 11

2.2.1 NoC Topologies 15

2.3 Optical Networks 16

2.4 Hierarchical & Bus Interconnects 18

2.5 Transmission Lines in Existing Interconnects 22
### 3 Simulation Tools

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Sonnet</td>
<td>25</td>
</tr>
<tr>
<td>3.2 ADS</td>
<td>27</td>
</tr>
</tbody>
</table>

### 4 Injection-Locked Clocking

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 ILC Overview</td>
<td>33</td>
</tr>
<tr>
<td>4.2 Clock Generation and Distribution</td>
<td>34</td>
</tr>
<tr>
<td>4.3 Challenges in Conventional Clocking</td>
<td>36</td>
</tr>
<tr>
<td>4.4 On-Chip Clocking Networks</td>
<td>37</td>
</tr>
<tr>
<td>4.5 Injection-Locked Clocking</td>
<td>39</td>
</tr>
<tr>
<td>4.5.1 Injection-Locked Oscillators</td>
<td>39</td>
</tr>
<tr>
<td>4.5.2 Clocking using ILOs</td>
<td>41</td>
</tr>
<tr>
<td>4.5.3 Power Savings</td>
<td>42</td>
</tr>
<tr>
<td>4.5.4 Skew Reduction and Deskew Capability</td>
<td>44</td>
</tr>
<tr>
<td>4.5.5 Jitter Reduction and Suppression</td>
<td>45</td>
</tr>
<tr>
<td>4.5.6 Potential Applications</td>
<td>45</td>
</tr>
<tr>
<td>4.6 Case Study and Experimental Setup</td>
<td>46</td>
</tr>
<tr>
<td>4.6.1 Case Study</td>
<td>46</td>
</tr>
<tr>
<td>4.6.2 Circuit Simulation Setup</td>
<td>48</td>
</tr>
<tr>
<td>4.6.3 Architectural Simulation Setup</td>
<td>50</td>
</tr>
<tr>
<td>4.7 Experimental Analysis</td>
<td>51</td>
</tr>
<tr>
<td>4.7.1 Jitter and Skew</td>
<td>51</td>
</tr>
<tr>
<td>4.7.2 Chip-Wide Power Impact of ILC</td>
<td>53</td>
</tr>
<tr>
<td>4.8 Summary</td>
<td>55</td>
</tr>
</tbody>
</table>

### 5 Transmission Line Basics

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Transmission Line Properties</td>
<td>60</td>
</tr>
</tbody>
</table>
# Transmission Line Design Study

6. Transmission Line Design Study

6.1 Transmission Line Overview .................... 68

6.2 Physical Design .................................... 69

6.2.1 Transmission Line Topology .................. 71

6.2.2 Attenuation & Crosstalk ....................... 71

6.2.3 Aggregate Throughput .......................... 73

6.3 Circuit Design ........................................ 74

6.3.1 Transmitter & Receiver ......................... 74

6.3.2 SerDes & PDR .................................... 78

6.3.3 Isolation Switch ................................. 81

6.3.4 Signal draining ................................... 81

6.4 System-Level Impact of Transmission Line Links .................. 82

6.4.1 Architecture Design ............................. 82

6.4.2 Architectural Comparison of Transceivers ............ 82

# Transmission Line Bus Architecture

7. Transmission Line Bus Architecture .............. 85

7.1 On-Chip Network Overview ...................... 85

7.2 Globally Shared-Medium On-Chip Interconnect .... 86

7.2.1 Transmission Line Link Design Space ........... 87

7.2.2 Traffic Demand .................................. 90

7.2.3 Bus Architecture ................................. 92

7.3 Analysis of TLLB in Small to Medium CMPs .......... 95

7.3.1 Experimental Setup .............................. 95

7.3.2 Traffic and Performance Analysis ............... 97

7.3.3 Power Savings ................................. 101
7.3.4 The Impact of Bundling .................................................. 102
7.3.5 Performance in Scaled-up Systems ................................. 103

7.4 Analysis of Baseline Bus in Larger-Scale CMPs .................. 105
7.4.1 Application Characteristics ........................................... 106
7.4.2 Techniques for Increasing Effective Bus Throughput ............. 107

7.5 Increasing Raw Link Throughput ....................................... 108

7.6 Increasing the Utilization Efficiency .................................. 113
7.6.1 Partitioning .............................................................. 113
7.6.2 Wave-based arbitration ................................................. 115
7.6.3 Segmentation ............................................................ 118
7.6.4 Local links ............................................................... 122

7.7 Optimizations on the Use of Buses ................................. 125
7.7.1 Invalidation acknowledgement omission ......................... 126
7.7.2 Limited multicasting .................................................... 126
7.7.3 Fuzzy invalidation ...................................................... 127
7.7.4 Boolean bus ............................................................. 130
7.7.5 Coherence-aware arbiter .............................................. 131

7.8 TLLB 64-core Evaluation ................................................. 133
7.8.1 Performance Benefits .................................................. 133
7.8.2 Costs ...................................................................... 135
7.8.3 Comparison of Cost Effectiveness ................................... 136
7.8.4 Example Design Points ................................................. 137
7.8.5 Scaling Up .............................................................. 141

7.9 Summary ..................................................................... 142

8 Future Directions .............................................................. 144
8.1 Quality of Service .................................................. 144
8.2 TLL Allocation ...................................................... 146
  8.2.1 Serialization Exploration ..................................... 146
  8.2.2 Circuit-Switched Networks ..................................... 148
8.3 Hybrid TLLB and Packet-Switching Designs ......................... 148

9 Conclusions ............................................................. 151

Bibliography ............................................................ 154
## List of Tables

2.1 Characteristics of various types of interconnect normalized to a single bit on a bus. \( n \) is the number nodes in a row and column. ................................. 13

4.1 ILC system configuration. ....................................................... 51

5.1 The criteria necessary to determine model type. \( \lambda \) is the wavelength, \( t_r \) is the rise time of the signal, and \( t_t \) is the transmission latency of the signal [41; 104; 106]. ................................................................. 58

6.1 Transmission line characteristics. .............................................. 76

6.2 Transceiver characteristics. Note that in the digital configuration, the transmitter latency is incurred every hop. The SERDES results are based on the fastest data rate (from analog transmission circuit). ................................................................. 76

7.1 16-core system configuration. .................................................... 96

7.2 Benchmarks used. ................................................................. 97
7.3 Cache controller transitions for L1 and L2 cache. The rows are the current state, the columns are the events/requests, and each entry contains an `<action/next state>` pair. Impossible cases are marked “error” and “z” means the event cannot currently be processed, and in some cases, the incoming request will be reinterpreted as a different one due to race. M, E, S, and I are stable states of L1 cache controller and DM, DS, DV (Valid with no sharers), and DI are stable states of L2 directory controller. Transient states are denoted by the pair of previous and next stable state. Transient states waiting for a data reply are superscripted with D and those waiting for just an acknowledgment are superscripted with A. All request events (Req) are followed by request type *i.e.*, (Sh: read in shared mode, Ex: read in exclusive mode, Upg: upgrade request, Dwg: downgrade request, and Mem: memory access request). [141].
List of Figures

2.1 A broad interconnection network may connect CPUs, memory, I/O, or more. 7

2.2 Various types of interconnects (assuming a 16-node system). Each circle is a network node/router and the lines are the links. In the case of (e), the layout is logical, not necessarily the physical layout of a butterfly topology. 12

2.3 A generic virtual-channel packet-switched network router. 14

2.4 An optical interconnect system, often with off-chip lasers, requiring electrical-to-optical and optical-to-electrical conversion, as well as a propagation medium. 17

2.5 A free-space optical interconnect. A GaAs substrate containing photonic devices is laid on top of a CMOS substrate. Optical signals are sent from the VCSEL through a micro-lens. The light is then reflected through free-space off of the mirrors until it reaches its destination, passes through another lens, and is converted back to an electrical signal through a photo-diode. [142] 17

2.6 Microrings can be designed to couple to a specific wavelength, creating a WDM medium which provides high throughput. 18

2.7 Early processors contained digital buses to connect multiple components, such as CPUs, memory, and I/O ports. 19

2.8 A top-level view of the hierarchical bus/ring structure proposed by Udipi et al. [128]. 20
2.9 A state diagram for a MSI protocol, showing the transitions between modified, shared, and invalid states. ....................................................... 21

3.1 A Sonnet screenshot displaying the top level view of a group of coplanar strips. The top layer and bottom layers are gold, and the middle layers are $SiO_2$ with a dielectric constant of 3.0. ....................................................... 26

3.2 A Sonnet screenshot displaying the top level view of a group of coplanar strips. The top layer and bottom layers are gold, and the middle layers are $SiO_2$ with a dielectric constant of 3.0. ....................................................... 27

3.3 Sonnet’s S-Parameter analysis (signal and crosstalk). The axes have been enhanced for easier reading. .......................................................... 28

3.4 The ADS circuit “network” environment, in this case, showing the construction of a longer transmission line out of shorter Sonnet-based transmission line S-parameters. ....................................................... 29

3.5 The setup of an ADS S-parameter simulation of a transmission line, based on Sonnet S-parameters. .......................................................... 30

3.6 The ADS frequency response of 2 transmission lines. The attenuation (top) and crosstalk (bottom) of the lines are shown, along with trace markers. ............ 31

3.7 A differential amplifier used for transmitters and receivers of the analog transceiver design. ............................................................. 32

4.1 Conventional global clock distribution, showing an H-tree topology with interconnects and clock buffers [43]. ....................................................... 37

4.2 (a) Beat and injection locking phenomenon when an oscillator is driven by an input single-frequency signal. (b) locking range. ........................................ 40

4.3 (a) A generic model of an injection-locked oscillator (ILO). (b) a divide-by-2 ILO based on a common differential LC oscillator. ........................................ 40
4.4 Phase transfer functions for divide-by-2 ILOs in Figure 4.3-(b). \( \eta \equiv \frac{I_{inj}}{I_{osc}} \) is the injection ratio, \( \omega_0 \) is the free-running oscillation frequency, \( \Delta \omega \equiv \omega - \omega_0 \) is the frequency shift, and \( Q \) is the LC tank quality factor. 41

4.5 Injection-locked clocking scheme. 42

4.6 Voltage gain of an inverter and an injection-locked oscillator at different input signal voltage levels. 43

4.7 The (a) conventional clock tree (shown without clock grids) and (b) proposed ILC tree and associated voltage swings. 44

4.8 Jitter in the ILC design (b) is reduced compared to (a) as jitter-injecting components can be removed. 45

4.9 Illustration of the three different configurations of global clock distribution. Each configuration is designated according to its clocking network: XGM, IGM, and IM\( ^{\prime} \). 46

4.10 Circuit-level simulation setup. (a) Clock distribution network modeling. (b) Jitter simulation. 49

4.11 Schematic of (a) the test chip and (b) a divide-by-2 ILO used. 52

4.12 Deskew capability of ILC in the test chip, where \( V_{diff} = V_{t1} - V_{t2} \). 53

4.13 Breakdown of processor power consumption with different clock distribution methods. 54

5.1 Transmission lines as seen in day-to-day life, as power lines. 57

5.2 The (a) RC and (b) RLC equivalent models of a wire. \( R \) and \( L \) are the series resistance and inductance per unit length and \( G \) and \( C \) are the shunt conductance and capacitance per unit length. 58

5.3 (a) A demonstration of the charging of a traditional wire. (b) A pulse traversing a transmission line. 60

5.4 The maximum bandwidth and achievable throughput of (a) 5mm and (b) 75mm lines. 61
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5</td>
<td>An equivalent circuit model of a transmission line, including voltage and current definitions.</td>
<td>62</td>
</tr>
<tr>
<td>5.6</td>
<td>A transmission line terminated with (a) an open circuit, (b) a short circuit, and (c) a load impedance $Z_L$.</td>
<td>63</td>
</tr>
<tr>
<td>5.7</td>
<td>Common transmission line topologies, along with a legend of materials. Ground lines are also marked.</td>
<td>66</td>
</tr>
<tr>
<td>6.1</td>
<td>Illustration of transmission line link system design space.</td>
<td>69</td>
</tr>
<tr>
<td>6.2</td>
<td>Top-level view of 16-core interconnect layout. The solid line is a physical bidirectional ring, and the dotted line is a bi-directional terminated bus.</td>
<td>70</td>
</tr>
<tr>
<td>6.3</td>
<td>Cross-section of (a) microstrip lines and (b) coplanar strips. The dotted lines in (a) represent inter-digitated MSL.</td>
<td>71</td>
</tr>
<tr>
<td>6.4</td>
<td>Per-line bit rate as a function of wire pitch. I-MSL, or inter-digitated microstrip lines, is similar to CPW where the ground line is the same width as the signal line. The I-MSL is essentially a MSL design where and identical line is placed halfway in between signal lines and grounded.</td>
<td>72</td>
</tr>
<tr>
<td>6.5</td>
<td>Crosstalk as a function of wire pitch. I-MSL, or inter-digitated microstrip lines, is similar to CPW where the ground line is the same width as the signal line.</td>
<td>72</td>
</tr>
<tr>
<td>6.6</td>
<td>Aggregate bit-rate as a function of the number of lines in a 2.5mm space. I-MSL, or inter-digitated microstrip lines, is similar to CPW where the ground line is the same width as the signal line.</td>
<td>73</td>
</tr>
<tr>
<td>6.7</td>
<td>Test chip of transmission line links.</td>
<td>74</td>
</tr>
<tr>
<td>6.8</td>
<td>General schematic for the transmission line link interconnect.</td>
<td>75</td>
</tr>
<tr>
<td>6.9</td>
<td>A digital transmitter and receiver in a fully digital system.</td>
<td>77</td>
</tr>
<tr>
<td>6.10</td>
<td>A digital transmitter and analog receiver for a mixed-signal system.</td>
<td>77</td>
</tr>
</tbody>
</table>
6.11 A fully analog system, using differential amplifiers for the transmitter and either a chain of inversely scaled differential amplifiers or a differential amplifier and current-mode latched sampler for the receiver. “S2D” is serial-to-differential circuit, “D2S” is differential-to-serial circuit, which accounts for the transitions between serial signals from the core to differential signals in the CPS transmission lines.

6.12 (a) A serializer diagram for 8-to-1 bit serialization, with (b) showing detail of the 2-to-1 multiplexer.

6.13 (a) A serializer diagram for 1-to-8 bit serialization, with (b) showing detail of the 1-to-2 multiplexer.

6.14 (a) A block diagram for a clock and data recovery system. (b) A block diagram of a PLL, often used for a clock recovery circuit.

6.15 (a) A block diagram for the phase and data recovery system. This system requires a training period to set the phase select decision circuit. (b) An example of a delay line using a multiplexer and a line of inverters.

6.16 The per-application breakdown of different transceiver designs. The digital components are used a baseline.

6.17 A comparison of each transceiver type’s area, energy-per-bit, power, system energy, system delay. The digital transceiver is used as a baseline.

7.1 Link diagram with differential transmitter and receiver. D2S is a differential-to-single-ended circuit, converting the differential signals to a full swing digital signal.

7.2 Measurement setup and waveforms of transmission line circuit simulations. A, B, and C (as labeled in the design diagram shown in Figure 7.1) correspond to transmitter output, receiver amplifier input, and final latch output, respectively.

7.3 Node structures: (a) single-core nodes; and (b) clustering two cores into a node on the interconnect.

7.4 Overview of the bus-based communication subsystem.
7.5 Illustration of bus operation timing. Three nodes \((i, n,\text{ and } j)\) are shown. Tx and Rx indicate transmitting and receiving ports respectively. Note that pulses are sent using a much higher clock rate and thus each logical communication cycle can transmit multiple bits per TLL.

7.6 (a) Percentage of L2 accesses that are remote. The 3 configurations are 1, 2, and 4 cores per node. (b) Speedup due to profiling and clustering. The bar on the left is for 1 core per node, the right bar is for 2 cores per node. The baseline in this case is a 16-core mesh with round-robin data distribution.

7.7 Speedup of TLL bus system over the respective (16- or 8-node) mesh-based system. The left bar in each group represent 16 node configuration and the right bar, 8 node. Note the y-axis does not start at 0.

7.8 The 16-node and 8-node system performance compared to a latency-only, contention-free ideal.

7.9 Average energy savings for the 16-core TLL bus compared to NoC.

7.10 (a) Speedup of the 16-node system with bundling of 2 and 3, over the system without bundling. Note the y-axis does not start at 0. (b) Overall packet latency relative to a non-bundled system. The left and right bar correspond to a bundling of 2 and 3 respectively.

7.11 Relative performance of a 64-core system. For the TLL bus configurations, a bundle of 3 is used.

7.12 Performance of TLL bus relative to idealized contention-free, low-latency interconnect.

7.13 Performance of baseline TLL bus and NoC (both 2-cycle and ideal 1-cycle routers) normalized to wire-delay-only ideal interconnect. The 3 groups represent, from left to right, the benchmarks with increasing room for performance improvement for the TLL bus.

7.14 The maximum bandwidth and achievable throughput of 75\text{mm} lines.

7.15 Block diagram of links using 4-PAM and FDM.
7.16 Transmission line frequency response. As the frequency approaches the boundaries of the spectrum, there are noticeable noises due to artifacts of the simulation setup. ................................................. 110

7.17 The performance of doubling available throughput using frequency division multiplexing or 4-level pulse amplitude modulation. .................................................. 111

7.18 The potential performance window for doubling throughput with encoding techniques. The value plotted is a result of the formula: $100 \times \frac{Perf_{Optimization} - Perf_{Base}}{Perf_{Optimization} - Perf_{Ideal}}$. The G1, G2, G3, and Total values are the arithmetic averages. .................................................. 112

7.19 Detailed performances of different bus partitioning options over the baseline bus design. ................................................. 114

7.20 The range and average speedup of different bus partitioning options over the baseline bus design. ................................................. 114

7.21 Wave-based arbitration would allow multiple signals to propagate without physical segmentation. ................................................. 116

7.22 The per-application performance of wave-based arbitration. ................................................. 117

7.23 The potential performance window for wave-base arbitration. The value plotted is a result of the formula: $100 \times \frac{Perf_{Optimization} - Perf_{Base}}{Perf_{Optimization} - Perf_{Ideal}}$. The G1, G2, G3, and Total values are the arithmetic averages. ................................................. 117

7.24 The chip layout with 4 segments. Each segment is connected by either a pass-gate switch or an amplifier pair. ................................................. 119

7.25 The performance impact of using 2 and 4 segments, compared to using a 1 un-segmented bus. Note that the y-axis does not start at 0. ................................................. 119

7.26 The potential performance window for segmentation. The value plotted is a result of the formula: $100 \times \frac{Perf_{Optimization} - Perf_{Base}}{Perf_{Optimization} - Perf_{Ideal}}$. The G1, G2, G3, and Total values are the arithmetic averages. ................................................. 120

7.27 The per-application performance of a 2- and 4-segment bus, connected using simplified routers, over a baseline of a single unified bus. ................................................. 121
7.28 (a) A conventional router with 4 VCs, also pictured in Figure 2.3 [105]. (b) Our much simplified digital link controller. 123

7.29 The performance impact of using 1-dimensional and 2-dimensional nearest neighbor routers, compared to TLL bus baseline. Note that the percentages above the bar are how often the traffic is nearest neighbor for the 1-dimensional case. 124

7.30 The potential performance window for local links. The value plotted is a result of the formula: $100 \times \frac{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Base}}}{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Ideal}}}$. The G1, G2, G3, and Total values are the arithmetic averages. 125

7.31 Relative performance over baseline bus when using invalidation acknowledgement omission (left), and multicasting (right). Note the y-axis does not start at 0. 127

7.32 The potential performance window for invalidation acknowledgement omission and limited multicasting. The value plotted is a result of the formula: $100 \times \frac{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Base}}}{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Ideal}}}$. The G1, G2, G3, and Total values are the arithmetic averages. 128

7.33 The per-application performance of fuzzy invalidation. 129

7.34 The potential performance window for fuzzy invalidation. The value plotted is a result of the formula: $100 \times \frac{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Base}}}{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Ideal}}}$. The G1, G2, G3, and Total values are the arithmetic averages. 129

7.35 The per-application of a cache utilization study (over baseline), using fuzzy invalidation to mark “stale” lines, but still sending messages to remove the impact of traffic and timing on the system. 130

7.36 The performance breakdown of the Boolean bus optimization for applications with significant lock activity (more than 1 lock per 10k instructions). 131

7.37 The per-application performance of using a Boolean bus. 131

7.38 The per-application performance of using a coherence aware arbiter. 133
7.39 The potential performance window for the coherence-aware arbiter. The value plotted is a result of the formula: $100 \times \frac{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Base}}}{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Ideal}}}$. The G1, G2, G3, and Total values are the arithmetic averages. .......................... 133

7.40 The performance impact of the techniques discussed. The bars show the geometric mean of relative performance over a baseline bus on all application, while the I-beams show the range from all applications. Note that the y-axis does not start at 0. .................................................. 134

7.41 The relative network energy, chip energy, and energy-delay product of applying each technique discussed in a 64-core, 16-node system. ......................... 137

7.42 The effect of a few configurations. FIMP is short for the combination of fuzzy invalidation, invalidation acknowledgement omission, multicasting, and partitioning. (a) The relative performance with geometric mean and range from all applications. An ideal interconnect is shown as a reference. (b) The relative network energy, total chip energy, and energy-delay product. .......................... 138

7.43 Packet latency vs. offered traffic of various interconnect configurations. Note that the model of flattened butterfly assumes no increase in router delay. .... 139

7.44 The average injection rate of particular application mapped to the predicted uniform random traffic generator. Red represents the highest injection rate, around 3 flits/cycle. Red represents the range of Group 3’s offered load, yellow corresponds to Group 2, and green to Group 1. .......................... 141

7.45 The performance of various configurations under escaladed traffic environment, all normalized to baseline bus. .................................................. 142

8.1 A logical snapshot of a prioritized output queue, with a high priority message being inserted from the node. .................................................. 146

8.2 Results for a preliminary QoS algorithm. The baseline is a 16-node, 64-core system, as in Chapter 7. .................................................. 146

8.3 A TLL allocation scheme with a single line connecting 1 transmitter to multiple receivers. .................................................. 147
8.4 A sampling of circuit switched topologies. The “S” represents a circuit-switch to connect segments/branches to link remote nodes. While this could be a router, the focus here is to move away from packet-switching, so the switch is assumed to be a simple circuit switch. The size of the segments and switch placement are meant to be an example, not a fixed or proven quantity. The top 2 rows are for 16 nodes, and the last row is for 64 nodes. The proposed buses are shown in the upper left and upper middle figures.
Foreword

While I am the author of this thesis dissertation, the work presented would not have been possible without the benefit of collaboration with various people in the Electrical and Computer Engineering department. First and foremost, I would like to thank Professor Michael Huang, my advisor, for his expertise and advice throughout my graduate studies and research.

For the work presented in Chapter 4, exploring the use of injection locked oscillators to produce a reliable clock, I must acknowledge Professor Hui Wu, Dr. Lin Zhang, and Berkehan Ciftcioglu, who provided the device design and analysis for the ILOs, as well as valuable insight for the clocking scheme. Michael Huang and Alok Garg provided collaborative efforts in the architectural design and analysis, and along with Prof. Wu and Dr. Zhang, we produced 2 papers, one in IBM’s PACC in 06, and another in TVLSI 2008.

For Chapters 6 and 7, portions of which resulted in publications in ISLPED 2011, ISCA 2011 and JETCAS 2012, it is necessary to acknowledge not only the work of Professor Huang and Professor Wu who provided guidance and insight, but also Jianyun Hu, who helped to guide the design of the high-speed circuits and transmission lines. Jie Xu and Övünç Kocabas also provided support for some of the circuit design for Chapter 6.
Introduction

As the number of transistors on a chip continues to increase, on-chip communication becomes a more important facet of architectural design. Traditional electrical wires, typically driven by digital components using simplistic digital signals have issues to address in the scaling chip multiprocessor market, specifically latency and energy. Global wire latency remains relatively constant, translating to a larger relative latency for even moderately-sized systems. In order to ensure signal quality, digital repeaters and packet-switching routers must be added to facilitate the transmission of long distance communications, contributing further to the latency and energy issues.

Current research focuses on a few categories of solutions, each with unique benefits and limitations. The current convention is the use of packet-switching networks topologies to provide the interconnect backbone for chip-multiprocessors. A packet-switched network-on-chip (NoC) provides in-field scalability, the ability to use commercial-off-the-shelf components, and high aggregate throughput. However, a NoC also requires higher power routers and potentially long latencies for long distance communication. Another state-of-the-art solution for interconnects uses on-chip optics. Research is currently being proposed to use either waveguides or free-space optics to provide a high-throughput, low-energy, low-latency medium for on-chip communication. On the other hand, optics also has issues that prohibit its immediate use as an interconnect backbone. First, current optical components are not easy to integrate into a standard silicon CMOS process, making it more difficult to fabricate with the current technolo-
gies without sacrificing electro-optical conversion efficiency. Additionally, while some on-chip lasers exist, most often, off-chip lasers are used to provide the optical power, shifting the onus of energy efficient operation off-chip, but not removing it from the system.

Instead of relying on energy-inefficient packet-switching, or not-yet-fully-developed technologies, we can take advantage of current CMOS technologies to improve the on-chip network. The improving transistor performance no longer directly translates to processor performance as it once did. Increasing frequency results in overwhelming power consumption, and thus, the frequency of an entire microprocessor is constrained. However, as the transistors scale, the increasing performance can be exploited for some circuits, such as the communication backbone and the clock generation and distribution network. The goal of this work is to leverage the improving technology for benefits in microarchitectural design. The increasing quality of the transistors at high frequencies makes it possible to incorporate high-speed communication circuits. Rather than push the limits of traditional interconnection techniques, we can use the technological advances in high-speed circuits to alleviate these latency and power issues. In some cases, it is close to a drop-in replacement, with little change to the design of the architecture. In other cases, the architecture and high-speed analog components must be co-designed, to ensure optimal performance from both regimes. Overall, we hope to provide an alternative on-chip interconnect backbone which has sufficiently high throughput, low energy consumption and low propagation delay, exploiting the CMP environment and the improving transistor performance.

In this thesis, I will focus on the development and utilization of transmission lines and the associated transceiver circuits, in order to provide a high-speed communication network for architectural optimization. The first step to creating a high-speed communication network is providing a fast, reliable, energy-efficient clock distribution network. I will explore injection-locked clocking, which is a high-quality, high-speed clocking scheme, providing a reliable globally synchronous clock signal at lower power, by both removing the necessity for large clock distribution networks, as well as reducing timing margins, allowing for a chip-wide reduction of supply voltage, translating directly to energy and power savings, when compared to traditional clock networks. Traditionally, clock generation and distribution requires over-provisioning of both circuits and wires (such as the global clock grid, high-power PLLs, and clock buffers)
and timing margins (to account for skew and jitter). And as on-chip clock often account for up to 30% of the overall chip power, injection-locked clocking can reduce the need for over-provisioning, and in turn, the power.

Once high-speed, low-power clocks can be provided, it is possible to enhance the communication substrate \((i.e.,\text{ the interconnect network})\). Traditional on-chip networks for chip multiprocessors consist of complex packet-switching routers and digital repeaters as links. Packet-switching routers are often large and require high power. Additionally, multiple hops result in non-trivial latency and energy in long distance communication. Of course, packet-switched interconnects provide scalable throughput, and are a valid design point. The question becomes, “Are the benefits of scalable throughput worth the costs for small and medium-sized chip multiprocessors?”

An alternative to the packet-switched interconnect can be provided by using properly designed transmission lines driven by high-frequency analog circuits, made possible by improving transistor technology. These links can be used as a basis for a shared-medium interconnect, which presents challenges to the architect, such as coping with inherent throughput-scalability deficiencies, while also providing unique advantages, such as high-speed, atomic, point-to-point communication, which the architecture can exploit. Even a simple, shared-medium bus, in a moderately-sized CMP, can provide high performance at significantly lower energy costs than the packet-switched alternative.

The transmission line link shared-medium bus provides a significant amount of raw throughput, but this throughput is not scalable. As the number of cores, and in turn the amount of traffic, increases, it is necessary to use more sophisticated techniques to improve scalability. By sacrificing some energy savings and increasing complexity, we can both increase the available throughput and reduce the need for high throughput. Therefore, the transmission-line link bus can function as a high-performance interconnect backbone for larger systems, while still maintaining significant energy savings.

Transmission lines and high-speed circuits provide certain characteristics, such as low-latency and low-power communications. These not only bring about opportunities in the network structure, but can also be used to optimize higher-level functions, such as the coherence
substrate. For example, messages sent on the TLL bus are atomic and transmitted in a bounded amount of time, unlike packet-switched interconnects. We can leverage this capability to co-design the network and the rest of the system. The last portion of the thesis will investigate the higher-level architectural design opportunities that arise from a communication network consisting of transmission lines and high-speed circuits.

**Thesis organization**

The remainder of this thesis will be organized as follows: Chapters 2 and 3 review detailed background on related work proposed in the community and describes the tools used to design and characterize transmission lines.

In Chapter 4, I present an on-chip clocking scheme using injection-locked oscillators for use in microprocessors. Such a design provides a low-power, high-quality (i.e., low skew) clock as a simple replacement for conventional PLL-driven clocks, providing a drop-in replacement for conventional microprocessors and a foundation for synchronous high-speed communication.

Next, Chapter 5 reviews transmission line theory, and Chapter 6 develops and explores a transmission line design space, targeted at a microarchitectural designer. Rather than focus the design space only on the circuit-level decisions, this exploration of transmission lines and transceiver circuitry takes into account both circuit- and architectural-design parameters, in order to optimize a transmission line for use as an on-chip shared-medium interconnect.

Once the transmission-line links have been explored, we can use them to form a globally shared-medium on-chip interconnect for use in a chip multi-processor, as in Chapter 7. The shared-medium interconnect removes the more complex, power-hungry routers from the network, providing a low-latency and low-power alternative for moderately sized CMPs. First, a small to medium scale system is presented with a simple network architecture. Following that, I present a network architecture which uses more sophisticated optimizations in order to (a) improve the scalability of the transmission line link bus architecture, and (b) take advantage of distinct opportunities provided by the transmission line link technology.

Finally, Chapter 8 will discuss some potential future directions for transmission line link based communication.
Thesis statement

Microarchitectural design can take advantage of opportunities provided by improving fabrication technology, specifically using on-chip high-speed transmission lines and communication circuits, to increase system performance and reduce overall power and energy.
Chapter 2

Interconnection Networks

Digital systems, and in particular, microprocessors, are made up of 3 major components: computation, memory, and communication [38]. In the general sense, each of these components can be made up of a variety of sub-categories. Memory may include local or shared cache hierarchies, or off-chip memory. Computation can be digital arithmetic units or analog components. Communication includes, but is not limited to, connections between neighboring functional units (e.g., an adder to a register), global communication between cores, and/or chip-to-chip interaction (e.g., CPU to memory). Figure 2.1 illustrates a generic digital system.

Figure 2.1: A broad interconnection network may connect CPUs, memory, I/O, or more.

In microprocessors, those components correlate to a CPU, cache/memory hierarchy, and the connecting fabric. As memory and computation scale to smaller feature sizes and faster overall
speeds, the inherent need for communication may become an increasingly pressing bottleneck. The integration of more components only exacerbates the problem, as more cores and memory storage are included on a single chip. While much of this communication is done locally between adjacent components (a core to its private L1 or shared L2 slice), a growing component of this on-chip communication is global, from core to core, or core to remote memory. This thesis will focus on the global communication fabric, which is commonly referred to as the interconnection network, rather than the local communication (more on this in Chapter 7).

With the integration of multiple cores on a single die, proposals of advanced interconnection have emerged. These proposals range from networks-on-chip (NoC) [4; 12; 17; 37; 42; 49; 54; 55; 57; 71; 73; 74; 78; 79; 87; 89; 90; 91; 96; 99; 113; 114; 127; 135] to optical interconnects [5; 31; 34; 50; 56; 67; 76; 77; 102; 103; 119; 120; 130; 131; 132; 133; 142] or RF interconnects [13; 14; 23; 24; 28; 29; 30; 62; 63; 86; 92; 100; 111; 116].

2.1 Common Terminology & Basic Interconnect Components

In order to effectively compare interconnect designs, it is necessary to review common language and terminology. For the purposes of the work discussed in this thesis:

- When 2 components want to communicate, the source creates a packet, and injects it into the communication substrate to transmit to its destination. A packet is a mechanism for sending a message. A packet can contain a memory address, some data from memory, or both, and also holds some control bits for the components to categorize and decode for operation. If a packet only contains address information and control bits, it is called a control or meta packet. When a packet contains a line of data, it is a data packet.

- A flit is a flow-control digit. A packet contains a head flit, a body flit and tail flit. The head denotes the beginning of a packet, containing routing and control information, the body contains the payload of the packet, and the tail denotes when the entire packet has been received.

- A phit, or physical digit, is a component of a flit. This is due to the physical size of the transmission medium. A phit is the size of the link width.
• **Total latency** refers to the time it takes for a flit or message to traverse the interconnect, from the time of injection to the time it is processed at its destination. Total latency consists of three major components:

  – *Propagation latency*: the time it takes for a bit to traverse the network medium (e.g., wires, waveguides, optical waveguides).
  
  – *Queuing latency*: the time a packet waits in buffersqueues before (or after) traversing the network. In the case of a packet-switched network, these queues exist not only at the source and destination nodes, but within each router as well.
  
  – *Serialization latency*: when a packet's size is larger than the channel’s width, the packet is broken into multiple pieces (phits), resulting additional time for a total packet traversal. For example, if a packet is 128 bits, and the channel is 32 bits-per-cycle wide, the serialization is 4 cycles.

• **Throughput** is the bit-per-cycle or bit-per-second capability of the network (e.g., 3 Terabits-per-second or 26 Gigabits-per-cycle). Often in the computer architecture community, throughput is synonymous with *bandwidth*. However, to avoid ambiguity, we refer to bandwidth as the frequency response of the medium, referred to in the wireless and analog design communities as *analog signal bandwidth*, measured in hertz. This is not in agreement with much of the architecture community.

There are a number of relevant throughput characteristics:

  – *Aggregate or peak throughput* refers to the number of bits which can be sent across the entire fabric in a given instant.

  – *Bisection throughput* is the throughput cross the middle (and narrowest part) of the system. In other words, if the network were to be cut in 2 parts of equal size, the bisection throughput is the minimum number of wires/links that you would cut. The bisection throughput is important for applications in which each node communicates with all other nodes.

  Aggregate throughput in a bus is equal to the bisection throughput. While a NoC has high, scalable, aggregate throughput which is higher than a bus, the bisection throughput of a bus
is high, which may be a better indicator of network performance for uniform random traffic. Table 2.1 lists some common topologies (discussed later) and the associated throughputs. Throughput, in general, is a result of the number of nodes and links in the system, the latency of the propagation and processing, and the composition of the messages and traffic. The number of nodes is a result of the CMP architecture. In the common case, the number of interconnect taps is equal to the number of cores in the system. In terms of a packet-switched network, this takes all intermediate hops into account.

Now that we have established a common terminology, we can more easily discuss the details of various network designs. We will review the three major components of a communication network: links, topology, and routing/flow control.

**Links**

The links of an interconnect system are the medium that carries the packet signal, as well as associated circuitry. A simple example of a link is a wire. In a conventional, digital system, a wire, whether buffered/repeated or not, carries a digital 0 or 1 from a source point to a destination. The link is a physical component, and is independent of topology or routing. The link design, as we will refer to it, includes not only the wire, or equivalent connection medium, but also contains the driver and immediate load on the wire (e.g., a digital repeater to amplify/receive the signal at the destination). Link design varies widely between interconnect designs. Optical links are made up of waveguides, electrical-optical (EO) or OE converters, and (de)modulators. In a transmission line topology, the links are the transmission line itself, along with the transceivers. In a traditional digital network, the links are the wires which connect the routers. These wires (and repeaters) have a large degree of design freedom, and can be designed for faster speeds, area, or energy efficiency [32; 97].

**Topology**

The interconnection network is made up of multiple links, laid out in a particular pattern, called the topology. The topology of the interconnect is the arrangement of the nodes and links and can be chosen independently of the link design. For example, a mesh or grid network (see
Figure 2.2-(a)) can be created using optical links instead of wires, and a large bus can be created using exclusively slow, narrow links. A packet is delivered by following a particular path in the topology, hopping through the shared channels or links from the source node to the destination. A topology is chosen based on several factors, including, but not limited to, intended traffic patterns, router complexity, distance of communication, and energy/power requirements. To make an analogy, if the links are roads connecting cities, than the topology is a map which describes how to construct those roads and depends on the city’s intended traffic pattern.

Routing and flow control

Routing and flow control, to extend the analogy, are the directions used to traverse the roads, and the traffic rules and signaling that control the movement and congestion on the roads. Routing determines which path a particular packet will use to reach its destination, deciding between multiple possible paths. Routing algorithms vary and can include minimal path routing (to reduce complexity and average latency), dimensional ordering (to alleviate deadlock concerns), or more adaptive routing (to maximize link efficiency). In the case of a conventional bus, routing is more simplistic, as there is typically only 1 path to traverse.

Flow control decides which packets are allowed to use a particular network component at a given time. For example, if multiple packets are waiting to send messages across a shared resource, the flow control resolves any conflict by ordering the access. In our proposed bus structure, we refer to the flow control unit as the arbiter, which dictates which node gets full rights to send on the bus, and for how long. Flow control is an important design choice, especially under heavy traffic conditions, as it influences fairness and potential queuing delay. If a flow control unit starves a particular resource, or ignores a particular node’s input, it may lead to significant performance implications, both for the network and the system as a whole.

2.2 Packet-Switched Networks-On-Chip

Packet switched architectures come in a variety of topologies, but a broad description is provided below. Figure 2.2-(a) shows a mesh topology. Each node contains a router which is
connected by digitally repeated wires. Traditional wires require a full-swing signal to transmit a '0' or '1'. In the domain of packet-switched on-chip interconnect, there are various proposals to optimize the canonical design. Muralimanohar et al. use varied wires for different purposes to reduce the long-latency, high-power communication [97; 98]. In general, these wires can be designed for specific purposes, trading latency for area or energy. Additionally, placement of the repeaters has a significant impact on the performance of these global wires. Digital differential signals can be used to lower the voltage swing and power of global wires, at cost to speed and wire count.

Global wires are only a medium for signal propagation. Overall, as lines grow in length and width, the result is less resistance and increased inductive effects, until finally the wire can and should be modeled as a transmission line. The main differences between a transmission line and a digital wire, other than the width and length of the wires, are (a) how these wires are driven and (b) how the lines should be accurately modeled. Traditional transmission lines are typically driven by analog drivers, while conventional digital wires are driven by simple full swing inverters. Chapter 6 will study these designs in depth.

Figure 2.2: Various types of interconnects (assuming a 16-node system). Each circle is a network node/router and the lines are the links. In the case of (e), the layout is logical, not necessarily the physical layout of a butterfly topology.
<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Aggregate Throughput</th>
<th>Bi-section Throughput</th>
<th>Average Hop Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ring</td>
<td>(n^2)</td>
<td>(n/2)</td>
<td>(n^2/2)</td>
</tr>
<tr>
<td>Mesh</td>
<td>(3n^2)</td>
<td>(n)</td>
<td>(n/3)</td>
</tr>
<tr>
<td>Torus</td>
<td>(4n^2)</td>
<td>(2n)</td>
<td>(n/4)</td>
</tr>
<tr>
<td>Flattened Butterfly</td>
<td>(6n^2)</td>
<td>(4n)</td>
<td>1-2</td>
</tr>
<tr>
<td>Hypercube</td>
<td>(4n^2)</td>
<td>(2n)</td>
<td>1-2</td>
</tr>
</tbody>
</table>

Table 2.1: Characteristics of various types of interconnect normalized to a single bit on a bus. \(n\) is the number nodes in a row and column.

The wires compose just one component of the packet-relay network. Simply redesigning the wires themselves may result in some significant savings, changing the overall latency or energy of the signal transmission. The larger contributor to the energy of the system is that of the router. Even simple routers have significant power consumption. Figure 2.3 shows the general layout of a router for either a mesh or a torus. Virtual channels and large crossbar switches provide the NoC with significant throughput, but require large area and power/energy. Crossbar switches connect \(n\) inputs to \(n\) outputs. In a typical mesh or torus network, a router needs 5 inputs and 5 outputs (north, east, south, west, and node). A crossbar switch arbitrates and connects the appropriate input to the intended output, without collisions. Virtual channels offer extra dedicated buffers which protect the router from stalling or slowing down when a single packet cannot be routed. In other words, if a particular flit is stored in a buffer, and cannot be removed due to congestion, then another buffer can be made available at that input, allowing more packets to enter the router. These buffers, switch and virtual channel arbiters, as well as the crossbar itself all add area, energy, and complexity to the network. And while these optimizations contribute to increased performance, the accompanying costs may become overwhelming.

Plenty of work has been done to lowering these costs, by optimizing the router latency and power. For example, Sanchez et al. analyze different packet-switched topologies in order to find the optimum design [113]. Kim changes the router design to reduce area and power overheads associated with the on-chip routing[72].
Figure 2.3: A generic virtual-channel packet-switched network router.

**Single-cycle routers**

A large subset of cutting edge of NoC research focuses on the latency of the router itself. Assuming either a standard mesh or torus network topology, the router can often become a costly component for overall transmission latency, especially at long distances. Mullins *et al.*, for example, uses speculation to hide the decision-making pipeline steps [96]. In the best case scenario, this router can achieve single-cycle router latency. However, in a congested network and/or with an imperfect prediction scheme, the router must account for penalty cycles, potentially hindering the router. Matsutani *et al.*, in [87] describe a low latency, speculative router which adapts its prediction scheme based on the current network state.

There has also been a some work on non-speculative, single-cycle routers proposed in the community [54; 55; 78; 90]. Kumar *et al.* present a redesigned router, both at an architectural and circuit level, focusing on reducing latency of each component, while not drastically increasing the power consumption or required circuit area [78]. Hayenga *et al.* proposed SCARAB, a single cycle adaptive routing and buffer-less router, reducing both the router latency and the dynamic energy consumption, by shifting buffer responsibilities to the processor side, minimizing the complexity of the router itself [54]. Hayenga and Lipasti, in a later work, present a new arbitration/hardware interaction. Rather than using a crossbar switch which buffers and pipes packets from the input port to the output ports, the crossbar is constructed with XOR gates. In an uncontested network, there is no need to arbitrate, and the packet will simply pass through.
When there is more than 1 packet at different outputs waiting to use the crossbar, rather than arbitrating for the switch, the packets are simply XORed together. The arbiter concurrently decides which packet should have been granted access, and eliminates it from the input queue. When the packets reach either the destination, or a point in which their paths diverge, the packets are XORed again to disambiguate them. This greatly reduces the input queuing delay, but does shift some of that delay to the back-end, as a packet now must wait for all XORed packets to arrive before it can be disambiguated and freed [55]. Michelogiannakis et al. exploit “packet-chaining” in order to reduce the routing time for small packets. As a brief example, if a packet traverses the crossbar switch from input port 2 to output port 1, if there is another packet waiting to traverse that same path, or another non-interfering path, the packet can be “chained” with the previous one, reusing the already allocated switch path, rather than wasting a cycle waiting for another path to be allocated [90]. While these routers offer faster latencies without drastic increase in energy, the overall energy consumption of the entire NoC architecture is not directly addressed.

2.2.1 NoC Topologies

While a mesh is a simple and effective topology, it is only one of the many topologies of a packet-switched network-on-chip. In general, the topologies are often classified into 2 categories: torus and butterfly.

Torus Topologies

The torus family of interconnects is denoted as k-ary, n-cube. The total number of nodes \( N = k^n \), where \( n \) is the dimensional grid and \( k \) is the number nodes in each dimension and channels between neighbors. Tori with a low dimensionality have short wires, minimizing the impact of the wires on operation. For local communication patterns, tori have a low total latency and high throughput. They also provide good load balancing. Torus wires can also be bidirectional. However, torus topologies do incur a large hop count for long distance communication. Figure 2.2-(a) through (d) illustrate various torus topologies: (a) 4-ary, 2-mesh; (b) 4-ary, 2-cube; (c) 16-ary, 1-cube; (d) 2-ary, 4-cube.
In the commercial chip space, most designs with more than a handful of cores still use conventional electrical interconnect, such as Intel’s 80-core ring topology [129] or IBM’s Cell Broadband Engine’s Element Interconnect Bus [8], which despite its name is in reality a collection of packet-switched rings.

**Butterfly Topologies**

Figure 2.2-(e) and (f) show a couple of examples of the other topology classification: butterflies. The notation to describe butterfly networks is as follows: $k$-ary, $n$-fly consists of $k^n$ nodes and $n$ stages of $k^{n-1}kXk$ intermediate nodes. Another way to describe a $k$-ary, $n$-fly is that there are $n$ stages of radix-$k$ switches. Reducing the number stages results in a more complex, higher-radix switch design, and vice versa, a greater number of stages allows for simpler switches with a lower radix. Butterflies typically have lower hop count than tori.

**2.3 Optical Networks**

A large set of modern research has focused on the use of optical devices to replace or enhance the interconnect. These works tend to focus heavily on the link design, but topology and routing are also considered. The general layout of an optical system is shown in Figure 2.4. A laser, often off-chip, provides the necessary optical power to drive the propagation medium, either free-space or waveguides. First we will review some recent work in free-space optics [142]. Rather than use a waveguide to distribute light, point-to-point lasers paths are done using on-chip lasers and mirrors to guide signals through free-space. On-chip lasers are low energy, and do not shift power constraints off-chip. Figure 2.5 shows a free-space optics interconnect and on-chip VCSELs (Vertical Cavity Surface Emitting Lasers) to form a point-to-point interconnect based on high-speed optics.

Waveguide materials are laid out in a similar fashion to wide wires, but with a different propagation material. In recent work, micro-rings are used to couple specific wavelengths of light in and out of the medium. Figure 2.6 illustrates a number of rings each operating at a different wavelength. The devices tune to specific wavelengths and a medium carries said multiple
Figure 2.4: An optical interconnect system, often with off-chip lasers, requiring electrical-to-optical and optical-to-electrical conversion, as well as a propagation medium.

Figure 2.5: A free-space optical interconnect. A GaAs substrate containing photonic devices is laid on top of a CMOS substrate. Optical signals are sent from the VCSEL through a micro-lens. The light is then reflected through free-space off of the mirrors until it reaches its destination, passes through another lens, and is converted back to an electrical signal through a photodiode. [142]

wavelengths without interference, allowing for wavelength division multiplexing, increasing the available throughput of the links. Wavelength-division multiplexing merges multiple optical signals, at different wavelengths, into the same medium. These rings are currently in research, and have integration issues with current CMOS microprocessors, making them less desirable in the short term.

One specific design that merits special mention is the Corona interconnect [132], which uses these microrings to modulate and receive optical signals across an optical waveguide acting as an atomic bus. By manipulating multiple microrings, operating at different wavelengths (i.e., WDM), the network can provide significant throughput and speed-of-light propagation speeds
over long distances. Even in moderately sized CMPs, the attenuation caused by multiple rings operating on the same wavelength can degrade the performance of the system. Conceding these constraints, the Corona system, which is essentially an optical bus operating at multiple frequencies, could provide high throughput and high speed interconnection. It does however, suffer from problems inherent for many optical systems, including sensitivity to temperature variations and ease-of-integration issues.

2.4 Hierarchical & Bus Interconnects

Early processors used buses for on-chip communication. Figure 2.7 shows a general bus structure, connecting a CPU, memory, and input/output control for a microprocessor. As the speed of the processor increased and the number of interconnected components grew, the bus’s inability to provide high throughput became a bottleneck. Larger-scale multiprocessors (e.g., SGI Orion [83]) and other parallel systems (e.g., IBM SP2 [7]) use packet-switching routers that can provide different connection topologies and configurations depending on the system’s size. While earlier systems used I/O buses to connect processors with routers, further integration allows the chip to contain both processor core and router for interconnection [94]. Modern systems, such as InfiniBand and HyperTransport, provide system-level and board-level shared bus communication structures to connect multiple chips and boards together [59; 88].

Hierarchical structures have been proposed to avoid relying on full-fledged packet-switching architecture [39; 128]. It is worth noting that such a design does not necessarily eliminate packet switching. The authors refer to the structures as a bus, but we wish to clarify that it still contains packet-routers and cannot perform operations atomically. An operation is considered atomic when it appears to occur instantaneously and simultaneously across the entire system. In a conventional notion of a bus, when a packet is sent, it is received, with no
intermediate steps. With a hierarchical system, the transition from one level of the hierarchy to the next removes the possibility of atomicity. For instance, the hubs connecting the “bus” segments need to buffer packets and arbitrate for the next segment, essentially serving as a router, albeit with fewer input and output ports \[128\], than a full mesh/torus/fly topology, and thus is not a typical bus structure.

While early buses were simple atomic structures, connected CPUs to memory and I/O devices, as we have already discussed, traditional digital wires could not sufficiently accommodate large scale CMPs. A compromise between a digital-bus system and a full-fledged packet-switched mesh was proposed by Udipi et al. \[128\]. Given a 16-node system, the nodes are connected first by local buses, and then the local buses connect to a global bus (the bus, in this case, can also be viewed as a ring). Figure 2.8 illustrates the idea. While locally there is no need for packet-routing, globally at least 2 routers must be traversed. Additionally, the long wires that make up the global ring/bus can be a source of delay and energy loss, as they are still digitally driven, requiring repeaters and large drivers. It is certainly possible to design these lines with these long distances in mind to either lower latency, or lower energy consumption. Udipi et al. focus on energy consumption, sacrificing propagation latency.

Another important difference between this related work and the architectural setup of this thesis is the cache coherence protocol. In our work, we utilize a directory protocol, described
Figure 2.8: A top-level view of the hierarchical bus/ring structure proposed by Udipi et al. [128].

in a more detail below and in Section 7.3. Udipi et al. instead use snoop-based protocol, also described below.

**Directory-based cache coherence**

In a directory protocol, the lower level caches contain the state of the each line (in this case, we will assume directory states are held in the L2). One simple protocol, identified by the abbreviation MSI, allows lines to have 3 global stable states: modified, shared, and invalid. If a line is invalid, it can no longer be used by any caches, and must be reloaded from a lower level of memory. In shared state, the line is clean, and may be held by more than one L1 cache. Each of the shared versions of this line in the L1 is identical to the others and to the lower level versions. In modified state, only one L1 contains a valid copy of the particular line, which assumed to have been written. When a modified line is requested, it is necessary to invalidate and writeback the dirty version. Figure 2.9 shows a state diagram for a MSI protocol.

There are also transient states for the lines. These states are explicitly differentiated, but instead the line is marked as transient until the state is stable. For example, if line A is marked as shared and is shared by multiple nodes, and then a node requests write-permission to line A, each of the sharing nodes is sent an invalidation message. Each of the sharing nodes processes this request, invalidates the local copies, and then responds to the L2 directory with an acknowledgement that the invalidation has been processed. The requesting node is then sent
line $A$ in modified state. Between the time of the invalidation messages and the collection of all acknowledgments, the line is kept in transient state. This is necessary to ensure that no other operations are performed on line $A$, in order to keep the global memory ordering.

There are more complex coherence protocols. A MESI protocol adds an exclusive state. If a line is clean, but not shared by more than 1 node, it is exclusive, reducing the time for write-requests. Another option is MOESI, which adds an owner state. An owner holds a valid copy of a line, but lower levels of cache can have incorrect data. In this case, the hierarchy is not inclusive, saving communication to lower levels. There are more coherence protocols, and much further detail for each.

All cache transactions in this case are handled by the directory. By centralizing the accesses to a single point, we can assure global ordering. However, there is some inefficiency in the system. In the previous example, the directory does not hold any valid and clean version of the line. Thus, by sending a request to the directory, and waiting for the directory to send back valid data, we have added “hops” through the network which expend energy and take time to traverse the network and process at each node. If instead, we broadcast the write-request to all nodes,
and remove the directory from the critical path, the overall latency of the request-reply stream is reduced. This system is referred to as a snoop-based protocol.

**Snoop-based cache coherence**

A snoop-based (or snoopy) protocol alleviates the latency overhead of multi-hop communication introduced by the directory. Instead, a request is sent to all nodes. The request is then processed by each node, and acknowledgements are sent to the requester. The possible states of the lines, such as MSI, or MESI, are similar to that of the directory protocol. The difference is how the requests are handled. However, while the snoop-based protocol is more latency efficient, it suffers from high energy overhead. For each request, each message is processed $n$ times, if $n$ is the number of nodes connected to the interconnect backbone. This means more energy is consumed for each message, even if that node is not involved in the transaction.

Udipi et al. use a filter to remove some of the snoop-based traffic which may otherwise overwhelm the interconnect. The router keeps state of sharers, and if an address is not being shared globally, then there will be no need to route the packet to remote nodes. This saves not only on traffic, but significantly reduces the energy spent on any particular snoop-request, lowering the overhead of snooping as a protocol scheme. These savings, along with the savings of localizing much of the traffic, lowers the energy of the entire system, and has equitable performance compared to a traditional mesh network.

### 2.5 Transmission Lines in Existing Interconnects

There has been some past research proposing the use of transmission lines in an on-chip environment. Transmission line mechanisms have been explored by Beckmann and Wood to enhance the communication in L2 caches [13; 14]. Recent proposals use RF circuitry as an acceleration mechanism to supplement the conventional mesh network [28; 29]. To supply sufficient bandwidth, the transmission line is used as a multi-band medium, at the cost of complexity in transceiver design. Finally, transmission lines are also used in building fast synchronization mechanisms [100].
Beckmann and Wood first presented transmission lines as a means to connect remote L2 banks [13; 13]. It is meant as an alternative to non-uniform cache access (NUCA) systems. In a chip multiprocessor, the L2 cache is large. Accessing different lines in the cache may incur different latencies, as some are more distant. While lines close to the controller are fast, the lines which are farther away slow down the overall latency of all accesses. To alleviate the impact of the longer latency accesses, we can bank the L2, either locally or in a distributed fashion, and assume different access times for each bank. The large latency is often a result of long wire delays across the chip. A NUCA architecture allows some banks of the L2 to operate faster than others, hiding the long wire latencies of more distant L2 banks. Beckmann and Wood offer an alternative, replacing the long digital wires with transmission lines, lowering the latency impact of these remote banks, taking advantage of the faster transmission time. In these works, the TLs are used only as a communication fabric between the local L1 and the shared L2 caches, not as a backbone for all core-to-core communication.

Chang et al. use the transmission lines for the interconnect backbone, as a supplement to an already established packet-switched mesh [28; 29; 30]. The transmission lines provide high-speed “highways” to avoid long packet delays. There is an important distinction between these works and the ideas proposed in this thesis: in this thesis, all inter-node traffic in our work is placed on the transmission line, while in [28; 29], the traffic is distributed on the transmission lines and the underlying mesh network. Congestion and the source and destination determine what route a particular packet takes. Essentially, the transmission lines can be considered as a special packet-switched link, which can be used to offload some of the mesh traffic, by bypassing mesh routers and shorter digital links. Another important aspect to note is that Chang et al. use frequency division multiplexing in order to increase the overall throughput. By modulating a signal to a higher frequency and feeding this signal through mixers/filters, the throughput of the narrow TL medium can be increased. This presents significant issues for the design of the transceiver, as higher frequencies require more power to transmit and re-constitute the signal at the receiver. More details on this are presented in Section 7.4.

Another possible use for the high speed transmission lines is for high priority synchronization primitives. Oh et al. use the TLs to provide high-speed barrier synchronization. Each transmission line is connected to \( n \) nodes. When a barrier starts, each node pushes a “1” onto
the TL. When a barrier is reached, the node stops transmitting. When the signal on the TL is “0,” each node knows it can continue on past the barrier. This works brings up a point which we will explore: specializing the TLs for particular types of traffic (e.g., synchronization, data lines, control messages) can result in performance/energy improvements which lend insight into how the entire network interacts with the traffic generated by the system.

These designs rely on point designs for the transmission line systems, or depend on prior independent designs. In this work, we attempt to co-design the transmission lines, transceiver circuits, and architecture in order to optimize the entire design stack.
Simulation Tools

Rather than making assumptions about the transmission lines and the transceiver circuits, it is important to understand the performance of these components before using them in the architectural analysis. In order to analyze the circuits, we use Sonnet, which is a first-principle electromagnetic simulator [1]. We use Sonnet to provide the behavioral model of the transmission lines, which we then use in ADS, a circuit simulator created by Agilent Technologies, to determine the overall behavior of the circuits and system. In this chapter, the procedure for the circuit design and analysis is presented.

3.1 Sonnet

In simple terms, Sonnet requires a map of the layout of materials in the system. For example, for a microstrip transmission line, the design would require 3 metal layers: a top layer for the strip, a middle layer which is dielectric, and a bottom layer for the ground plane. The top layer and bottom layer would be made of a conductive material (e.g., gold, copper), and the middle layer consists of a dielectric (e.g., silicon dioxide). Figure 3.1 shows the setup from a top level view, and Figure 3.2 shows the setup from a 3D view.

First, it is necessary to choose a layout workspace, which is defined by the “box,” shown in Figure 3.1 as the white box. Essentially, this lays out the Cartesian grid in which we will layout the materials. Figure 3.1 shows only the top layer, which contains the gold wires, and
Figure 3.1: A Sonnet screenshot displaying the top level view of a group of coplanar strips. The top layer and bottom layers are gold, and the middle layers are SiO$_2$ with a dielectric constant of 3.0.

the underlying middle layer, which is a slab of dielectric. These materials are selected through the properties specified in Sonnet’s environment. The goal of our Sonnet analysis is to provide S-parameters of the system to input into a circuit simulator for further evaluation. In the figures shown here, we are exploring coplanar strips, which will use differential signaling. Therefore, the paired lines can be marked as negative pairs. For example, a pair of wires can be marked with the ports “1” and “-1” to indicate that they will be connected together (in our case, connected to either terminal of a matched impedance). This is done to simplify the analysis, both in Sonnet, and later in ADS, when more ports translates to more data and more time. In a similar manner, the other layers should be added according to the requirements. A final version of the coplanar
strip layout is shown in 3D form, in Figure 3.2. Of course, here we only discuss a simple straight line. Sonnet is capable of much more complex structures. The straight transmission line is shown for simplicity.

![Fig. 3.2: A Sonnet screenshot displaying the top level view of a group of coplanar strips. The top layer and bottom layers are gold, and the middle layers are SiO₂ with a dielectric constant of 3.0.]

Once the device, in this case, the transmission lines, are completely described in the workspace, we must analyze the structure, in our case to find the S-parameters. The S-parameters describe the port-to-port interaction of a device or system. For example, the \( S_{2,1} \) describes the frequency response of a system between ports 1 and 2. Figure 3.3 shows the analysis environment for Sonnet.

There are a few caveats to keep in mind when designing these transmission line systems. Longer lines require longer testing time. Testing a 75mm line could take days or more to run a full spectrum analysis from 0-50GHz. Instead, a shorter line can be designed, and then the S-parameter component replicated to emulate a longer line (see Figure 3.4). Additionally, while Sonnet can display frequency spectrum results. However, to test circuits along with the TLs, a circuit simulator is necessary. The resulting S-parameters are then transferred to ADS for final circuit evaluation.

### 3.2 ADS

ADS, from Agilent Technologies, is a circuit simulator for analog, RF, and digital systems. One of the key attributes for transmission line analysis is the ability to import a generic S-
Figure 3.3: Sonnet’s S-Parameter analysis (signal and crosstalk). The axes have been enhanced for easier reading.

parameter “black-box.” ADS imports the Sonnet-generated S-parameter model for analysis. Figure 3.4 shows the ADS environment, in this case displaying a series of concatenated S-parameter components, each using the same model from Sonnet for a given transmission line design.

S-Parameter Analysis

The ADS simulation environment has far more capabilities than will be covered in this thesis. Instead, we will explore a few examples of transmission line evaluation. The performance across the frequency spectrum can be assessed using the “S Parameter” simulation option in ADS (we will briefly discuss “Transient” analysis used for circuit simulation later). This option requires a few inputs/components before running: (a) port terminations, in this case shown as resistances, and (b) the frequency range (and resolution), in this case from 0 GHz to 50 GHz, at a step of 100 MHz. The terminations should be matched to reduce the influence of reflections in the system. This can be determined by test-and-error, or by mathematically determining the approximate values. The frequency range is a factor of what window is of interest for the particular application. Sonnet’s S-parameter model is frequency dependent, and thus the ADS
frequency range should be the same. ADS has native transmission line models which can be used, but are less accurate than Sonnet, as ADS is based on simple mathematically extracted models. Hence for this work, we choose to use Sonnet to create realistic transmission line models. Figure 3.5 shows the ADS implementation of the S-parameter simulation using the Sonnet input files, and Figure 3.6 shows the results window.

For the purposes of this thesis, the most important result of the simulation is the frequency analysis for the input/output of the signal lines and the crosstalk between the signal lines and the neighboring aggressors. In Figure 3.6, we show 2 different widths, to illustrate the difference in the frequency response for multiple sizes. The y-axis is shown in decibels, but can also be shown in formats such as magnitude, phase, absolute value, and more. To examine bandwidth, we use the decibels option, in order to investigate the 3-dB point of the graph, measured from the 0 GHz magnitude to the frequency which is 3-dB less. Additionally, we show the crosstalk for such lines.
It is also worth noting that there are a variety of measurement tools within ADS. Here we show the use of a simple trace marker which displays the x and y coordinates for any point on the graph, making for easier reading.

**Transient analysis**

In a similar fashion, we can also do transient analysis. Specifically, we use transient analysis to design and evaluate the circuit performance for the transceiver. To show detail, Figure 3.7 shows the design of a differential amplifier used in the transmitter and receiver of our final design, discussed more in Chapter 6.

There are some tricks to remember when doing a transient analysis such as this. For example, it is important to use random inputs to avoid any periodicity which may adversely affect the output, hiding possible issues. Additionally, to mimic a real-life system, noise sources should
Figure 3.6: The ADS frequency response of 2 transmission lines. The attenuation (top) and crosstalk (bottom) of the lines are shown, along with trace markers.

be inserted on the supply voltage lines and inputs. Finally, ADS offers a number of simple meters (voltage and current) which prove useful in the final analysis, for correctness, timing, and power/energy calculations.

Overall, ADS has an extremely large range of capabilities, including timing and frequency analysis, as well as a library of components which can be used for designing the intended circuit. These components can be used as part of the standard library, or can be linked to an external model or netlist, as is usually the case for transistors. Rather than rely on assumptions
Figure 3.7: A differential amplifier used for transmitters and receivers of the analog transceiver design.

or “black-boxes” for the circuits and devices, we choose to use ADS and Sonnet to provide realistic analysis of the simple and practical designs.
Chapter 4

Injection-Locked Clocking

4.1 ILC Overview

High-speed circuits, including those driving transmission lines, require high-frequency, high-quality clocks, often an order of magnitude higher than the CPU’s computation clock. Distributing high quality clock signals is one of the most challenging tasks in high-performance microprocessors. Clock circuits account for an overwhelming amount of total power consumption in multi-GHz processors. Unfortunately, deteriorating clock skew and jitter make it difficult to reduce power dissipation. A new global clocking scheme, injection-locked clocking (ILC), is proposed to combat clock skew and jitter. This new scheme uses injection-locked oscillators as the clock receivers. It can achieve better power efficiency and jitter performance than conventional buffered trees with the additional benefit of built-in deskewing. Unlike other proposed clocking schemes, ILC is fully compatible with conventional clock distribution. In this chapter, a quantitative study based on circuit- and microarchitectural-level simulations is conducted to verify the performance and power consumption improvements using ILC compared to conventional clocking.
4.2 Clock Generation and Distribution

Clock distribution is a crucial aspect of modern multi-GHz microprocessor design. Improving technology, which provides higher quality devices at higher frequencies helps to minimize the overall impact the clock distribution network on the system, but can still be improved, using novel circuit techniques. Conventional distribution schemes are more or less monolithic in that a single clock source is fed through hierarchies of clock buffers to eventually drive almost the entire chip. This raises a number of challenges. First, due to irregular logic, the load of the clock network is non-uniform, and the increasing process and device variations in deep sub-micron semiconductor technologies further adds to the spatial timing uncertainties known as clock skews. Second, the load of the entire chip is substantial, and sending a high quality clock signal to every corner of the chip necessarily requires driving the clock distribution network “hard,” usually using the full swing of the power supply voltage. Not only does this mean high power expenditure, but it also requires a chain of clock buffers to deliver the ultimate driving capability. These active elements are subject to power supply noise, and add delay uncertainty – jitter – which also eats into usable clock cycle. Jitter and skew combined represent about 18% of cycle time currently [95], and that results in indirect energy waste as well. Skew and jitter will be discussed often in this chapter, and thus, should be explicitly defined:

- Skew is the time difference between edges of a signal propagating through the network to two different points on the chip. In other words, due to spatial and circuit differences in the network, some regions receive a signal edge earlier than others, creating a slight timing shift in events. For example, if 2 latches at different points on a chip are awaiting the arrival of a clock edge, that edge may reach one latch before the other. Skew is consistent from cycle to cycle, not varying with time.

- Jitter is the difference in signal arrival from an ideal system. In simpler words, if an event is expected at a particular time, noise will shift the event earlier or later. This varies from cycle to cycle, and is largely a result of noisy elements being on the signals path (i.e., the more circuits on the path, the higher chance of jitter).
For a fixed cycle time budget, any increase in jitter and skew reduces the time left for the logic. To compensate and make the circuitry faster, the supply voltage is raised, therefore increasing energy consumption. Conversely, any improvement in jitter and skew generates timing slack that can be used to allow the logic circuit to operate more energy-efficiently.

As commercial microprocessors are rapidly becoming multi-core systems, monolithic clock distribution will be even less applicable. In the era of billion-transistor microprocessors, a single chip is really a complex system with communicating components and should be treated as such. In communication systems, synchronizing clocks is also a rudimentary and crucial task. In this chapter, the concept of injection locking is reviewed and a microprocessor clock distribution network is developed using the ILOs.

Injection locking is the physical phenomenon where an oscillator “locks on” to an external stimulus (a periodic signal) and fundamentally synchronizes with the input when the frequency of the input signal is close enough to the oscillator’s native frequency or its (sub)harmonics. Recent circuit implementation of injection-locked oscillators (ILO) not only demonstrated superb gain and noise rejection, but also showed flexible frequency multiplication and division capabilities and phase adjustment capabilities. Using ILOs, the (global) clock distribution of a microprocessor can be improved substantially. For example, all logic macro blocks can be clocked by independent ILOs connected to a low-swing global clock input signal. Compared to the traditional approach where a local clock buffer is being driven by a full-swing clock signal directly driven from a central source, ILOs enable much lower power expenditure on the global level and eliminate multiple levels of clock buffers, which in turn, reduces clock jitter. Additionally, one can further reduce clock skew time leveraging phase shift capabilities of the state-of-the-art design of ILOs.

Apart from the technical advantages, injection-locked clocking is also a non-intrusive technology. There is no need to change the processor architecture or the design methodology the way (partially) asynchronous designs do. All in all, ILOs promise to bring significant advantages to current and future high-speed microprocessors and open up opportunities to design novel clocking schemes. Given the significant investment in IPs and design tool chains in the synchronous regime and the fact that high-end microprocessors routinely spend 30% or more
on clock distribution, injection-locked clocking (ILC) is a promising solution to meet the increasing challenge of clock distribution.

This chapter discusses a few possible scenarios of using injection locking for clock distribution, reviewing the results of a detailed quantitative analysis comparing some options of ILC designs with conventional approaches in terms of power consumption. Due to the scarcity of detailed reports on processor clock distribution, especially its power consumption, in the public domain, our study is constrained to a few ILC options that are suboptimal. Even using these limited options, simulation results suggest that power consumption of a high-end processor reduces from 40.7W to 33.9W, a 17% reduction. This clearly shows the potential of ILC.

4.3 Challenges in Conventional Clocking

Figure 4.1 shows a typical conventional clock distribution scheme. The global clock is generated by an on-chip phase-locked loop (PLL) from an off-chip reference clock, usually a crystal oscillator at tens of MHz. The global clock is distributed using an H-tree, which consists of both distribution lines and clock buffers, and then further distributed by local clock distribution networks. In order to minimize the global clock skew, the global clock-distribution network has to be balanced by meticulous design of the distribution lines and buffers. This practice puts a very demanding constraint on the physical design of the chip. Even so, the ever-increasing process variations with each technology generation still results in greater challenges in maintaining a small skew budget. Another current practice is to use a grid instead of a tree for clock distribution, as shown in the upper-left local clock region in Figure 4.1. A grid has a lower resistance than a tree between two end nodes, and hence can reduce the skew. At the same time, a grid usually has much larger parasitic capacitance (larger metal layers) than an equivalent tree and therefore takes more power to drive. Passive and active deskew methods [45; 81; 109; 125] have also been employed to compensate for skew after chip fabrication. This approach increases the chip complexity, manufacturing cost, and in the case of active deskew, power consumption and jitter.

Jitter poses an even larger threat to microprocessor performance and power consumption. The global-clock PLL and clock-distribution network generate noise, and hence contribute to
global clock jitter. But the main culprit is usually the noise coupled from other circuits, such as power supply noise, substrate noise, and cross-talks. Short-term jitter (cycle-to-cycle jitter) can only be accounted for by adding a timing margin to the clock cycle, and hence degrades performance. Unlike skew, jitter is very difficult to compensate for due to its random nature. In order to reduce jitter, the interconnect wires in the global clock distribution network needs to be well shielded from other noise sources, usually by sandwiching them between $V_{DD}$/ground wires and layers. Shielding inevitably increases the parasitic capacitance of the clocking network, which means more and larger clock buffers, and hence larger power dissipation to drive them. In turn, having more buffer stages introduces another source of jitter, and the situation deteriorates quickly with faster clock speed. It is evident that current skew and jitter reduction techniques almost always result in higher power consumption. A better clocking scheme with less jitter and skew directly translates into power savings for a given performance target.

4.4 On-Chip Clocking Networks

There have been intensive research efforts in recent years to address the challenges in high-speed clocking from different disciplines, including clockless design (asynchronous circuits), optical interconnect, and resonant clocking, to name a few. Each of these alternative solutions has its own technological issues to be addressed.
Optical interconnect potentially offers smaller delays and lower power consumption than electrical ones, and is promising for the global clock distribution network [48; 68; 95]. However, there are still great challenges in its silicon implementation, particularly for on-chip electrical-optical modulators [22]. Wireless clock distribution, proposed in [51; 84], suffers substantial overhead in chip area and power consumption due to on-chip clock transceivers.

Among the proposed electrical solutions, a family of synchronized clocking techniques, such as distributed PLLs [52; 107], synchronous distributed oscillators [93; 126], rotary clocking [137], coupled standing-wave oscillators [101], and resonant clocking [27] have been proposed to improve the performance of global clock distribution. In [26; 27], on-chip inductors are added to all of the local nodes of the global clock distribution tree, and hence turn it into a single large resonator. As discussed in Section 4.5.1, resonance improves power efficiency. Therefore, this technique reduces dc power dissipation and lowers jitter in the global clock distribution network. It is a good step in the right direction. However, it does not provide deskew capabilities like injection-locked clocking. The more stringent layout constraints due to on-chip inductors could even aggravate the problem of skew.

In [52; 107], an array of PLLs is constructed using a voltage-controlled oscillator (VCO) and loop filter at each node, and a phase detector between adjacent nodes. Each PLL generates the local clock in the particular clock domain, which is synchronized with others through the aforementioned phase detectors at the clock domain boundaries. The conventional notion of a global clock is removed in this scheme, and hence it promises lower jitter. The drawbacks are that a) the global skew is still a problem since deskewing only happens locally, and b) the sensitive analog circuits in a PLL (phase detectors, loop filters, ring oscillators) are vulnerable to noise in the hostile environment of digital circuits.

In [93; 101; 126; 137], an array of oscillators are connected to the global clock distribution network, and thus are synchronized by coupling. The resulting oscillator array becomes a distributed oscillator. The difference is that in [137] the oscillator array is a one-dimensional loop, and the phase of oscillators change linearly along the array, similarly to a distributed VCO [139], which was based on traveling-wave amplification [47]. In [101], the oscillator array generates a standing-wave pattern on the network, i.e., each oscillator has the same phase. Essentially all these techniques use a distributed oscillator with interconnects as its resonator. A distributed
oscillator suffers the problem of phase uncertainty due to mode locking [52; 107; 126]. This is evident in that similar topologies can be used for either traveling-wave [137] or standing-wave oscillation [101]. Another problem is that jitter tends to be worse than conventional clocking since the global clock is now generated on chip using lossy passive components, without the clean reference clock from the off-chip crystal oscillator. It is worth noting that [26] unintentionally adds injection locking to distributed oscillator clocking and demonstrated good jitter performance.

Overall, all these promising technologies face significant technical difficulties and require dramatic changes in process technologies, design methodologies, or testing methods, and hence will face significant resistance in adoption. In comparison, injection-locked clocking is highly desirable because it is fully compatible with existing IC infrastructures as well as current design and testing methodologies.

4.5 Injection-Locked Clocking

4.5.1 Injection-Locked Oscillators

Injection locking [6; 82] is a special type of forced oscillation in nonlinear dynamic systems (also known as synchronization). Suppose a signal of frequency \( \omega_i \) is injected into an oscillator (Figure 4.2-(a)), which has a self-oscillation (free-running) frequency \( \omega_0 \). When the difference between \( \omega_i \) and \( \omega_0 \) is large, “beats” of the two frequencies are observed. As \( \omega_i \) approaches \( \omega_0 \), the beat frequency (\(|\omega_i - \omega_0|\)) decreases. When \( \omega_i \) approaches \( \omega_0 \), the beats suddenly disappear, and the oscillator starts to oscillate at \( \omega_i \) instead of \( \omega_0 \). The frequency range in which injection locking happens is called the locking range (Figure 4.2-(b)). Injection locking also happens when \( \omega_i \) is close to the harmonic or subharmonic of \( \omega_0 \), i.e., \( n\omega_0 \) or \( \frac{1}{n}\omega_0 \). The former case can be used for frequency division, and the latter for frequency multiplication.

An injection-locked oscillator (ILO) can be considered as a simple first-order PLL (Figure 4.3-(a)), in which non-linearity of the oscillator core functions as a phase detector. For example, in a typical divide-by-2 ILO (Figure 4.3-(b)) [108], the oscillator core (consisting of \( M_1 \), \( M_2 \) and \( M_t \)) also serves as a single-balanced mixer for phase detection. Because of the
simple structure, ILOs consume much less power than a full-blown PLL and can operate at extremely high clock speeds [138]. The fact that the built-in “phase detectors” are mixer-based also explains why ILOs can operate at the harmonic and subharmonic frequencies of the input signal.

Once locked to the input signal, the output of ILOs will maintain a determined phase relative to the input signal (Figure 4.4). The phase difference from the input signal to the output is determined by the injection signal strength, the frequency shift from its free-running oscillation frequency, and the frequency characteristics of the oscillator resonator. As shown in Figure 4.4, the phase shift $\varphi$ is a monotonic function of the frequency shift $\Delta \omega$, and the function is quite linear within the locking range except when close to the edges. Tuning the free-running fre-
quency of the oscillator also tunes the phase of the output signal \[^{145}\]. Converting to time, this means tunable delay for the output signal. This phase transfer characteristics can be utilized to achieve deskew between different clock domains with no need for other deskew circuits.

\[ \varphi = \frac{I_{\text{inj}}}{I_{\text{osc}}} \]

\[ \eta = 0.5 \quad \eta = 1 \quad \eta = 1.5 \quad \eta = \infty \]

\[ \omega = \omega_0 \quad \Delta \omega \equiv \omega - \omega_0 \]

\[ Q \]

Figure 4.4: Phase transfer functions for divide-by-2 ILOs in Figure 4.3-(b). $\eta \equiv I_{\text{inj}}/I_{\text{osc}}$ is the injection ratio, $\omega_0$ is the free-running oscillation frequency, $\Delta \omega \equiv \omega - \omega_0$ is the frequency shift, and $Q$ is the LC tank quality factor.

### 4.5.2 Clocking using ILOs

This chapter presents a new clocking scheme as shown in Figure 4.5. Similar to conventional clocking, the global clock is generated by an on-chip PLL and distributed by a global tree. The difference is that the proposed scheme uses injection-locked oscillators (ILOs) to regenerate local clocks, which are synchronized to the global clock through injection locking (see Section 4.5.1). Another difference is that most global clock buffers in conventional clocking are removed because the sensitivity of ILOs is much greater than digital buffers (see detailed discussion below). Essentially, ILOs are used as local clock receivers, similar to the idea of clock recovery in communication systems. Note that this is different from resonant clocking \[^{26}\], where all the oscillators are coupled together (see Section 2). Further, ILOs can be constructed as frequency multipliers \[^{69}\] or dividers\[^{108; 140}\], and hence this scheme enables local clock domains to have higher ($n \times f_0$) or lower clock speed ($f_0/m$) than the global clock ($f_0$). Such a global-local clocking scheme with multiple-speed local clocks offers significant improvements.
over conventional single-speed clocking scheme in terms of power consumption, skew, and jitter.

![Injection-locked clocking scheme](image)

**Figure 4.5: Injection-locked clocking scheme.**

### 4.5.3 Power Savings

Injection-locked clocking can lead to significant power savings in high-speed microprocessors. The benefits come from several aspects of the design. First, the possible combination of a low-speed global clock and high-speed local clocks can reduce the power consumption in the global clock distribution network. In the conventional approach, this would require multiple power-hungry PLLs for frequency multiplication. An ILO consumes much less power than a PLL because of their circuit simplicity [138]. This will become more evident as more cores are added on-chip.

Second, ILOs have higher sensitivity than buffering inverters. As a synchronized oscillator, an ILO effectively has very large voltage gain when the injection signal amplitude is small, while the gain of an inverter is much smaller (Figure 4.6). This can be easily understood if we realize that synchronization in an ILO is usually achieved in tens to hundreds of clock cycles, and hence in each clock cycle only a small amount of injection locking force is needed. Meanwhile, a digital inverter needs to change its state twice in every clock cycle. Therefore, the global clock signal amplitude can be much smaller in the new clocking scheme, which means less power loss on the parasitic capacitance and resistance of the global-clock distribution network. This will be increasingly attractive as the wire loss becomes a dominant factor as the process technology scales further.
Figure 4.6: Voltage gain of an inverter and an injection-locked oscillator at different input signal voltage levels.

Further, the number of clock buffers in the global clock distribution can be reduced. In conventional clocking, in order to minimize jitter generated by digital buffers, the global clock signal needs to be driven from rail to rail throughout the whole network, and in turn many clock buffers are inserted. In injection-locked clocking, ILOs can achieve good jitter performance with small input signal amplitude (see Section 4.5.5). Therefore, the global clock signal amplitude no longer needs to be full $V_{dd}$ swing, and few (or none at all) clock buffers are needed on the global tree. Reduced number of clock buffers directly translates into lower power consumption. Figure 4.7 illustrates the conventional buffer tree and associated voltage swings versus the proposed ILC tree with reduced swing. Note that no buffers are used in Figure 4.7-(b) between the global PLL and ILOs, allowing the signal to degrade before being amplified by the local ILOs.

More importantly, because injection-locked clocking significantly lowers skew and jitter in the global clock, the timing margin originally allocated can be recovered, and used for circuit operation. This can enable faster clock speed or can be traded for lower power supply voltage ($V_{dd}$), and save power dissipation from not only the clock distribution network, but all the logic gates on the chip. In Section 4.6 below, a quantitative case study demonstrates the power savings from all aspects.
Figure 4.7: The (a) conventional clock tree (shown without clock grids) and (b) proposed ILC tree and associated voltage swings.

4.5.4 Skew Reduction and Deskew Capability

Because the number of buffers is reduced in the new clocking scheme, skew due to mismatch of clock buffers is reduced compared to conventional clocking. More importantly, ILC provides a built-in mechanism for deskew. From Section 4.5.1, the phase difference between the input and output signals of an ILO can be tuned by adjusting its center frequency. This phase tuning capability enables ILOs to serve as built-in “deskew buffers”, and conventional deskew architectures can be applied directly. For example, similar to active deskewing in conventional clocking, phase detectors can be placed between local clock domains to check skew, and then tune corresponding ILOs. Removing dedicated deskew buffers not only saves power, but also reduces their vulnerability to power supply noise. Note that ILC deskewing is different from the distributed PLL approach [52; 107], where phase detectors have to be added between all adjacent clock domains for frequency synchronization, and then possibly for deskew. In injection-locked clocking, frequency synchronization is achieved by injection locking, and the phase detection is used for deskew only. In other words, injection-locked clocking with deskew tuning is a dual-loop feedback system, and therefore provides both good tuning speed and small phase error.
Because of the excellent built-in deskew capability of ILOs, it can be expected that an injection-locked clock tree has much more freedom in its physical design (layout).

4.5.5 Jitter Reduction and Suppression

Injection-locked clocking can significantly reduce jitter in global clock distribution networks. First, reduced number of global clock buffers also means less sensitivity to power supply and substrate noise, and hence less jitter generation and accumulation (see Figure 4.8). Second, because of the design freedom in layout, the clock network can be placed where there is minimal noise coupling from adjacent circuits and interconnects. In addition, similar to a PLL, an ILO can suppress both its internal noise (low-pass filtering) and input signal noise (high-pass filtering). So it can possibly lower the input signal jitter at its output [53; 138]. Overall, injection-locked clocking is likely to achieve better jitter performance than conventional clocking.

![Figure 4.8: Jitter in the ILC design (b) is reduced compared to (a) as jitter-injecting components can be removed.](image)

4.5.6 Potential Applications

With the numerous technical advantages, a stand-alone ILC-based clocking network can be used to improve high-end microprocessors and the design process in many ways:

First, ILC reduces jitter and skew compared to a conventional clocking network. This reduces cycle time and therefore allows a faster clock speed. As technology scaling improves transistor performance but does not reduce jitter and skew (which actually increase), the improvement in clock speed will be more pronounced over time. Although further increasing
whole-chip clock speed finds limited practical appeal in today’s setting, it may still be effective in certain specialized engine inside a general-purpose architecture, such as high-speed interconnect backbones.

Second, using ILC, clock distribution for a multi-core system is a natural extension from a single-core system. A conventional clocking scheme would require adding chip-level PLLs. PLLs are bulky and particularly vulnerable to noise and hence usually placed at the very edge of a chip. In future multi-core systems, it represents a significant challenge to place PLLs and route high-speed clock signal to the destination cores. In contrast, in ILC, a single medium-speed global clock signal is distributed throughout the chip and locally, each core can multiply the frequency according to its need.

Third, even in a single-core architecture, different macro-blocks can run at different frequencies. This is referred to as the multiple clock domain (MCD) approach [64; 118]. The ILO can locally multiply (or divide) the frequency of the single global clock. One significant advantage of using ILC to enable multiple clock domains is that the local clocks have a well-defined relationship as they are all synchronized to the global clock. As a result, cross-domain communication can still be handled by synchronous logic without relying on asynchronous circuits. Note that although ILOs are not as flexible as PLLs in frequency multiplication, they are sufficient for MCD processors as only a few frequency gears are needed for practical use [147].

### 4.6 Case Study and Experimental Setup

#### 4.6.1 Case Study

![Figure 4.9: Illustration of the three different configurations of global clock distribution. Each configuration is designated according to its clocking network: XGM, IGM, and IM’.](image)

Figure 4.9: Illustration of the three different configurations of global clock distribution. Each configuration is designated according to its clocking network: XGM, IGM, and IM’.
This section will quantitatively review some benefits of injection locked clocking in a straightforward setting, a single-core processor running at a single clock frequency. As high energy consumption and the resulting heat dissipation issue become a dominant engineering challenge in high-end microprocessors, the energy benefit of using ILC becomes more pronounced, as shown in this case study. The experiments compare processors that only differ in the global clock distribution, some using conventional clocking and some using ILC. Due to the limited availability of detailed characterization of clocking network in the literature, the choice of the clocking network in ILC is limited and very closely resembles that of the baseline processor. Note that this is far from the optimal ILC design for the given processor, but is sufficient to demonstrate the significant benefit of ILC nonetheless.

The baseline processor is Alpha 21264, which has the most details in public domain on its clock distribution network \[11; 18\]. In this processor, an on-chip PLL drives an X-tree, which in turn drives a two-level clocking grid containing a global clock grid and several major clock grids. The major clock grids cover about 50% of the chip area and drive local clock chains in those portions. The remaining part of the chip is directly clocked by the global clock grid. The densities of the two levels of grids are different. This configuration is illustrated in Figure 4.9-(a).

In the first configuration using ILC, only the very top level of the clock network is replaced. Most of the buffers in the X-tree are removed and the final level of buffers (a total of 4) replaced with ILOs. The rest of the hierarchy remains unchanged (Figure 4.9-(b)). Note that in contrast to the Alpha implementation, only low-swing signals are sent on the global X-tree. Clearly, this reduces the energy consumption of the top level clock network. Furthermore, as discussed before, clock jitter and skew will also reduce, which can be converted into energy reduction by slightly reducing the supply voltage to capitalize on the timing slack.

While such a simple approach of using ILC as a drop-in replacement already reduces energy consumption, it is hardly exercising the power of ILC. As discussed before, numerous ILOs can be distributed around the chip to clock logic macro-blocks. Thanks to the built-in deskew capability, it is possible to avoid using power-hungry clock grids altogether. However, to faithfully model and compare different approaches, parameters (e.g., capacitance load of individual logic macroblocks) are necessary for circuit-level simulation which could not be found in the liter-
nature. As a compromise, in the second configuration, only the global clock grid is removed. Instead, we use a set of ILOs to directly feed one single level of grids. The reason this is not done in the Alpha is that to meet the skew and jitter target, a single-level grid would need higher density, as well as stronger and more numerous drivers that together consume far more power than the two-level approach [11]. In an ILC, however, the skew and jitter performance is much improved. With this configuration, the clock network load can be derived based on results reported in [11; 18] and technology files. Since the chip areas not covered by major clock grids are directly clocked from the global clock grid, this single level of grids consist of all the major clock grids and the portion of the global grid that directly feeds logic circuit (Figure 4.9-(c)).

To evaluate the benefits of injection-locked clocking, both circuit- and architecture-level simulations are done on the baseline processors with each clock distribution configuration in Figure 4.9. In order to reflect the state of the art, the global clock speed is scaled from 600MHz to 3GHz, and correspondingly the process technology from 0.35\( \mu \)m to 0.13\( \mu \)m. The validity of scaling is verified using Pentium 4 Northwood 3.0GHz processor as the reference.

### 4.6.2 Circuit Simulation Setup

ADS was used to evaluate power consumption and jitter performance of the clock distribution network with different configurations. The simulations are based on extracted models of the clock distribution networks, including buffer sizes, interconnect capacitance, and local clock load capacitance. Then the distribution network model was applied in the circuit simulation with ILOs and clock buffers constructed using SPICE models of transistors. The circuit model for the baseline chip (Figure 4.9-(a)) is shown in Figure 4.10-(a). Capacitance values for global and major grids are calculated based on the reported chip dimension, grid structure and grid density. They are then scaled to 0.13\( \mu \)m technology. Clock load was calculated based on its reported power consumption, and also scaled to 0.13\( \mu \)m technology because it represents the logic transistors. All buffer sizes are derived from the reported power consumption. The model for the IGM configuration (Figure 4.9-(b)) was different from Figure 4.10-(a) only in the first stage, where the buffered X-tree was replaced by a passive X-tree driving four ILOs. Similarly
the circuit model for IM’ configuration (Figure 4.9-(c)) was modified from IGM, by removing the global grid stage, and adjusting the capacitance of the remaining stages accordingly.

Since jitter is largely introduced by power supply and substrate noise through clock buffers, a noise voltage source with a Gaussian distribution is inserted to the power supply node, as shown in Figure 4.10-(b). Transient simulation was used to calculate the voltage and current waveforms along the clock distribution. Output clock waveform was analyzed statistically to get the distribution of the clock period. Jitter at the output was then calculated based on this distribution. Jitter was first measured in the baseline conventional clocking configuration, and the noise source amplitude was determined by matching measured jitter with reported value in [81], 35ps. The same noise voltage source was then used in the subsequent jitter simulation for the ILC configurations, and the results are compared to the baseline configuration. This approach is actually pessimistic considering the target jitter number (35ps) is among the lowest in conventional clocking reported [95]. The source jitter from on-chip PLL is represented using
a built-in ADS model of clock with jitter, and the clock jitter is chosen to be 5ps, which is consistent with jitter of on-chip PLLs published.

### 4.6.3 Architectural Simulation Setup

A modified version of SimpleScalar[21] toolset is used for architectural simulations, simulating the Alpha ISA, modeling a chip with one core. Both dynamic and leakage power were investigated in detail. Wattch [19] evaluates the dynamic energy component, and model the conventional clock tree in detail following the configuration of [11]. There is one important change to the simulator. When using Wattch’s built-in scaling to scale to the target technology point, the global clock power (24W) is much higher than reported from our circuit-based power analysis and scaling (9.17W). At the time of this writing, it is not possible to pin-point the reason for the apparently different scaling assumptions. To stay on the conservative side, Wattch’s clocking model is replaced with the circuit simulation-based results. This means global clock distribution accounts for only 23% of overall power, which is significantly lower than reported results [11; 60]. Therefore, the benefits of ILC reported in this work are likely to be very conservative.

Leakage power is temperature-dependent and was computed based on predictive SPICE circuit simulations for 0.13µm technology using BSIM3 [20]. Device parameters, such as $V_{th}$, are based on the 2001 International Technology Roadmap for Semiconductors and IBM 0.13µm CMOS technology file. Temperature (for leakage calculations) is modeled with HotSpot [121] using the floorplan of our modeled single-core processor, adapted from the floorplan of Alpha 21364.

Table 7.1 lists all the parameters for processor (based on Alpha 21264) along with process specifications. The quantitative analysis use highly-optimized Alpha binaries of all 26 applications from the SPEC CPU2000 benchmark suite. 100 million instructions are simulated after fast-forwarding one billion instructions.
Table 4.1: ILC system configuration.

<table>
<thead>
<tr>
<th>Processor core</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue/Decode/Commit width</td>
<td>4 / 4 / 4</td>
</tr>
<tr>
<td>Issue queue size</td>
<td>16 INT, 16 FP</td>
</tr>
<tr>
<td>Functional units</td>
<td>INT 4+1 mul/div, FP 1+2 mul/div</td>
</tr>
<tr>
<td>Branch predictor</td>
<td>Bimodal and Gshare combined</td>
</tr>
<tr>
<td>- Gshare</td>
<td>1K entries, 10 bit history</td>
</tr>
<tr>
<td>- Bimodal/Meta table/BTB</td>
<td>4K/4K/1K (2 way) entries</td>
</tr>
<tr>
<td>Branch misprediction penalty</td>
<td>7+ cycles</td>
</tr>
<tr>
<td>ROB/Register(INT,FP)</td>
<td>64/(80,72)</td>
</tr>
<tr>
<td>LSQ</td>
<td>32, 2 search ports, 1-cycle port occupancy, 2-cycle latency</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory hierarchy</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 instruction cache</td>
<td>64KB, 2-way, 32B line, 2 cycles</td>
</tr>
<tr>
<td>L1 data cache</td>
<td>64KB, 2-way, 32B line, 2 cycles</td>
</tr>
<tr>
<td>L2 unified cache</td>
<td>2MB, 4-way, 32B line 15 cycles</td>
</tr>
<tr>
<td>Memory access latency</td>
<td>250 cycles</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Process specifications</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Size / Frequency</td>
<td>0.13µm / 3GHz</td>
</tr>
<tr>
<td>$V_{dd} / V_t$</td>
<td>1.5V / 0.43V</td>
</tr>
</tbody>
</table>

4.7 Experimental Analysis

4.7.1 Jitter and Skew

In the circuit simulation, the PLL source jitter is set to 5ps, and the value of the added power supply noise source is chosen so that the output clock jitter for the baseline processor (Figure 4.9-(a)) is 35ps, as described in section 4. There are 30ps of jitter added along the clock distribution network, which comes from the power supply noise coupled through the buffers. For the clock speed of 3GHz, the overall jitter in the baseline processor therefore corresponds to 10.5% of the clock cycle. In the case of ILC with the IGM configuration (Figure 4.9-(b)), under the same power supply noise and source jitter, the output clock jitter is lowered to 15ps (a 57% reduction). This translates into recovering 6% of a clock cycle at 3GHz, a significant performance improvement. As described in Section 3.5, the jitter reduction can be attributed to the reduced number of clock buffers and good noise rejection of ILOs. When ILOs are used to directly drive the local clock grids without the global grid as in IM′ configuration (Figure 4.9-(c)), thanks to the further reduction in the buffer stages, jitter is lowered to 12ps, or 66% lower.
than the baseline. This clearly demonstrates that ILC can achieve better jitter performance than conventional clocking.

In the current study, it is assumed that built-in deskew capability of ILOs can reduce the skew to below 15ps, or 10ps savings in timing margin compared to the baseline processor (without any deskew). This estimate is consistent with the results using existing deskew schemes [95], and hence quite reasonable. In fact, ILC should lead to even lower skew as discussed in Section 3.4, which can be supported by a test chip measurement shown below.

**Test chip**

A test chip was designed and fabricated to verify the jitter reduction and deskew capability of ILC [144]. As shown in the schematic of the test chip (Figure 4.11-(a)), a 3-section H-tree mimics the global clock distribution network in real microprocessors, and the leaves of the H-tree are four divide-by-2 ILOs, which divide the input 10GHz clock signal into 5GHz local clocks. The differential outputs of ILOs then drive four open-drain differential amplifiers, which are directly connected to output RF pads for measurements. The differential divide-by-2 ILO used in the test chip is shown in (Figure 4.11-(b)). NMOS transistors biased in the inversion region are used as varactors to tune the ILO center frequency, which in turn changes the phase of the local clocks for deskewing purposes.

The deskew capability is measured in the test chip by tuning the control voltage $V_t$ of two ILOs. The result is shown in Figure 4.12. The whole deskew curve shows that a wide skew range of up to 80ps can be compensated by the built-in deskew capability. Because of the
continuous deskew characteristics of ILC, the deskew resolution of ILC depends on the skew measurement and control circuit. Under current skew measurement and control circuit the skew can be controlled at 7ps or less \cite{124}. Thus, the assumption of 15ps skew in an ILC system is valid.

4.7.2 Chip-Wide Power Impact of ILC

Baseline processor

The power consumption of the baseline processor ranges from 30.4W to 50.4W with an average of 40.7W. The power can be divided into three categories: global clock distribution power, leakage, and the dynamic power of the rest of the circuit. The breakdown of the power is shown in Figure 4.13. The global clock is unconditional and consumes 9.17W or about 23%.

ILC configurations

For IGM (Figure 4.9-(b)), power savings come from two factors. First, the power consumed in the top-level X-tree is reduced from 1.72 to 1.56W because the reduction of the total levels of buffers used and the lowered voltage swing on the X-tree. Second, as explained earlier, jitter and skew are both improved when using ILC: a 20ps reduction in jitter and 10ps in skew are achieved in this configuration. These total savings of 30ps increases the available cycle time.
for logic from 273ps to 303ps. This, in turn, allows a reduction in $V_{dd}$ without affecting the clock speed. The following voltage-delay equation from [115] was used to calculate the new $V_{dd}$, which is 1.415V, reduced from 1.5V.

$$t = \frac{C}{k'(W/L)(V_{dd} - V_t)} \left[ \frac{2V_t}{V_{dd} - V_t} + \ln \left( \frac{3V_{dd} - 4V_t}{V_{dd}} \right) \right]$$

The power reduction for the tested applications ranges from 3W to 5.2W with an average of 4.1W or 10.1%. The reduction is mainly due to the lowering of supply voltage. Considering the minimal change and the conservativeness in the entire evaluation process, the result is very encouraging.

Though using ILC as a drop-in replacement of the top-level clock distribution tree already shows notable power savings, as explained before, it is not fully exploiting the potential of the new clocking paradigm. The second ILC configuration, $I'M$ (Figure 4.9-(c)), further reduces clock distribution power by reducing the size of the grid. For $I'M$, the global clock power is reduced to 5.9W (from 9.17W in the baseline XGM) and the combined jitter and skew reduction is 33ps, which allows us to scale $V_{dd}$ to 1.41V. The overall effect is an average of 6.8W (17%) total power reduction. Compared to IGM, $I'M$ further reduces power by 2.7W or 7%.

![Figure 4.13: Breakdown of processor power consumption with different clock distribution methods.](image)

The results of using different clocking structures are summarized in Figure 4.13. In this comparison, all configurations achieve the same cycle time. The density of the grids and the driving capabilities are determined using circuit simulation. The design point where energy is minimized is chosen. For reference, it is necessary to also show the result of replacing the two
levels of grids by a single grid in the conventional configuration. Note that this grid is different from the \( M' \) grid as it needs higher density and larger buffers to achieve the same overall cycle time target. This grid is designated as \( G' \), and the configuration \( XG' \). The same methodology is used to compute its jitter performance, clocking load, and power consumption.

From the results, it is clear that ILC significantly improves power consumption. It is also clear that using a single-level grid per se is not the source of energy savings for \( IM' \): using a single grid in the conventional design leads to a significant 7.9W of extra power consumption.

Overall, it is clear that ILC can be introduced to a processor in various levels of ease. With minimum design intrusion, when only the very top level of the clock tree is modified to use injection locking, energy reduction is already significant (10%), thanks to the lowered jitter and skew. When the clock grid is further optimized, the power savings become more pronounced (17%). All these are achieved without affecting performance or the design methodology of the processor.

### 4.8 Summary

Thanks to the high sensitivity, good noise rejection, and built-in deskewing capability of injection-locked oscillators, the proposed injection-locked clocking scheme can significantly improve skew and jitter performance of a multi-GHz clock distribution network. Reduced number of clock buffers, and recovered timing margin from skew and jitter lead to substantial power savings for the whole processor. Initial results from circuit and architectural simulations confirmed the initial analysis. Further, the benefits of this new clocking scheme should be even greater when it is applied to high-performance multi-core microprocessors and other high-performance system-on-a-chip (SoC) systems. Also, as ILOs have the ability to lock onto harmonics of the injected frequency, high frequency clocks can be achieved without stressing the PLL and clocking network. This capability also allows for local ILOs to run at frequencies higher than the base frequency, independently of other ILOs on the network.

Injection locked clocking is certainly an important application of new fabrication and circuit technology and can stand alone. However, it is also an important enabling technology for high-
speed, on-chip communication. Chapters 6 and 7 will explore this concept in more detail, following a brief review of some background.
Chapter 5

Transmission Line Basics

Injection locked clocking provides reliable, high-speed clocks across a large chip, such as a chip multiprocessors. This, in turn, alleviates the synchronization burden (i.e., use of clock and data recovery circuits) of using a high-speed shared communication medium, in this case, a transmission line link-based bus. These buses can operate an order of magnitude faster than a typical core frequency, and thus ILC is necessary to ensure a quality clock network. In this chapter, we will review some of the basics of transmission line theory and operation, before using these high speed lines in a communication network.

Figure 5.1: Transmission lines as seen in day-to-day life, as power lines.

Transmission lines are, in the simplest terms, very long wires, the most commonly seen version connecting power plants to homes, as seen in Figure 5.1. In this case, we will discuss on-chip wires transmission lines. A wire, made of a conductive metal, carries current when presented with a voltage between the ends. Such a wire is considered “invisible” when the signal at any position on the wire is considered to have the same value at a given time instance [41;
Table 5.1: The criteria necessary to determine model type. $\lambda$ is the wavelength, $t_r$ is the rise time of the signal, and $t_t$ is the transmission latency of the signal \cite{41,104,106}. In other words, a traditional wire, modeled as an RC circuit (see Figure 5.2-(a)), is typically short enough to ignore long transient signals.

![Figure 5.2: The (a) RC and (b) RLC equivalent models of a wire. $R$ and $L$ are the series resistance and inductance per unit length and $G$ and $C$ are the shunt conductance and capacitance per unit length](image)

As we scale these wires to larger and larger dimensions, the propagation delay from end to end becomes non-negligible. Additionally, as the wires lengthen, it is often necessary to widen these lines to reduce the overall resistance of the line (wider lines, in general, result in less resistance). As the wires grow in width and length, the inductive effect of these lines becomes more dominant, especially at higher frequencies. This transition to higher frequencies and wider/longer lines has a two-fold effect. First, it requires a change in the model used by designers. If an RC model is used without the inductive effects, properties like reflections and bandwidth limitations could go unaccounted for. Table 5.1 reviews a set of criteria that distinguishes between the RC model of a wire, and the RLC model, shown in Figure 5.2-(b). To summarize the table, if the time to propagate along the line is significantly longer than the rise time of the signal, then it should be modeled as a transmission line, using an RLC circuit.

The aforementioned second effect of these wider/longer lines is as follows: now they are operating as traditional transmission lines, meaning that the speed of propagation through the wire is the speed-of-light (in a medium). The benefit of the increase in the inductive properties of the line is that rather than pushing a voltage and waiting for the signal to fully charge the...
line, the line can carry a high-quality electro-magnetic wave, which travels at the speed of light. Figure 5.3 illustrates this point. The signal speed of the transmission line is now dominated by the LC delays, which are small, instead of the RC delays, which are larger. As Beckmann and Wood succinctly described it, “The speed of the incident wave across a transmission line is analogous to the speed of a ripple moving across water in a bathtub, while the latency across conventional RC wires is analogous to changing the water level of the bathtub [13].” A transmission line can carry signals at much higher speeds, and thus make a better fit for long global interconnects.

An on-chip network depends not only on the speed of the interconnects signal, but just as importantly, relies on high throughput density. A packet-switched network with thin digital wires has high throughput density because the thinner wires allow for more wires. However, for longer wires, it is necessary for digital wires to insert repeaters, or suffer signal degradation and increasingly lower speeds. At a 5mm length, thinner wires provide high throughput without suffering in overall line bandwidth. At 75mm, the achievable throughput is significantly higher for wider lines. Figure 5.4 shows the maximum bandwidth (the 3-dB frequency of an individual line times the number of lines) and the achievable throughput (the Shannon-Hartley maximum throughput) for the number of lines in a given space. The Shannon-Hartley channel capacity theorem is:

\[ C = B \times \log_2(1 + S/N), \]

where \( C \) is the channel capacity, \( B \) is the bandwidth, \( S \) is the signal strength, and \( N \) is the strength of the noise.

In this example, the lines are coplanar strips (topologies are discussed further in Section 5.2) driven differentially, in a maximum aggregate width (sum of the pitch of all parallel lines) of 2.5mm. As the number of lines in the space increases, the available wire pitch per line decreases. For short wires, clearly, more lines directly correlates to more throughput, as the bandwidth of the lines remains high. However, for longer lines, the throughput peaks, suggesting that simply adding more lines in a given aggregate width no longer improves aggregate throughput.

Unfortunately, designing an on-chip transmission line is not as simple as taking a standard global wire and (a) widening it, or (b) changing the equivalent model. Properly designed transmission lines require a strong signal return path, isolation from crosstalk, bandwidth density considerations, and more. If we were to simply expand a standard wire, it would likely be a poor transmission line, having a weak return path (the general ground path). Chapter 6 will ex-
explore a targeted design space, in order to design on-chip transmission lines for the specific use as a globally shared-medium interconnect, and will provide more concrete examples of some of these concepts. The rest of this section will instead give a higher level overview of the design of on-chip transmission lines.

5.1 Transmission Line Properties

In this section, we will review transmission lines basic operation. Figure 5.2-(b) reviewed the circuit model of a transmission line. A more detailed view for voltage and current analysis is presented in Figure 5.5. Using Kirchhoff’s voltage and current laws, we can obtain the circuit model equations:

\[ v(x, t) - R \Delta x i(x, t) - L \Delta x \frac{di(x, t)}{dt} = v(x + \Delta x, t) \]

\[ i(x, t) - G \Delta x v(x + \Delta x, t) - C \Delta x \frac{dv(x + \Delta x, t)}{dt} = i(x + \Delta x, t) \]
Figure 5.4: The maximum bandwidth and achievable throughput of (a) 5mm and (b) 75mm lines.

By dividing these equations by $\Delta x$ and taking the limit as $\Delta x$ approaches 0, the remaining equations are as follows, first in the time domain, then in the frequency domain:

\begin{align*}
\frac{dv(x, t)}{dx} &= -Ri(x, t) - L \frac{di(x, t)}{dt} \quad (5.3) \\
\frac{di(x, t)}{dx} &= -Gv(x, t) - C \frac{dv(x, t)}{dt} \quad (5.4)
\end{align*}
Figure 5.5: An equivalent circuit model of a transmission line, including voltage and current definitions.

\[
\frac{dV(x)}{dz} = -(R + j\omega L)I(x)
\]

\[
\frac{dI(x)}{dz} = -(G + j\omega C)V(x)
\]

Equations (5.5) and (5.6) can be solved for \(V(x)\) and \(I(x)\) as:

\[
\frac{d^2V(x)}{dx^2} - ((R + j\omega L)(G + j\omega C))V(x) = 0
\]

\[
\frac{d^2I(x)}{dx^2} - ((R + j\omega L)(G + j\omega C))I(x) = 0
\]

resulting in the solutions:

\[
V(x) = V_0^+ e^{-\gamma x} + V_0^- e^{-\gamma x}
\]

\[
I(x) = I_0^+ e^{-\gamma x} + I_0^- e^{-\gamma x}
\]
where $\gamma = \sqrt{(R + j\omega L)(G + j\omega C)}$.

Using the solution in Equation (5.10) for Equation (5.6) results in the following:

\begin{equation}
I(x) = \frac{\gamma}{R + j\omega L} (V_0^+ e^{-\gamma x} + V_0^- e^{-\gamma x})
\end{equation}

Combining Equations (5.10) and (5.11), we can find the characteristic impedance of the line. This characteristic impedance becomes important in the overall design of the system, as this impedance is used to terminate the lines to ensure minimal reflection and maximum power transfer.

\begin{equation}
Z_0 = \sqrt{\frac{R + j\omega L}{G + j\omega C}}
\end{equation}

In a lossless model, $R$ and $G$ can be ignored, and only $L$ and $C$ need to be considered, and the equation reduces to $Z_0 = \sqrt{L/C}$.

**Reflections**

When a transmission line is terminated, the properties of the line versus those of the termination determine how the signal is absorbed and/or reflected. Figure 5.6 shows the 3 relevant cases of transmission line termination: open circuit, short circuit, and $Z_L$ (*i.e.*, any load of real and/or imaginary components).

![Figure 5.6: A transmission line terminated with (a) an open circuit, (b) a short circuit, and (c) a load impedance $Z_L$.](image)
Reflection is defined as the percentage of the incident wave that is not absorbed by the terminating load, and is typically represented by $\Gamma$.

\begin{equation}
\Gamma = \frac{V_{\text{reflected}}}{V_{\text{absorbed}}} = \frac{Z_L - Z_0}{Z_L + Z_0}.
\end{equation}

When the end of the line is an open circuit (i.e., no load termination at all), as in Figure 5.6-(a), $Z_L$ is infinite, resulting in a $\Gamma = 1$, meaning all of the wave’s energy is reflected. With a short circuit, as with Figure 5.6-(b), $\Gamma = -1$, meaning all of the wave’s energy is reflected and inverted.

The average power at the load is represented by:

\begin{equation}
P_{av} = \frac{1}{2} \frac{|V_0^+|}{Z_0} (1 - |\Gamma|^2),
\end{equation}

so to maximize the power transfer, it is necessary to set $\Gamma = 0$, or $Z_L = Z_0$. In this condition, the load is called a “matched” load. When designing the transmission lines in a real-world environment, matching the load of the on-chip transmission lines will minimize reflection and maximize the energy passed through the load. In many cases, the load is purely resistive, matching the real portion of the characteristic impedance, keeping reflection low without the real-world burden of capacitances and inductances being placed on-chip.

Before the electromagnetic (EM) waves reach the load, they must propagate along the transmission lines, and depending on the topologies and materials, the characteristics of such propagation changes. Here, we review the various propagating modes.

- Transverse EM waves: waves where neither the electric nor the magnetic field oscillate in the direction of the propagation. TEM mode is not only an easier mathematical model for analysis, but also results in less loss than non-TEM wave propagation.

- Transverse electric mode: the electric field waves do not oscillate in the direction of propagation, but the magnetic field waves do.
- **Transverse magnetic mode:** the magnetic field waves do not oscillate in the direction of propagation, but the electric field waves do.

- **Quasi-TEM mode:** EM fields oscillate mostly perpendicular to the direction of propagation, but there are small components that propagate in the same direction as oscillation. This is sometimes referred to as a “hybrid mode.” Quasi-TEM is a consequence of propagating in multiple mediums (like microstrips, described below).

### 5.2 Transmission Line Topologies

For the purposes of this thesis, we will focus only on on-chip transmission lines, ignoring off-chip technologies, such as coaxial cables. There are many common topologies of on-chip transmission line, with some being variants on a similar design. Figure 5.7 shows some of the most common in general terms [41]. Each topology, as would be expected, comes with advantages and disadvantages:

- **Stripline (Figure 5.7-(a)):** Because a stripline is symmetrical in both the horizontal and vertical directions, waves propagate entirely in TEM mode. A stripline is one of the oldest designs, as it closely resembles a coaxial cable (a common transmission line topology for non-on-chip communication). Striplines require 3 levels of metal, which makes the layout less attractive. A similar topology uses differential lines (with no ground plane), resulting in good isolation from neighboring lines (often called *embedded differential lines*).

- **Embedded Differential Lines (Figure 5.7-(b):** Embedded differential lines have similar characteristics to a stripline with one important difference: the use of differential signaling significantly reduces crosstalk. Crosstalk is the influence neighboring lines’ signals have on a line, or pair of lines. Crosstalk is a large source of noise in on-chip transmission lines, and thus is often an important design constraint. Differential signals, for example transmitting on a pair of embedded differential lines, have minimal crosstalk. Additionally, the lack of a ground plane simplifies the layout (only needs one metal layer surrounded by dielectric, rather than multiple metal layers).
Figure 5.7: Common transmission line topologies, along with a legend of materials. Ground lines are also marked.

- Microstrip (Figure 5.7-(c)): Microstrips are simple in terms of geometric layout, but the quasi-TEM behavior makes modeling more difficult and increases loss. As frequency increases, the loss increases significantly, while noise influence from neighboring lines also increases. This limits the overall bandwidth of microstrips, as the signal-to-noise ratio (SNR) quickly decreases, resulting in either errors or the necessity of more complex and power-hungry transceiver designs.

- Coplanar Waveguide (Figure 5.7-(d)): The above topologies all rely on multiple levels of metal to provide a return path. Coplanar waveguides, often abbreviated as CPWs, on the other hand require only a single layer of metal. The grounding plane being interleaved between signal lines provides a strong return path and isolation between signal lines. CPW provides better loss characteristics and lower fabrication costs than microstrip and
are a more attractive solution for on-chip transmission lines. Similar to the other planar designs, CPW behaves in a quasi-TEM mode, as do coplanar strips.

- Coplanar Strips (Figure 5.7(e)): One of the main disadvantages of each of the above geometries is that each is driven single-endedly (when used as an interconnect). Coplanar strips (or CPS) are made up of 2 adjacent lines of the same dimensions on the same metal layer, with no other paths. These lines can be used in a single-ended manner, grounding one line, essentially creating a specific geometry of coplanar waveguides. However, in terms of crosstalk isolation, it may be more effective to drive these lines differentially. Chapter 6 will illustrate this in greater detail. CPS have a strong return path for long lengths, making them a particular favorable choice for long lines.
Chapter 6

Transmission Line Design Study

One challenge to continued scaling of chip multiprocessors is the ability to provide efficient communication between cores. Injection-locked clocking provides a globally synchronous clock to all points on the chip in a power-efficient manner. Reliable and synchronous clocking support lays a solid foundation for high-speed synchronous communication.

6.1 Transmission Line Overview

As the number of cores integrated into a single chip steadily increases, an important component in chip multiprocessors (CMPs) is the on-chip interconnect. For a number of reasons, packet-switched interconnect is often accepted as the de facto solution [44; 85]. A packet switched network offers numerous advantages such as throughput scalability and modularity. However, it is not without drawbacks. Routers are complex structures that occupy significant chip real-estate and consume significant power [94]. Repeated packet relaying adds latency to communication and can be an important performance issue, especially for simpler topologies with large network diameters such as ring or mesh. These disadvantages are upfront costs paid even when the applications do no need scalable throughput. As such, alternative architectures should be explored. Transmission line based interconnects are a promising candidate.

A transmission line (TL) allows high signaling rate, speed-of-light propagation velocity and can potentially provide sufficient throughput for a range of CMPs such that packet relaying can
be avoided altogether. TL-based designs have already been used in numerous ways including in the context of microprocessors, but the specific design used is often studied and described in an ad-hoc fashion. A TL link has a large degree of freedom in designing the channel medium, the coding scheme, and the circuitry in the signaling chain and offers a vast range of trade-offs between costs and benefits. There is a lack of comprehensive design space studies to help architects navigate the design space and make optimal system-wide trade-offs.

This chapter takes a first-step effort exploring the design space of TL circuitry. As illustrated in Figure 6.1, this design space can be roughly broken down into three regions based on the transceiver circuitry. It is worth noting that future work will almost certainly push the envelope of all designs, despite efforts to evaluate optimal designs from each category.

![Figure 6.1: Illustration of transmission line link system design space.](image)

### 6.2 Physical Design

With ever improving transistor performance, a communication system can achieve a data rate of tens of Gb/s per line and an aggregate data rate of Tb/s over on-chip global transmission lines. In medium-sized CMPs, the global network connecting different cores can be entirely based on a multi-drop transmission line system (illustrated in Figure 6.2 and Figure 6.8) allowing packet-switching-free communication that is both energy-efficient and low-latency. This chapter focuses on circuit- and system-level analyses in such a context. Clearly, transmission lines can be used in other ways in the on-chip interconnect.
From the system’s perspective, a channel’s latency, throughput, and energy efficiency are of primary interest. In a transmission-line channel, the signal propagation latency is largely determined by the length of the line, as the propagation velocity is simply the speed of light in the medium \( c/\sqrt{\mu_r \varepsilon_r} \), which is roughly 6ps/mm for CMOS technologies where \( \varepsilon_r = 3.0 \) is assumed, and will likely decrease over time as low-K dielectric materials improve. Modern CMP dies are relatively stable in dimensions (about 2cm on each side). A multi-drop transmission line loop meandering through a 16-tile CMP therefore measures about 75mm in length, as in Figure 6.2, and a corresponding worst-case propagation delay of about 440ps. If a closed loop is used, the worst-case distance and delay becomes 40mm and 235ps, respectively. Transceiver circuitry will also add some delay. Nevertheless, the overall transmission latency is only a few cycles even for multi-GHz cores. As such, channel throughput is the key speed metric and can impact the serialization and queuing delay of the packet latency. Channel throughput and energy per bit in turn depend on the transmission line physical properties, as well as the transceiver circuitry.

![Figure 6.2: Top-level view of 16-core interconnect layout. The solid line is a physical bi-directional ring, and the dotted line is a bi-directional terminated bus.](image-url)
6.2.1 Transmission Line Topology

While there are more transmission line structures, the most common ones for on-chip interconnect are microstrip lines (MSL), coplanar waveguides (CPW) and coplanar strips (CPS). The latter two have similar characteristics, and CPS lines lead to higher interconnect density. Hence microstrips and CPS lines are the focus in this work. Figure 6.3 shows a cross section of each and the main parameters in their physical design. Microstrip lines are often chosen for their simplicity and are typically used with pure digital transmitters and receivers (inverters). In contrast, coplanar strips, paired with differential signaling provide extra robustness.

![Figure 6.3: Cross-section of (a) microstrip lines and (b) coplanar strips. The dotted lines in (a) represent inter-digitated MSL.](image)

6.2.2 Attenuation & Crosstalk

To understand the characteristics of the channel formed by these different transmission lines in isolation, it is necessary to idealize the active circuitry and estimate the maximum channel throughput (bit-rate) purely based on the characteristics of the lines. This is performed using a pair of industrial grade simulators. Sonnet [1] is used to obtain S-parameter profiles given the transmission line dimensions; and Advanced Design System (ADS) is used to take the resulting attenuation and crosstalk characteristics into account and perform transient analyses to estimate achievable data rate. All simulations were done using noisy environments, including aggressor lines to simulate crosstalk between neighboring lines.

Given the same pitch size (W+G in Figure 6.3), varying the gap and spacing yields different attenuation and crosstalk. Sweeping through the space to identify the optimal metal strip width and necessary spacing in each configuration (MSL or CPS) helps put these sizing decisions into broader context. The results are plotted in Figures 6.4 and Figure 6.5.
Figure 6.4: Per-line bit rate as a function of wire pitch. I-MSL, or inter-digitated microstrip lines, is similar to CPW where the ground line is the same width as the signal line. The I-MSL is essentially a MSL design where and identical line is placed halfway in between signal lines and grounded.

Figure 6.5: Crosstalk as a function of wire pitch. I-MSL, or inter-digitated microstrip lines, is similar to CPW where the ground line is the same width as the signal line.

Clearly, as the pitch size increases, crosstalk lowers for both configurations. However, crosstalk remains high for MSL in absolute terms. In contrast, CPS is subject to much less crosstalk, thanks to the differential signaling. Without the cost of running a pair of differential strips, MSL potentially provides good throughput at the low end of the pitch scale (< 25µm), but the throughput saturates very fast. This saturation is mainly due to crosstalk. For illustration,
Figure 6.6: Aggregate bit-rate as a function of the number of lines in a 2.5mm space. I-MSL, or inter-digitated microstrip lines, is similar to CPW where the ground line is the same width as the signal line.

the maximum throughput of MSL without crosstalk is also plotted. This is done by not injecting any signals as noise sources. As we can see, the difference is significant: with crosstalk, the maximum capacity drops from about 60Gb/s to about 20Gb/s.

One simple approach to reduce crosstalk is to use an inter-digitated organization of the strips, alternating signal lines and ground lines that provide some shielding.\(^1\) Figures 6.4 and 6.5 suggest that I-MSL offers less protection against crosstalk and a somewhat lower throughput than CPS. CPS is chosen for this work, in order to narrow the search.

### 6.2.3 Aggregate Throughput

Intuitively, wider metal strips (which lower attenuation) and larger spacing (which lowers crosstalk) both help improve single-channel throughput, but not necessarily throughput density. Since practical transmission lines are already much wider than typical digital (RC) wires, optimal use of metal space is important.

In Figure 6.6, the total pitch of all transmission lines is limited and the number of lines is varied to obtain the aggregate throughput of the system. Assuming a 2cm×2cm CMP divided

\(^1\)Compared to the more generic notion of Co-Planar Waveguide (CPW) in which the width of the shielding line and its distance to a signal line are free variables, the inter-digitated organization places a shielding line equal in width to the signal line equal-distance to the two neighboring lines.
into sixteen 5mm $\times$ 5mm tiles, the total width can be limited to 2.5mm, or half of the tile’s width. Note that this is a rather arbitrary limit and not a fundamental constraint.

As we can see, the bandwidth peaks at about 60 lines for both configurations and CPS offers a maximum of 1.9 Tbps aggregate throughput. This is a substantial amount of raw bandwidth. It is entirely conceivable that a medium-scale CMPs relies only on transmission lines to provide a shared-medium global interconnect. It is worth noting that when the transmission circuitry is taken into account, the actual throughput can change in either direction: slower transistors can limit throughput, and equalization circuitry can compensate for the channel bandwidth limitation. The optimal number of lines, as a result, can also change.

We have also taped out a test chip (Figure 6.7) to further validate the circuit parameters obtained from our circuit simulations.

![Figure 6.7: Test chip of transmission line links.](image)

### 6.3 Circuit Design

#### 6.3.1 Transmitter & Receiver

The transmission circuitry design space is equally vast and unlikely to be explored exhaustively in one iteration. This work focuses on designs that are relatively simple and can be easily integrated with digital CMOS circuits. Note that transmission circuit design is not orthogonal to
the design of the physical line. For instance, differential signaling naturally pairs with coplanar strips.

Figure 6.8 shows the general schematic of a single transmission link (surrounded by neighboring links) with transmission circuits. In general, the transmission circuit can be as simple as inverter-chain based fully digital circuits and as it becomes more sophisticated, it allows faster data rates at generally reduced per bit energy costs.

![General schematic for the transmission line link interconnect.](image)

**Figure 6.8:** General schematic for the transmission line link interconnect.

**Digital**

Probably the simplest design is to use a chain of (large) inverters (Figure 6.9) to drive the TL (microstrip) “strongly” so that the attenuated signal still arrives at the receiver discernible by the same style of inverter chain (albeit with smaller sizes to reduce the load on the TL). Even with this simple link design, transmission lines can achieve a transmission rate of 10Gb/s over a 40mm TL. Unfortunately, when the line is used as a multi-drop medium and when other circuit elements are included in the simulation, the signal degradation is so severe that the system no longer works regardless of transistor sizing. A simple remedy is to repeat the transmitter at each node. Such repeated TL becomes uni-directional and adds significant gate delays on top of propagation delay. Indeed, the gate delay, at 30ps (Tables 6.3.1 and 6.2), is comparable to propagation delay for each segment of the TL, and thus doubles the total latency. Note that at about 5mm apart, the repeaters are inserted far more sparsely than in typical digital wires.
Propagation

Single Segment: 28.9 ps; Round-trip: 461.9 ps

Line Dimensions

56 lines, 45µm pitch; Length: 5mm per segment

Table 6.1: Transmission line characteristics.

<table>
<thead>
<tr>
<th>Component</th>
<th>Transmitter Side</th>
<th>Receiver Side</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bit-Rate (Gb/s)</td>
<td>Power (mW)</td>
<td>Latency (ps)</td>
</tr>
<tr>
<td>Digital</td>
<td>10</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>Mixed</td>
<td>17</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>Differential</td>
<td>26.5</td>
<td>3.1</td>
<td>22</td>
</tr>
<tr>
<td>Latched Sampler</td>
<td>26.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SERDES</td>
<td>-</td>
<td>1.6</td>
<td>750</td>
</tr>
<tr>
<td>PDR</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6.2: Transceiver characteristics. Note that in the digital configuration, the transmitter latency is incurred every hop. The SERDES results are based on the fastest data rate (from analog transmission circuit).

Mixed

The limitation of an all-digital link is that the signal at the receiver needs to maintain full swing. An analog receiver using current source amplifiers (see Figure 6.10) obviates the need of a full-swing signal and allows two benefits: First, the transmitter area and power can be decreased substantially. Second, the more forgiving receiver allows a faster bit rate. As Table 6.2 describes, mixed signal design increases the per-line bit-rate by 70%, from 10 to 17 Gb/s. The energy/power of the mixed signal design is higher for a single hop when compared to the fully digital design. However, as the digital design potentially requires adding repeaters, the energy per bit increases. The mixed signal transceiver does not require buffers and thus the average energy per bit is significantly lower. The area and latencies are slightly larger, but similar to the digital design.

Differential

Finally, the transmitter can adopt (analog) differential signaling over coplanar strips (Figure 6.11). A standard CMOS differential amplifier is used in this design. No special RF devices, like inductors, are used for better integration. The receiver is a chain of differential amplifiers scaled using inverse scaling [112], allowing for high bandwidth and low power. The differential amplifiers are gated, and can be turned off when inactive, saving power/energy.
Differential signaling offers much better rejection of noise and permits faster data rate and lower power on the transmitter side. On the other hand, the receiver needs more amplification stages, which add more area and power. Nevertheless the overall per-bit energy is low (Table 6.2). In this case, 5 stages were chosen somewhat arbitrarily. A range of 3 to 5 stages of amplifiers were each tested, with 5 stages providing the best performance at 26.5\( Gb/s \). Further exploration of this receiver would likely improve both performance and energy, so estimates provided here will be conservative. This receiver generally uses inverse scaling in order to maintain the high bandwidth. Sackinger and Fischer used inverse scaling to provide a high bandwidth [112]. The input capacitance of a stage directly impacts the bandwidth of the previous stage. By decreasing the size of successive amplifier stages, we increase the gain across the spectrum, improving the maximum output of the receiver. In other words, by decreasing the load that an amplifier stage needs to drive (i.e., the next amplifier stage), the performance of the driving stage is improved.

One alternative to the chain of amplifiers is current-mode logic (CML) latched sampler, similar to the one presented in [25]. As shown in Figure 6.11, the latched sampler uses a cross-coupled latch immediately after a differential amplifier, which resulted in economy of circuit
Figure 6.11: A fully analog system, using differential amplifiers for the transmitter and either a chain of inversely scaled differential amplifiers or a differential amplifier and current-mode latched sampler for the receiver. “S2D” is serial-to-differential circuit, “D2S” is differential-to-serial circuit, which accounts for the transitions between serial signals from the core to differential signals in the CPS transmission lines.

and still permits high data rate. Depending on the number of latches used, this circuit can subsume some of the deserialization functionality. In the extreme case, enough latches can be used to obviate any deserialization, greatly shortening the latency at some power cost. A latched sampler does require low-skew clocks, provided by circuit technologies such as injection locked clocking [143].

6.3.2 SerDes & PDR

Faster transistor speeds in modern and future generation CMOS technologies are an important contributor to the performance of a transmission line link bus (TLLB). On-chip TLL-based interconnect will operate at many times the core frequency, making serialization and deserialization (SerDes) necessary. Typically, multiple stages of 2:1 MUX/DEMUX are used as SerDes. Figures 6.12 and 6.13 illustrate these (de)serializers and (de)multiplexers. These are designed using high-speed digital circuits but still introduce non-trivial delays as the simulations show (Table 6.2).

Clock and data recovery (CDR) is another necessary component to ensure the transmitters and receivers can properly communicate, and is independent of transceiver design: After a distance-dependent propagation delay, the transmitted pulses do not align with the receiver’s
Figure 6.12: (a) A serializer diagram for 8-to-1 bit serialization, with (b) showing detail of the 2-to-1 multiplexer.

Figure 6.13: (a) A serializer diagram for 1-to-8 bit serialization, with (b) showing detail of the 1-to-2 multiplexer.

clock. In a conventional system, clock and data recovery (CDR) is necessary, as a globally synchronous clock is difficult to build. CDR often requires high power and complex circuits to locally synchronize the clock. However, with the ILC scheme developed in Chapter 4, the CDR
is replaced by a phase and data recovery circuit (PDR). The magnitude of phase delta depends on the sender and can be quickly determined by sending and receiving a short test sequence in an initial calibration step. Data recovery circuit use the clock with the modified phase to ensure correct latching. PDR circuits tend to be simpler than CDR circuits. Figures 6.14 and 6.15 demonstrate an example of each type of circuit [9].

![Diagram](image)

Figure 6.14: (a) A block diagram for a clock and data recovery system. (b) A block diagram of a PLL, often used for a clock recovery circuit.

Because a clock recovery circuit requires complex circuits, such as PLLs, the energy overhead and complexity of the entire CDR is high. When the ILC scheme is introduced, local clocks are globally synchronized, effectively replacing the per-node PLL, the need for a full-fledged CDR is removed. Instead, a simple delay line can be used. The delay line is made up of delay stages, in this case, CMOS inverters, which delay the signal by some time interval. A training sequence, run once, calibrates the amount by which each node must shift its phase to align the global data and local clock edges. This is a one-time cost, and not on the critical path. The delay line is a simpler, more energy-efficient method of data recover, but does require a globally synchronous clock, such as the one provided by the injection-locked clocking scheme proposed in Chapter 4.5.
6.3.3 Isolation Switch

Because of the large metal area required to route TLLs, it is necessary to share the lines among nodes. To prevent excessive loss and limit noise of inactive nodes, a switch is needed between the transceiver circuit and the transmission line tap.\footnote{Such a switch is also used in wireless systems to allow transmitter and receiver to time-share the antenna and is referred to as the T/R switch \cite{65}.} When the switch is on, it must allow the signal to pass through with low loss and low distortion. When off, the switch must allow very little energy to be passed through in either direction. In 32nm technology, both of these goals can be accomplished reasonably well using a standard CMOS pass-gate structure. Additionally, the receivers and transmitters are power gated when not in use.

6.3.4 Signal draining

When the link forms a ring, we need to ensure that that a signal does not traverse the loop and overlap with a subsequent packet causing interference. This is achieved by having nodes that

---

Figure 6.15: (a) A block diagram for the phase and data recovery system. This system requires a training period to set the phase select decision circuit. (b) An example of a delay line using a multiplexer and a line of inverters.
are outside the shorter path between the transmitter and the receiver turn their receiver to drain mode during transmission. In drain mode the isolation switch is turned on to siphon energy from the transmission line. The amplifiers, however, are turned off, since the information is useless. At each node, impedance tuning is done to minimize reflection. As a result, when the isolation switch is on, the node absorbs 50% of the energy propagated thus far. In other words, after 6 draining nodes, the signal energy in the transmission line is reduced to about 1% of the original signal, no longer a significant source noise. Furthermore, an extra cycle is padded to every transmission to allow the previous packet to drain out before starting the next packet.

For the line architecture, rather than the ring, no signal draining mode is necessary as the ends of the lines have matched terminations. This is done to absorb signals as they reach the ends of the bus, with maximum power transfer and minimal reflection. The extra cycle of padding is still necessary to ensure signals do not interfere, but all the receivers that are not involved with the transmission are simply turned off. The bus/line topology will be assumed for the remainder of the work, unless otherwise specified.

### 6.4 System-Level Impact of Transmission Line Links

In order to understand the ultimate impact at the system level, the multi-drop transmission line links discussed above can be used to build a bus-like global interconnect for a CMP. Compared to a conventional packet-switched interconnect, such a transmission-line link bus (TLLB) does not have packet relay or routing. But unlike a conventional bus (with implied broadcast capabilities), different nodes on a TLLB merely share the same transmission line medium for point-to-point communication. Such a bus needs a few architectural elements to function. More in-depth discussion of the architectural design of TLLB can be found in Section 7.2 and [24].

#### 6.4.1 Architecture Design

#### 6.4.2 Architectural Comparison of Transceivers

In order to make an informed decision on which transceiver topology to use, we will review some architectural results, using each of the designs. The digital transceiver, as discussed in
Table 6.2, can operate at a maximum of 10Gb/s. The mixed signal transceiver operates at 17Gb/s, and the differential designs can operate at 26.5Gb/s. Figure 6.16 shows the results of an architectural analysis, comparing the use of each of these transceivers in a 16-core chip multiprocessor. The baseline in this case is the digital transceiver As the figure shows, using the slower transceivers results in performance degradation for the total system. While for many applications the difference is minute, for a few benchmarks, using the slower bit-rate drastically increases transmission time, and thus, queuing delay. The mixed signal transceiver closes the gap between digital and full analog, but still leaves room for improvement. Switching the analog receiver from differential amplifiers to CML latches increases performance slightly, as deserialization stages can be reduced or even removed. On average, the mixed signal transceiver performs 1.08x faster than digital, and the analog transceivers perform 1.15x and 1.17x faster than the digital transceiver. It is worth noting that changing the number of lines or allocation of given lines may have a significant effect on the performance of any of these systems, but for simplicity, the same layout is used for each.

Figure 6.16: The per-application breakdown of different transceiver designs. The digital components are used as a baseline.

Of course, system performance is not the only factor for which the designer must account. Energy and area also play an important role. Figure 6.17 demonstrates the overall average comparison of the transceiver design, in terms of area, average energy-per-bit, power, total energy, and total system performance. Note that the SERDES is simplified in the digital and mixed designs, as it requires less serialization stages. As with Figure 6.16, the digital transceiver is used
as a baseline. The systems are listed in order of total system performance, as well as transceiver complexity. Overall, the analog transceivers are larger, but operate with lower overall energy and power, as well as a decrease in system delay and energy. The difference between the 2 analog receivers is minimal for the total system (around 2% in terms of delay and energy).

Figure 6.17: A comparison of each transceiver type’s area, energy-per-bit, power, system energy, system delay. The digital transceiver is used as a baseline.
Transmitton Line Bus Architecture

Analysis shows that with straightforward optimizations, the traffic between different cores can be kept relatively low. This in turn allows simple shared-medium interconnects to be built using the communication circuits and transmission lines developed in Chapter 6. This architecture offers extremely low latencies and can support a large number of cores without the need for packet switching, eliminating costly routers.

7.1 On-Chip Network Overview

Mainstream microprocessors already include a handful of high-performance cores in each chip. As the scale increases, a natural component in these chip-multiprocessors is a high-performance on-chip interconnect. While conventional designs (reviewed in Chapter 2) used in multiprocessors are valid candidates, the on-chip environment offers a different set of constraints and opportunities for new designs and optimizations.

Because of the relatively limited market, traditional parallel machines often use commercial, off-the-shelf components such as microprocessors, chip-sets, and routers. They also use scalable designs that can be configured (in the field) for different sizes [7; 83]. Packet-switched networks fit the bill for the interconnect needs: existing routers can be directly used in multiple system designs and configurable routing tables allow easy customization for a scalable interconnect.
With chip-multiprocessors, each implementation only needs to deal with a fixed configuration. Thus, there is room for niche designs that do not scale to large configurations. Furthermore, future general-purpose chips are not necessarily destined to all be “many-core” designs. Increasing core count can lead to diminishing returns of utility. Extra transistor budget can find fruitful deployment in storage, specialized accelerators, and continued integration of traditionally discrete system components. Even given a large number of cores built in a chip, the actual communication demand is not automatically high. Consider a fairly common use of a multi-core chip: as a throughput engine processing largely independent tasks. There is little inherent need for inter-core communication.

In short, not all general-purpose chips need high, scalable on-chip communication bandwidths. And providing scalable bandwidths is not free. Recent studies have warned about the potential costs of powerful interconnects [80] and argued for the continued use of (optimized) digital buses with extensions of limited packet-switching for better scalability [128].

Considering these factors, there may be candidate designs for on-chip interconnect that trade off high bandwidth and scalability (and the associated cost) for improvements in other metrics of interest, such as latency and energy efficiency. As the on-chip interconnect offers a large design space and is related to many different aspects of chip-multiprocessor design, navigating the design space is an inherently imprecise process that requires iterative, community efforts. This chapter attempts to present a case for an interconnect that is free of packet switching/relay, explicit or implicit. Instead of packet switching, a shared medium can be used as a communication channel. In particular, this allows us to take advantage of communication circuit technologies driving on-chip transmission lines. Transmission lines can provide tens of Gb/s of signaling rate and a propagation velocity approaching the speed of light, all at an energy cost that is a fraction of a conventional packet-switching interconnect. It is the goal of this chapter to provide evidence that such a design would be a serious option for general-purpose chips.

7.2 Globally Shared-Medium On-Chip Interconnect

As microarchitecture becomes more complex, data communication occurs more and more frequently and is becoming increasingly explicit. Earlier systems used wires to carry data from
one logical unit to another without drawing designers’ attention. Gradually, wire delays were accounted for and pipeline stages were added explicitly to carry data. The difficulties in routing wires have long prompted researchers to call for a more general-purpose communication substrate to carry standardized, containerized payloads (packets/flits) as opposed to provisioning ad hoc data passages. This argument is aptly summarized into the slogan “route packets, not wires” [37; 117]. As chips integrate more and more cores, packet-switched interconnect seems to be accepted by many as the default solution for inter-core fabric.

This model has many appealing aspects. A common fabric provides an economical way of supporting different pathways between connected entities. Different types of payloads reuse the same passageway. However, there are limitations to packet-switching as well. Every stop the payload goes through has non-trivial handling that adds latency and energy overhead and demands significant hardware router infrastructure. Packetization also adds overhead at the source and destination that only becomes negligible when the distance traveled is sufficiently long. Indeed, in existing chip-multiprocessor designs, on-chip packet-switched fabric only serves as the backbone network connecting multiple “nodes”. Within a node, a variety of fabrics, such as crossbars and point-to-point links, are used to connect components. As such, how many nodes are being connected and how much traffic is there between these nodes are important factors to decide interconnect design. For certain chips, a more suitable implementation of the backbone may be a shared-medium system that delivers sufficient bandwidth without the delay and energy overhead of packet relays. In particular, transmission lines and appropriate transceiver circuitry provide the opportunity to build high-bandwidth interconnect without packet switching or relay.

7.2.1 Transmission Line Link Design Space

In this chapter, coplanar strips are used, as they utilize the space of the top metal layer more efficiently than the microstrips or coplanar waveguides (see Chapter 6; basic differential transmitters and receivers, scaled inversely, are also used without any equalization [112]. To see if such a setup is sufficient for building a backbone interconnect, some simple assumptions about dimensions must be made. Assuming a 2cm×2cm chip divided into 16 nodes, the longest dis-
tance between any 2 nodes on a single bus meandering through all nodes would be about 7.5 cm long.

Figure 7.1: Link diagram with differential transmitter and receiver. D2S is a differential-to-single-ended circuit, converting the differential signals to a full swing digital signal.

Figure 7.1 shows a schematic of the differential-pair design point of a transmission-line link (TLL). Simulations show that a data rate of 26.4Gb/s can be achieved (Section 6) for a pair of transmission lines with a total pitch (including spacing) of 45µm. A simulated transmission system shown in Figure 7.2-(a) was evaluated at that frequency in a realistic and noisy environment, and the signal waveforms are shown in Figure 7.2-(b). In this setup, the signal-line pair being tested is surrounded by other lines as crosstalk noise sources. A voltage noise source is also introduced at the differential amplifiers’ supply voltage.

Within 2.5mm of space, this pitch allows up to 55 pairs to be laid out, allowing 1.45Tb/s of total bandwidth (we will use 45 pairs, or 1.2Tb/s). The total area of active circuitry associated with each pair of transmission lines at each node is about 1,200µm². In a 16-node chip-multiprocessor, the total combined area for active circuitry is less than 1mm², or about 0.2% of total chip area. Depending on various factors impacting traffic demand, a straightforward backbone interconnect based on transmission line links can be a good design option for general-purpose chip multiprocessors.
Figure 7.2: Measurement setup and waveforms of transmission line circuit simulations. A, B, and C (as labeled in the design diagram shown in Figure 7.1) correspond to transmitter output, receiver amplifier input, and final latch output, respectively.
7.2.2 Traffic Demand

Node structure

In conventional multiprocessors, the packaging of the microprocessor dictates the boundary of nodes to be interconnected. A processor core is packaged with its local caches into a chip. Therefore, any traffic between the core and the on-chip caches does not go through the interconnect. Sometimes, multiple chips are part of one node and share the same router.

With chip-multiprocessors, there is more flexibility to determine what on-chip communication uses the packetized interconnect. A baseline assumption often made in literature is that a chip consists of tiles, each with a core, a L1 cache, and a slice of a globally shared L2 (last-level) cache. In such a system, if a L1 miss occurs, the access will result in a packet injected into the interconnect if the address maps to a remote node. Otherwise, the L1 miss is served by the local L2 bank. Intuitively, it pays to have a direct interface between the L1 and its local L2 slice, rather than forcing the local request to also go through the router. This avoids adding a whole sequence of unnecessary overheads – packetizing/depacketizing and router pipeline delay – for local traffic. And in certain cases (e.g., single-threaded applications when the data is mapped locally to the L2 bank) almost all traffic is local. In short, it is worthwhile to add an ad hoc channel between a core’s L1 and its local L2 slice.

In a similar vein, there may be a benefit to clustering a small number of cores and L2 slices into a node (and concentrate interconnect demand). In such a system, the backbone network only makes a stop at every node. This organization of cores requires an intra-node fabric (e.g., crossbar) that connects multiple L1 caches and the L2 cache banks in the node.

Figure 7.3: Node structures: (a) single-core nodes; and (b) clustering two cores into a node on the interconnect.
In terms of performance, clustering adds extra latency for accesses from a L1 cache to the nearest L2 bank (e.g., Figure 7.3-(b) Core0 to L2_0) that would otherwise have been connected with a smaller intra-node fabric (Figure 7.3-(a)). However, it makes accessing neighboring cache banks within the node (e.g., Figure 7.3-(b) Core1 to L2_0) somewhat faster than without clustering, as packetization is avoided. Moreover, it reduces the number of hubs a long-distance packet needs to traverse through and reduces the amount of traffic on the inter-node fabric. The extra cost of a larger intra-node fabric offsets the savings due to a lower number (and thus cost) of hubs for inter-node fabric. As Kumar et al. pointed out, sharing more than a few ways with crossbars quickly becomes prohibitive in cost [80]. Intuitively, the sweet spot would be a small number of cores per node (e.g., 1-4). Even a small degree of clustering will reduce the number of nodes and the total traffic on the backbone.

**Minimizing horizontal traffic**

To sustain high-speed processing, each core demands sufficient “vertical” bandwidth to fetch data from lower levels in the memory hierarchy all the way up to the core. Ideally, this vertical bandwidth is being provided by dedicated links between different levels of caches in the core’s node. However, depending on the address mapping, the data may be physically located on a cache in a remote node, incurring demand for “horizontal” bandwidth. Much research has been done to optimize the location of data to avoid unnecessary horizontal traffic. For instance, data can be mapped either statically or dynamically to the node where it is most often accessed or migrated there at run-time [10; 33; 70]. Such optimizations are important in their own right and will, as a side effect, significantly reduce the demand on the backbone, further strengthening the appeal of shared-medium, relay-free solutions.

**On-chip accelerators**

As the transistor budget keeps increasing while the power budget continues to be tight, on-chip accelerators are increasingly popular [66]. Because of their custom-designed nature, accelerators are typically far more energy-efficient than general purpose cores. At the same time, they can have much lower duty cycles due to their special-purpose nature. More prevalent use of
accelerators potentially leads to lower pressure on the interconnect scalability. Furthermore, accelerators use more hardwired logic and are more likely to exhibit simpler, more predictable access patterns such as streaming patterns [58]. Shared-medium structures are a good match for streaming larger segments of data.

In summary, communication in a chip-multiprocessor is carried out on a collection of fabrics; many architectural factors impact how much traffic depends on the backbone. Hence, sacrificing scalability of the backbone to achieve better energy efficiency and latency can be a viable alternative.

### 7.2.3 Bus Architecture

As discussed in Chapter 6, even with simple transmitter and receiver circuitry, transmission line links (TLLs) can offer high data rates. Figure 7.4 shows an overview of the interconnect sub-system. Each node uses a high-speed communication circuit to deliver packets over shared transmission lines connecting all nodes. Note that unlike the conventional notion of a bus that often implies broadcast capability, our bus is merely a shared medium that allows point-to-point communication. Prior to the transfer of payload data on the bus, two setup operations are performed.

![Figure 7.4: Overview of the bus-based communication subsystem.](image)

**Arbitration**

The use of a shared-medium bus structure requires an arbitration mechanism. A design can either use explicit permission granting or use carrier sensing for collision detection. The former is chosen for this work. The two approaches offer almost opposite trade-offs. Collision de-
tection offers good latency in a contention-free environment but utilizes the bandwidth poorly: a synchronized/slotted system cannot provide more than 36% of the channel capacity [110]. Granting explicit permission allows high utilization of the bandwidth at the expense of possible extra latencies, which better suits our bus. While any implementation of a permission granting system works, this design included a centralized system which can be thought of as a centralized token ring. Because the ring is centralized, the “token” can quickly pass to the next requester.

In theory, any central arbiter has a scalability limit. In practice, we find that the central arbiter is not a concern before we reach the saturation of the bus itself. This is to a large extent due to the simplicity of design – it is essentially just a priority encoder for, say, 16 bits in a 16-node system. Larger, far more complex priority encoders are used in the timing-critical store-forwarding circuit inside the core. Furthermore, when we use techniques such as segmentation (discussed later in Section 7.4) to improve the throughput of the bus, the scale of the arbiter actually decreases as each segment is smaller. We have measured a straightforward, unoptimized synthesis of a 16-node arbiter and compared it to the synthesized router used in a packet-switched interconnect [105]. The router’s overall delay is 4.3x that of the arbiter (1.65ns vs. 0.38ns). The router is also much larger (10x), consumes far more power (20x), and is used more frequently (per flit-hop).

The request and grant signals are transferred over transmission lines similar to those used to build the bus. Such transfers take additional latency (modeled faithfully in this study) that will only be exposed when the bus is lightly loaded.

**Receiver wake-up**

For energy efficiency, the receivers operate in two modes. When the message is intended for a node, its receiver transfers energy from the transmission line to the detector. On the other hand, when the message is intended for another node, the node is set to cause minimum loss for the through signal. For this reason, a setup step is performed immediately before payload data transmission to “wake up” the intended receiver, while other receivers remain in the off (and high isolation) mode. This setup is done in a pipelined fashion and the timing is shown in Figure 7.5.
**Turn-around time and bundling**

After the transmission of the payload, the bus will be idle for a period of time to allow the signal to “drain” from the links. Even in the short distance of on-chip transmission lines, the wave’s propagation delay is not negligible. The amount of time needed to wait before another node can start to use the bus to transmit depends on the distance between the current transmitting node and the next scheduled to transmit. In most cases, a full cycle of turn-around time is enough. In the extreme case, a two-cycle turn-around delay is needed.

Note that in the special case of the same node transmitting another packet there is no need for such a turn-around period. Thus for better utilization of the bus bandwidth, this design uses a policy that allows *bundling*: sending multiple packets for each bus arbitration. The impact of bundling is quantified in Section 7.3.4.

To summarize the timing of the bus’s operation, Figure 7.5 shows an example of a few packets transmitted over the bus. In this figure, node $i$ sent two packets, one each to node $m$ and $n$. In the background, the arbiter passes on the grant to node $j$ after accounting for the total...
bus occupancy of node $i$, which includes the time for the draining of the signal (2 cycles in this case).

**Partitioning the bus**

A simple way to get high throughput out of the bus structure is to use a wide bus that minimizes serialization latency. For example, a 32-byte cache line payload can be sent in one processor cycle over a bus with 32 data links operating at a data rate 8 times the computing clock speed. Clearly, a wide bus is wasteful for smaller payloads such as requests. In a shared-memory architecture, meta packets are common (about 60% in our suite of applications). Having another, smaller bus for meta packets is a clear option. In fact, with relatively small costs, it is possible to have multiple buses for meta packets. They can be used to increase throughput, or to support different types of requests such as in Alpha GS320 [46] (which prevents fetch deadlocks and eliminates the need to use NACK in their protocol). For simplicity, for this chapter, the bus consists of a single bus for meta packets and another one for data packets.

### 7.3 Analysis of TLLB in Small to Medium CMPs

#### 7.3.1 Experimental Setup

Architectural simulations of the proposed design were performed using an extensively modified version of SimpleScalar [21]. PopNet [2] is used to model the packet-switched network, while extra support was added to model the TLL bus. The details of the setup are listed in Table 7.1.

The cache coherence substrate for the architectural simulations is a directory-based MESI protocol with transients faithfully modeled both at the L1 and at the directory controller. The two state machines combined handle a total of 13 transient states and 57 legal transitions (excluding deferred handling), as described in Table 7.3 [141].

The simulation uses a set of diverse multi-threaded applications to test the designs. These applications are compiled using a cross-compiler to generate Alpha binaries. The limitation of the cross-compiler prevents us from running certain applications. Table 7.2 lists the applications used. Abbreviations are used in the data figures, and the corresponding abbreviation is in
### Processor core

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch/Decode/Commit</td>
<td>8 / 5 / 5</td>
</tr>
<tr>
<td>ROB</td>
<td>128</td>
</tr>
<tr>
<td>Issue Q/Reg. (int,fp)</td>
<td>(32, 32) / (112, 12)</td>
</tr>
<tr>
<td>LSQ(LQ,SQ)</td>
<td>64 (32,32) 2 search ports</td>
</tr>
<tr>
<td>Branch predictor</td>
<td>Bimodal + Gshare</td>
</tr>
<tr>
<td>Gshare</td>
<td>8K entries, 13 bit history</td>
</tr>
<tr>
<td>Bimodal/Meta/ BTB</td>
<td>4K/8K/4K (4-way) entries</td>
</tr>
<tr>
<td>Br. mispred. penalty</td>
<td>at least 7 cycles</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process spec.</td>
<td>Feature size: 32nm, Freq: 3.3 GHz, $V_{dd}$: 1 V</td>
</tr>
</tbody>
</table>

### Memory hierarchy

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 D cache (private)</td>
<td>16KB, 2-way, 32B line, 2 cycles, 2 ports</td>
</tr>
<tr>
<td>L1 I cache (private)</td>
<td>32KB, 2-way, 64B line, 2 cycle</td>
</tr>
<tr>
<td>L2 cache (per-core, shared)</td>
<td>128KB slice/core, 8-way, 64B line, 15 cycles, 2 ports</td>
</tr>
<tr>
<td>Intra-node fabric delay</td>
<td>2-core node: 1-cycle, 4-core node: 3-cycle</td>
</tr>
<tr>
<td>Memory latency</td>
<td>at least 250 cycles, 4 ports in 16-core system</td>
</tr>
<tr>
<td>Network packets</td>
<td>Flit size: 72-bits</td>
</tr>
<tr>
<td>Mesh interconnect</td>
<td>data packet: 4 flits, meta packet: 1 flit</td>
</tr>
<tr>
<td>Transmission line link (each node)</td>
<td>4 VCs; 3-cycle router; buffer: 5x12 flits</td>
</tr>
<tr>
<td>Bit Rate</td>
<td>26.4 Gb/s, 8 bits per CPU cycle</td>
</tr>
<tr>
<td>Transmission latency</td>
<td>2 cycles (worst-case)</td>
</tr>
<tr>
<td>Data link</td>
<td>36 links for data, 9 for meta</td>
</tr>
<tr>
<td>Outgoing queue</td>
<td>12 packets</td>
</tr>
<tr>
<td>Overhead</td>
<td>2 cycles each for (de)serialization, 30ps propagation delay per hop, 1 cycle for token request, 1 cycle for token grant/wake-up</td>
</tr>
</tbody>
</table>

Table 7.1: 16-core system configuration.

parentheses next to the application names. Inputs for each application are listed along with a brief description of the application. Each application is fast-forwarded past the initialization. An offline profile is used to determine data page mapping.

### Coherence protocol

Traditionally, a bus-based system uses a snoopy coherence protocol. But such an association is not fundamental and is perhaps inappropriate for a transmission line based implementation: First, leveraging analog circuit and transmission lines, a bus can support a rather large number of processor cores. Fanning out snooping requests to a large number of cores incurs significant energy overhead in cache controllers and is undesirable. Second, while a conventional digital bus can support broadcast primitives in a straightforward (but costly) way, broadcast operations are more demanding on analog transmission line designs, especially if the fan-out is large.
Table 7.2: Benchmarks used.

<table>
<thead>
<tr>
<th>Splash-2 [136]</th>
<th>Parsec [16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>barnes (ba)</td>
<td>blackscholes (bl)</td>
</tr>
<tr>
<td>cholesky (ch)</td>
<td>fluidanimate (fl)</td>
</tr>
<tr>
<td>fft (ff)</td>
<td>Other Benchmarks [35; 40]</td>
</tr>
<tr>
<td>fmm (fm)</td>
<td></td>
</tr>
<tr>
<td>lu (lu)</td>
<td></td>
</tr>
<tr>
<td>ocean (oc)</td>
<td></td>
</tr>
<tr>
<td>radiosity (rs)</td>
<td></td>
</tr>
<tr>
<td>radix (rx)</td>
<td></td>
</tr>
<tr>
<td>raytrace (ry)</td>
<td></td>
</tr>
<tr>
<td>water-sp (ws)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>n-body simulation (16K particles)</td>
<td>financial analysis/calculation</td>
</tr>
<tr>
<td>sparse matrix factorization (tk15.O)</td>
<td>(16K options)</td>
</tr>
<tr>
<td>complex 1-D fft computation (64K points)</td>
<td>animation (5 frames, 35K)</td>
</tr>
<tr>
<td>fast n-body simulation (16K particles)</td>
<td></td>
</tr>
<tr>
<td>matrix factorization</td>
<td></td>
</tr>
<tr>
<td>(512x512 matrix, 16x16 blocks)</td>
<td></td>
</tr>
<tr>
<td>simulation of ocean currents</td>
<td></td>
</tr>
<tr>
<td>(256x256 matrix)</td>
<td></td>
</tr>
<tr>
<td>graphics (large room)</td>
<td></td>
</tr>
<tr>
<td>integer sort algorithm (1M integers)</td>
<td></td>
</tr>
<tr>
<td>3-D rendering (car.env)</td>
<td></td>
</tr>
<tr>
<td>molecular dynamics (512 molecules)</td>
<td></td>
</tr>
</tbody>
</table>

Of course, both issues can be addressed. Snooping overhead can be mitigated by incorporating elements similar to a directory-based protocol that filters out nodes known to be unrelated to the current transaction. A special broadcast-capable bus can be demanded from the circuit layer. It is unclear whether these fixes are more cost-effective than avoiding broadcast with a directory-based protocol. In this study, we opt to assume a directory-based protocol in the baseline design. Table 7.3 shows the directory states (stable and transient) for our protocol [141].

### 7.3.2 Traffic and Performance Analysis

This analysis focuses on multi-threaded applications where there is a fundamental demand for horizontal communication. The applications can serve as stress-test workloads for our system which is more bandwidth constrained. The L1 miss rate of these applications ranges up to 61
Table 7.3: Cache controller transitions for L1 and L2 cache. The rows are the current state, the columns are the events/requests, and each entry contains an <action/next state> pair. Impossible cases are marked “error” and “z” means the event cannot currently be processed, and in some cases, the incoming request will be reinterpreted as a different one due to race. M, E, S, and I are stable states of L1 cache controller and DM, DS, DV (Valid with no sharers), and DI are stable states of L2 directory controller. Transient states are denoted by the pair of previous and next stable state. Transient states waiting for a data reply are superscripted with D and those waiting for just an acknowledgment are superscripted with A. All request events (Req) are followed by request type i.e., (Sh: read in shared mode, Ex: read in exclusive mode, Upg: upgrade request, Dwg: downgrade request, and Mem: memory access request). [141].

misses per thousand instructions (MPKI). As a frame of reference, larger, server workloads generally have a MPKI of up to 75. [15; 122].

Traffic impact of page placement:

A significant body of research exists to reduce unnecessary remote accesses by trying to map data close to the threads that frequently access the data. The solutions range from simple heuristics to map pages (e.g., first-touch) to sophisticated algorithms that migrate data on the fly. Such optimizations not only improve performance on their own by reducing average latencies, but also serve to reduce horizontal traffic. This research uses a simple model as a proxy of a “middle-of-the-road” solution to localize data. Specifically, the last-level cache is shared and page interleaved. Off-line profiling assigns pages the color that matches the color of the node where the pages are accessed most frequently.
Figure 7.6-(a) shows that simple techniques can already cut down on unnecessary horizontal traffic. Without data mapping optimizations, using round-robin data distribution in an $n$-node system, each L1 miss has a $1/n$ chance of being served locally. Hence, one would expect remote traffic to be roughly 94%, 88%, and 75% respectively for 16, 8, and 4 node systems. With even a simple profiling technique, the percentage of remote accesses drops to 53%, 46%, and 35%, respectively.

Figure 7.6: (a) Percentage of L2 accesses that are remote. The 3 configurations are 1, 2, and 4 cores per node. (b) Speedup due to profiling and clustering. The bar on the left is for 1 core per node, the right bar is for 2 cores per node. The baseline in this case is a 16-core mesh with round-robin data distribution.

The performance impact of such data mapping on a canonical mesh interconnect is shown in Figure 7.6-(b). Note that the 16-node organization has 1 core linked to its own L2 slice. The 8-node organization clusters 2 cores into a single node. The result is a longer latency for using the intra-node fabric to access the cache slices local to the node, but a decrease in the number
of remote accesses that use the backbone interconnect. The decrease in horizontal traffic and increased locality results in a speedup of more than 2x over a baseline with round-robin page allocation. Clearly, better data placement is an important optimization in its own right, and the sophistication and effect of the technique will only increase over time. The important side effect of traffic reduction alleviates the problem of the simpler shared-medium relay-free interconnect, such as our design.

Performance comparison

While the TLL bus has a more limited aggregate bandwidth, it offers a better latency in general and in particular for packets between far apart nodes. Figure 7.7 compares the execution speed of this interconnect (with a bundling factor of 3) with a mesh. In this experiment, the chip-multiprocessor has 16 cores and is organized into 16 or 8 nodes. At this scale, the limit in bandwidth is seldom a problem for any application and, in general, more than compensated for by the superior latency. Even the more bandwidth demanding applications, such as em3d, mp3d, and ocean, perform comparably to mesh, especially in an 8-node configurations. On average, applications run faster on the TLL bus than on the mesh by 1.15x in the 16-node and 1.17x in the 8-node configurations respectively.

![Figure 7.7: Speedup of TLL bus system over the respective (16- or 8-node) mesh-based system. The left bar in each group represent 16 node configuration and the right bar, 8 node. Note the y-axis does not start at 0.](image)

An idealized interconnect system was also designed, and it was verified that the TLL bus performs close to this upperbound. The ideal interconnect models propagation latency for the
Manhattan distance from the source to destination, but ignores contention and possible router delay. For instance, the 8-node system can achieve 91% performance of the ideal system, as shown in Figure 7.8.

![Figure 7.8: The 16-node and 8-node system performance compared to a latency-only, contention-free ideal.](image)

As can be seen in Figure 7.6-(b), even though the intra-node fabric becomes slower as the node size increases, the benefit of having a smaller network in general outweighs the cost of slower intra-node accesses. In a mesh-based system, clustering helps improve performance by 4%. Just as with the case of better data placement, these optimizations reduce the demand on the backbone interconnect and has a slightly more significant benefit (6%) in the TLL bus system.

To summarize, even though bus architectures face bandwidth scalability challenges, in modest-scale chip-multiprocessors and when natural steps are taken to improve performance, much of the disadvantages of TLL bus are mitigated and the benefit becomes more pronounced.

### 7.3.3 Power Savings

One of the main disadvantages of canonical mesh networks is the high power and energy consumption [39; 72; 97; 128]. On average, the network power accounts for around 20% of the total system’s power. In contrast, the TLL bus uses no relay or energy-intensive routing. The power consumption of TLL bus is low in both absolute and relative terms. An entire link con-
sumes 12.7mW while active (Table 6.2 shows power of individual components). Even when all lines are working all the time, the total power is around 600mW. When idling, the power consumption is even lower. Leakage in the communication circuit is estimate to be around 10µW per node [3], essentially negligible.

Comparing the energy consumed by the TLL bus to the power statistics from the network power model, Orion [134], there is a reduction in network energy of about 26x. With this reduction, the energy spent in the interconnect is less than 1% of the total energy consumption. The system level impact in terms of the energy savings can be seen in Figure 7.9.

![Figure 7.9: Average energy savings for the 16-core TLL bus compared to NoC.](image)

### 7.3.4 The Impact of Bundling

As discussed in Section 7.2.3, the turn-around time also wastes bus bandwidth and can be mitigated with bundling. So far, the design has used a bundling factor of 3, i.e., each node can send up to 3 packets before yielding the bus. Figure 7.10 shows the impact of varying the bundling factor from 1 (no bundling) to 3. As we can see, the performance generally increases when the bundling factor increases. Without bundling, much bandwidth is wasted due to turn-around, and so there is a noticeable performance increase with a bundling of 2. However, too much bundling can be detrimental to performance as well (e.g., in the case of tsp and lu). Figure 7.10-(b) shows the average overall packet latency for a bundling of 2 and 3 compared to
no bundling. On average, bundling of 2 and 3 saves 13% and 20% respectively of the latency and improves performance by 2.0% and 3.4% respectively.

Figure 7.10: (a) Speedup of the 16-node system with bundling of 2 and 3, over the system without bundling. Note the y-axis does not start at 0. (b) Overall packet latency relative to a non-bundled system. The left and right bar correspond to a bundling of 2 and 3 respectively.

### 7.3.5 Performance in Scaled-up Systems

While many-core chips will fill a certain market niche, a significant fraction of general-purpose chip multiprocessors may have only a relatively modest number of cores. The proposed design works well in such an environment. As the number of cores increases beyond a threshold, the viability of our current design will decrease. A limited scalability test is conducted with a 64-core system organized into 2- or 4-core nodes (32 nodes, 2 cores each; and 16 nodes, 4 cores
each), using the exact same bus design as before. Figure 7.11 summarizes the performance result compared to the (scaled-up) mesh-based design with the same clustering.

Figure 7.11: Relative performance of a 64-core system. For the TLL bus configurations, a bundle of 3 is used.

As the system grows in size, the probability of the bus becoming a bottleneck increases. In a few cases (e.g., fft and radix), the performance of the TLL bus is significantly worse than the conventional mesh interconnect (Figure 7.11). On the other hand, when the bandwidth is not a bottlenecked resource, the latency advantage over mesh becomes even more pronounced. As a result, the performance gap between the bus-based and mesh-based systems widens for many applications (e.g., fmm and shallow). On average, the TLL bus performs 16% and 25% better than mesh for a 32- and 16-node system, respectively. Clearly, simply having better aggregate bandwidth scalability is not enough. A packet-switched interconnect (including segmented bus) segments wires to allow simultaneous traffic, improving overall bandwidth at the expense of latency. The result can also be a serious performance issue for chip-multiprocessors.

In other words, a bus architecture should not be written off as a possible solution for on-chip interconnect. After all, no design is truly scalable in all respects. The sacrifice in latency in some packet-switched interconnects can be an even more serious performance problem, not to mention the significantly higher energy cost.

To better understand the limitation of bus-based system, the TLL bus is also compared it to an idealized interconnect system using conventional digital wires. In this system, no band-
width limitation or contention is modeled for the interconnect. A packet’s delay is calculated as 0.03mm/ps based on the latency-optimized wires in [97].

![Graph](image)

**Figure 7.12:** Performance of TLL bus relative to idealized contention-free, low-latency interconnect.

Figure 7.12 shows the performance of the TLL bus in 32-node and 16-node configurations (both have 64 cores) normalized to that of the ideal interconnect. As we can see, while 7 out of 18 benchmarks perform within 10% of the idealized case, the limited bandwidth shows significant limitation in a number of applications where performance can be improved several folds. Nevertheless, the bus system achieves 67% and 72% of the idealized performance, for 32- and 16-nodes respectively, showing a somewhat graceful degradation beyond its intended usage range. Recall, in a 16-core 8-node system, the bus can achieve 91% of the ideal’s performance.

### 7.4 Analysis of Baseline Bus in Larger-Scale CMPs

In order to improve the performance of the larger system, it’s necessary to investigate more sophisticated architectural techniques. In the small- to moderately-sized systems, the performance of a simple TLL bus is enough to provide performance which is close to ideal. The goal of the next few sections is to provide architectural solutions for improving the performance of larger systems, focusing on improving the available throughput and reducing the amount of traffic.

There are a number of benefits of increasing bus throughput to support a larger number of cores (e.g., 64 cores): if the bus can support an entire chip of cores, no packet switching is
necessary. The communication substrate design will be simplified. Moreover, it offers unique opportunities to simplify and optimize the shared-memory coherence substrate. And for server chips, which are often used in a virtualized environment, creating logical partitions with the size of 32 to 64 cores, it will be ideal for the bus to support the entire partition. Even if a bus does not cover the entire chip, but is used only within a group of cores with packet-switching connecting groups, the larger the group size the bus can support, the smaller the diameter of the packet-switched interconnect and fewer hops are needed.

In summary, high-speed buses that can competently handle workloads in larger scale systems (e.g., with 64 high-performance cores) provide important practical advantages. In the following sections, a number of approaches to improve the overall throughput of a transmission line based bus are explored and the cost of achieving higher throughput is analyzed.

### 7.4.1 Application Characteristics

We first analyze the characteristics of the benchmarks. So far we have treated the applications without classifications. However, as we add optimizations, the applications will refined differently depending on their traffic needs. In a first testbed, the 64-cores are clustered into 16 nodes, 4 cores each. The cores in the same cluster share the interconnect circuitry. It is worth noting that among our benchmarks, some are already performing well on the baseline TLL bus without any throughput enhancing techniques. Since their performances already come close to running on an idealized interconnect, there is little room for further improvement. To more clearly understand the impact of the techniques discussed so far, we have divided the benchmarks into 3 groups (G1-G3) with increasing potential performance benefits of throughput enhancement. This grouping is done by comparing the performance of applications under three different types of interconnects: a baseline TLL bus, a (concentrated) mesh NoC (with both a 2-cycle router and an idealized 1-cycle router), and an ideal interconnect modeling only the latency of aggressively configured metal wires [97]. Figure 7.13 illustrates this classification.

In G1, the benchmarks have low throughput demand that is well met by the baseline bus and the performance is within 90% that of ideal interconnect. These applications will see little, if any, performance improvement from optimizing the interconnect.
In G2, even though there is a significant performance gap between the baseline bus and ideal network, the bus still performs better or within 10\% of the NoC. In fact, the bus outperforms the NoC on average. Only when we use the 1-cycle ideal router do we see the NoC slightly outperforming the bus. Clearly, the latency advantage of the bus is important. For these applications, improving throughput will not help if it comes at a significant cost of latency.

Finally, in G3, the NoC clearly outperforms the bus, suggesting ample room for improvement when the bus throughput increases.

The significant performance increase from the 1-cycle aggressive NoC implies that as the NoC system improves its latency, the suggested optimizations will be even more important to larger scale systems, in order for the TLL bus to perform comparably. These optimizations may result in more router/network energy compared to a baseline TLLB but significantly improve the performance. At the same time, the overall energy of the bus-based system can be substantially lower than either NoC configuration.

### 7.4.2 Techniques for Increasing Effective Bus Throughput

Given a basic design, we can increase the throughput of the bus via a number of simple techniques at the circuit or architecture level, or with a combination of both. It is worth noting that
some optimizations are a unique result of the shared-memory chip multiprocessor environment, including its traffic properties, and are not necessarily applicable to bus topology in general. These techniques can be broadly categorized into three groups (a) increasing the underlying supply of raw throughput, (b) improving the utilization efficiency, and (c) co-optimization with the cache coherence protocol layer to reduce traffic demand. Of course, sometimes a particular mechanism defies exact categorization and can fall into more than one group.

### 7.5 Increasing Raw Link Throughput

Perhaps the first thought that comes to mind about increasing the throughput of a bus is to increase the raw throughput of each individual link. Intuitively, these approaches are more or less brute force approaches. Nevertheless, we analyze some options and evaluate their efficiency later in Section 7.8.

The potential of link throughput and inherent channel bandwidth of the transmission lines are high, thanks to the well-controlled on-chip environment and the relatively short distances of on-chip transmission lines. Figure 5.4-(b), in Section 5, illustrates an experiment to determine the aggregate bandwidth and potential throughput provided by an array of transmission lines, and has been replicated in Figure 7.14.

![Figure 7.14: The maximum bandwidth and achievable throughput of 75mm lines.](image)
To review Section 5, in this experiment, we limit the total pitch of the transmission lines to 2.5mm but vary the width, gap (between the pair of differential lines), and spacing (between two neighboring pairs) of the transmission lines. The length of the lines is set to 7.5cm, assuming a meandering shape going through the centers of sixteen $5mm \times 5mm$ tiles forming a $2cm \times 2cm$ chip. We then use EM and circuit simulators (see Chapter 3 for details about the tools) to estimate the 3dB bandwidth of the transmission lines and aggregate the bandwidth for the entire array. We also model noise coupled from neighboring aggressor lines and power supply noise in transceiver circuitry and estimate overall signal-to-noise ratio (SNR). This can give us the theoretical limit on the channel capacity.

Note that this experiment provides an approximate understanding of the potentials and cannot be used as a bound: in practical operating environments, the presence of thermal noise creates the noise floor that is not captured by the analysis of SNR. On the other hand, the bandwidth limit can be overcome, to a certain extent, using various circuit techniques. The bandwidth can also be expanded using multi-band design. Nevertheless, this analysis shows that on-chip transmission lines do possess relatively abundant bandwidth and can support a substantial amount of throughput.

There are many coding strategies to increase the raw throughput. For on-chip communication, however, we are likely limited to simpler variations. We first turn to 4-PAM (Pulse Amplitude Modulation) which doubles data rate over OOK. The additional circuit includes a digital-analog converter (DAC) for the transmitter and an analog-digital converter (ADC) for the receiver (Figure 7.15). These elements not only increase energy but also add latency on the packet transmission path. In order to minimize the latency impact, we use it only for data packet buses.¹

Second, we investigate Frequency Division Multiplexing (FDM). FDM allows us to use higher frequency bands on the same physical media. The attenuation in these bands can be high and it increases with frequency. When used as global buses, the higher-frequency bands quickly become too lossy and thus inoperable. A simple calculation can illustrate the problem. Assume we have a 10GHz channel spacing and use 6 such channels [29], Figure 7.16 shows the

¹One can even use global intelligence about traffic conditions to bypass 4-PAM when traffic demand is low in order to further minimize latency and energy overhead. This part of the design space is not explored.
Figure 7.15: Block diagram of links using 4-PAM and FDM.

Figure 7.16: Transmission line frequency response. As the frequency approaches the boundaries of the spectrum, there are noticeable noises due to artifacts of the simulation setup.

The frequency response of our transmission lines in the needed spectrum (between dc and 50GHz). At 50GHz, the attenuation is around 9dB. Furthermore, mixers introduce non-trivial noise figure (or degradation of SNR introduced by a component), especially for high-frequency operations. Even with bipolar designs, the noise figure can be around 10dB per mixer [61]. The combined effect of two mixers and the transmission line itself can amount to 29dB (800x), not to mention the filter’s loss. A rough interpretation is that in the 50GHz channel, the power of the transmitter and the sensitivity of the receiver need to increase a combined 800 times to achieve the same SNR as when using the baseband without mixers, which takes about 30 times more power on each side. Clearly, the higher frequency channels are exceedingly expensive to use in long on-chip transmission lines. They are intended for much shorter communications [29].
For this study, we use only two bands. The circuit support includes mixers for both the transmitter and the receiver side and a filter for the receiver side (Figure 7.15). Accurately estimating the power costs of the supporting circuitry is challenging. These non-trivial analog components need to be designed, tested, and fine-tuned to work at the required specifications. For this study, we use a simplifying analysis to estimate the minimum power cost to support frequency-division multi-band transmission. We use the design similar to [29] but adapted to the baseline system design. Specifically, the bandwidth of the baseband is 16GHz, the channel spacing is 30GHz, and the first band starts at dc.

In this case, the encoding is used to simply double the throughput of the data bus. Rather than increase the complexity of the arbiter, data packets are individually encoded, decreasing the impact of serialization. For example, rather than a data packet needing 4 compute cycles to traverse the transmission line, only 2 are needed. Figure 7.17 shows the results of using either frequency-division multiplexing or pulse-amplitude modulation to double the throughput of each line. On average, the system performance increases by 1.13x over a non-encoded baseline. Group 3, the group with the highest throughput requirements, benefits by up to 1.8x, and with an average of over 1.3x. These performance improvements do, of course, come at a potentially high cost, in terms of energy, area, and transceiver complexity, but with such significant performance benefits, may be an attractive optimization.

![Doubled Throughput through Encoding](image)

Figure 7.17: The performance of doubling available throughput using frequency division multiplexing or 4-level pulse amplitude modulation.

In order to remove the influence of the application characteristics, we can choose to look only at the window of improvement between the baseline bus and the ideal interconnect
(latency-only, contention-free). In other words, we can determine how much of the gap between the baseline bus and ideal network is closed by each optimizations. To determine this, we plot the performance of the encoding techniques, using the formula $100 \times \frac{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Base}}}{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Ideal}}}$. As the performance of the optimization approaches ideal, the metric will approach 100%. If the technique has little or no impact, it will produce a 0%, which is the baseline performance. If the optimization degrades performance, the value of this metric is negative. Figure 7.18 plots this metric for the encoding optimizations. For example, cholesky, whose baseline performance is within 10% of ideal performance, has little room to improve, and therefore, its average improvement is not accurately portrayed in Figure 7.17. However, when we consider only the gap between baseline and ideal performance, cholesky excels, closing the gap by 50%. On the other hand, mp3d which shows high overall improvement, shows less significant improvement in its potential performance window, indicating that while the encoding techniques improve performance, the gap below ideal performance is still large. Overall, the encoding techniques close the overall gap by almost 30%, on average. We will use this “gap” or “potential performance window” metric to supplement traditional results throughout this analysis.

![Figure 7.18: The potential performance window for doubling throughput with encoding techniques. The value plotted is a result of the formula: $100 \times \frac{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Base}}}{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Ideal}}}$. The G1, G2, G3, and Total values are the arithmetic averages.](image-url)
7.6 Increasing the Utilization Efficiency

While the underlying global transmission lines support a very high data rate, using them to shuttle around short packets found in a shared-memory system can cause significant under-utilization. First, the relatively long line means that a packet can take a long time to “drain” from the transmission line (the worst case propagation delay in our bus is 440 ps). A simple arbitration that waits for the bus to drain is one source of under-utilization. Second, packets destined for a near neighbor are a poor match to the global line structure. A number of techniques can address these issues.

7.6.1 Partitioning

A straightforward option is to partition the same number of underlying links into more numerous, but narrower, buses. In a narrower bus, longer serialization reduces the waste due to draining. To minimize extra transmission delays due to packet serialization, we can limit serialization to the data bus and use critical-word-first data marshaling.

An interesting side effect of partitioning the wide data bus into narrower buses is that the finer granularity allows us to better balance the load of the two types of buses. Instead of using a single 1-flit-wide meta bus and a 4-flit-wide data bus, we can use two 1-wide meta buses and three 1-wide data buses, which best match the average traffic distribution.

Our baseline bus uses 9 and 36 pairs of differential transmission lines for the meta and data bus respectively. Partitioning these lines into five independent buses makes data buses narrower. The longer serialization latency and occupancy of the bus better amortizes the time wasted waiting for the signal to drain out of the bus. This can be measured in busy utilization, which we define as the percentage of busy time (there are some requests in the system) the bus is actually transferring packets (as opposed to in turn-around). In the baseline, data bus utilization is around 60%. After partitioning, and thus serializing long packets, the utilization improves to close to 90%.

Additionally, having narrower buses allows us to better balance the resource and have the throughput supply matched closer to the demand of meta vs. data packets. We experimented
5 different configurations with 1 to 4 meta buses (the rest being data buses) and a final configuration where all 5 buses are mixed purpose. In all cases, the buses are cache line address interleaved. Figure 7.19 shows detailed breakdown of each partitioning choice, relative to the baseline bus. Figure 7.20 compares the average and range of each partitioning choice.

Figure 7.19: Detailed performances of different bus partitioning options over the baseline bus design.

Figure 7.20: The range and average speedup of different bus partitioning options over the baseline bus design.
The first thing to notice is that in all cases, *some* applications are doing (far) worse than the baseline bus. Understandably, the narrower buses incur longer serialization latencies that slow down the access to non-critical words. This goes to underscore the fact that throughput often comes with a latency cost and that over-provisioning throughput can backfire and degrade performance.

A second thing to notice is that better balancing of resources is more important to performance than the better amortization of turn-around time. The 4D+1M configuration provides the same throughput distribution as the baseline bus, but with better amortization of turn-around time. On average, the effect is only a few percent of improvement. On the other hand, the best static partitioning (3D+2M) improves performance by about 13% over 4D+1M. Note that it is the best partitioning only in the sense that on average it is better than other configurations. Different applications prefer different configurations. For example *raytrace* seems to prefer a wider data bus, and shows degradation for all of the newly partitioned lines. This implies that the serialized packets, which incur longer latencies for any non-critical block, hinder the requesting core, as it waits for the longer penalty. This is important to note, because while these particular allocations seem to translate to high performance, they are by no means ideal for each application.

Dynamic allocation – having all buses being general-purpose – in theory achieves best throughput utilization. Interestingly, it almost consistently degrades performance. This is due to longer queuing delays for critical meta packets when they queue up behind long data packets. This suggests that better QoS control can potentially further improve the performance.

Overall, the range of performance is a bit of surprise: without any change to the underlying raw throughput supply, simply reorganizing the buses can improve some applications’ performance by 2X. This indicates that better understanding and matching the application’s demand is perhaps no less important than scaling up the throughput supply.

### 7.6.2 Wave-based arbitration

Another mechanism to reduce the impact of the draining latency is to allow waves to coexist on the transmission lines. When waves meet, they travel on without impacting each other, only
creating a superposition where they meet. In the example shown in Figure 7.21, two far apart nodes send each other a pulse train. The two trains cross each other over inactive nodes and do not interfere with each other when they reach their respective receiver.

![Wave-based arbitration diagram](image)

Figure 7.21: Wave-based arbitration would allow multiple signals to propagate without physical segmentation.

In theory, we can send multiple pulse trains on the links so long as no two trains cross over at an active receiver or transmitter. In practice, we send at most two such trains and use a simple rule to pick a second pair of transmitter and receiver (\(T_x_1\) and \(R_x_1\)) that do not interfere with the already-selected first pair (\(T_x_0\) and \(R_x_0\)). The distance between \(T_x_0\) and \(T_x_1\) and between \(R_x_0\) and \(R_x_1\) need to be, in our case, larger than half the total length of the bus. The distance between the transmitter and receiver pair does not have a requirement, as long as the 2 transmitters (and 2 receivers) are sufficiently distance. This is important, as benchmarks with largely local traffic will likely see large benefits. We tested the design of such an arbiter and found that it does not affect cycle-level performance.

Figure 7.22 shows the performance of the applications using wave-based arbitration, and Figure 7.23 shows the performance compared to the performance gap between baseline and ideal. In a single unified bus, long queuing delays can occur as a distant node awaits the arbi-
Figure 7.22: The per-application performance of wave-based arbitration.

Figure 7.23: The potential performance window for wave-based arbitration. The value plotted is a result of the formula: $100 \times \frac{Per_{Optimization} - Per_{Base}}{Per_{Optimization} - Per_{Ideal}}$. The G1, G2, G3, and Total values are the arithmetic averages.

Wave-based arbitration allows these distant nodes to communicate concurrently without fear of interference at the receivers. Of course, not all distant node traffic can be sent simultaneously. Rather, the transmitters and receivers must satisfy the distance requirements. In cases where throughput is not an issue (e.g., group 1), there is no gain, as expected. In groups 2 and 3, there are still a few applications which see no performance improvement, due to the traffic pattern. However, for
some applications, such as mp3d and ocean, which require more throughput, and have largely local traffic, the ability to send messages simultaneously at the extreme ends of the bus has a significant performance impact, peaking at almost 1.20x performance compared to baseline. Group 3 as a whole averages 1.10x improvement, which is non-trivial.

Additionally, as we analyze the performance windows in Figure 7.23, we see that for a few individual applications, the performance gap to ideal is decreased by up to 50%, and by over 10% on average.

Overall, however, wave-based arbitration does not work with all other optimizations. For example, if we were to segment the bus, as we will describe below, then the number of opportunities to exploit wave-based arbitration decreases past the point of usefulness. Therefore, despite its noticeable impact over a baseline, as we add other optimizations, wave-based arbitration becomes incompatible.

7.6.3 Segmentation

In addition to increasing the temporal efficiency of the bus, we can improve its spatial utilization. One benefit of packet-switched interconnect is that multiple transmissions can happen simultaneously at different parts of the fabric. A similar effect can be achieved if we divide the transmission line into a few segments, similar to the electrical segmented bus in [146]. When a node is communicating with another node within the same segment, it only needs to arbitrate for the segment of the bus, leaving other segments free for other independent transmissions. When the communication crosses multiple segments, the transmitter needs to obtain permissions for all segments and the segments will be connected to act as a single transmission line.

Note that such electrical segmentation is fundamentally different from buffered buses which are essentially ring interconnects. Our segmentation does not change the architectural model of a global bus: delivery of a packet about an address does not overlap with that of another packet on the same address. Those packets are globally serialized. Maintaining such feature allows significant simplification of the coherence protocol [46] and other optimizations.

Electrically, the segments can be connected in two ways, as in Figure 7.24. First, a pass-gate can be used to form a passive, bi-directional connection. In this case, the pass-gate adds a little
bit of attenuation and signal distortion. We find the impact to be acceptable when the number of segments is low.

Second, two separate uni-directional amplifiers can be used to connect neighboring segments. The cost of this approach is the power consumption for the amplifiers. However, with these amplifiers, the source transmitter power can be lowered somewhat since the signal travels at most the length of one segment and is essentially repeated at the segment boundary.

For arbitration, we use one local arbiter for each segment. Each arbiter has request and grant ports to all local nodes as well as to other arbiters. Intra-segment arbitration is completely local to the segment arbiter. Inter-segment communication requires two-stage arbitration, where the sender’s local arbiter request for the use of other segments.

Figure 7.24: The chip layout with 4 segments. Each segment is connected by either a pass-gate switch or an amplifier pair.

Figure 7.25: The performance impact of using 2 and 4 segments, compared to using a 1 unsegmented bus. Note that the y-axis does not start at 0.
Segmentation works well in cases where traffic demonstrates locality. We can see from Figure 7.25, using 2 or more segments, some applications can improve performance by 1.4-1.6x. At first glance, on average, the impact is much lower, especially for the 2-segment configuration. However, when we look instead at the performance gap, as in Figure 7.26, we see that for some benchmarks, the 2 segment case improvement is significant, and perhaps better than the 4 segment case. For Group 3, the 4 segment case closes around 40% of the performance gap, with a maximum of over 60%. For both cholesky and shallow, both Group 1 applications, we see around 40% decrease in the performance gap. For all applications, the 4 segment case closes the ideal performance gap by 20%. In absolute terms, the 4 segment case outperforms the 2 segment case, and thus will be assumed for future optimization comparison, but the 2 segment case also shows merit, and should not be discarded.

![Figure 7.26: The potential performance window for segmentation. The value plotted is a result of the formula: $100 \times \frac{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Base}}}{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Ideal}}}$. The G1, G2, G3, and Total values are the arithmetic averages.](image)

In the 2-segment case, an average of 40% of all traffic uses both segments. This translates to a small opportunity for performance improvements. However, when we increase to 4 segments, a ring structure guarantees no packet needs to access all segments. There are more chances for multiple segments to work for different packets simultaneously, therefore increasing effective throughput. In the 4-segment cases, we observe that each bridge is activated between 12% and 30%. For G1 applications, segmentation increases performance by about 1.4x and 1.11x for the 4-segment and 2-segment cases respectively. For the other groups of applications, there is
seldom a net performance degradation, even though both arbitration and propagation latency increase.

In terms of energy cost, the exact result depends on the implementation of the bridges: a bridge with a pass gate incurs very little energy overhead itself but attenuates the signal a little bit. The downside is that cascading too many such switches will impact both the data rate and the overall energy. In a 4-segment configuration with ring, at most two switches are used for any packet. Our simulations show that the impact of two switches is small enough to be neglected. The amplifier-based bridge has better loss and isolation characteristics but comes at a cost. Each bridge consumes an additional 2.8mW of power, or about 90% of the transmitter’s power.

It is also possible to use routers to connect the segments together, creating a simple packet-switched bus. For completeness, this was tested as an option, despite the desire to remove routers. However, in this case, the router only has 2 inputs/outputs (similar to the one in Figure 7.28). This method does not have the same performance implications as the switch-segmented buses discussed already. Figure 7.27 shows the performance results for a router-connected segmented bus.

![Figure 7.27](image)

**Figure 7.27**: The per-application performance of a 2- and 4-segment bus, connected using simplified routers, over a baseline of a single unified bus.

For a 2-segment case, the router-based segmentation performs only 1.03x better than baseline, similar to the switch-segmented bus. For group 3 applications, in the 4-segment case, the router-based segmentation outperforms baseline by 1.24x, as compared to the 1.36x from the
switched segments. This is partly due to the extra time it takes for local traffic to receive the token, as the routers now act as an extra node in the segment. In a broader view, these routers also remove the atomicity of the bus, which, as we aggregate optimizations, would become a significant hindrance. The lack of atomicity, along with the slightly degraded performance compared to circuit switched segmentation, motivates the use of circuit switching over packet switching when segmenting the bus.

7.6.4 Local links

In shared-memory programs, there are intrinsic reasons behind near-neighbor communications that result in local packets. An extreme form of that locality is nearest-neighbor communication. A globally-shared bus topology delivers 100% of its total throughput as its bisection throughput.\(^2\) This allows the bus to have a much lower total throughput compared to alternatives and yet still satisfy real workloads competently. But a global bus is a poor match for nearest-neighbor communication patterns. Adding dedicated links just for neighbor communication is one way to mitigate the problem.

Such communication is very well supported by packet-switched interconnect with a topology like a ring or a mesh. Since the communication distance is so short, even simplistic digital signaling is reasonably efficient.

One way to exploit such locality is to use digital links to connect nearest neighbors and thus provide extra throughput capacity. An extreme form of such hybrid design is to have a complete packet-switched network in tandem with transmission line based buses. However, such a design is overkill and increases both area and energy overhead.

We propose a different form of hybrid design where the digital links are never used to relay packets. In other words, only packets that are destined for connected nearest neighbors are sent over these links. All other packets are sent over the main buses. With such limitation, the transceiver circuit is much simpler than a conventional router. First, there is no routing in this controller. Second, there is no need for virtual channels (VCs) and VC allocation. Finally, since this extra fabric is only in a supplemental role, the “strength” of the logic can be much

\(^2\)Compared to 1/6, 1/4, and 1/3 for (2D) mesh, torus, and flattened butterfly topologies in a 4 × 4 network.
reduced without significant impact on overall system performance. For instance, we can link neighbors in one dimension only and significantly reduce the size of crossbar and number of buffers. A similar reduction in radix in a conventional router would increase network diameter and potentially seriously degrade performance.

We avoid any packet switching and the associated complexities on these local links. Furthermore, since these links are not intended to suit all traffic patterns, we simply use a ring. Such links can be built with just digital links since the distance is relatively small. If transmission lines are used for local links, the pitch needed is much smaller than that of the global lines.

![Diagram](a)

**Figure 7.28**: (a) A conventional router with 4 VCs, also pictured in Figure 2.3 [105]. (b) Our much simplified digital link controller.

Figure 7.28 shows and the proposed digital link controller (compared to the conventional router discussed in Chapter 2). Synthesis results of the conventional router [105] and our design show that the area reduces by more than 6x, and the power reduces by close to 3x. Both designs were compiled and synthesized in the Cadence RTL compiler using a 45nm design process [123].
Hybrid links use digital links for nearest-neighbor communication and transmission line links (TLL) for everything else. We experimented with two links to two nearest neighbors along the bus or four links to neighbors in two dimensions. A summary of the performance impact is shown in Figure 7.29, along with the percentage of nearest-neighbor traffic. The performance improvement of linking neighboring in 1 dimension is around 17%, again with group 1 applications showing the most improvements (27%). Adding another dimension showed a negligible improvement (1-2%). On average, over a third (35%) of all traffic is to the nearest neighbors (in 1 dimension), while some applications have a much higher ratio of local traffic (em3d 62%, jacobi 54%, and ocean 87%). Additionally, Figure 7.30 illustrates the impact on the performance gap between the baseline bus and ideal network. The figure shows that using local links closes 25% of the gap between the baseline performance and the ideal network, and in some cases, closes the window by close to 60%.

![Figure 7.29: The performance impact of using 1-dimensional and 2-dimensional nearest neighbor routers, compared to TLL bus baseline. Note that the percentages above the bar are how often the traffic is nearest neighbor for the 1-dimensional case.](image)

On average, the energy per bit using NoC is about 20x higher than that of using the baseline bus. This high relative cost is the result of multi-hop relaying and high energy cost at the router. In our case, the digital links are only used in nearest neighbor (1-hop) communication and the controller is much reduced. A simple analysis can be used to estimate the energy/bit in hybrid links.
In NoC digital links, $E = h \times (E_R + E_L)$, where $E$ is overall average energy per bit, $h$ is average hop count (about 2.4), and $E_R$ and $E_L$ are router and link energy respectively. For easier calculation, all energy results are normalized to energy per bit on TLL. In hybrid links, if $x\%$ of bits are sent through digital links, the total energy cost is $E_H = x\% \times (E_C + E_L) + (1 - x\%) \times 1$, where $E_C$ is the energy per bit on the controller. Our energy estimation (from ADS, Orion, etc.) show that $E_R = 6$, $E_L = 2$, and thus $E = 19.2$. As previously mentioned, the simplified controller consumes about 3x less power based on synthesis result. Thus, $E_C = 2$. Combining these parameters, we can get $E_H = 2.1$. In other words, using digital nearest-neighbor links roughly doubles the per bit energy cost.

### 7.7 Optimizations on the Use of Buses

Unlike its off-chip counterpart, an on-chip interconnect is not subject to certain interface requirements such as those dictated by the pins of the chip. Evaluating a bus only as a backward-compatible, drop-in replacement for a packet-switched interconnect would underestimate its potential to help optimize the entire stack. Given the bus’s unique properties, we can convey certain information much more efficiently.
7.7.1 Invalidation acknowledgement omission

One example is the opportunity to omit invalidation acknowledgements. In a coherent shared-memory system, the knowledge of store completion is needed in implementing memory barriers or write atomicity (our system supports Alpha consistency model with write atomicity). With a packet-switched network, protocols rely on explicit invalidation acknowledgements to provide the knowledge of completion. If the interconnect offers certain capability to help infer the delivery, an explicit acknowledgement can be avoided [46]. A traditional bus is one such case. Protocols rely on the commitment of carrying out received invalidation requests instead of acknowledgement [36]. In other words, the nodes ensure that the invalidation will be logically ordered before any out-going transactions and this commitment effectively serves as an implicit, instantaneous acknowledgement.

Note that interconnects such as the Element Interconnect Bus for IBM Cell processors [8] are essentially rings, despite the name. These “buses” relay packets and cannot omit invalidation acknowledgements. Our system always delivers packets end-to-end in an atomic bus transaction. This is true even with the segmentation discussed earlier, since the segments are electrically joined into a single bus before the packet is transmitted in an atomic transaction.

7.7.2 Limited multicasting

While transmission lines are most often used for point-to-point communications, they can be designed to allow multicast operations. In our system, supporting a small number of simultaneously operating receivers is relatively easy. Our circuit simulation shows that if two receivers are turned on, there is a tolerable 5% additional attenuation for the signal at the more distant receiver. Multicasting finds natural usage in sending out invalidations to multiple nodes. We choose to support 2-way multicasting only. While the traffic reduction due to 2-way multicasting may not be dramatic, it drastically cuts the latency and queuing delays during traffic bursts resulting from invalidations of widely held data.

In general, the traffic reduction is limited, but so is the extra circuit support needed for the techniques. Furthermore, these techniques do not cause any performance degradation. On average, omitting invalidation acknowledgement and multicasting reduce meta bus traffic by
12% and 5% respectively. The traffic reduction is highly uneven among the applications and so is the performance impact.

Figure 7.31: Relative performance over baseline bus when using invalidation acknowledgement omission (left), and multicasting (right). Note the y-axis does not start at 0.

As can be seen in Figure 7.31, these minor techniques can sometimes improve application performance by 20%. On average, G2 and G3 applications see a 13% performance benefit, comparable to that achieved when doubling the data bus throughput with either 4-PAM encoding or frequency-division multiplexing. And as Figure 7.32 shows, invalidation acknowledgement omission closes almost 90% of the gap between shallow’s baseline and ideal, and closes 20% of the average gap for all applications, while the addition of limited multicasting pushes that total to 25% of the overall gap.

7.7.3 Fuzzy invalidation

In some cases, we can communicate compressed information without incurring costs elsewhere in the system. In the case of a NoC, a packet shorter than a flit reaps no benefit in traffic savings. But with a transmission line bus, a narrower width directly reduces bus footprint.

We can send shortened messages to convey invalidation addresses. One approach is to use lossy compression that reduces the precision about the lines to invalidate. Taken to an extreme, our design uses 1 bit to represent a group of special lines. We find that a sizable fraction of cache lines are used only immediately after they are brought in and never again until eviction or invalidation. If a line shows this behavior consistently, it is a candidate for such imprecise
or fuzzy invalidation (FI or $\phi$) as the risk of invalidating the line prematurely is low. We model a simple implementation that only identifies lines not accessed again after being brought in. When such a line is evicted, with a certain probability (25% in our case) the cache will notify the directory about its $\phi$-readiness. When a $\phi$-line is fetched to L1 cache, the line’s $\phi$-bit will be set. The cache controller uses this bit to flash-invalidate all $\phi$-lines upon receiving a fuzzy invalidation command. When the directory serves a write request to a $\phi$-line, it sends the $\phi$-command by sending a pulse over a special broadcast transmission line.

A more sophisticated approach could use frequency-division multi-accessing to allow directories to send $\phi$-commands in a more targeted way, but for these experiments, more complex schemes were not tested.

Figure 7.33 illustrates the per-benchmark breakdown of using fuzzy invalidation. The average performance improves by over 1.04x when fuzzy invalidation is utilized, with a peak of 1.4x. Figure 7.34 also shows the impact of fuzzy invalidation within the potential performance improvement window between the baseline and ideal networks. While Group 1 and Group 3 show little performance gap improvement, Group 2 applications performance gap is closed by 20%, with a maximum of 40%. Group 3 applications are typically more reactive to changes in overall throughput, but show less impact for optimizations that focus on reducing the latency.
There are 2 main benefits of using a specialized bus. First, there are less invalidations that need to be sent through the conventional network, reducing traffic demand and decreasing the time a pending request must wait for invalidations and acknowledgements. Second, the cache utilization is improved, as dead lines are no longer taking up space in the cache. This effect does not have as much impact as the former, as we can see by comparing Figure 7.33 with the results shown in Figure 7.35, which uses fuzzy invalidation, but still sends messages, to mimic the optimization’s effect on cache utilization. As the figure shows, the impact of the
cache utilization is minimal (1.01x over baseline, with a high of 1.1x) compared to the full optimization (1.04x average, with a maximum of 1.4x). This implies that fuzzy invalidation optimizes the architecture because of the simplified coherence for a subset of cache lines, rather than simply optimizing the useful cache lines in the local L1.

Figure 7.35: The per-application of a cache utilization study (over baseline), using fuzzy invalidation to mark “stale” lines, but still sending messages to remove the impact of traffic and timing on the system.

7.7.4 Boolean bus

Similar to fuzzy invalidation, we can build a narrow specialized bus to support transfers of Boolean values (0s and 1s), which are commonly used in synchronizations. To simplify the design, the Boolean bus is only used to send a data reply when the line is Boolean, i.e., all but the least significant bit are zero. Our software library that implements locks and barriers spaces the synchronization variables into single-word lines and uses load-link instructions to suggest the hardware to send special Boolean requests. When serving such a request, a simple 0-test is performed to decide whether the Boolean bus is used to send the reply.

Not all applications will benefit from the use of a Boolean bus in this form, as not all applications contain a substantial number of locks. For the applications which do not contain locks, there is no performance change. For the relevant applications, the performance increases by around 4%, with a maximum of over 1.06x and no performance degradation. Figure 7.36 shows
only relevant applications (*i.e.*, applications with a significant number of locks). Figure 7.37 shows a per-application breakdown of the performance for all applications. Future applications which include more synchronization instructions would show more improvement. Additionally, as some benchmarks have little room for improvement, the average impact is hampered further.

![Figure 7.36: The performance breakdown of the Boolean bus optimization for applications with significant lock activity (more than 1 lock per 10k instructions).](image)

![Figure 7.37: The per-application performance of using a Boolean bus.](image)

### 7.7.5 Coherence-aware arbiter

Another possible optimization takes advantage of the arbiter’s centrality in all transactions. All coherence traffic must traverse the arbiter at least tangentially, in order to send a request message. When these coherence messages are sent, the bus grant request can also include some
extra state (if the message is making a line exclusive or transient). For example, a request for an exclusive line can be marked in the arbiter and subsequent requests for that line can be handled by the arbiter, thus eliminating the immediate need for the bus. The directory will then handle the message off the critical path, gather invalidations and acknowledgements. In other words, we can shift some of the coherence burden to the arbiter.

Of course, if the arbiter holds the state at too fine a granularity, then it will become complex and slow. Instead, we can hold coherence information at the slice level. By recording whether a L2 cache slice has any line that is currently in transient or exclusive state, the arbiter can shorten the wait time for a miss request. Any time a line is marked as transient or currently being written to, the arbiter can be notified in parallel, as a message traverses the bus. By sending a notification to the arbiter (a single bit per cache slice), the arbiter can keep a global state of the L2. For example, if slice A has no pending transactions to any of its lines, the arbiter is aware of that fact, and when an upgrade request is sent for a particular line, the requester can probe the arbiter which can respond as to whether the line is in a stable, shared state. If the requester is handling a miss, the baseline system would need to be used. If the arbiter has no state indicating that a line is currently in a transient or exclusive state, then the arbiter responds to the requesting node that the write update can occur without waiting for the subsequent invalidations.

These invalidations still occur in the background, but the write request can be serviced without the extra delay. In systems where not many lines are held in an exclusive state, the critical path latency for these types of requests can be drastically reduced. In a system with many exclusive lines, the system will default back to the baseline, never preemptively allowing a line to upgrade.

As the arbiter is the center point for all messages, it maintains global ordering. On average, this optimization improves performance by only 1.04x over baseline with a maximum of 1.17x (see Figure 7.38). This performance, along with the increase in complexity for the otherwise simple arbiter suggests it as an interesting, but in the end, negligible, optimization, and thus, is not explored when the optimizations are accumulated. Figure 7.39 shows that for Group 2, in particular, the coherence aware arbiter significantly closes the gap between the baseline and ideal networks, by an average of 1.1x and with a peak of 1.18x improvement.
Figure 7.38: The per-application performance of using a coherence aware arbiter.

Figure 7.39: The potential performance window for the coherence-aware arbiter. The value plotted is a result of the formula: $100 \times \frac{\text{Perf}_{\text{Optimization}} - \text{Perf}_{\text{Base}}}{\text{Perf}_{\text{Ideal}}}$ – $\text{Perf}_{\text{Ideal}}$. The G1, G2, G3, and Total values are the arithmetic averages.

7.8 TLLB 64-core Evaluation

7.8.1 Performance Benefits

We have described many different ways of improving the throughput. Which ones ought to be pursued in a practical design depends on many factors, some of which hard to quantify. Below, we will first show the performance of these techniques in isolation.
In Figure 7.40, we sort the techniques by decreasing mean performance improvement. For brevity, we only show the (geometric) mean and the range of relative performance. As a frame of reference, we also include the result from the ideal interconnect, which clearly shows the ample performance headroom as well as significant variability among applications. Two general observations can be made from this summary figure, which we elaborate below: 1) raw throughput is not as critical as intuitively expected; and 2) even simple coherence optimizations can be fairly effective.

![Relative Performance](image)

Figure 7.40: The performance impact of the techniques discussed. The bars show the geometric mean of relative performance over a baseline bus on all application, while the I-beams show the range from all applications. Note that the y-axis does not start at 0.

**Impact of throughput**

Throughput is a metric used routinely to characterize a network’s performance. This can be a misleading oversimplification in the context of on-chip interconnect for CMPs. In a CMP, traffic is a direct result of cache misses. Various data dependencies and limits on buffers or outstanding requests constrain the packet-level parallelism. This is different from, say, file transfers where more throughput can always be readily leveraged. Once over a certain threshold, throughput only has a secondary impact as it affects latency indirectly through queuing and serialization.
delays. As a result, more throughput supply is only beneficial when the latency impact of obtaining more throughput is small.

We can see this effect in Figure 7.40. Increasing the raw throughput (in this case doubling it via either 4-PAM encoding or 2-band FDM) provides similar benefits as the techniques that merely try to incrementally improve the utilization efficiency. These techniques (e.g., partitioning) carry little latency and energy costs. In contrast, a NoC achieves high throughput at a more significant latency cost due to repeated packet relays. The high latency is then being mitigated with complex, speculative router designs that further drive up energy cost. Eliminating or at least reducing the reliance on packet switching in on-chip interconnect design is a direction that deserves more attention.

**Simple coherence optimizations effective**

Another set of techniques reduces traffic demand by leveraging the properties of a bus. These include invalidation acknowledgement omission, fuzzy invalidation, and multicasting. These techniques can make a non-trivial performance impact (e.g., up to 1.3x for fuzzy invalidation), although they do not directly increase the nominal throughput of the interconnect. Note that in some cases, the benefits will increase when programs start to use these underlying mechanisms (e.g., Boolean bus) for more purposes.

### 7.8.2 Costs

The costs of these techniques include extra circuit support and run-time energy expenditure. The techniques can be grouped based on these costs:

- **Little to no cost:** Partitioning and invalidation acknowledgement omission require only a different way of organizing resources and need no new circuits.

- **Some circuit cost:** Multicasting, fuzzy invalidation, the Boolean bus, coherence-aware arbitration and wave-based arbitration require some support from the circuit, but there is little or no run-time energy cost.
• With circuit and energy costs: The remaining techniques incur some circuit costs and energy costs.

In segmentation, the energy cost depends on the implementation of the bridges: a bridge with a pass gate incurs very little energy overhead itself but attenuates the signal a little bit. We conservatively assume a more costly amplifier-based bridge, each consuming about 90% of the transmitter’s power. On average, we observe about 40% of the packets cross the bridge in 2 segments, and about 65% cross one or two bridges in 4 segments.

Local links implemented with transmission lines do not add any energy overhead and in fact use less powerful drivers. They do require slightly more area to be devoted to transmission lines even though each link uses narrower transmission lines. Conservatively, we assume local links using digital wires that do not take metal area for the global transmission lines. Our synthesis results show that factoring in the controller, transmitting over local digital links costs about 4 times the energy as that over the global buses. On average, about 35% of data traffic is off-loaded to the local links.

Finally, providing raw throughput, especially through FDM, is a more energy intensive option. We estimate the PAM design to double the energy per bit of an OOK link. FDM requires an increase in the transmitter and receiver power in order to compensate for the increased attenuation on the higher frequency band and the noise figure, introduced by mixers. We assume a noise figure of 5dB per mixer, a 6dB increase in attenuation, and thus a compensation of 8dB on both transmitter and the receiver side.

7.8.3 Comparison of Cost Effectiveness

We now summarize the high-level cost benefit analysis of each individual mechanism. Some optimizations have been filtered out (wave-based arbitration, coherence-aware arbiters, Boolean bus) because of the underwhelming performance improvement or the inability to integrate with some more effective mechanisms. This analysis does not tease out the synergy or overlap between multiple techniques when deployed together. But the analysis still gives a reasonable picture of what some of the first steps we should take to increase effective throughput.
Figure 7.41: The relative network energy, chip energy, and energy-delay product of applying each technique discussed in a 64-core, 16-node system.

Figure 7.41 shows energy-related metrics for each individual technique applied in isolation, all normalized to baseline bus. From left to right, the bars are ordered by increasing network energy. The first five techniques have very little energy overhead and in fact some save network energy by sending fewer packets. Starting with (4-way) segmentation, the last four techniques have noticeable energy increases in the network. But the performance benefit reduces energy spending elsewhere in the system (e.g., clock distribution), so the chip wide energy is actually reduced.

### 7.8.4 Example Design Points

Given this array of the techniques, a chip designer can put together a set of them to suit the needs of the chip. In Figure 7.42, we show four such configurations. The first configuration combines partitioning and invalidation acknowledgement omission. These two techniques not only have little costs, but have significant performance impacts. The next configuration adds multicasting and fuzzy invalidation. At this point, there is no change in the nominal throughput of the bus and no increase in energy of the network. The performance is already 1.22x that of the baseline bus, 1.17x faster than a mesh NoC, and higher than a mesh NoC with idealized 1-cycle routers.

In the next two configurations, we progressively add segmentation and 4-PAM. The system performance improves to 1.39x. An ideal interconnect is only 1.06x faster. While the network
Figure 7.42: The effect of a few configurations. FIMP is short for the combination of fuzzy invalidation, invalidation acknowledgement omission, multicasting, and partitioning. (a) The relative performance with geometric mean and range from all applications. An ideal interconnect is shown as a reference. (b) The relative network energy, total chip energy, and energy-delay product.

energy is much higher, the faster speed compensates partly and the chip energy is still about 20% lower than the baseline bus. Note that these results are the average of all applications. Within G3 applications, the benefits are much more obvious: 1.89x speedup at an average of 23% less energy.

Clearly, the effectiveness of these techniques is highly dependent on the application behavior. As we already saw, G1 applications will not see much performance gain even if the interconnect is ideal. Therefore, it would be helpful to have some dynamic adjustment to turn
on power-hungry communication mechanisms only when there is significant performance benefit to be gained.

**Comparison with NoC:**

As we have shown, with some enhancements, the effective throughput can be increased with low energy costs. Compared to the bus, a NoC solution starts from a high-throughput design point. But the high throughput comes at the expense of energy intensity (NoC’s network energy is 15x that of baseline bus) and higher latency and may not necessarily translate to high application performance. Figure 7.43 shows the experiment that uses injected traffic (uniform random) to measure packet latency in different configurations. We can see that techniques discussed significantly extend the saturation point of the bus without increasing packet latency at low load. While NoC has a higher saturation point, the common-case latency is worse.

![Figure 7.43: Packet latency vs. offered traffic of various interconnect configurations. Note that the model of flattened butterfly assumes no increase in router delay.](image)

Note that the uniform random traffic patterns show the best cases for the NoC configurations, whereas a bus architecture is much less sensitive to the traffic pattern. When we use execution-driven simulations, the benefits becomes much more obvious. In G3 applications,
where the baseline bus lags significantly behind NoC in performance (0.75x), the improved bus now is 1.4x times faster than NoC.

For a better understanding of the application behavior, we map the application injection rate, seen here in Figure 7.44. This lends insight into why some benchmarks benefit more than others from changes in the network architecture. As the figure shows, the applications tested, all of which have inherently inter-node communication, have an offered load that is not high enough to take advantage of the higher throughput scalability offered by the NoC. This figure lends proof to what was assumed about the characterization of the benchmarks: for the baseline TLLB, Group 3 has too much offered traffic for the TLLB to perform well; hence, it performs significantly worse than the NoC. Group 1 has minimal offered traffic and thus the traffic can benefit from the lower packet latency. Group 2 is in a more tenuous position, in which the traffic offered ranges around the intersection of the baseline TLLB and the NoC performance. As we added optimizations to the TLLB, we extend the available throughput past what is needed by Groups 2 and 3, resulting in performance improvement. Furthermore, after applying these optimizations, which clearly meet the throughput needs for our applications, the TLLB still demonstrates significant energy savings over the NoC.

Of course, there are optimizations to improve the latency of the router and to minimize network diameter. But these come at even higher energy costs and can have limited effectiveness. For example, when we idealize the router delay to 1 cycle, the improved bus still shows a speedup of 1.19x over all applications (Figure 7.42-b). It is only when we use a flattened butterfly topology with the idealized 1-cycle router, that the NoC is outperforming the improved bus by 1.04x.

In practice, these speculative, look-ahead routers can only achieve 1-cycle routing delay in the best case. And using higher radix routers (to enable topologies like flattened butterfly) do not fundamentally change the total routing delays, but only reduces hop at the expense of increasing per-hop router delay [75]. Our models of NoC, especially with flattened butterfly topology, are only capturing the benefits not the performance costs – nor any energy costs. These models are providing an increasingly loose upper-bound for their performance potentials.
Figure 7.44: The average injection rate of particular application mapped to the predicted uniform random traffic generator. Red represents the highest injection rate, around 3 flits/cycle. Red represents the range of Group 3’s offered load, yellow corresponds to Group 2, and green to Group 1.

### 7.8.5 Scaling Up

It is a little tricky to study the impact of these techniques in an even larger system. The parallelism of the applications, the simulation environment, and the data set all start to reach or pass their fidelity limit and will contribute significant noise towards the measurements. So instead of trying to simulate more cores and threads, we do the following two things to escalate only the traffic on the network. First, we turn off the data placement optimization. Second, we make each core a standalone node. We hope to use this environment only to shed some light on the techniques’ impact in a larger scale environment, not to predict precise performance or energy gains.

Figure 7.45 shows the comparison of the NoC, bus, and a few configurations of improved bus. All results are shown as normalized to baseline bus. In such an environment, the baseline bus is under far more pressure and, at less than half of the performance of ideal interconnect, it
is significantly slower than NoC on average. The several improved bus designs all outperform the NoC, while using 9-25x less energy in the network.

With these analyses, including the limited scaling study, some insights can be obtained:

1. A bus architecture can be augmented with various techniques to be a viable solution even for large-scale CMPs.

2. Sometimes, these techniques come at a non-trivial cost in interconnect energy efficiency. Nevertheless, compared to the NoC approach, the energy cost is still much smaller.

3. Applications demonstrate a diverse set of behaviors that call for an adaptive control mechanism that can increase throughput on demand at the cost of extra energy overhead. NoC, on the other hand, operates at a point that provides high throughput at a significant cost of energy and latency.

7.9 Summary

In this chapter, we first introduced a transmission line link-based bus to use as a drop-in replacement for the traditional NoC. Through simple circuit techniques and minimal optimizations, this shared communication medium provides significant energy advantages over the NoC, while also
providing more than sufficient performance for a 16-core system. However, scaling such a simple system to 64-cores requires more insight.

Next, we have discussed an array of techniques to enhance throughput of transmission line buses via increasing the utilization efficiency, leveraging the bus properties and transmission line capabilities to reduce traffic demand, and to directly increase the raw link throughput. Among these techniques, those that increase the raw throughput often carry a higher energy cost for the same performance benefit. Even so, the energy cost is still far lower than that of using a NoC.

In a 64-core, 16-node system, when a number of techniques are applied, the performance of the system is improved by 1.39x and is 1.34x faster than the same system using a mesh NoC. This performance improvement is achieved with a 2x increase in interconnect energy, but a decrease of 12% of chip energy compared to the baseline bus thanks to faster execution. Compared to the mesh NoC, the network energy is still 8x lower, while the chip energy is 36% lower. In a limited scaling study where interconnect traffic is escalated, the throughput-augmented bus continues to outperform NoC almost consistently across all benchmarks.

In summary, transmission line-based links are a readily available mechanism to provide high-speed low-power communication. Using these links to build bus structures is a compelling alternative to NoC and other technologies such as on-chip photonics, which are far from being practical in the near term. The effective throughput of bus can be significantly increased with simple, practical designs. As a result, a bus architecture can support chip-multiprocessors at the scale of 64 cores competently with a much better energy profile than NoC. These findings bring into doubt the necessity of heavy-duty packet switching for on-chip interconnect in the foreseeable future.
Chapter 8

Future Directions

There are still unexplored options that should be fully tested in order to maximize the potential of the transmission line link bus architecture. For example, as with many interconnects, providing quality-of-service is a common design goal. Determining which messages are a priority, and which aren’t, and providing different levels of service across the TLLB structure could have significant performance implications. Another broad future direction is that of varying topologies. A shared bus was chosen for its simplicity, and the meandering layout was one implementation of a global bus. However, there are myriad choices. Some potential directions are listed here.

It is important to note that while some of the future directions, as well as some already proposed ideas, focus on the scalability of the TLLB system, scalability should not necessarily be viewed as the main goal. While it is an important design consideration, the energy, latency, and performance of a system of a given size should also be examined.

8.1 Quality of Service

Not all messages in a system necessarily need equal priority. We have seen some evidence of this when manipulating the transmission line bus allocations and partitions. For example, providing meta packets with a dedicated channel, rather than queuing behind data packets has a significant impact on performance. Additionally, synchronization communication benefits from a dedicated bus, as we saw with the Boolean bus. Similar steps can be taken within each general,
non-dedicated bus structure. In addition to synchronization primitives, some other messages, on demand miss replies for example, should be given higher priority, as resources of the CMP may be idle if these messages have not arrived. Conversely, some messages need lower priority. For example, loads which are more likely to be mispredicted, writebacks, and prefetch loads are not on the critical path, and thus can be delayed with little performance impact. Prioritizing all messages would offer these high priority messages less waiting time, while the low priority messages could increase their wait time. An analogy can be made to an emergency room: some patients clearly need help sooner than others. However, no patient should sit idle forever.

To demonstrate some of the potential for providing quality-of-service (QoS) algorithms, we present a simple experiment. If a message is a synchronization message, it is marked as high priority. If it is a writeback, prefetch load miss, or a deeply speculative load miss (a load which occurs after many unexecuted branches), then it is given low priority. All other messages are left as is. When a node is waiting to send a message, high priority messages will be sent before waiting low priority messages. If a message has been queued for long enough (for example, 10 messages have been sent without it moving, or perhaps 100 cycles), then its priority is increased. Figure 8.1 illustrates the state of an output queue from a node. Of course, logic must be added to ensure that a waiting prefetch miss or miss reply, or a speculative miss, does not become high priority while it is waiting (say all branches are resolved, or an on-demand miss occurs to a previously prefetched line).

Figure 8.2 demonstrates that even a simple design such as this can have non-trivial effect for some applications. Overall, providing QoS should result in no degradation, and while for a few individual benchmarks, we see negative performance impact, on average for all benchmarks, the QoS design provides a 1.02x increase in performance, with a maximum of 1.13x.

Further investigation into (a) what constitutes high or low priority, and (b) what is the ideal algorithm for promotion/demotion, and (c) how prioritized messages traverse the queue and network would potentially yield significant performance benefits. For example, allocating a particular set of lines to carry high priority messages could free these messages from being queued behind relatively unimportant packets. Providing such a “critical path” bus may take advantage of better allocation of the transmission line links.
Figure 8.1: A logical snapshot of a prioritized output queue, with a high priority message being inserted from the node.

Figure 8.2: Results for a preliminary QoS algorithm. The baseline is a 16-node, 64-core system, as in Chapter 7.

8.2 TLL Allocation

8.2.1 Serialization Exploration

So far, the transmission lines have been allocated to increase parallelization, and balance available throughput and serialization latency. Originally, all buses were wide enough to send entire
packets within a single computation cycle. As the network is scaled for throughput, the data buses are narrowed to increase throughput, allowing control packets to transmit in a single cycle, and data packets to transmit in just a few cycles. It is possible to continue to narrow the bus, increasing the serialization latency and utilization. Additionally, instead of a global medium where each transceiver is on the same line, perhaps buses can be dedicated to a particular transmitter or receiver. For example, each bus could handle 1 transmitter and 15 receivers, decreasing contention for the line. This would difficult to accomplish without drastically reducing the number of lines per bus.

A rudimentary and extreme example would be 45 lines, each acting as its own bus. Sending 8 bits per clock cycle, a 72-bit control packet would require 9 transmission cycles, and a 288-bit data packet would require 36 cycles. One possible allocation scheme is shown in Figure 8.3, where each line has a single transmitter and multiple receivers.

To truly understand the design choices for transmission line link allocation, we must study the impact of dedicating lines to particular nodes and transmitters, and of different groupings (e.g., 4 bus groups each with 4 transmitters, or 8 bus groups each with 2 transmitters).

![Figure 8.3: A TLL allocation scheme with a single line connecting 1 transmitter to multiple receivers.](image)
For large data packets, the increase in serialization, especially in a single line case, would be
dramatic, and possibly detrimental. Data packets are typically done at the granularity of cache
lines. If the granularity of the data requests is reduced, for example, to a single cache block
(rather than 4 blocks which make up a cache line), the allocation of the available transmission
lines can be re-evaluated, without concern for drastically increasing the serialization latency.
Additionally, word-based granularity is more feasible with fast, energy-efficient network struc-
tures.

8.2.2 Circuit-Switched Networks

Of course, there are many more topologies and layouts possible for transmission line links. In
this work, a circuit switched network was proposed, but kept in the same shape as the bus.
The segmented bus is one form of a circuit-switched network, using transmission line buses.
Figure 8.4 shows a number of options for 16 nodes (including the proposed design, for com-
parison), as well as 2 new options for 64 nodes. Each is intended to reduce the overall latency,
while providing maximum bandwidth. It is important to note that these topologies are linked
together using some form of circuit-switching, not packet-switching. This is important to keep
atomicity, as well as avoid creating a reduced packet-switched architecture, as in [128].

These new topologies’ performances are likely a result of the traffic pattern of a given ap-
lication. If traffic is mostly nearest neighbor, creating local buses which can be connected
to create a long bus only when needed would have high performance, while, similar to the
proposed segmented bus, uniform traffic may result hindered performance.

8.3 Hybrid TLLB and Packet-Switching Designs

The future directions described above use the TLLB as the sole interconnect. However, in
many-core designs, the bus will hit a fundamental limit as the traffic overwhelms the throughput
capabilities of the TLLB. All of the proposed optimizations are an attempt to push that limit
farther into the future, allowing simpler interconnect designs for larger systems. However, at
some point, the throughput demand will be significantly higher than the bus’s capability. At this
Figure 8.4: A sampling of circuit switched topologies. The “S” represents a circuit-switch to connect segments/branches to link remote nodes. While this could be a router, the focus here is to move away from packet-switching, so the switch is assumed to be a simple circuit switch. The size of the segments and switch placement are meant to be an example, not a fixed or proven quantity. The top 2 rows are for 16 nodes, and the last row is for 64 nodes. The proposed buses are shown in the upper left and upper middle figures.

point, a combination of different interconnects may be an attractive option. We have seen hybrid designs before [128]. In this case, a merging of packet-switching and the transmission line bus will be able to provide on-chip communication which is fast, energy efficient, and scalable. For example, partitions of 32 to 64 cores can share a single TLLB, while these partitions can
be connected together using packet-switched interconnects. Such a design would provide fast, low-power local communication without hindering the throughput of the larger global system. The underlying TLLB would also reduce the complexity of the packet-switched interconnect, reducing overall hop count and keeping the diameter of packet-switched layer at a minimum. Further study could develop such a system, making a hybrid design could extend the lifetime of the TLLB system into the many-core era, handling well over the already proposed 64-cores.

Overall, in order to create a scalable interconnect solution for many-core systems, a more in-depth evaluation of the available topologies will likely be necessary.
Conclusions

Transistor scaling and improving process technologies no longer translates directly to increased processor speed as it has in the past. And the growing number of cores integrated on chip puts pressure on the communication network structure of the system. It is possible to exploit the improving transistor performance in order to incorporate high-speed analog communication (i.e., RF) circuits, for both on-chip clocking and interconnect backbone. The faster transistors allow for higher sensitivity and higher frequencies for on-chip analog circuits.

First, an alternative to traditional H- or X-tree clocking networks, typically driven by a PLL, was proposed. These networks are power hungry, often requiring up to 30% of the total chip power. By using analog circuits, called injection-locked oscillators, we can reduce the complexity of the clocking network, removing intermediate buffers. Additionally, the de-skew capability and low jitter of the injection-locked clocking scheme, results in smaller noise margins, reduced up to 30ps in a 3GHz network. By reducing the margins, a higher percentage of a given clock cycle can be dedicated to logic, allowing for a reduction in the required $V_{dd}$ (by more than 5% chip-wide). As shown in Chapter 4, the injection-locked clocking scheme results in close to 20% savings of the overall chip power, for a single processor system.

High-speed analog communication circuits can also be used to provide a high performance, shared-medium interconnect backbone. Chapter 6 takes a first-step towards exploring the vast design space, including transmission lines (including topologies and physical parameters) and transceiver circuits. Experiments show that a single transmission line can provide high band-
width and throughput, with even simple encoding, and multiple lines can provide significant aggregate throughput (more than a terabyte), at high frequency and with low attenuation and dispersion over a reasonably wide spectrum. The link, including the transmission lines and transceiver circuits, can consist of standard components and practical circuits, while providing a 26.4GHz shared-medium, point-to-point interconnect substrate, with a total aggregate throughput over 1Tb/s.

Next, an interconnection network based on these links was presented. Specifically, the speed-of-light propagation delay links provide opportunities for globally shared-medium point-to-point communication. Chapter 7 shows that even with a simple architectural design, these TLLs can provide more than comparable performance, at much lower energy costs, when compared to packet-switched interconnect structures, which provide scalable throughput at the expense of latency and energy. In particular this simple design shows great potential as an alternative to packet-switched interconnects for small and moderately sized chip multiprocessors, with more than 15% performance improvement, with up to 26x network energy savings, for a 16-core system.

The transmission line link bus structure offers significant throughput, but unlike packet-switched architectures, an increase in the number of cores does not necessarily result in throughput scalability. As a result, as more cores are integrated and the throughput requirements increase, it is necessary to increase the available throughput and/or decrease the traffic demand on the network. To fit this need, this work explored more sophisticated techniques, increasing the scalability of the system, allowing the TLL bus structure to scale to larger systems. This scaled version improves the overall performance (1.42x over NoC), and in particular, increases the performance of benchmarks with a need for high throughput, while maintaining lower overall energy costs (1.6x lower than NoC).

In the future, we plan to explore alternative line allocation schemes as well as techniques for improving the quality-of-service for the interconnect backbone.

Overall, incorporating high-speed, high-quality, and low-power analog circuits into the design of on-chip communication circuits for chip multiprocessors provide significant opportu-
nities, drastically reducing overall energy while not compromising system performance, and therefore can set the stage for improvement in future microprocessor design.
Bibliography


[60] IBM. IBM PowerPC 970FX RISC Microprocessor Data Sheet, April 2006. SA14-2760-06.


