ENERGY SCALABILITY OF ON-CHIP INTERCONNECTION NETWORKS

by

THEODOROS K. KONSTANTAKOPOULOS

Diploma ECE, University of Patras, Greece, 2000
M.S. EECS, Massachusetts Institute of Technology, 2002

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2007

Author .....
Department of Electrical Engineering and Computer Science
May 25, 2007

Certified by .....
Anant Agarwal, Professor
Department of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

ENERGY SCALABILITY OF On-Chip Interconnection NETWORKS

by

THEODOROS K. KONSTANTAKOPOULOS

Submitted to the Department of Electrical Engineering and Computer Science
on May 25, 2007, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

ABSTRACT

On-chip interconnection networks (OCN) such as point-to-point networks and buses form the communication backbone in multiprocessor systems-on-a-chip, multicore processors, and tiled processors. OCNs consume significant portions of a chip's energy budget, so their energy analysis early in the design cycle becomes important for architectural design decisions.

Although innumerable studies have examined OCN implementation and performance, there have been few energy analysis studies. This thesis develops an analytical framework for energy estimation in OCNs, for any given topology and arbitrary communication patterns, and presents OCN energy results based on both analytical communication models and real network traces from applications running on a tiled multicore processor. This thesis is the first work to address communication locality in analyzing multicore interconnect energy and to use real multicore interconnect traces extensively. The thesis compares the energy performance of point-to-point networks with buses for varying degrees of communication locality. The model accounts for wire length, switch energy, and network contention. This work is the first to examine network contention from the energy standpoint. The thesis presents a detailed analysis of the energy costs of a switch and shows that the estimated values for channel energy, switch control logic energy, and switch queue buffer energy are 34.5pJ, 17pJ, and 12pJ, respectively.

The results suggest that a one-dimensional point-to-point network results in approximately 66% energy savings over a bus for 16 or more processors, while a two-dimensional network saves over 82%, when the processors communicate with each other with equal likelihood. The savings increase with locality. Analysis of the effect of contention on OCNs for the Raw tiled microprocessor reports a maximum energy overhead of 23% due to resource contention in the interconnection network.

Thesis Supervisor: Anant Agarwal
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I feel privileged to have worked with so many talented people during my time at MIT. I would have never achieved any of this without the support, guidance, and direction from my advisor, Prof. Anant Agarwal. His supervision helped me move many academic steps ahead. Working with him has been a great intellectual experience. Anant was great in making me look at the big picture in times where I was lost in the details of research. My research has greatly benefited from his technical guidance and his steady stream of enthusiasm.

I would also like to thank Prof. Anantha Chandrakasan and Prof. Arvind for serving as members of my committee. They were always there for me when I needed them. They provided valuable feedback on my thesis and helped with the clarity of the presentation from the view of an external reader.

Many thanks to all the members of the Raw group. I have spent a lot of time with them working in different projects, also doing things outside research. Several people have contributed directly to the work in this thesis and I would like to point out their contribution. Paul Johnson ran the benchmarks and provided me with the trace information that I used in the thesis for the model validation. Jason Miller has been an extremely helpful labmate and officemate. He was always there to answer all my questions on the Raw microprocessor. His help is greatly appreciated. Jonathan Eastep spent a significant amount of his time going over my thesis and providing valuable feedback on all aspects of it. He has been extremely helpful in pointing out ways to present the results of my research in a more clear and straightforward way.

Special thanks to my good friend Duke Xanthopoulos for providing valuable feedback on many aspects of the thesis. Our discussions have really helped me all this time. Petros Boufounos has been there for me whenever Matlab was giving me nightmares. Paul-Peter

Sotiriadis has been very helpful in the initial parts of my research when I first started thinking about the models and the framework.

I couldn’t thank more my parents for everything they have done for me. They have always been encouraging me to try different things in life, respecting my decisions and helping me achieve my goals. I am extremely grateful for that. All members of my family have been extremely supportive; they were a constant force of strength all seven years I spent at MIT.

Throughout my life at MIT, I was privileged to meet so many people that I could call my friends. Anna has been a great support all this time and she was there to help me bring some balance in my life. A lot of people had a positive impact in my life: Aggelos, Angelina, Anna, Anna, Christina, Christina, Duke, Eleftheria, Giorgos, Kostas, Margarita, Maria-Katerina, Nikolas, Nikol, Pavlos, Petros, Ramin, Sebastien, Thodoros, Vasilis, Yiannis, Yiannis.

Last but not least, I would like to thank all the MIT faculty and the administrative stuff for doing a perfect job in providing excellent standards of education and a very intellectually stimulating environment.

TABLE OF CONTENTS

Chapter 1 Introduction 17

1.1 Contributions 24

1.2 Thesis Roadmap 27

Chapter 2 Background 31

2.1 The Raw Microprocessor [1] 32

2.1.1 Overview and Design Philosophy 32

2.1.2 The Processing Core 33

2.1.3 Communication Networks 34

2.1.4 Raw Implementation 35

2.1.5 Energy Summary 36

2.2 Tiled Architectures 37

2.2.1 The Stanford Smart Memories Project [26] 37

2.2.2 The UWM Multiscalar [30] 38
2.2.3 The U. of Washington WaveScalar [28] 39
2.2.4 The UT-Austin TRIPS architecture [29] 40
2.2.5 The UC Davis Synchroscalar [31] 41
Summary 41

Chapter 3 A Framework for Energy Analysis 43
3.1 Energy Analysis Framework 44
3.2 Workload Model 44
3.3 Bus Interconnection Model 45
3.4 Point-to-Point Interconnection Network Model 46
3.5 Traffic Distributions 48
3.5.1 Communication Energy Cost 48
3.5.2 Uniform Distribution 49
3.6 Energy Comparison of the Two Systems 51
3.7 Localized Traffic Distributions 54
3.7.1 Linear Decay 55
3.7.2 Exponential Decay 57
3.7.3 Step Distribution 59
3.7.4 Truncated Linear Decay 61
3.7.5 Truncated Exponential Decay 63
3.7.6 Savings Comparison for Different Distributions 64
Summary 66

Chapter 4 Two-Dimensional Interconnection Networks 69
4.1 Interconnection Network Model for 2-D Systems 69
4.2 Communication Energy Cost 71
4.3 Uniform Distribution 73
4.4 Energy Comparison of a 2-D Network with 1-D Network and Bus 74
4.5 Localized Traffic Distributions 77
4.5.1 Uniform Distribution 80
4.5.2 Linear Decay 80
4.5.3 Exponential Decay 81
4.5.4 Step Distribution 81

4.5.5 Truncated Linear Decay 82
4.5.6 Truncated Exponential Decay 82
Summary 83

Chapter 5 Energy Savings and Model Validation Using Network Traces 85
5.1 Network Traces 85
5.2 Trace Communication Statistics 89
5.3 Energy Savings 90
5.4 Model Validation 94
Summary 97

Chapter 6 Switch Energy and Wire Length 99
6.1 Switch Energy 100
6.2 Switch Energy and Varying Wire Lengths 100
6.3 High-Order Dimension Mapping 102
6.4 Calculation of Matrices D and H 103
6.5 Energy Comparison for High-Dimensional Networks 105
Summary 108

Chapter 7 Contention Energy Analysis for Multi-Dimensional Networks 111
7.1 Switch Model 111
7.2 Life of a Message in the Network 115
7.3 Calculation of Contention Probability 119
7.3.1 Contention Probability In One-Dimensional Networks 128
7.3.2 Contention Probability In Bus-Based Networks 129
7.4 Energy Consumption Assuming Uniform Distribution 131
7.5 Energy Consumption Using Network Traces 135
Summary 142

Chapter 8 Future Work 145
8.1 Advantages of Distributed vs. Centralized Hardware Resources 145
8.1.1 Cache Architecture 146
8.1.2 Energy Savings for Distributed Caches 148
8.2 Network Topologies 151

8.3 Traces 151
8.4 Technology Scaling Effect....152
8.5 Energy Performance of Tiled Processor Architectures 154

Chapter 9 Related Work 155

Chapter 10 Conclusions....161

Nomenclature....165

Appendix A Calculation of Energy Costs....169
A.1 Link Capacitance Energy Cost Estimation 169
A.2 Crossbar and Control Logic Energy Cost Estimation 171
A.3 Input Queue Buffer Energy Cost 172

Appendix B Communication Patterns in Raw Applications 177
B.1 APDCM 178
B.2 AES 178
B.3 AES_FIX 179
B.4 BTRIX....179
B.5 CHOLESKY....180
B.6 FPPPP 180
B.7 JACOBI 181
B.8 JACOBI_BIG 181
B.9 LIFE....182
B.10 MXM....182
B.11 SHA....183
B.12 SWIM 183
B.13 TOMCATV....184
B.14 VPENTA....184

Appendix C Three-Dimensional Interconnection Networks 185
C.1 Interconnection Energy for 3-D Systems 185
C.2 Uniform Distribution....187

LIST OF FIGURES

Chapter 1

Fig. 1.1: Transistor Trend [20]. Number of transistors used in microprocessors over the last twenty five years. 18
Fig. 1.2: Frequency Trend [20]. Reported operating frequency for different microprocessors over the last twenty five years. 19
Fig. 1.3: Power Trend[20]. Reported microprocessor power consumption over the last twenty five years. 20

Chapter 2

Fig. 2.1: The Raw Microprocessor. A Raw processor consists of a 2-D array of uniformly replicated tiles, each containing a MIPS-style processor and two types of network routers to connect it to the neighboring tiles. 33
Fig. 2.2: Raw Die Photo 36

Chapter 3

Fig. 3.1: Energy Analysis Framework. The framework considers the network topology, the workload model, and energy estimates of the network components to calculate the energy dissipated in the interconnection network. 44
Fig. 3.2: Workload Model. There are N processors in the network. Each processors wants to transmit M data words. The processor communication patterns are specified by the traffic model. 44
Fig. 3.3: Bus-Based Model. There are N processors in the system. Each processor wants to transmit M data words.....45
Fig. 3.4: Point-to-Point Interconnection Network Based Model. There are N processors in the network. Each processor wants to transmit M data words. The switch is responsible for re-transmitting the data or sending them to the attached processor. 46
Fig. 3.5: A One-Dimensional Bi-Directional Point-to-Point Network with no End-Around Connections. 47
Fig. 3.6: Energy Costs for a Network Hop. The total energy consists of the energy dissipated in the link that connects two neighboring processors and the energy dissipated in the switch.52
Fig. 3.7: Energy Savings for Uniform Distribution (1-D vs. Bus). 54
Fig. 3.8: Linear Decay Probability Distribution. Probability of Communication Between Processors P1, P4, and P10 with all the Processors in the Network. 56
Fig. 3.9: Percentage Energy Savings for 1-D vs. Bus for Linear Decay Distributions for Three Pairs of (b, a). 57
Fig. 3.10: Exponential Decay Probability Distribution. Probability of Communication Between Processors P1, P4, and P10 with all the Processors in the Network. 58
Fig. 3.11: Percentage Energy Savings for 1-D vs. Bus for Exponential Decay Distributions for Three Pairs of (b, d). 59
Fig. 3.12: Exponential Decay Probability Distribution. Probability of Communication Between Processors P1, P4, and P10 with all the Processors in the Network. 60
Fig. 3.13: Percentage Energy Savings for 1-D vs. Bus for Step Distributions for Three Different Radii. 61
Fig. 3.14: Truncated Linear Decay Probability Distribution. Probability of Communication Between Processors P1, P4, and P10 with all the Processors in the network. 62
Fig. 3.15: Percentage Energy Savings for 1-D vs. Bus for Truncated Linear Decay Distributions for Three Different Radii.....62

Fig. 3.16: Truncated Exponential Decay Probability Distribution. Probability of Communication between Processors P1, P4, and P10 with all the Processors in the Network. 63
Fig. 3.17: Percentage Energy Savings for 1-D vs. Bus for Truncated Exponential Decay Distributions for Three Different Radii. 64
Fig. 3.18: Savings Comparison for All Distributions. 65

Chapter 4

Fig. 4.1: 2-D Point-to-Point Network Model: N Processors, each transmitting M data words on a 2-D mesh network. There are X processors in each row and Y processors in each column. 69
Fig. 4.2: Energy Consumption Savings Assuming Uniform Distribution (Bus vs. 2-D Mesh). 76
Fig. 4.3: Energy Consumption Savings Assuming Uniform Distribution (1-D Mesh vs. 2-D Mesh. 77
Fig. 4.4: Energy Consumption Savings Assuming Uniform Distribution (1-D Mesh vs. Bus and 2-D Mesh vs. Bus). 80
Fig. 4.5: Energy Consumption Savings Assuming Linear Distribution (1-D Mesh vs. Bus and 2-D Mesh vs. Bus). 80
Fig. 4.6: Energy Consumption Savings Assuming Exponential Distribution (1-D Mesh vs. Bus and 2-D Mesh vs. Bus). 81
Fig. 4.7: Energy Consumption Savings Assuming Step Distribution (1-D Mesh vs. Bus and 2-D Mesh vs. Bus). 81
Fig. 4.8: Energy Consumption Savings Assuming Truncated Linear Distribution (1-D Mesh vs. Bus and 2-D Mesh vs. Bus). 82
Fig. 4.9: Energy Consumption Savings Assuming Truncated Exponential Distribution (1-D Mesh vs. Bus and 2-D Mesh vs. Bus). 82

Chapter 5

Fig. 5.1: Number of Messages Transferred from One Processor to Other Processors in the Network for sha and btrix. 88
Fig. 5.2: Distributions of Communication Patterns for All Applications. 90
Fig. 5.3: Savings Over Different Numbers of Tiles for the 14 Applications when No

Contention is Assumed in the Network Switches.....92
Fig. 5.4: Benchmark Locality Characteristics. Average Distance in Number of Hops for Each Application. 94
Fig. 5.5: Energy Savings (2-D vs. Bus) for Benchmarks and Modeled Communication Patterns. 95

Chapter 6

Fig. 6.1: 64-Tile System Implemented in Three Dimensions. There are 4 Processors in Each Dimension. 102
Fig. 6.2: Mapping of the 64 Tiles Into Two Dimensions. The Logical Distance Between P1 and P49 is 3 Hops and the Physical Distance is 12 “Physical Hops”. 103
Fig. 6.3: Energy Comparison for 2-D, 3-D and 4-D Networks with 256 Nodes.....107

Chapter 7

Fig. 7.1: Typical Microarchitecture of An On-Chip Network Switch for 2-D Mesh. 112
Fig. 7.2: Physical Placement of the Crossbar, the Control Circuitry, the Input Queue Buffer, the Channels for the Dynamic Network in Raw. 114
Fig. 7.3: Expected Energy Consumption of a Message for One Network Hop.....117
Fig. 7.4: Example of a Message Route on a 2-D Network.....118
Fig. 7.5: Process Describing the Total Expected Energy Cost for Moving a Message from Pi to Pj.....119
Fig. 7.6: Channel Utilization at a Switch. The channel utilization r is composed of three components: ri (messages generated/consumed by the node processor), rc (messages continuing to this direction through the switch), and rs (messages switching to this direction at the switch). (a) shows how r is composed and (b) shows how r is decomposed.....122
Fig. 7.7: Probability of Contention for Uniform Distribution.....124
Fig. 7.8: Probability of Contention for Uniform Distribution when the Queue is Empty.125
Fig. 7.9: Approximations of the Probability of Contention when the Queue is Empty for Uniform Distribution. 127
Fig. 7.10: Probability of Contention for a Two- and One-Dimensional Network, and a Bus-Based System. 131
Fig. 7.11: Total energy consumption in a 8-by-8 system for different values of channel utilization. Each processors transmits 1000 words and processors communicate with equal likelihood with

each other. 133
Fig. 7.12: Energy Overhead due to Contention for the Uniform Distribution. 135
Fig. 7.13: Probability of Contention for Uniform Distribution in the Raw Static Switch. ... 136
Fig. 7.14: Communication Pattern: mxm. The graphs shows the total number of messages generated by the processors in the system and their destinations. 139
Fig. 7.15: Communication Pattern: btrix. The graphs shows the total number of messages generated by the processors in the system and their destinations. 140
Fig. 7.16: Upper and Lower Energy Overhead Bounds for Benchmarks Compared to the Energy Overhead Assuming Uniform Distribution..... 141

Chapter 8

Fig. 8.1: a) Systems with centralized cache structures result in long interconnection buses within the cache for accessing the contents of a sub-bank. b) In tiled architectures large structures are distributed across the chip area, saving power when the compiler exploits data locality, compared to centralized structures. 148
Fig. 8.2: Exponential Decay Probability Distribution. 150

Appendix A

Fig. A.1: Message Path from the East Output of a Tile to the Switch Input Port of the Neighboring Tile on the East. 170
Fig. A.2: Schematic of Dynamic Output Block. The block includes the CrossBar block, control logic for the CrossBar and for the control signal to the input block. 174
Fig. A.3: Schematic of Dynamic Input Block. The block includes the Input Buffer, control logic for the Input Buffer and the generation of route_req and tail signals. 175

Appendix B

Fig. B.1: Communication Pattern: adpcm. 178
Fig. B.2: Communication Pattern: aes. 178
Fig. B.3: Communication Pattern: aes_fix. 179
Fig. B.4: Communication Pattern: btrix. 179
Fig. B.5: Communication Pattern: cholesky. 180

Fig. B.6: Communication Pattern: fpppp. 180
Fig. B.7: Communication Pattern: jacobi. 181
Fig. B.8: Communication Pattern: jacobi_big. 181
Fig. B.9: Communication Pattern: life. 182
Fig. B.10: Communication Pattern: mxm. 182
Fig. B.11: Communication Pattern: sha. 183
Fig. B.12: Communication Pattern: swim. 183
Fig. B.13: Communication Pattern: tomcatv. 184
Fig. B.14: Communication Pattern: vpenta. 184

Appendix C

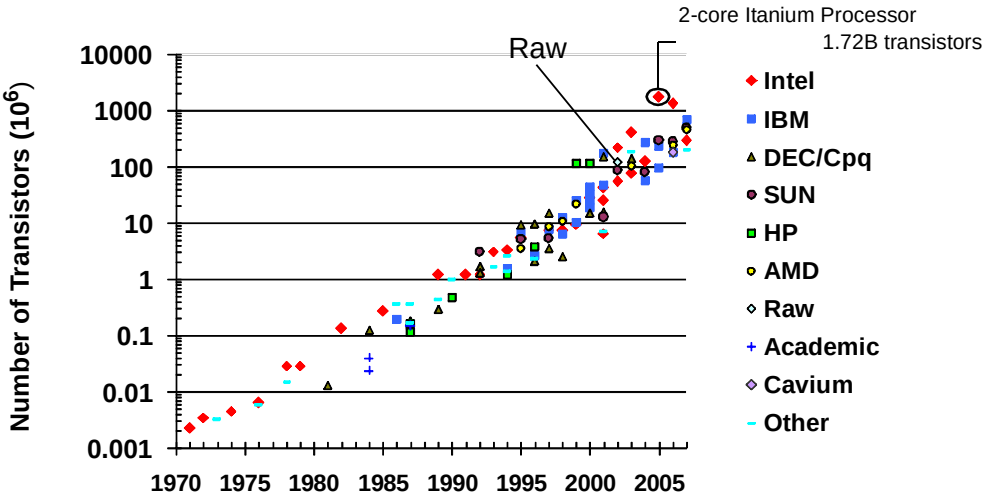
Fig. C.1: Normalized Energy - Uniform Distribution for 1-D, 2-D and 3-D mesh networks with same numbers of processors. 188

CHAPTER 1

INTRODUCTION

Microprocessor performance has increased dramatically in the last three decades. Moore's Law predicted that the number of transistors in integrated circuits would double every eighteen months. Fig. 1.1 illustrates Moore's Law [22]. The minimum feature size in production integrated circuits has continued on an exponential decline since the first integrated circuit appeared. Moreover, the average selling price per transistor has fallen over six orders of magnitude in the last thirty years [23]. The graph reveals that Moore's Law still holds, despite the escalating technological challenges, which means that the availability of transistors and resources will grow for microprocessor designers. Previously, computer designers have taken advantage of these resources largely to build systems with centralized structures such as superscalar and pipelined processors with caches, due to the simpler programming model compared to systems with distributed structures.

Figure 1.1: **Transistor Trend [20]. Number of transistors used in microprocessors over the last twenty five years.**

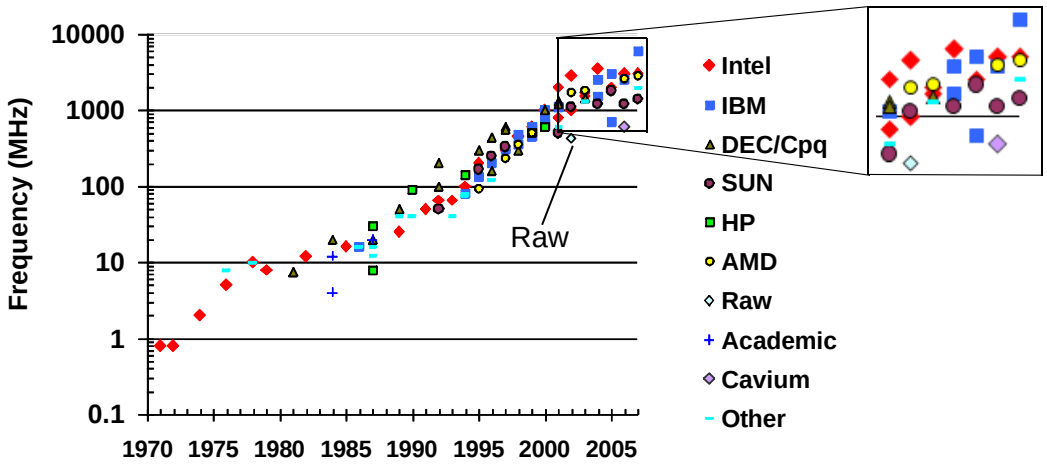


Source: ISSCC Proceedings 1971-2007

Contrary to the transistor trend shown in the previous graph, the operating frequency trend (Fig. 1.2) reveals that the frequency has reached a maximum in 2004 [21], with the exception of IBM's Power6 microprocessor [59]. Moreover, predictions for the operating frequency in future microprocessors indicate that it has leveled off. The perception of building deeper and more complex pipelines for complex superscalar machines has been mainly driving the microprocessor industry, leading to the increase in the operating frequency.

Power and energy consumption issues, however, have become a limiting factor in modern processors and make the need for different directions in computer architecture imminent. Power and wire constraints will drive processors to explicitly parallel modular architectures ([63], [10]). These constraints have led to the development of several research projects aiming to explore scalable designs such as the MIT Raw microprocessor [1], the Stanford Smart Memories project [26], the Stanford Merrimac-Streaming Supercomputer [27], the MIT Scale project [25], the U. of Wisconsin Multiscalar [30], the UW WaveScalar [28], the UT Austin TRIPS [29], the UC Davis Synchroscalar [31].

Figure 1.2: **Frequency Trend [20]. Reported operating frequency for different microprocessors over the last twenty five years.**

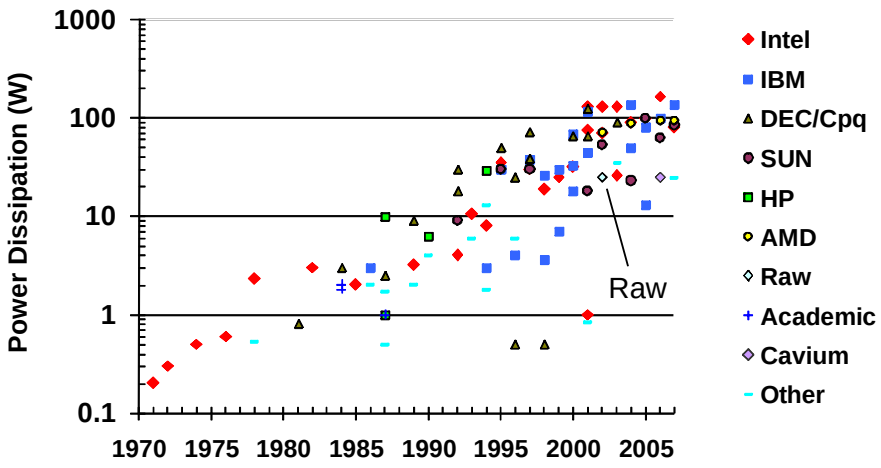


Source: ISSCC Proceedings 1971-2007

Modern superscalar processors with large out-of-order instruction issue widths, register renaming units, multi-level caches, and other performance-enhancing features have begun to yield diminishing returns on performance. An even more important issue than the operating frequency wall is the tremendous amount of power that such structures typically consume, giving rise to today's power-hungry commercial processors ([9], [15], [16], [17]). The power in modern microprocessors reached a barrier in 2000, as Fig. 1.3 clearly shows. This graph presents the reported power dissipation of microprocessors in the last twenty-five years.

Power consumption and wire delays have limited the continued scaling of centralized systems [63], while making multicore architectures increasingly popular. To emphasize the above statement, Intel cancelled the next step in the Pentium 4 processor line (code name Tejas) in 2004 because of power issues, and they are refocusing on multicore processors to hit new power and performance targets. They have recently announced [49] an 80-tile network-on-chip architecture with a processing engine on each tile. All the major semiconductor companies follow this trend toward multicore solutions; this trend will continue all the way to tiled processor

Figure 1.3: **Power Trend[20]. Reported microprocessor power consumption over the last twenty five years.**



Source: ISSCC Proceedings 1971-2007

architectures as it is one of the most power efficient ways to exploit technology scaling. Tiled architectures use point-to-point interconnection networks for data communication; this thesis shows that point-to-point networks is an energy efficient way to communicate data in systems with multiple cores.

A number of recent studies have shown that implementing architectures that attempt to break up large centralized structures into smaller, more localized ones seems to be an effective way to alleviate both the performance and power scaling issues. Taylor et al. [2] demonstrated that the long wire delay caused by these large centralized structures is effectively mitigated by a tiled architecture. Additionally, Zyuban and Kogge [6] address the above-mentioned power issues by proposing a distributed architecture that is inherently lower-power without compromising performance, while showing that centralized superscalar architectures, even when optimized for energy efficiency, are inherently energy-inefficient.

Tiled processor architectures have been proposed as a way to allow the performance of high-performance microprocessors to scale along with processor designers' exponentially-increasing

transistor budgets. They are distributed processor architectures aimed at exploiting fine-grain concurrency, even from a single programmer-specified thread. Tiled architectures reduce power by breaking up computation into multiple independent tasks, which can potentially decrease power consumption without sacrificing performance [7]. They typically incorporate a number of tiles that are replicated across the chip and connected via an on-chip network, thus simplifying VLSI design complexity. Furthermore, intelligent compilers exploit data locality, allowing computation to be scheduled and positioned in such way that communication costs are low, thereby reducing overall power consumption.

Wire delays, power limitations, and complexity limitations are moving VLSI processor designs towards tiled architectures. Tiled processor architectures attempt to mitigate these issues by organizing the processor resources in a more power-efficient manner. For instance, they typically distribute large caches and register files across the chip, keeping the size of each individual structure small. Compilers optimized for and targeted toward tiled architectures exploit data locality and localized computation and can eliminate heavy network activity for operand communication across tiles.

As the demand for on-chip bandwidth between different processing elements in multicore systems scales up, point-to-point interconnection networks are becoming a necessity for on-chip communication. Riding this wave, as the number of processing elements scale up in multicore and tiled processor architectures, the study of on-chip interconnection networks (OCN) — the medium for processor-to-processor or memory communication — becomes extremely important. Although network performance has been extensively studied previously (e.g., [13], [51], [52], [12], [53]), the power and energy of OCNs have not been explored as rigorously. As the energy consumption in OCNs increases [54], energy estimation tools that can provide a comparison of different architectures and applications for various network traffic patterns early in the design cycle become extremely useful to the computer architect.

This thesis proposes an energy analysis framework for on-chip interconnection networks that can serve as a basis to model (a) multi-dimensional point-to-point networks or buses that transfer data between processors, (b) OCNs that connect distributed resources such as caches on chip, or for that matter (c) networks that communicate data between different components, such as ALUs and register files.

This work shows that point-to-point interconnection networks have significant energy advantages over bus-based networks. Our framework demonstrates how energy savings depend on the number of nodes in the network and the degree of communication locality. We present our analysis for a one-dimensional point-to-point network and a bus-based network and show that the one-dimensional OCN results in approximately 66% energy savings over a bus for 16 or more processors, even when the communication patterns are uniformly distributed.

Increasing the network dimensionality to two dimensions results in additional energy savings. We show that for uniformly distributed communication patterns moving from one to two dimensions results in energy savings of $O(\sqrt{N})$, where N is the number of processors in the system. Applications that exhibit communication locality result in significantly greater energy savings. As an example, the results that use the analytical model and are confirmed with traces of real benchmarks show that the energy of a 2-D OCN can be 10 times lower than that of a bus for 16 processors when the applications show communication locality (e.g., ADPCM), and about 5 times lower better when there is poor locality (e.g., btrix).

With the advent of tiled architectures [1], [26], [28], [56], [31], compilers become increasingly responsible for balancing computational parallelism and communication locality. As such, the resulting communication patterns of applications produce widely differing on-chip network energy consumption. Using network traces from a set of benchmarks compiled and run on a tiled processor, we quantify the degree to which applications with greater communication locality are more energy-efficient.

The concept of communication locality is increasingly important in two-dimensional realizations of on-chip networks. The importance occurs because there is a significant trade-off between the energy savings that result from the smaller logical distance between processors and the increased energy dissipation due to the greater wire lengths [12] and hence greater capacitance between processors, as the higher dimensions are mapped into the plane. We investigate this trade-off by comparing the total energy consumption of systems with the same number of processors but with a different dimensionality. For example, we show that for a chip with 256 processing cores, a 2-D mesh is more energy efficient than a 3-D OCN under the condition that the energy of the switch logic is no greater than 0.5 times the energy of a network channel connecting a pair of physically adjacent cores.

Communication locality depends on the examining benchmark as well as on the partitioning and placement of data on the different tiles of the system. Early-stage energy and power estimation of multicore chips is extremely important in providing compilers information in determining the most efficient code partitioning and data and instruction placement across the cores.

Network contention of network resources results in message delays and increased energy dissipation in the switch, when messages are written into queueing buffers waiting to be serviced by a specific output port. This thesis examines the effect of contention on the energy dissipated in interconnection networks. We derive a closed-form solution for the energy for various channel utilization values assuming processors communicate to each other with equal likelihood. We used energy estimates for the energy dissipated in the interconnection networks in the Raw microprocessor to quantify the energy overhead and showed that the maximum amount of additional overhead paid is 23.3%. Additionally, using network traces we estimate the lower and upper bounds for the energy dissipation in the communication network for the different applications that we examine.

1.1 Contributions

This thesis proposes a practical analytical model to describe the energy characteristics of on-chip networks, demonstrating the importance of computation-communication locality and providing a simple framework for application experts, architects, compilers, and run-time systems to reason about energy-centric properties of their codes and systems.

We demonstrate the use of probability distributions to model various network traffic patterns and show, using these distributions as well as network traces, that the energy dissipation in on-chip point-to-point networks is inversely proportional to the distance that messages travel in the network. The examination of the communication characteristics of network traces from applications running on a tiled multicore processor suggests that probability distributions is an effective and accurate way of modeling on-chip traffic patterns. Therefore, our framework can be used to estimate the energy consumption in on-chip interconnection networks without running the applications.

We present a contention energy analysis and derive a set of closed-form equations for the probability of contention in buses, one-, and two-dimensional networks. The analysis suggests that in point-to-point networks the most important and common source of contention for new messages entering a network switch is messages in the switch waiting to be serviced. We show that there is an upper limit on the energy cost expended due to contention and calculate this cost for a tiled multicore processor as 23%.

We estimate the energy dissipation of the different components in the switch using extracted capacitance values from the layout of a tiled processor. We follow the path of a message from the output of one tile to the output of the neighboring one and estimate the energy dissipation in the channels that connect the two tiles, the logic circuitry that generates the control signals required for routing the message from the switch input to the output, and the input queue buffer where the message is stored when it suffers contention.

This thesis differs from previous work on energy analysis of OCNs [36], [39], [41], [42], by making the following unique contributions:

(1) Energy Analysis Framework

This work proposes an energy analysis framework for on-chip interconnection networks that can serve as a basis to model: (a) multi-dimensional point-to-point networks or buses that transfer data between processors; (b) OCNs that connect distributed resources such as on-chip caches; or for that matter (c) networks that communicate data between different components such as ALUs and register files.

We develop a framework for the energy analysis of OCNs that can be used with any topology (buses, tori, meshes, etc.) and arbitrary communication patterns (uniform, truncated exponential, patterns measured from a real workload, etc.).

We present a set of probability distributions to model various traffic patterns that closely match the communication characteristics of various benchmarks. These distributions can be used to represent any level of locality among the processors in the system and have a true point-to-point nature.

We propose the use of the analytical framework to compare the energy performance of a centralized multi-bank cache organization (or any centralized structure for that matter) to a tiled architecture that arranges cache memories across the total chip area.

(2) Extensive Use of Real Network Traces

In the analysis, we extensively use real network traces from benchmarks running on the Raw microprocessor to compare the energy performance of OCNs and to validate the analytical model. Using both the analytical model and network traces, we quantify the positive

impact of communication locality on the energy dissipated in the on-chip interconnection network.

If we can estimate what the communication among tiles would look like, we can model these communication patterns and have interconnection network energy reports using our framework without running the applications and recording the network traces. Benchmark simulation can be expensive and not extremely useful if the designers are in the phase where they are making architectural decisions.

(3) Contention Energy Analysis

We analyze the effect of contention on OCNs to the total energy consumption and propose message scheduling advice that reduces the energy overhead of network delays. We derive a closed-form solution for the energy dissipation on the network for various values of channel utilization and calculate the energy overhead of contention on the total energy dissipation in point-to-point interconnection networks.

Additionally, using information collected from applications running on the Raw multicore processor we present upper and lower bounds of the energy dissipated on the raw network for scalar operand communication.

(4) Network Hardware Components Energy Estimates

The thesis presents a detailed analysis of the energy costs of a switch. We follow a low level approach in our methodology based on capacitance values from the Raw microprocessor dynamic networks. For wiring and metal capacitance values, we use the extracted capacitance values generated by the IBM ChipEdit capacitance extractor tool for the final layout of the Raw microprocessor. For the cell input and output capacitances we use the values provided by IBM for their cells in the SA-27E process. We show that the measured

values for channel energy, switch control logic energy, and switch queue buffer energy are 34.5pJ, 17pJ, and 12pJ, respectively.

(5) Network Switch Analysis

We investigate the effect on energy of two-dimensional realizations of high-order networks (three- and four-dimensional networks) and examine the effect of switch energy and wire lengths under various traffic pattern assumptions, highlighting the trade-offs between the logical and physical distance between processors in the system.

We examine how the choice of the most energy-aware network topology is based on the communication locality inherent on the application by comparing the total energy consumption of a two-, three, and four-dimensional network for applications that exhibit different locality characteristics.

(6) Analysis Results Comparing Buses, 1-D, and 2-D Networks

The framework compares the energy of buses, 1-D, and 2-D point-to-point networks, including the impact of wire lengths and related capacitance, and communication locality. We quantify the energy savings moving from bus-based architectures to point-to-point topologies for applications that exhibit no communication locality, as well as other applications with various levels of locality.

1.2 Thesis Roadmap

The remainder of the thesis is organized as follows:

- **Chapter 2** gives a background on the MIT Raw Microprocessor and presents other existing tiled processor architectures.

- **Chapter 3** describes our energy analysis framework and presents an analysis of the energy advantages moving from a bus-based system to a one-dimensional point-to-point interconnected system. We present two simple interconnection models and investigate the power performance for various communication patterns. We derive closed-form equations that describe the energy savings of the point-to-point networks compared to bus-based systems and investigate the effect of locality of communication on the total energy dissipation. In our analysis, we use different probability distributions to model various locality characteristics for the traffic patterns on the interconnection network.
- **Chapter 4** moves the analysis of Chapter 3 to two-dimensional interconnection systems. We analyze the communication costs for a uniform and for various localized traffic patterns. We provide formulas that explain the relation between the energy dissipated on a two-dimensional point-to-point interconnection network and one-dimensional point-to-point networks as well as bus-based systems.
- **Chapter 5** provides a validation of our model using network traces from benchmarks running on the Raw [1] microprocessor for different tile configurations. We examine the different locality characteristics of the network traces and their effect on the total energy dissipated.
- **Chapter 6** examines the energy characteristics of high-dimensional networks. We enhance our model to accommodate any realization of high-order networks, after they are mapped into a two-dimensional substrate. We investigate the effect of the logical and physical distances between two processors on the total energy for communication, assuming traffic patterns that allow localized and non-localized communication.
- **Chapter 7** presents an analysis of the effects of network contention to the total energy dissipation in on-chip interconnection networks. In our analysis we use energy costs for

major components of a network router, that are based on an actual fabricated switch, the network switch of the Raw microprocessor. We calculate the probability of contention on a specific output port assuming different channel utilization patterns. Furthermore, we provide energy estimates for the total energy consumption on the interconnection network, taking into account the probability of contention on every node of the system assuming a uniform distribution for the communication among processors. We extend our analysis to provide upper and lower bounds of the energy consumption in the Raw networks for different benchmarks running on a 16-tile configuration.

- **Chapter 8** describes possible future directions of this study.
- **Chapter 9** presents the related work on high-level power and energy estimation tools and describes the differences between each work compared to the work presented in this thesis.
- **Chapter 10** concludes the thesis.
- **Appendix A** describes the methodology for estimating the energy costs for the various components of the network switch for the Raw microprocessor. These energy costs are used in the contention analysis in Chapter 7.
- **Appendix B** presents graphs with the communication patterns evident in the applications that we examine in this thesis.
- **Appendix C** presents an analysis for the energy estimation for a point-to-point network implemented in three dimensions.

CHAPTER 2

BACKGROUND

The energy analysis of interconnection networks has a direct application to tiled processor architectures. With process scaling and increasing number of cores in multicore systems, point-to-point interconnection networks are becoming a necessity for on-chip communication, as the demand for on-chip bandwidth between different processing elements scales up.

This chapter presents existing tiled processor architectures that can utilize our framework to perform high-level calculations on the energy dissipated on the interconnection networks. We examine the following multiprocessor projects: the MIT Raw microprocessor [1], the Stanford Smart Memories project [26], the U. of Wisconsin Multiscalar [30], the UW WaveScalar project [28], the UT Austin TRIPS [29], and the UC Davis Synchroscalar project [31]. The reader familiar with the Raw microprocessor and tiled processor architectures can proceed to Chapter 3.

The first challenge of tiled architectures [34] is to determine the best allocation of silicon resources among the computing, memory and communication needs. The second challenge is determining the granularity and the number of the tiles of the system. Addressing these problems calls for several models. The optimal balancing and evaluation of all the design trade-offs require an architecture model, an application model as well as a cost model.

2.1 The Raw Microprocessor [1]

This section describes the MIT Raw Microprocessor. Raw is a research architecture design undertaken by the MIT Raw research team and was fabricated by IBM. We provide a more detailed description of Raw compared to the other tiled processor architectures, because in future chapters we will be using some of its network components for energy characterization as well as network trace information collected from various benchmarks running on the Raw microprocessor.

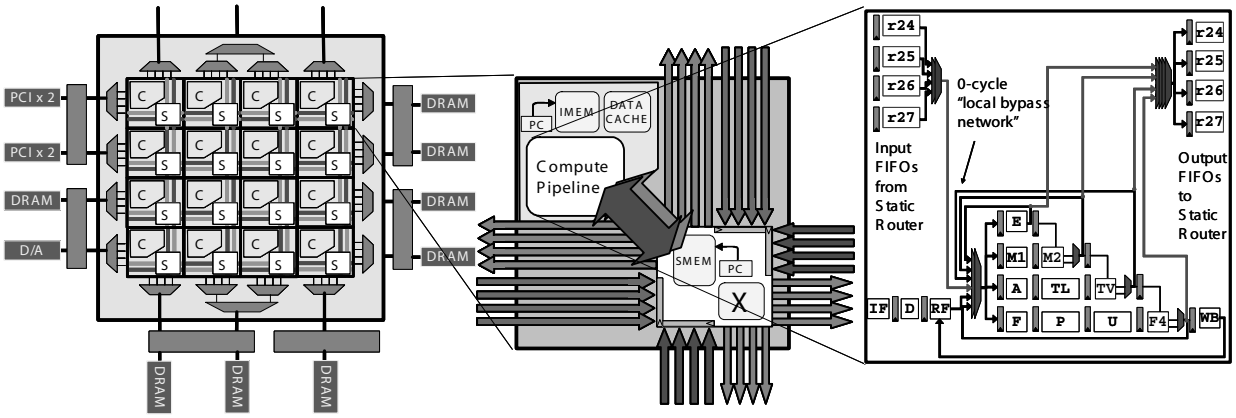
The network traces provide relevant information on the traffic volume, distance, and locality of communication. They are not particular reflective of contention.

2.1.1 Overview and Design Philosophy

The Raw architecture exposes the details of the hardware to the programmer (or the compiler) making parallelism explicit, instead of using expensive hardware structures to hide the true nature of the processor. For example, the software should specify separate instruction streams for each functional unit.

Additionally, Raw eliminates large, centralized structures for the purpose of scalability. In traditional superscalar processors, the area and delay of structures like bypass networks and register files grow with the square or cube of the issue width. As these structures grow, the wires within them grow to the point that a signal can no longer traverse them in a single clock cycle. In that case, additional pipeline stages should be added or the clock frequency must be reduced.

Figure 2.1: **The Raw Microprocessor. A Raw processor consists of a 2-D array of uniformly replicated tiles, each containing a MIPS-style processor and two types of network routers to connect it to the neighboring tiles.**



Raw on the other hand retains only the essential hardware, enabling the placement of many more functional units on chip than a superscalar can. Using distributed computational elements, the Raw architecture addresses scalability.

Besides the advantages of scalability and efficient use of die area, a Raw processor is significantly cheaper and easier to build than a monolithic superscalar. All of the tiles are identical and relatively simple. The effort required to build a Raw processor is essentially the same as that required to build a single tile.

2.1.2 The Processing Core

Fig. 2.1[1] illustrates the basic architectural characteristics of Raw. The tiles are interconnected by four 32-bit full-duplex on-chip networks, consisting of over 12,500 wires. Two of the networks are static (routes are specified at compile time) and two are dynamic (routes are specified at run time). Each tile is connected only to its four neighbors. Every wire is registered at the input to its destination tile, which means that the longest wire in the system is no greater than the length or

1.Reproduced with permission from Michael Taylor

width of a tile. The tile is sized so that a signal travels through a small amount of logic and across the tile in one clock cycle. This property ensures high clock speeds, and the continued scalability of the architecture.

The Raw chip is divided into an array of 16 identical uniformly-replicated programmable tiles. A tile contains an 8-stage, in-order, single-issue MIPS-style processing pipeline, a 4-stage single-precision pipelined FPU, a 32KB data cache, two types of communication routers, static and dynamic, and 32KB and 64KB of software-managed instruction caches for the processing pipeline and static router respectively. These tiles are general purpose in nature and each can run its own independent instruction stream.

The exposed ISA allows parallel applications to exploit all of the chip resources, including gates, wires and pins. While obviously conducive to data- and task-level parallelism, Raw supports instruction level parallelism by spreading computation across cores and scheduling operands over the low-latency static networks. The Raw compiler can do this automatically managing the effect of wire delays by orchestrating both scalar and stream data transport.

2.1.3 Communication Networks

The Raw processor has four 32-bit bi-directional on-chip mesh networks. There are two types of networks, *static* and *dynamic*. The two static networks are statically scheduled and explicitly managed. The two dynamic networks are routed dynamically at run-time.

Static Networks

The static networks provide a low-latency, high-bandwidth connection between the tiles. They are intended for known communication that can be statically scheduled at compile time. The static router can be directly programmed to control the flow of data on the static networks.

Each static network provides a 32-bit full-duplex network link between each tile processor, its next neighbor, and the other static network. These networks are register-mapped and integrated

into the bypass network in each processing pipeline. This results in high-speed communication on the static network with an ALU-to-ALU latency of 3 cycles, between two neighboring tiles.

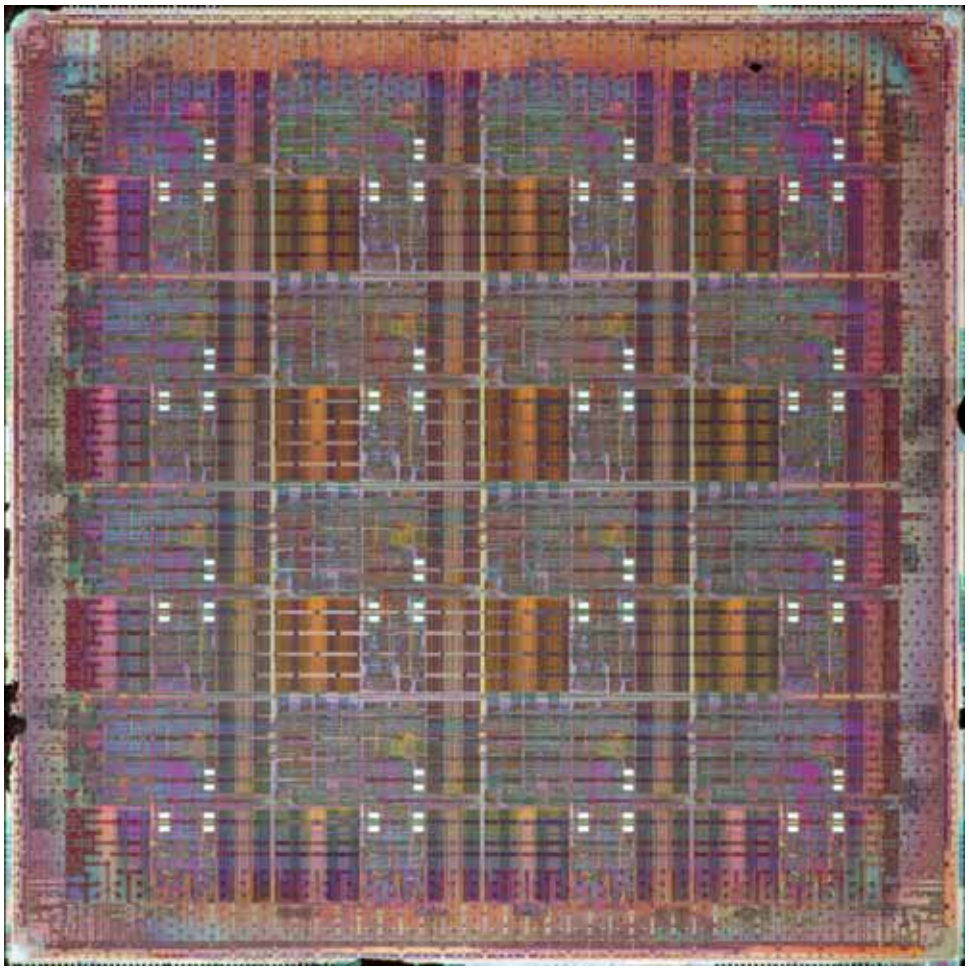
In the thesis we will be using trace information from the static networks for different benchmarks running on Raw. Those traces are for operand communication between tiles. There is an approximate cost of $504pJ$ [42] due to cache accesses but in our analysis we will not account cache energy.

Dynamic Networks

The second type of communication network on Raw is dynamic. There are two completely separate identical dynamic networks: the MDN (memory dynamic network) and the GDN (general dynamic network). The dynamic networks handle traffic which is not statically predictable such as cache misses, external interrupts, and data dependent communication patterns. The MDN is used for accessing on-chip resources including memory, the interrupt controller, and I/O devices. In contrast, the GDN may be used freely by an application. Since GDN usage is unrestricted, there is the potential for it to deadlock due to overflow of receive buffers, but the GDN has deadlock recovery mechanism.

The dynamic networks are dimension-ordered, wormhole-routed networks. Each message on the dynamic network consists of a header word with information about the message and up to 31 data words. Information specified in the header work includes the message length and type, as well as the destination tile. The messages travel in the network first in the X dimension and then in the Y dimension, causing the routers along the path to create a worm-hole for the remaining message. The routers forward the remaining words of a message until the whole message has passed by. Then, they are free to examine their input buffer queues to start servicing other messages.

Figure 2.2: **Raw Die Photo**



2.1.4 Raw Implementation

The Raw multicore processor was implemented in IBM's 180nm, 6-layer copper, CMOS 7SF standard-cell ASIC process. The choice of 16 tiles was determined by the die size available. The die area is 18.2mm X 18.2mm, although the tiles take up only 16mm X 16mm of this area. The larger die size was necessary to accommodate the column grid array (CGA) package in order to achieve a higher pin count. The package has 1657 total pins of which 1080 pins are available for use as high speed transceiver logic (HTSL) I/O pins. Fig. 2.2 shows a die photo of the entire 16 tile Raw processor.

2.1.5 Energy Summary

Kim et al. [42] present measurements of energy consumption in the Raw microprocessor. In their study they show that on idle state (when the clock is grounded) the chip draws a leakage current of $28mA$ and dissipates $45mW$.

Additionally, they examined the average current of an application with average instruction mix running on a single Raw tile, which showed that the compute power consumes a very small fraction of the power compared to the clock power. Therefore, they concluded that implementing clock-gating at the tile-level is highly desirable for tiled architectures.

The work in [42] continued examining the energy costs of communication over the two types of network in the Raw multicore processor. The measured numbers for the static and dynamic networks are $85pJ$ and $90pJ$, respectively. Both numbers are measured for a maximum toggle-rate per word; consecutive words injected to network would cause the channel lines to alternate on every cycle, so these values correspond to the maximum energy dissipation per cycle. This thesis describes a methodology in Appendix A that calculates the average and maximum energy costs per hop for the dynamic network; these values are $51.5pJ$ and $89pJ$, respectively.

A more detailed discussion of the microprocessor can be found in [1], [2], [3], [19], [63].

2.2 Tiled Architectures

In this section we present a brief overview of other academic projects on tiled processor architectures.

2.2.1 The Stanford Smart Memories Project [26]

The Stanford Smart Memories project is a research effort to design a single-chip computing element which provides configurable hardware support for diverse computing models and maps efficiently to future wire-limited VLSI technologies. Smart Memories is a partitioned, explicitly parallel, reconfigurable architecture for use as a future universal computing element. Finding a

single topology that fits well with all applications, which have different communication patterns and memory needs, is very difficult. With Smart Memories the appearance of the on-chip memory, interconnection network, and processing elements is tailored to better match the application requirements.

There are 64 tiles on a Smart Memories chip. Each tile contains processing, memory and communication resources. The tile processor is a 64-bit processor and the instruction and data accesses are interleaved on to tile crossbar on alternate phases of the clock. Four tiles form a quad. The 16 quads communicate over a global network. The intra-quad interconnect has four 64-bit phase-pipelined broadcast buses, which can be statically or dynamically allocated.

Smart Memories is designed to efficiently support different programming models to allow applications be programmed and run in the model that gives the best performance and programming ease. Different programming models are supported in the Smart Memories system by reconfiguring memory system to provide the memory access requirements for each model. The three major programming models are: the shared memory, multi-thread mode [48], the streaming mode, and the transactional coherence and consistency model.

2.2.2 The UWM Multiscalar [30]

Multiscalar processors use an aggressive implementation paradigm for extracting large quantities of instruction level parallelism from ordinary high-level language programs. A single program is divided into a collection of tasks by a combination of software and hardware. The tasks are distributed to a number of parallel processing units which reside within a processor complex. Each of the parallel processing elements operates on its task using its own program counter and physical copy of the single logical register file. Register results are dynamically routed among the many parallel processing units with the help of compiler-generated masks.

Memory accesses may occur speculatively without knowledge of preceding loads or stores. Addresses are disambiguated dynamically, many in parallel, and processing waits only for true data dependencies. Data dependencies are resolved by a combination of hardware and software, with hardware being given more responsibility compared to currently used instruction level parallelism paradigms.

2.2.3 The U. of Washington WaveScalar [28]

WaveScalar is a dataflow instruction set architecture and execution model designed for scalable, low-complexity/high-performance processors. The WaveScalar ISA is designed to run on an intelligent memory system. Each instruction in a WaveScalar binary executes in-place in the memory system and explicitly communicates with its dependents in dataflow fashion. Conceptually, a WaveScalar binary is the dataflow graph of an executable and resides in memory as a collection of intelligent instruction words. Each instruction word is intelligent, because it has a dedicated functional unit. In practice, since placing a functional unit at each word of instruction memory is impractical, an intelligent instruction cache called a WaveCache, holds the current working set of instructions and executes them in place.

WaveScalar architectures cache instructions and the values they operate on in a WaveCache, a simple grid of “alu-in-cache” nodes. By co-locating computation and data in physical space, the WaveCache minimizes long wire, high-latency communication.

The WaveCache is a grid of approximately 2K processing elements (PEs) arranged into clusters of 16. Each PE contains logic to control instruction placement and execution, input and output queues for instruction operands, communication logic, and a functional unit. Each PE also contains buffering and storage for 8 different instructions, bringing the total WaveCache capacity to 16 thousand instructions, which is equivalent to a 64KB instruction cache in a modern RISC machine. The input queues for each input require only one write and one read port and as few as

2 entries per instruction, or 16 entries total. The input queues are indexed relative to the current wave and a small, multi-ported RAM holds full-empty bits for each entry in the input queues. Matching logic accesses and updates the bits as new inputs arrive, obviating the need for content addressable memories.

Within a cluster, the processing elements communicate via a set of shared buses. Tiles within the same cluster receive results at the end of the clock cycle in which they were computed. Cluster size is one of the key architectural parameters of the WaveCache. Larger clusters require more wires and more area for intra-cluster communication, while smaller clusters increase inter-cluster communication costs.

For inter-cluster communication, the WaveCache uses a dynamically routed on-chip network. Each hop in the network crosses one cluster and takes a single cycle.

2.2.4 The UT-Austin TRIPS architecture [29]

The TRIPS architecture is an example of an Explicit Data Graph Execution (EDGE) architecture that supports a static placement, dynamic issue (SPDI) execution model. EDGE architectures, unlike RISC and CISC instruction sets, explicitly encode dependencies into individual instructions. This encoding permits dataflow-like execution without the hardware overheads of conventional out-of-order processors, in which the hardware must reconstruct dependences on the fly.

The TRIPS architecture is hierarchical with a system composed of multiple TRIPS chips, and each chip composed of multiple processing and memory elements. A TRIPS chip includes 8 processors, a collection of on-chip secondary memory arrays, and off-chip channels to external DRAM and other TRIPS chips. Each processor employs a Grid Processor Architecture consisting of an 8x8 array of ALUs, a local register file, local instruction and data caches, and control circuits. Both the grid processor and the memory arrays are configurable to enable efficient

execution of multiple program domains. The chip also includes a sensor network and a small embedded monitor processor to dynamically detect application behavior and changes in system behavior. This information is fed back to the runtime system, the application, and the compiler for on-line optimization.

TRIPS programs are compiled into graphs of predicated hyperblocks, each of which is represented internally as a dataflow graph, with instructions communicating directly though instruction-encoded dependences. Each hyperblock has a set of input and output registers, which is how communication occurs between them. The TRIPS architecture supports up to a maximum of eight 128-instruction hyperblocks executed on a processor core simultaneously, thus enabling a 1,024 instruction window.

2.2.5 The UC Davis Synchroscalar [31]

Synchroscalar is a tile-based architecture for embedded processing. It is designed to provide the flexibility of DSPs while approaching the power efficiency of ASICs. This goal is achieved by providing high parallelism and voltage scaling while minimizing control and communication costs. Specifically, Synchroscalar uses columns of processor tiles organized into statically-assigned frequency-voltage domains to minimize power consumption. Furthermore, while columns use SIMD control to minimize overhead, data-dependent computations can be supported by extremely flexible statically-scheduled communication between columns.

Summary

This chapter presented a brief overview of current academic tiled processor architecture projects. The most common feature of tiled architectures is the distribution of the silicon resources across the chip area. The lack of centralized control and centralized structures enables scalability for these designs.

Another major common characteristic of tiled architectures is the desire for data locality for improved performance and power dissipation savings. However, it is obvious from examining the diversity of the projects presented, that there is no “best” allocation of computing, memory or communication resources. The diversity of applications along with architectural trade-offs and design costs determine how VLSI resources are distributed.

CHAPTER 3

A FRAMEWORK FOR ENERGY ANALYSIS

This chapter develops a framework for energy analysis in on-chip interconnection networks. We present an analysis of two simple processor interconnection models, a bus-based model and a point-to-point network model. Initially we develop our model for a one-dimensional mesh network (we examine two-dimensional and higher-dimensional networks in Chapter 4 and Chapter 6) and present the advantages of point-to-point interconnection network systems compared to bus-based systems in terms of energy consumption, assuming different network traffic characteristics.

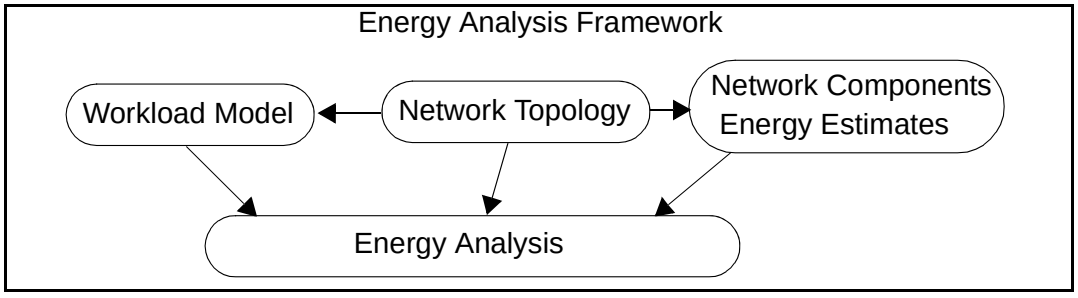
We present the workload model and the two interconnection models. Then we describe the network traffic patterns that we apply to the point-to-point model. For each pattern we calculate the total communication energy and compare it with the energy consumed in the bus-based model. We derive closed-form equations that describe the energy savings of the point-to-point

model assuming a uniform communication distribution (when processors communicate with equal likelihood) and investigate the effect of locality of communication on the total energy dissipation.

3.1 Energy Analysis Framework

Fig. 3.2 describes the energy analysis framework that we propose.

Figure 3.1: **Energy Analysis Framework. The framework considers the network topology, the workload model, and energy estimates of the network components to calculate the energy dissipated in the interconnection network.**



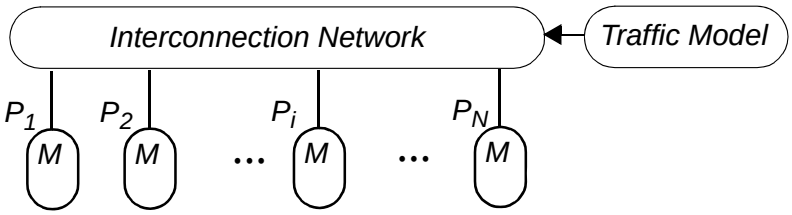
The framework takes into account the topology and size of the interconnection network (mesh, ring, torus etc. and the number of processors), the workload model (number of messages sent by each processor in the network and traffic patterns in the network), and estimates of the energy of the network components (energy expended in the channels and the switch) to calculate the energy dissipation in the interconnection network.

The network topology also feeds the workload model and the network hardware energy estimates components of the framework. For example, processors might not communicate frequently if they are not physically located close to each other. Additionally, the number of processors in a bus-based system affects the total energy cost for accessing the bus.

3.2 Workload Model

Fig. 3.2 describes our workload model.

Figure 3.2: **Workload Model. There are *N* processors in the network. Each processors wants to transmit *M* data words. The processor communication patterns are specified by the traffic model.**

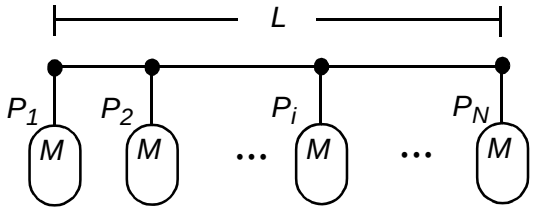


Our system consists of N processors $(P_1,...,P_N)$. The N processors are connected with an interconnection network. Our model does not exclude traffic patterns with unequal loads; however, we assume here for simplicity that each processor wants to transmit M data words as messages[1]. The M data words have as their destination other processors of the system. Applications that run on the system result in different data communication among processors. The different communication patterns are described by the traffic model.

3.3 Bus Interconnection Model

Fig. 3.3 depicts a simple bus-based machine model that we use in this analysis.

Figure 3.3: **Bus-Based Model. There are *N* processors in the system. Each processor wants to transmit *M* data words.**



In the bus model, when a processor transmits a data word, the data is available throughout the entire length L of the bus. The bus width is assumed to be a data word. After all the processors

1.We will mention how to model unequal traffic patterns, but will not analyze it in detail.

have sent their data, the total energy consumption E_{BUS} on the bus will be given by

$$E_{BUS} = M \cdot N \cdot E_L , \quad 3.1$$

where $E_L = \frac{1}{4} C_L V_{DD}^2$ is the average energy cost of accessing the bus, C_L is the total capacitance of the bus, and V_{DD} is the supply voltage.

The total energy supplied by an inverter for a low to high transition is $C_L V_{DD}^2$ [11]. Half of it is stored on the bus line as electrical capacitive energy and the remainder is dissipated as heat in the inverter output resistance. If the bus line is high, then no additional energy is required from the inverter to pull it high.

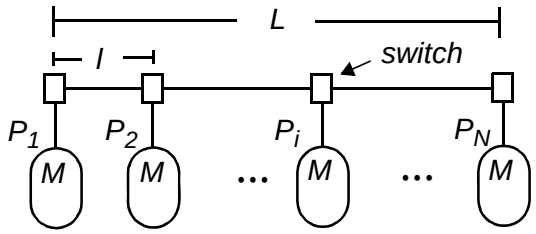
If the bus line is high and the inverter pulls it low, then there is no additional energy supplied by the inverter, but the stored energy on the bus line ($C_L V_{DD}^2 / 2$) is dissipated as heat in the inverter pull-down. If the bus line is low and the inverter pulls it low then there is no exchange of energy. The average of these cases is $C_L V_{DD}^2 / 4$. Even with repeaters, the total energy remains the same and only the bus delay changes.

3.4 Point-to-Point Interconnection Network Model

Fig. 3.4 shows the point-to-point model, where N processors are interconnected with a one-dimensional mesh network.

The total length of the bus is split into $N-1$ segments, each having length l and capacitance C_l; the width of each segment is also assumed to be a data word. When processor P_i wants to send data to processor P_j, the data are available only at the segments that connect P_i with P_j, and each switch is responsible for re-transmitting the data or sending them to the attached processor.

Figure 3.4: **Point-to-Point Interconnection Network Based Model. There are *N* processors in the network. Each processor wants to transmit *M* data words. The switch is responsible for re-transmitting the data or sending them to the attached processor.**



We will derive the general form for energy for any point-to-point network, and then specialize for a one-dimensional network.

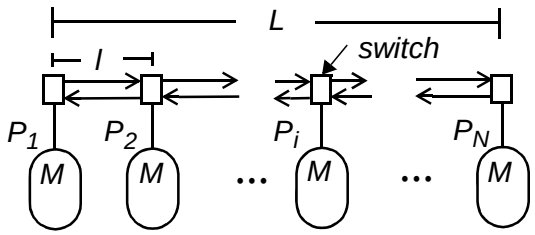
The total energy E_{P2P} when all processors have transmitted their data in any point-to-point network is given by

$$E_{P2P} = \sum_{i=1}^{N} \sum_{j=1}^{N} E_{i,j}, \qquad 3.2$$

where $E_{i,j}$ is the energy consumption due to transmitting of all data words that have as source processor P_i and destination processor P_j. The total energy as well as $E_{i,j}$ depend on the processor communication pattern. Eq. 3.2 applies to any one-, two- or high-dimensional point-to-point network. It is clear that $E_{i,j} = 0$, if P_i never sends a message to P_j.

For the remaining analysis, we assume that the network is bi-directional and that there are no end-around connections (Fig. 3.5). In other words, there are separate word-wide physical channels in both directions.

Figure 3.5: **A One-Dimensional Bi-Directional Point-to-Point Network with no End-Around Connections.**



3.5 Traffic Distributions

We examine six different probability distributions (uniform, linear decay, exponential decay, step, truncated linear decay, and truncated exponential decay) to model the traffic of the data for each processor. We investigate the effect of these network traffic distributions on the energy consumption for the point-to-point network model. We want to observe non-localized communication patterns, as well as localized ones. To model traffic patterns with various locality characteristics, we apply distributions that allow communication among all processors (linear decay and exponential decay), as well as distributions that allow communication among processors that are located within a specific radius (step distribution, truncated linear decay, and truncated exponential decay).

In Chapter 5 we incorporate into our framework traffic patterns taken from benchmarks running on the Raw microprocessor. At this point, our analysis ignores contention; doing so works in favor of the energy dissipation of the bus-based system; we show in Chapter 7 that contention in bus-based system is significantly greater compared to the contention in point-to-point networks. Even under the assumption of no contention, we will show that the energy efficiency of point-to-point networks is significant compared to bus-based systems.

3.5.1 Communication Energy Cost

If E_l is the energy cost of accessing a channel for one network hop and $p_{i,j}$ [1] is the probability that processor P_i communicates with processor P_j for a given data word, the expected energy cost $E_{i,j}$ of communicating data from processor P_i to processor P_j, for this network model, is given by

1.We define the probability $p_{i,j}$ as the probability that processor P_i communicates with processor P_j, where $p_{i,j}$ satisfies $\sum_{j=1}^{N} p_{i,j|i=I} = 1$, $p_{i,j} = 0$ for $i = j$, and $I = 1, 2, \ldots N$. The set of $p_{i,j}$ defines a communication probability matrix p.

$$E_{i,j} = M \cdot p_{i,j} \cdot E_l \cdot H_{i,j}, \quad 3.3$$

where each element $H_{i,j}$ (Eq. 3.4) shows the number of hops the data make when transmitted from processor P_i to processor P_j. The expected number of messages that P_i sends to P_j under the communication pattern described in the probability matrix p is $M \cdot p_{i,j}$. We assume that the probability $p_{i,j}$ is independent and identical for each data word sent from P_i to P_j. For a one-dimensional network matrix with no end-around connections H is given by

$$H = \begin{bmatrix} 0 & 1 & 2 & \dots & N-2 & N-1 \\ 1 & 0 & 1 & \dots & \dots & N-2 \\ 2 & 1 & 0 & 1 & \dots & \dots \\ \dots & \dots & 1 & 0 & 1 & 2 \\ N-2 & \dots & \dots & 1 & 0 & 1 \\ N-1 & N-2 & \dots & 2 & 1 & 0 \end{bmatrix}. \quad 3.4$$

For localized communication patterns, the communication probability $p_{i,j}$ for two processors in Eq. 3.3 decreases as the distance between the two processors increases.

Our analysis assumes that each processor wants to transmit M data words, which implies equal traffic loads. Our framework can model unequal traffic loads; to do so we modify Eq. 3.3 as

$$E_{i,j} = M_i \cdot p_{i,j} \cdot E_l \cdot H_{i,j}, \quad 3.5$$

where M is a vector and each element M_i holds the number of messages that P_i wants to transmit. In this case, the expected number of messages that P_i sends to P_j is $M_i \cdot p_{i,j}$.

3.5.2 Uniform Distribution

If there is no sense of communication locality in our system, a processor communicates with any other processor with equal probability. Therefore, in any point-to-point network, each message is equally likely to make $(1, 2, \dots, N-1)$ hops. So we can replace the communication probability $p_{i,j}$ of Eq. 3.3 with

$$p_{i,j} = \frac{1}{N-1}, \text{ for } i \neq j, \tag{3.6}$$

and get the following equation for the expected energy cost $E_{i,j}$ for the communication between processors P_i and P_j:

$$E_{i,j} = \frac{M}{N-1} \cdot E_l \cdot H_{i,j}. \tag{3.7}$$

The total expected energy cost of transmitting the data assuming uniform distribution (from Eq. 3.2) is

$$E_{P2P} = \sum_{i=1}^{N} \left(\sum_{j=1}^{N} E_{i,j} \right) = \frac{M}{N-1} \cdot E_l \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} H_{i,j}. \tag{3.8}$$

This formula is true for any point-to-point network. The matrix H captures the distance between processors in the network. In this chapter, we focus on one-dimensional point-to-point networks; we will use E_{1D} to denote the total expected energy E_{P2P} for one-dimensional networks. The next chapter discusses two-dimensional networks.

As shown in Eq. 3.4, matrix H is symmetric, thus we can compute the sum of the elements of the upper triangular matrix H_T (Eq. 3.9) and multiply by 2.

$$H_T = \begin{bmatrix} 0 & 1 & 2 & \dots & N-2 & N-1 \\ 0 & 0 & 1 & \dots & \dots & N-2 \\ 0 & 0 & 0 & 1 & \dots & \dots \\ \dots & \dots & 0 & 0 & 1 & 2 \\ 0 & \dots & \dots & 0 & 0 & 1 \\ 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix}. \tag{3.9}$$

We find the sum of the elements of matrix H_T, computing the sum of the elements on each diagonal.

$$\sum_{i=1}^{N} \sum_{j=1}^{N} H_{T_{i,j}} = \sum_{i=1}^{N-1} (N-i) \cdot i = \tag{3.10}$$

$$= \sum_{i=1}^{N-1} (N \cdot i - i^2) = \sum_{i=1}^{N-1} N \cdot i - \sum_{i=1}^{N-1} i^2 = \tag{3.11}$$

$$= N \cdot \sum_{i=1}^{N-1} i - \sum_{i=1}^{N-1} i^2 = N \cdot \left[\frac{N}{2} \cdot (N-1)\right] - \left[\frac{N}{6} \cdot (N-1) \cdot (2N-1)\right] = \tag{3.12}$$

$$= N \cdot (N-1) \cdot \left(\frac{N}{2} - \frac{2N}{6} + \frac{1}{6}\right) = \frac{N \cdot (N-1) \cdot (N+1)}{6}. \tag{3.13}$$

Therefore

$$\sum_{i=1}^{N} \sum_{j=1}^{N} H_{i,j} = 2 \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} H_{T(i,j)} = \frac{N \cdot (N-1) \cdot (N+1)}{3}. \tag{3.14}$$

Eq. 3.13 shows the sum of all distances between all pairs of processors. The total expected energy cost for a one-dimensional network from Eq. 3.8 is

$$E_{1D} = \frac{M}{N-1} \cdot E_l \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} H_{i,j} = \frac{M}{N-1} \cdot E_l \cdot \frac{N \cdot (N-1) \cdot (N+1)}{3} \tag{3.15}$$

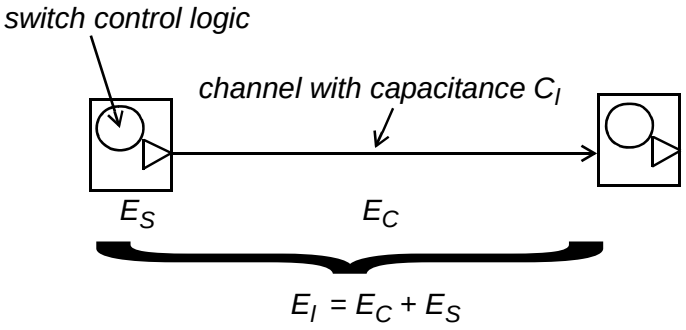
$$E_{1D} = M \cdot E_l \cdot \frac{N \cdot (N+1)}{3}. \tag{3.16}$$

This is the total energy to transmit M words per processor to other processors uniformly in a one-dimensional point-to-point network with bi-directional channels. Eq. 3.16 is consistent with the average distance $(\sim N/3)$ in a network assuming no end-around connections for large N [12]. Intuitively, this is the product of the number of words per processor (M), times the energy cost per network hop, times the number of processors in the network (N), times the average distance of a message $(\sim N/3)$.

3.6 Energy Comparison of the Two Systems

The ratio of the energy dissipated on the point-to-point network over the energy dissipated on the bus (Eq. 3.17 and Eq. 3.1) is

Figure 3.6: **Energy Costs for a Network Hop. The total energy consists of the energy dissipated in the link that connects two neighboring processors and the energy dissipated in the switch.**



$$\frac{E_{1D}}{E_{BUS}} = \frac{M \cdot E_l \cdot \frac{N \cdot (N+1)}{3}}{M \cdot N \cdot E_L} = \frac{N+1}{3} \cdot \frac{E_l}{E_L}, \qquad 3.17$$

where E_l is the energy cost of accessing one segment for one network hop and E_L is the energy cost of accessing the bus. E_l can be broken into two components: E_C, the energy for charging one bus segment of capacitance C_l, and E_S, the energy dissipated on the control logic of the switch (Fig. 3.6). Thus,

$$E_l = E_C + E_S. \qquad 3.18$$

E_S does not appear in the bus energy equation (Eq. 3.1) because there is no intermediate switching logic; therefore the energy required to charge the whole bus, E_L, is related to the energy of charging one segment, E_l, in the following manner

$$E_L = (N-1) \cdot E_C. \qquad 3.19$$

Using Eq. 3.17 and Eq. 3.18, Eq. 3.17 becomes

$$\frac{E_{1D}}{E_{BUS}} = \frac{N+1}{3} \cdot \left[\frac{E_C + E_S}{E_L}\right] \qquad 3.20$$

$$\frac{E_{1D}}{E_{BUS}} = \frac{N+1}{3} \cdot \left[\frac{E_L/(N-1)}{E_L} + \frac{E_S}{E_L}\right] \qquad 3.21$$

$$\frac{E_{1D}}{E_{BUS}} = \frac{N+1}{3} \cdot \left[\frac{1}{(N-1)} + \frac{E_S}{E_L}\right]. \qquad 3.22$$

Next, let's obtain asymptotic limits on the ratio.

For now, we assume that the energy overhead of the switch is small, relative to the required energy for accessing the full length of the bus (we investigate the effect of the switch power on the total energy of the network in Chapter 6). We are also underestimating the bus energy, because we do not consider the energy dissipated in the bus arbitration logic that replaces the switch in bus-based systems.

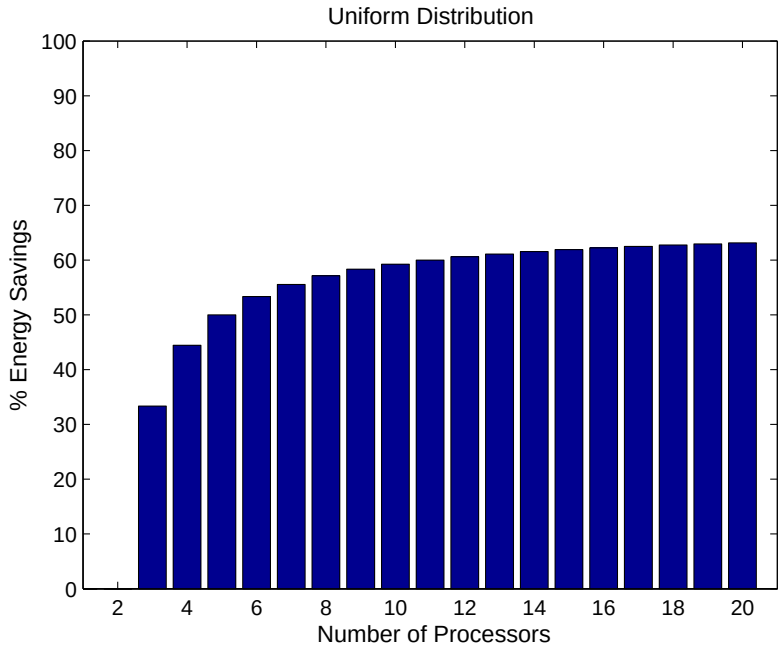
For large N, the previous equation becomes

$$\frac{E_{1D}}{E_{BUS}} = \frac{1}{3} \cdot \frac{N+1}{N-1} \approx \frac{1}{3}, \qquad 3.23$$

and shows that the energy dissipated in the bus is three times larger than the energy dissipated in the point-to-point network when processors communicate with equal likelihood. This makes intuitive sense because a bus requires charging its entire length, while a one-dimensional point-to-point network requires charging only a third of the entire length (recall that the average distance per message is $N/3$.

Fig. 3.7 shows the percentage savings for a system with a varying number of processors for the uniform distribution. We use percent energy savings as a comparison metric, which is defined as $\frac{E_{BUS} - E_{1D}}{E_{BUS}} \times 100$.

Figure 3.7: **Energy Savings for Uniform Distribution (1-D vs. Bus).**



As the number of processors in the system increases, the savings asymptotically approach the theoretical limit of 66% (from Eq. 3.23).

3.7 Localized Traffic Distributions

If we assume there is some sense of locality in our system, the data transmitted by processor P_i has greater probability to have destination processors close to P_i than processors located far from it. We apply five different probability distributions (linear decay, exponential decay, step, truncated linear decay, and truncated exponential decay) to model different communication patterns and locality characteristics. Recall that probability $p_{i,j}$ (Eq. 3.3) is the probability that processor P_i communicates with processor P_j and that the probability distributions are different for the communication patterns that we examine. We examine localized traffic distributions for a one-dimensional point-to-point network in this section, while we examine high-order networks in following chapters. Recall further, that for uniform communication $p_{i,j} = 1/(N-1)$, where N is the number of processors in the network.

3.7.1 Linear Decay

We examine the case in which the probability that two processors communicate drops linearly with the distance between the two processors. In this case the non-zero elements of the probability matrix $p_{i,j}$ [1] for a one-dimensional network are described in Eq. 3.24.

$$p_{i,j} = \frac{|b - (a \cdot |i - j|)|}{\sum_{j=1}^{i-1} |b - [a \cdot (i - j)]| + \sum_{j=i+1}^{N} |b - [a \cdot (j - i)]|}, \text{ for } i \neq j, \qquad 3.24$$

in which a and b are parameters of the model and define the amount of communication locality. The numerator in the previous equation provides the value for a linear function with coefficients a and b. Coefficient a is important as it defines the slope of the linear decay. The denominator is a normalizing factor that transforms the linear function into a probability mass function.

For example, the probability distribution of the communication between two processors, in a system with twenty processors, for three different processors ($i = 1$, $i = 4$, $i = 10$), for $b = N, a = 1$ looks like the graphs in Fig. 3.8.

1. As we define p in Eq. 3.3, $p_{i,i} = 0$

Figure 3.8: **Linear Decay Probability Distribution. Probability of Communication Between Processors P_1, P_4, and P_{10} with all the Processors in the Network.**

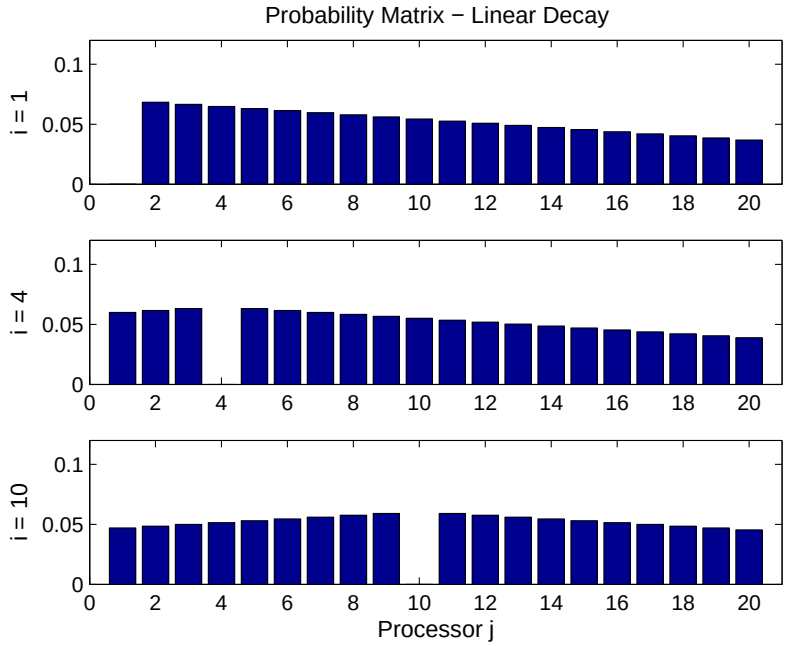
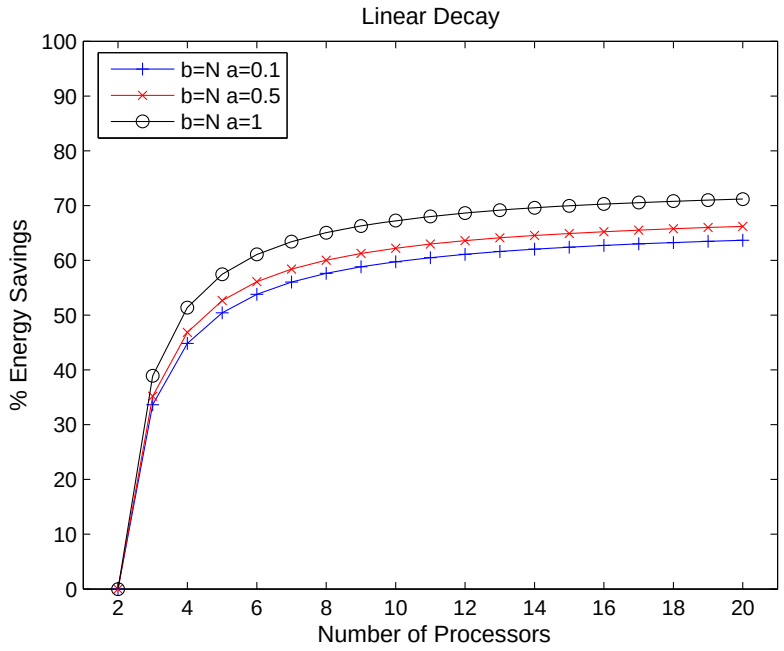


Fig. 3.9 presents the percentage energy savings for the 1-D point-to-point system compared to the bus-based system for the linear decay probability distributions for three different pairs of (b, a) for a different number of processors.

The probability values $p_{i,j}$ are used in Eq. 3.3 to calculate the expected energy consumption $E_{i,j}$ for the messages sent from processor P_i to processor P_j. The total energy dissipation for the communication in the network is calculated by Eq. 3.2, which is repeated below

$$E_{P2P} = \sum_{i=1}^{N} \sum_{j=1}^{N} E_{i,j}. \qquad 3.25$$

Figure 3.9: **Percentage Energy Savings for 1-D vs. Bus for Linear Decay Distributions for Three Pairs of (b, a).**



It is obvious that for more localized (increasing a) traffic patterns, the savings increase (71% for a system with twenty processors). The energy savings for a system with twenty processors and uniform traffic pattern (Fig. 3.7) are 63%; there is an additional 8% reduction in the energy even with low locality in the communication pattern.

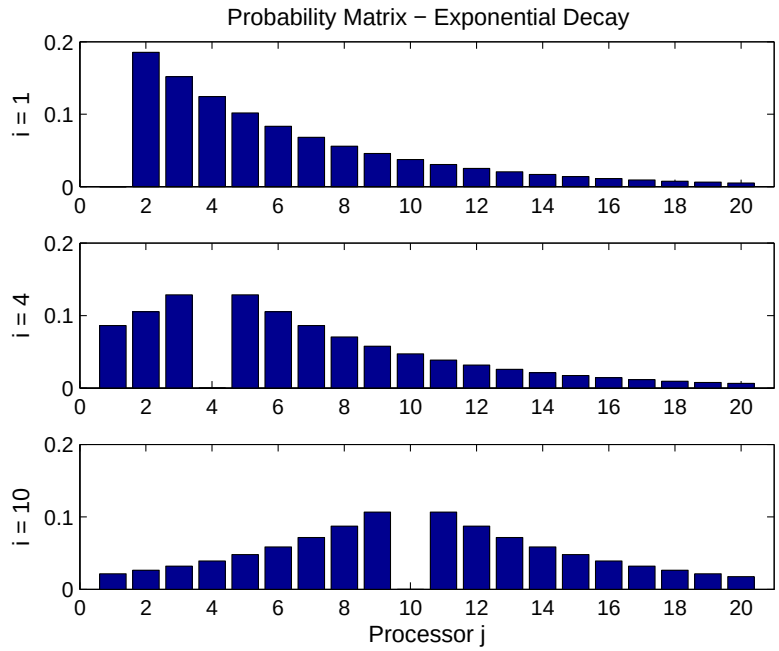
3.7.2 Exponential Decay

If we choose an exponential decay to describe the probability that data from processor P_i has destination processor P_j, then the elements of the communication probability matrix are given in Eq. 3.26.

$$p_{i,j} = \frac{\frac{1}{d} \cdot b^{-\frac{|i-j|}{d}}}{\sum_{j=1}^{i-1} \frac{1}{d} \cdot b^{-\frac{(i-j)}{d}} + \sum_{j=i+1}^{N} \frac{1}{d} \cdot b^{-\frac{(j-i)}{d}}}, \text{ for } i \neq j, \qquad 3.26$$

where d describes the average number of data hops and b is the base of the exponent. As was

Figure 3.10: **Exponential Decay Probability Distribution. Probability of Communication Between Processors P_1, P_4, and P_{10} with all the Processors in the Network.**

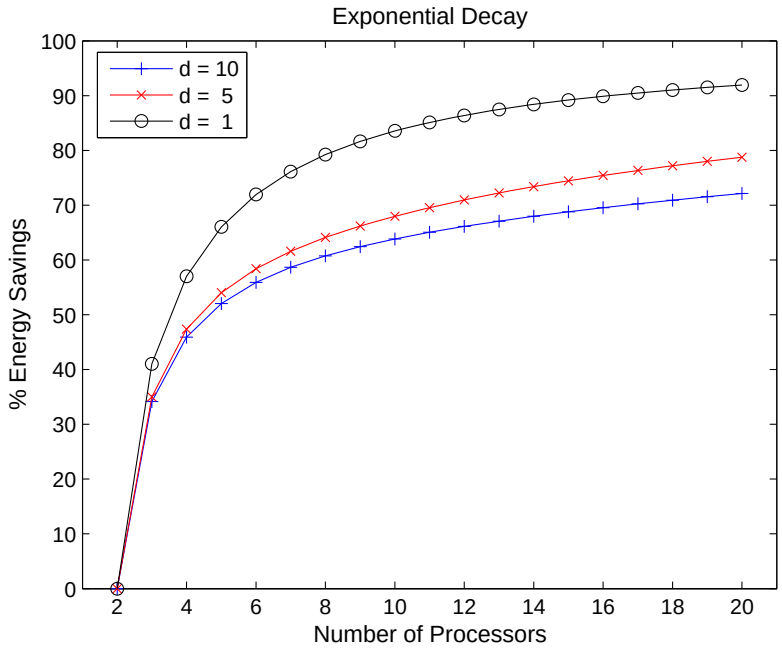


the case in the linear decay, the parameters d, b define the amount of communication locality in the system.

The probability distribution of the communication between two processors, in a system with twenty processors, for three different processors, looks like the graphs in Fig. 3.10.

The energy consumption ratio of the two systems for three different average distance values is shown in Fig. 3.11.

Figure 3.11: **Percentage Energy Savings for 1-D vs. Bus for Exponential Decay Distributions for Three Pairs of (b, d).**



The savings in this case increase significantly, as we move to extremely localized systems ($d = 1$). In a system with twenty processors the savings in this case are 92%, which is an increase of 29% compared to the case where processors communicate with equal likelihood with each other.

3.7.3 Step Distribution

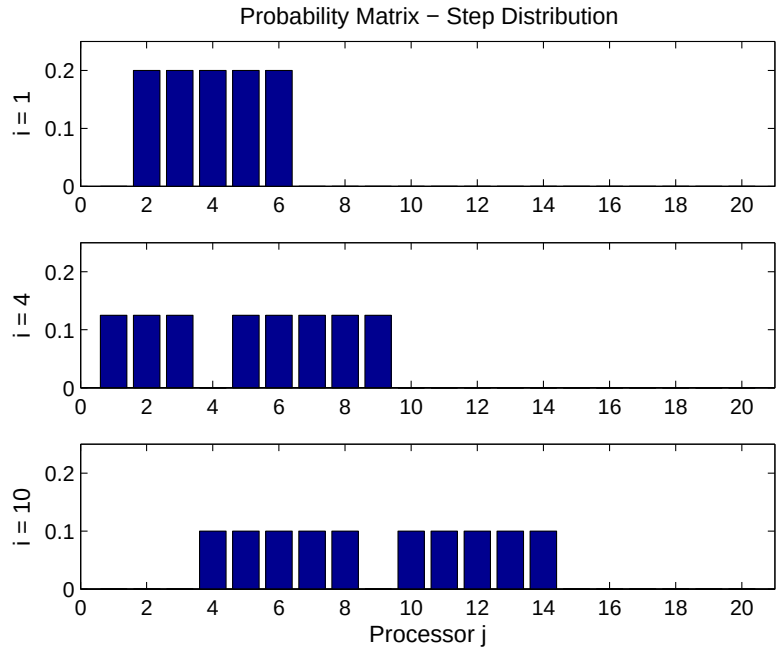
For the remaining analysis, we only allow communication among processors that are spaced within a specific distance r. We can express the step distribution (the case where P_i sends data with equal probability to processors within a radius r) with a probability matrix, where the non-zero elements of the matrix $p_{i,j}$ are described in Eq. 3.27.

$$p_{i,j} = \begin{cases} \dfrac{1}{r + (i-1)} & i \le r \\ \dfrac{1}{2r} & r < i < N - r \\ \dfrac{1}{r + (N-i)} & N - r \le i \end{cases}, \qquad 3.27$$

for $N > 2r$ and $|i - j| \le r$.

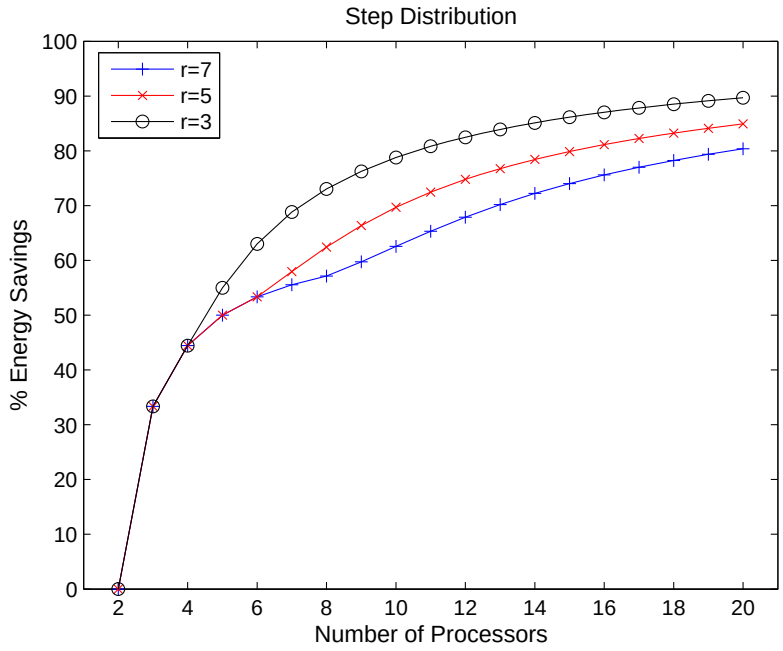
The probability distribution of the communication between two processors, on a twenty processor system and $r = 5$, for three processors, is shown below.

Figure 3.12: **Exponential Decay Probability Distribution. Probability of Communication Between Processors P_1, P_4, and P_{10} with all the Processors in the Network.**



The energy consumption ratio for the step distribution for different numbers of processors and different radii r is shown in Fig. 3.13.

Figure 3.13: **Percentage Energy Savings for 1-D vs. Bus for Step Distributions for Three Different Radii.**



The additional savings for the different radii are shown in networks with 4, 6, and 8 or more processors

3.7.4 Truncated Linear Decay

If the probability of the distance that the data travels is described by a truncated linear decay, the non zero elements of the probability matrix are given in Eq. 3.28:

$$p_{i,j} = \frac{|b-(a\cdot|i-j|)|}{\sum_{j=i-r}^{i-1}|b-(a\cdot|i-j|)| + \sum_{j=i+1}^{i+r}|b-(a\cdot(j-i))|}, \text{ for } i\neq j,\ |i-j|\leq r. \qquad 3.28$$

The probability distribution in a system with twenty processors, for three different processors, is presented in the graphs in Fig. 3.14.

Figure 3.14: **Truncated Linear Decay Probability Distribution. Probability of Communication Between Processors P_1, P_4, and P_{10} with all the Processors in the network.**

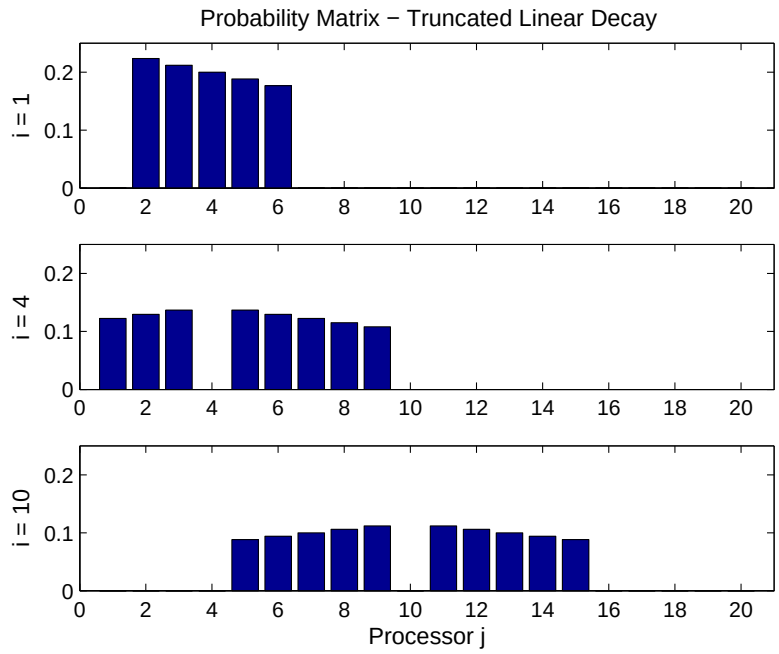


Fig. 3.15 shows the percentage energy savings.

Figure 3.15: **Percentage Energy Savings for 1-D vs. Bus for Truncated Linear Decay Distributions for Three Different Radii.**

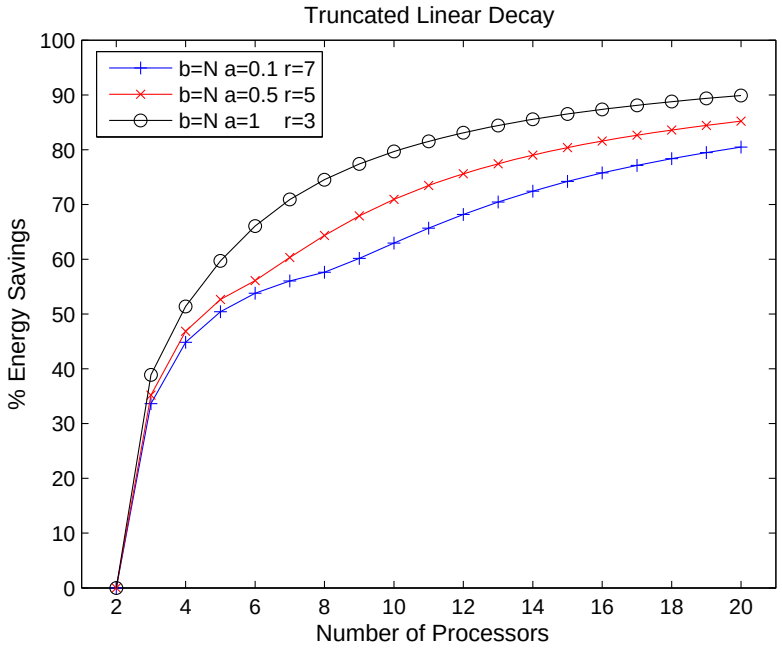
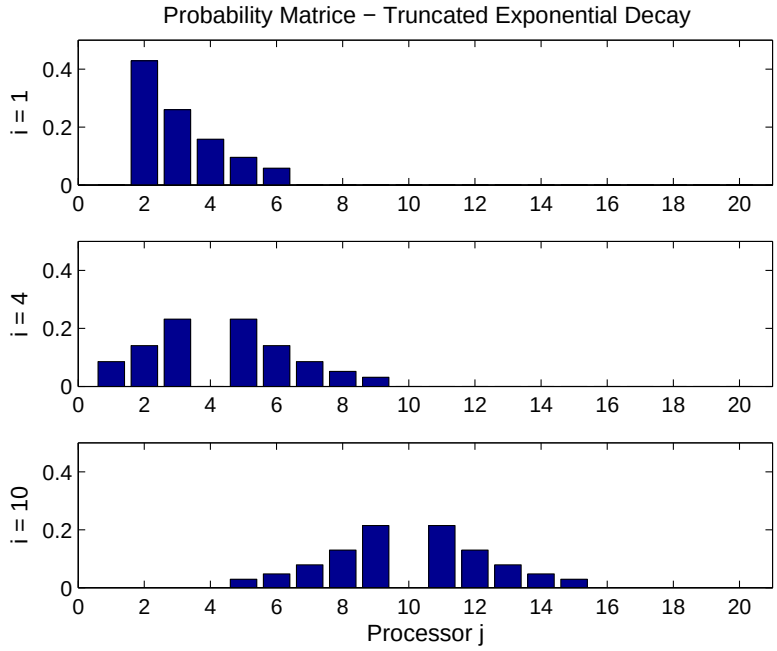


Figure 3.16: **Truncated Exponential Decay Probability Distribution. Probability of Communication between Processors P_1, P_4, and P_{10} with all the Processors in the Network.**



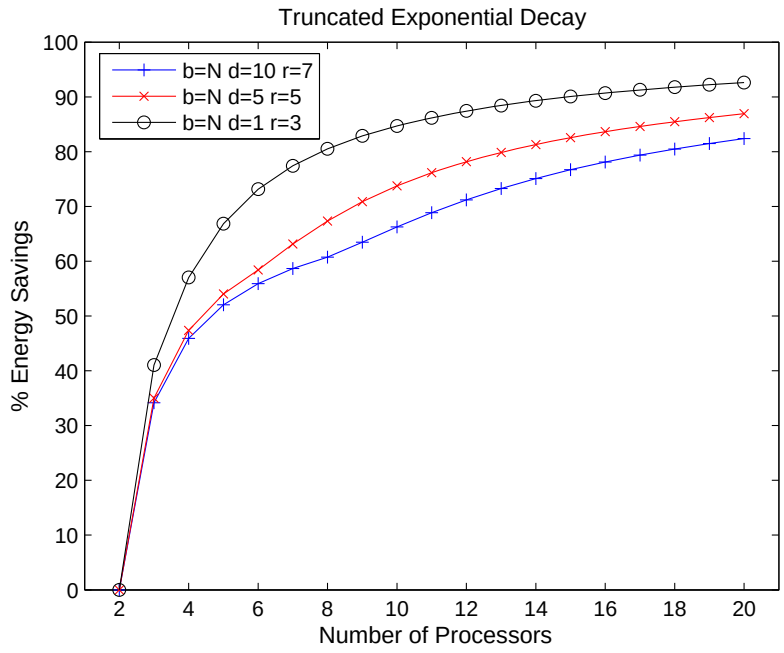
3.7.5 Truncated Exponential Decay

The truncated exponential decay has a probability matrix with non zero elements given in Eq. 3.29:

$$p_{i,j} = \frac{\frac{1}{d} \cdot b^{-\frac{|i-j|}{d}}}{\sum_{j=i-r}^{i-1} \frac{1}{d} \cdot b^{-\frac{(i-j)}{d}} + \sum_{j=i+1}^{i+r} \frac{1}{d} \cdot b^{-\frac{(j-i)}{d}}}, \text{ for } i \neq j, \ |i-j| \leq r \ . \qquad 3.29$$

The probability distribution in a system with twenty processors, for three different processors, is given in the graphs in Fig. 3.16.

Figure 3.17: **Percentage Energy Savings for 1-D vs. Bus for Truncated Exponential Decay Distributions for Three Different Radii.**



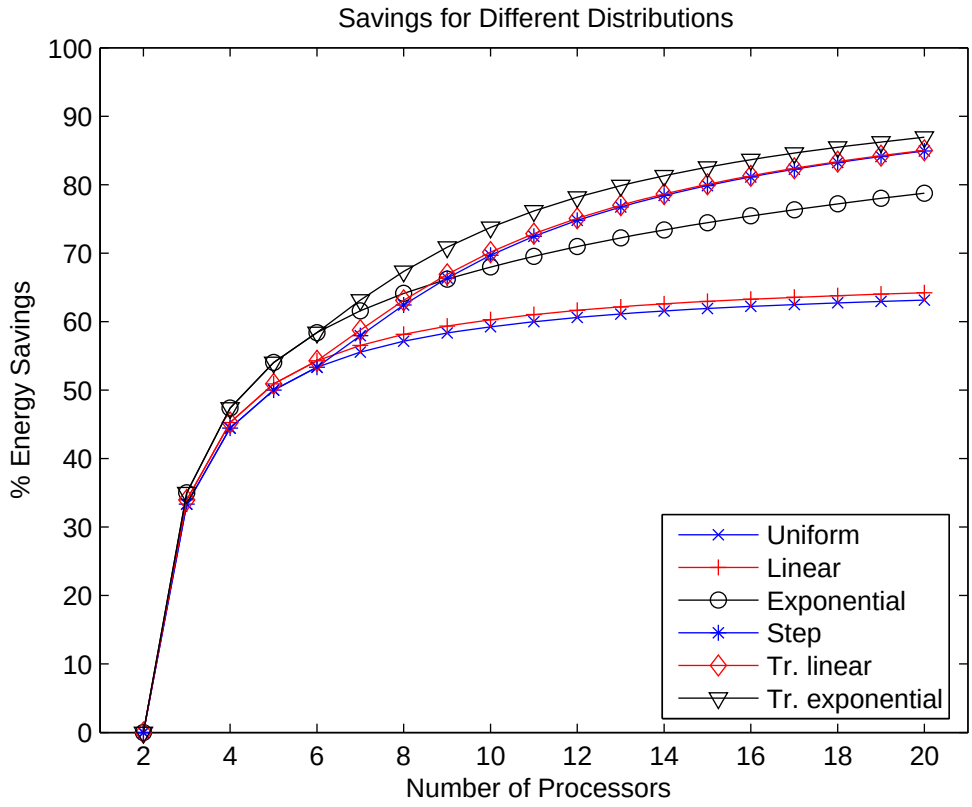
The percentage energy savings are shown in Fig. 3.17.

3.7.6 Savings Comparison for Different Distributions

As a summary, Fig. 3.18 groups the percentage energy savings of the point-to-point network over the energy consumed in the bus model for all six different distributions that we examined.

For the case of the truncated distributions, we select a radius equal to five as the maximum allowed distance for the transmitted data. The additional savings for the truncated distributions (step, truncated linear and truncated exponential) become evident in the systems with seven processors or more. As we move to systems with many processors, the energy savings of a point-to-point interconnection system increase significantly for greatly localized traffic patterns. Different communication locality patterns can have a significant effect on the energy performance of the point-to-point interconnection model. In systems with fifteen processors or more, the

Figure 3.18: **Savings Comparison for All Distributions.**



energy for the exponential distribution and the truncated distributions is half the energy for the linear distribution.

At this point, the model does not take contention for either the bus or the one-dimensional network into account. Contention in buses is significantly higher than contention in one-dimensional point-to-point networks. Even with no contention on the bus, it is evident that the energy savings of point-to-point networks are significant. We address contention of point-to-point networks in Chapter 7.

The probability distributions that we choose to examine included traffic patterns with no, low, and high locality. We have shown how important communication locality is, so applications that can be parallelized must exploit it to achieve reduced energy dissipation on the interconnection

network. Applications running on tiled architectures will not have traffic patterns that match exactly the patterns that we examine in this section. However, in Chapter 5 we investigate the energy dissipation for actual network traces, and we will show that in many cases the communication characteristics of the benchmarks can be modeled quite accurately with the probability distributions that we examined.

Summary

This chapter presented a detailed framework for the energy dissipation on different one-dimensional point-to-point network topologies and compared the energy performance of point-to-point networks with bus-based topologies. We presented a set of probability distributions to model various traffic patterns that closely match the communication characteristics of various benchmarks. These distributions can be used to represent any level of locality among the processors in the system.

We showed that for the uniform distribution the total expected energy for the point-to-point network is one third of the total energy for the bus-based system. The savings for more localized traffic patterns are even more considerable. For example, in the case where processors communication is modeled by an exponential decay probability distribution, the energy dissipation on the bus-based system can be 10 times larger compared to the energy consumption for a one-dimensional point-to-point network.

The framework and analysis that we presented in this chapter can be easily applied to two- and high- dimensional networks. The framework is true point-to-point in nature, in the sense that it individually calculates the expected energy dissipation for all possible communication between any pair of processors in the system. A set of probability distributions ($p_{i,j}$) that model the traffic patterns in the network, and information on the distance between the processors ($H_{i,j}$), for a given traffic load (M) and a given energy transfer cost E_l between two neighboring processors can estimate the total communication energy for uniform or arbitrary communication patterns.

The next chapter performs an energy analysis for two-dimensional networks. We apply the framework after we modify the communication probability matrix p and the distance matrix H to correspond to the probability of communication and the distance between two processors in two dimensions. Additionally, we compare the energy dissipated in two-dimensional networks to the energy dissipated in buses and one-dimensional networks, assuming similar levels of communication locality.

CHAPTER 4

TWO-DIMENSIONAL INTERCONNECTION NETWORKS

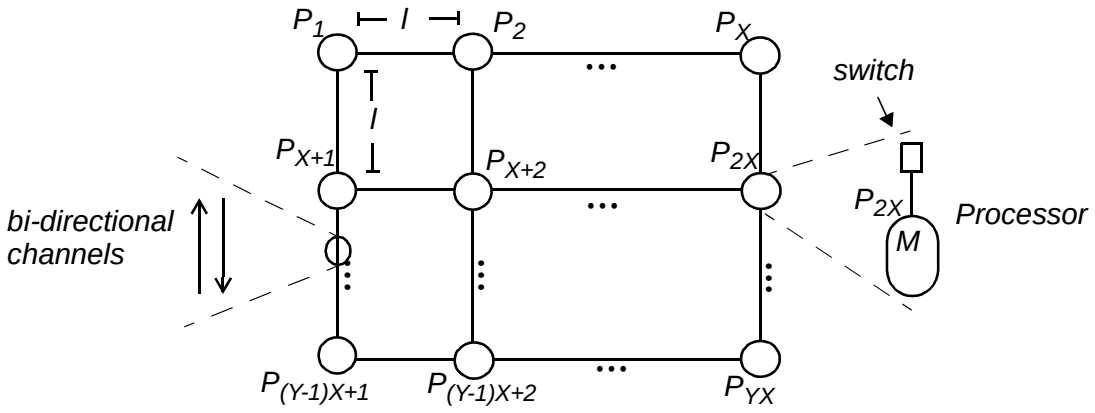
The analysis of the advantages of point-to-point interconnection networks in the previous chapter assumes a one-dimensional point-to-point mesh network topology. It is obvious however that, if the network dimensionality increases, the total energy savings will increase since the average and maximum communication distances decrease. For example, moving from one-dimensional to two-dimensional systems, decreases the average number of hops from $O(N)$ to $O(\sqrt{N})$ for a given number of processors.

This chapter expands the one-dimensional model to include any rectangular processor grid arrangement. We present a two-dimensional point-to-point interconnection model. Then we analyze the communication costs for uniform and for various localized traffic patterns, and derive the communication energy savings over the bus and the one-dimensional network.

4.1 Interconnection Network Model for 2-D Systems

Fig. 4.1 shows a two-dimensional point-to-point model.

Figure 4.1: **2-D Point-to-Point Network Model: *N* Processors, each transmitting *M* data words on a 2-D mesh network. There are *X* processors in each row and *Y* processors in each column.**



The N processors are interconnected with a two-dimensional mesh network (where X is the number of columns and Y is the number of rows in the network). When processor P_i wants to send data to processor P_j, the data is transmitted over only at the segments that connect P_i with P_j, and each switch is responsible for re-transmitting the data or sending it to the attached processor. We assume that each processor wants to send M data words. We also assume dimension-order routing [13] as a deterministic routing strategy. According to this strategy, messages travel all the distance on the first dimension before they switch to the second dimension.

As described in Section 3.4 on page 46, the total energy E_{P2P} consumed for all processors to communicate their data in a point-to-point network is given by

$$E_{P2N} = \sum_{i=1}^{N} \sum_{j=1}^{N} E_{i,j}, \qquad 4.1$$

in which the number of processors $N = X \cdot Y$ and $E_{i,j}$ is the energy consumption due to the transmission of all the data from processor P_i to processor P_j. As was the case with the one-

dimensional system, the total energy depends on the processor communication pattern. In our analysis we allow an unequal number of rows and columns (X can be different from Y), so that our model can support any rectangular interconnection network.

4.2 Communication Energy Cost

We modify the parameters of the model that we presented in the previous chapter to account for the communication of data between the processors in two-dimensional point-to-point network. These parameters include the communication probability matrix p and matrix H that captures the distance between every pair of processors in the network.

In the proposed interconnection model (Fig. 4.1) we assume an equal energy cost moving on each dimension[1]. If E_l is the energy cost for one network hop (on any dimension) and $p_{i,j}$ is the probability that processor P_i communicates with processor P_j, the expected energy cost $E_{i,j}$ for transferring the data originated by processor P_i to destination processor P_j, is given by (Eq. 3.3)

$$E = M \cdot p_{i,j} \cdot E_l \cdot H_{i,j}. \qquad 4.2$$

Each element of matrix H (Eq. 4.3) gives the number of hops between two processors as they are laid out in the plane. The processor IDs increase column-wise (Fig. 4.1). H is an $XY - by - XY$ symmetrical matrix and looks like:

1.This is a fair assumption. Tiles are square so the wires in the X and Y dimensions are of equal length.

$$H = \begin{bmatrix} \begin{bmatrix} 0 & 1 & \dots & X-1 \\ 1 & 0 & \dots & \dots \\ \dots & \dots & 0 & 1 \\ X-1 & \dots & 1 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 2 & \dots & X \\ 2 & 1 & \dots & \dots \\ \dots & \dots & 1 & 2 \\ X & \dots & 2 & 1 \end{bmatrix} & \dots & \begin{bmatrix} Y-1 & Y & \dots & X-1+Y-1 \\ Y & Y-1 & \dots & \dots \\ \dots & \dots & Y-1 & \dots \\ X-1+Y-1 & \dots & \dots & Y-1 \end{bmatrix} \\ \begin{bmatrix} 1 & 2 & \dots & X \\ 2 & 1 & \dots & \dots \\ \dots & \dots & 1 & 2 \\ X & \dots & 2 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 1 & \dots & X-1 \\ 1 & 0 & \dots & \dots \\ \dots & \dots & 0 & 1 \\ X-1 & \dots & 1 & 0 \end{bmatrix} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \begin{bmatrix} Y-1 & Y & \dots & X+Y-2 \\ Y & Y-1 & \dots & \dots \\ \dots & \dots & Y-1 & \dots \\ X+Y-2 & \dots & \dots & Y-1 \end{bmatrix} & \dots & \begin{bmatrix} 0 & 1 & \dots & X-1 \\ 1 & 0 & \dots & \dots \\ \dots & \dots & 0 & 1 \\ X-1 & \dots & 1 & 0 \end{bmatrix} \end{bmatrix}. \quad 4.3$$

Matrix H can be decomposed into Y^2 $X-by-X$ square matrices arranged in Y rows and Y columns. The elements of any of the $X-by-X$ square matrices show the distance among processors located on the same row with processors located on any same column.

Eq. 3.4 on page 49 shows the distance between two processors in the same column in the system. We present this equation again and define an $X-by-X$ matrix H_X.

$$H_X = \begin{bmatrix} 0 & 1 & \dots & X-1 \\ 1 & 0 & \dots & \dots \\ \dots & \dots & 0 & 1 \\ X-1 & \dots & 1 & 0 \end{bmatrix}. \quad 4.4$$

We also define an $X-by-X$ matrix with all elements equal to one:

$$A = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & 1 \end{bmatrix}. \quad 4.5$$

Matrix H can be written as

$$H = \begin{bmatrix} H_X & H_X + A & \dots & H_X + A \cdot (Y-1) \\ H_X + A & H_X & \dots & \dots \\ \dots & \dots & \dots & H_X + A \\ H_X + A \cdot (Y-1) & \dots & H_X + A & H_X \end{bmatrix}, \quad 4.6$$

which can be further decomposed to

$$H = \begin{bmatrix} H_X & H_X + A & \dots & H_X + A \cdot (Y-1) \\ H_X + A & H_X & \dots & \dots \\ \dots & \dots & \dots & H_X + A \\ H_X + A \cdot (Y-1) & \dots & H_X + A & H_X \end{bmatrix} = \tag{4.7}$$

$$= \begin{bmatrix} H_X & H_X & \dots & H_X \\ H_X & H_X & \dots & \dots \\ \dots & \dots & \dots & H_X \\ H_X & \dots & H_X & H_X \end{bmatrix} + \begin{bmatrix} 0 & A & \dots & A \cdot (Y-1) \\ A & 0 & \dots & \dots \\ \dots & \dots & \dots & A \\ A \cdot (Y-1) & \dots & \dots & 0 \end{bmatrix} = \overline{H} + \overline{A}. \tag{4.8}$$

Matrix H is essential in deriving a closed-form solution for the energy consumption of two-dimensional networks assuming a uniform distribution that describes the communication among the different processors in the system.

4.3 Uniform Distribution

First we examine the system in which processors communicate with each other with equal likelihood. The uniform communication probability $p_{i,j}$ of Eq. 4.2 is equal to $\frac{1}{XY-1}$ for every $i \neq j$. The equation for the expected energy cost (Eq. 4.2) for the communication between processors P_i and P_j becomes

$$E_{i,j} = \frac{M}{XY-1} \cdot E_l \cdot H_{i,j}. \tag{4.9}$$

So the total expected energy cost of transmitting the data is

$$E_{2D} = \sum_{k=1}^{N} E_k = \sum_{i=1}^{N} \left(\sum_{j=1}^{N} E_{i,j} \right) = \frac{M}{XY-1} \cdot E_l \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} H_{i,j}, \tag{4.10}$$

where $N = X \cdot Y.$

We find the sum of the elements of H,

$$\sum_{i=1}^{N} \sum_{j=1}^{N} H_{i,j} = \sum_{i=1}^{N} \sum_{j=1}^{N} \overline{H}_{i,j} + \sum_{i=1}^{N} \sum_{j=1}^{N} \overline{A}_{i,j} = \tag{4.11}$$

$$= Y^2 \cdot \sum_{i=1}^{X} \sum_{j=1}^{X} H_{X_{i,j}} + \left(\sum_{i=1}^{X} \sum_{j=1}^{X} A_{i,j} \right) \cdot sum \begin{bmatrix} 0 & 1 & \dots & Y-1 \\ 1 & 0 & \dots & \dots \\ \dots & \dots & \dots & 1 \\ Y-1 & \dots & \dots & 0 \end{bmatrix}, \quad 4.12$$

where $sum[M] = \sum M_{i,j}$. We manipulated Eq. 4.11 to come up with an expression that includes matrices H_X and A that we defined in Eq. 4.4 and Eq. 4.5 and are easy to calculate the sum of their elements.

Therefore,

$$\sum_{i=1}^{N} \sum_{j=1}^{N} H_{i,j} = Y^2 \cdot \frac{X(X-1)(X+1)}{3} + X^2 \cdot \frac{Y(Y-1)(Y+1)}{3} = \quad 4.13$$

$$= \frac{Y^2X(X-1)(X+1) + X^2Y(Y-1)(Y+1)}{3} = \frac{YX[Y(X-1)(X+1) + X(Y-1)(Y+1)]}{3} = \quad 4.14$$

$$= \frac{YX[Y(X^2-1) + X(Y^2-1)]}{3} = \frac{YX[YX^2 - Y + YX^2 - X]}{3} = \quad 4.15$$

$$= \frac{YX[X(YX-1) + Y(YX-1)]}{3} = \frac{YX(YX-1)(X+Y)}{3}, \quad 4.16$$

and the total expected energy cost (Eq. 4.1) is

$$E_{2D} = \frac{M}{N-1} \cdot E_l \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} H_{i,j} = \frac{M}{YX-1} \cdot E_l \cdot \left[\frac{YX(XY-1)(X+Y)}{3} \right] = \quad 4.17$$

$$= M \cdot E_l \cdot \left[\frac{YX(XY-1)(X+Y)}{3 \cdot (XY-1)} \right] = M \cdot E_l \cdot \frac{YX(X+Y)}{3}. \quad 4.18$$

This is the total expected energy consumption in a two-dimensional point-to-point network, with X processors on each row and Y processors on each row, after each processor sent M data words to other processors in the network. Processors communicate with equal likelihood and the energy cost for one network hop is E_l.

For $Y = 1$ the previous equation reduces to Eq. 3.16 which describes the total expected energy cost for a one-dimensional network and uniform communication patterns among the

processors in the system. Intuitively, this is the product of the number of words per processor (M), times the energy cost for one network hop, times the number of processors in the network (YX), times the average distance of a message in both dimensions ($\frac{X}{3} + \frac{Y}{3}$).

Additionally, we can show by substituting $Y = N/X$, differentiating Eq. 4.18 with respect to X, and setting to 0, that the energy in a two-dimensional network is minimized in a square system where $X = \sqrt{N}$.

In this case, the total expected energy consumption for a square mesh is

$$E_{2D} = M \cdot E_l \cdot \frac{2N \cdot \sqrt{N}}{3}. \quad 4.19$$

4.4 Energy Comparison of a 2-D Network with 1-D Network and Bus

The ratio of the energy dissipated on the two-dimensional mesh network over the energy consumed in the bus-based model is

$$\frac{E_{2D}}{E_{BUS}} = \frac{M \cdot E_l \cdot \frac{YX(X+Y)}{3}}{M \cdot XY \cdot E_L} = \frac{X+Y}{3} \cdot \frac{E_l}{E_L}. \quad 4.20$$

At this point in our analysis we ignore the switch energy cost and contention at the switch, focusing on the effect of communication locality on the energy of point-to-point interconnection networks.

With this assumption

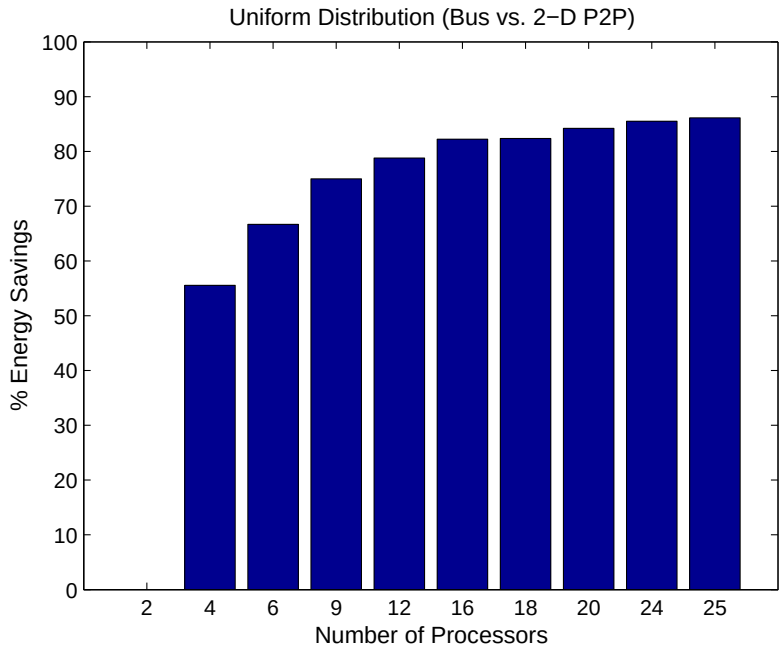
$$E_L = (XY-1) \cdot E_l, \quad 4.21$$

and Eq. 4.20 becomes

$$\frac{E_{2D}}{E_{BUS}} = \frac{1}{3} \cdot \frac{X+Y}{XY-1}. \quad 4.22$$

In a square ($X = Y = \sqrt{N}$) mesh with many processors the previous equation reduces to

Figure 4.2: **Energy Consumption Savings Assuming Uniform Distribution (Bus vs. 2-D Mesh).**



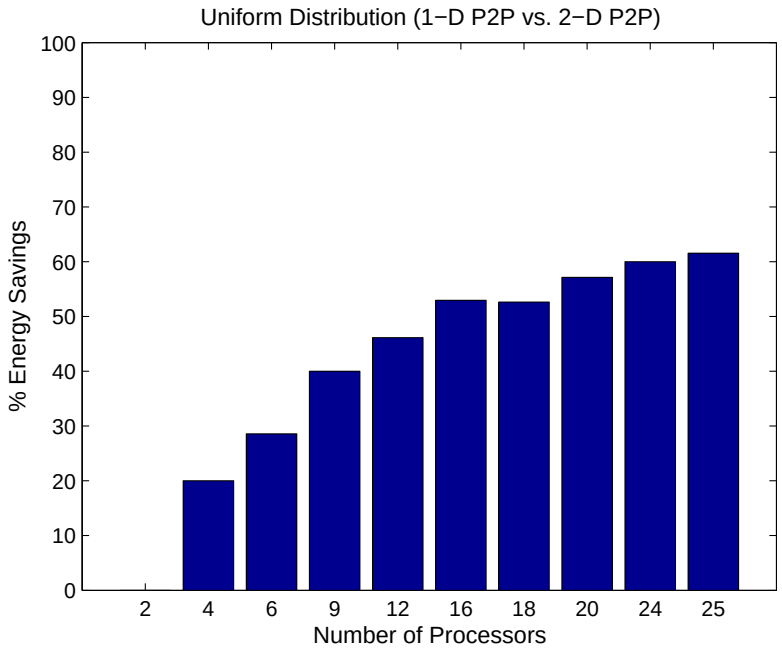
$$\frac{E_{2D}}{E_{BUS}} \approx \frac{2\sqrt{N}}{3N} \approx \frac{2}{3\sqrt{N}}, \qquad 4.23$$

and shows that for a square mesh network the energy savings are $O(\sqrt{N})$ compared to the energy dissipated on a bus. Intuitively, this is because the average distance that a message travels in a two-dimensional network is $O\sqrt{N}$ compared to the average distance in one-dimensional networks.

Fig. 4.2 shows the energy savings comparing the two systems. It is clear that, in the case of a two-dimensional mesh, the limit of the energy ratio in Eq. 4.22 is zero ($\lim_{N \to \infty} \frac{E_{2D}}{E_{BUS}} = 0$), while the limit in the one-dimensional network is $1/3$ (Chapter 3).

The energy savings for a square mesh network compared to a one-dimensional network are also expected to be $O(\sqrt{N})$ for large N, since there is a constant factor of $1/3$ that relates the energy dissipated on a bus-based system and a one-dimensional network.

Figure 4.3: **Energy Consumption Savings Assuming Uniform Distribution (1-D Mesh vs. 2-D Mesh.**



We validate this by comparing the energy of the two point-to-point networks for the same number of processors and plot in Fig. 4.3 the energy savings of the two-dimensional mesh compared to the one-dimensional mesh. The energy ratio of the two systems is

$$\frac{E_{2D}}{E_{1D}} = \frac{M \cdot E_l \cdot \dfrac{XY(X+Y)}{3}}{M \cdot E_l \cdot \dfrac{N \cdot (N+1)}{3}} = \frac{XY(X+Y)}{N \cdot (N+1)}. \qquad 4.24$$

For a square mesh the previous ratio becomes

$$\frac{E_{2D}}{E_{1D}} = \frac{2\sqrt{N}}{N+1} \approx \frac{2}{\sqrt{N}}, \text{ for large } N. \qquad 4.25$$

The previous equation proves that moving from a one-dimensional network to a network in two dimensions the energy savings are $O(\sqrt{N})$.

4.5 Localized Traffic Distributions

So far in this chapter, we focused on systems where processors communicate with each other with equal likelihood. We compared the total communication energy in two-dimensional networks with the energy expended in one-dimensional and bus-based networks. In the one-dimensional systems we also modeled the traffic patterns of the networks with probability distributions that describe different localized or non-localized traffic patterns.

As we did in the case of a one-dimensional point-to-point network, we examine the effect of communication locality on the total energy dissipated on the interconnection network in the case of a two-dimensional network. We modify the probability distributions $p_{i,j}$ (Eq. 3.3) that model the communication patterns between two processors in the network to represent different locality characteristics, as we did in the previous chapter.

We apply five different probability distributions (linear decay, exponential decay, step, truncated linear decay, and truncated exponential decay) to model different communication patterns and locality characteristics.

The equations for the probability distributions are two-dimension analogs of the ones described in Chapter 3 and are given below

- Linear Decay Probability Distribution

$$p_{i,j} = \frac{\left|b - (a \cdot H_{i,j})\right|}{\sum_{j=1}^{i-1} \left|b - (a \cdot H_{i,j})\right| + \sum_{j=i+1}^{N} \left|b - (a \cdot H_{i,j})\right|}, \text{ for } i \neq j. \qquad 4.26$$

- Exponential Decay Probability Distribution

$$p_{i,j} = \frac{\frac{1}{d} \cdot b^{-\frac{H_{i,j}}{d}}}{\sum_{j=1}^{i-1} \frac{1}{d} \cdot b^{-\frac{H_{i,j}}{d}} + \sum_{j=i+1}^{N} \frac{1}{d} \cdot b^{-\frac{H_{i,j}}{d}}}, \text{ for } i \neq j. \qquad 4.27$$

- Step Probability Distribution

$$p_{i,j} = \frac{1}{-1 + \sum_{j=1}^{N} 1}, \text{for } i \neq j \text{ and } \forall(i,j) \text{ such that } H_{i,j} \leq r. \quad 4.28$$

- Truncated Linear Decay Probability Distribution

$$p_{i,j} = \frac{\left|b - (a \cdot H_{i,j})\right|}{-b + \sum_{j=1}^{N} \left|b - (a \cdot H_{i,j})\right|}, \text{ for } i \neq j \text{ and } \forall(i,j) \text{ such that } H_{i,j} \leq r. \quad 4.29$$

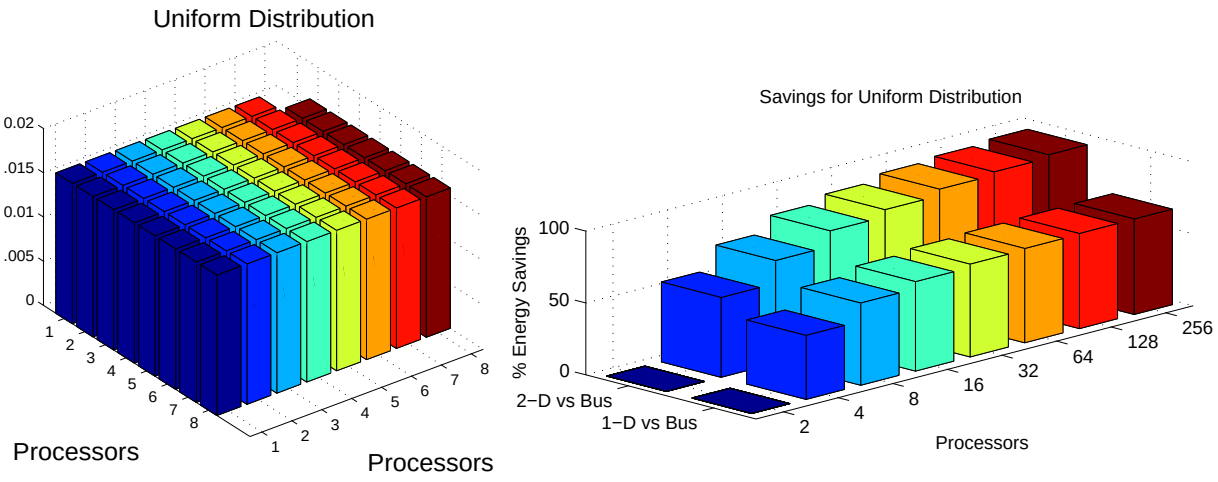
- Truncated Exponential Decay Probability Distribution

$$p_{i,j} = \frac{\frac{1}{d} \cdot b^{-\frac{H_{i,j}}{d}}}{-b + \sum_{j=1}^{N} \frac{1}{d} \cdot b^{-\frac{H_{i,j}}{d}}}, \text{ for } i \neq j \text{ and } \forall(i,j) \text{ such that } H_{i,j} \leq r. \quad 4.30$$

In the following sections, we graphically present the energy savings of the two-dimensional systems compared to one-dimensional and bus-based systems. We also present representative graphs of the communication probability values between one processor and the rest processors in the network.

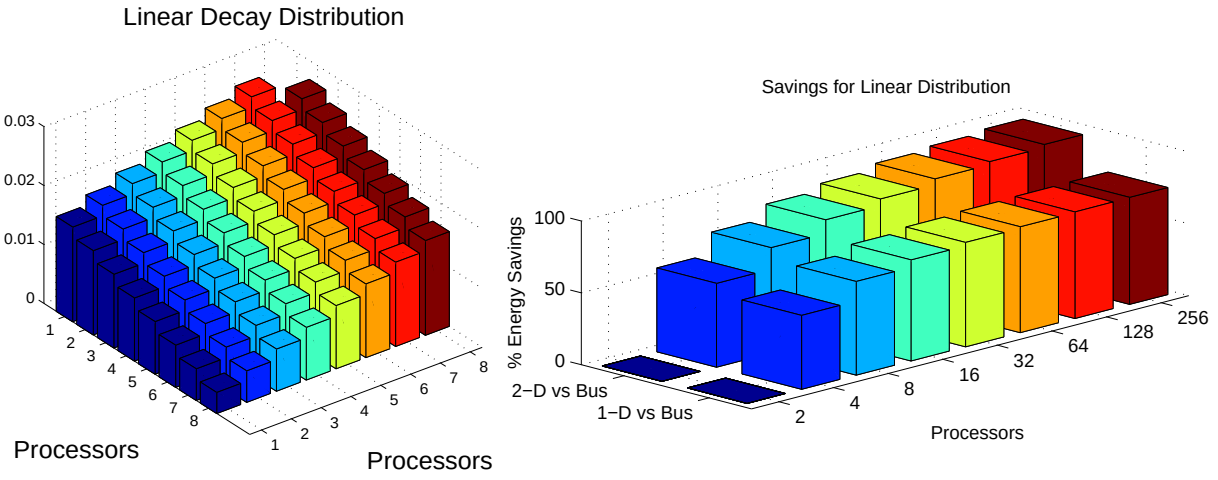
4.5.1 Uniform Distribution

Figure 4.4: **Energy Consumption Savings Assuming Uniform Distribution (1-D Mesh vs. Bus and 2-D Mesh vs. Bus).**



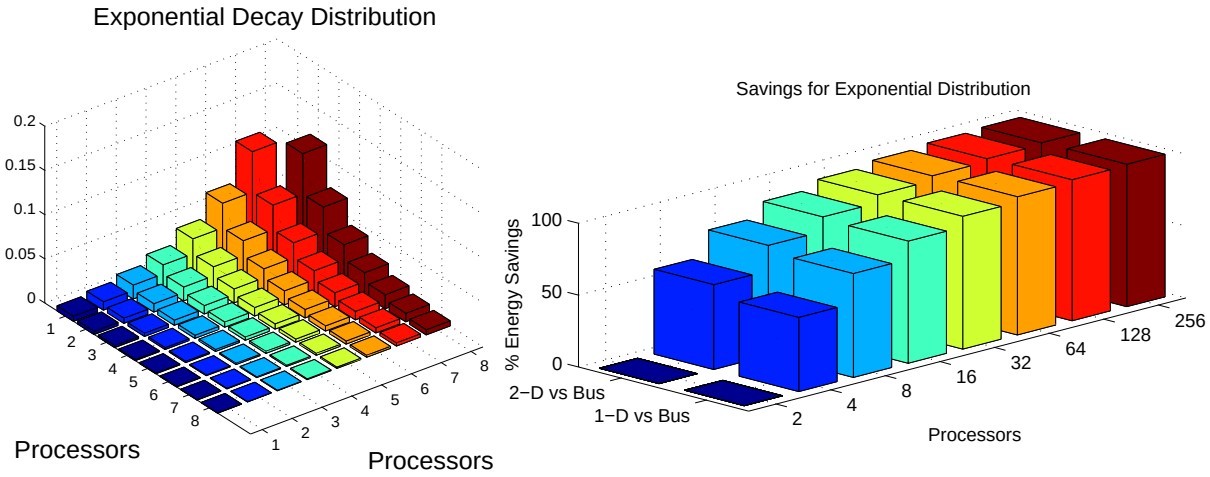
4.5.2 Linear Decay

Figure 4.5: **Energy Consumption Savings Assuming Linear Distribution (1-D Mesh vs. Bus and 2-D Mesh vs. Bus).**



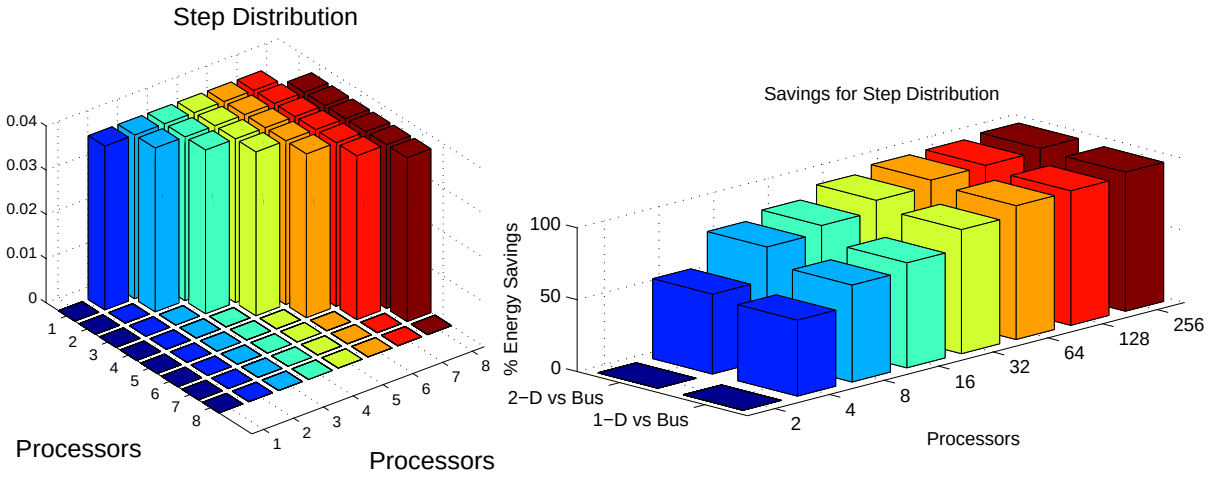
4.5.3 Exponential Decay

Figure 4.6: **Energy Consumption Savings Assuming Exponential Distribution (1-D Mesh vs. Bus and 2-D Mesh vs. Bus).**



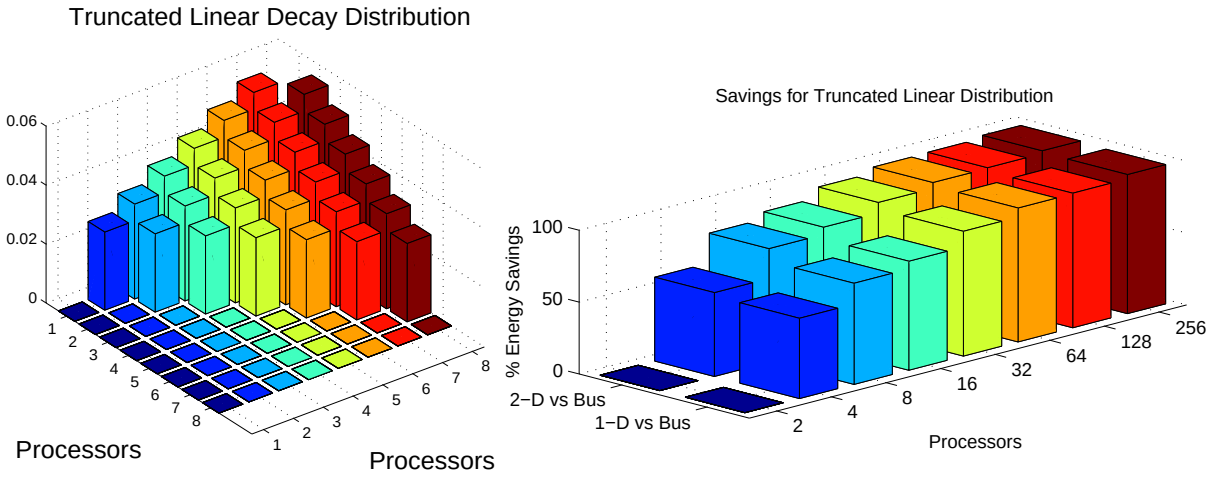
4.5.4 Step Distribution

Figure 4.7: **Energy Consumption Savings Assuming Step Distribution (1-D Mesh vs. Bus and 2-D Mesh vs. Bus).**



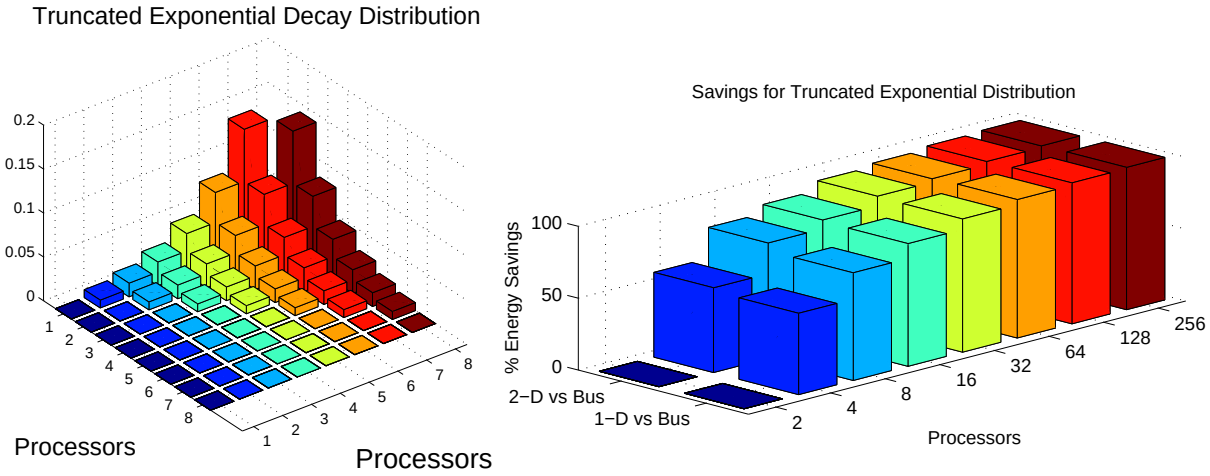
4.5.5 Truncated Linear Decay

Figure 4.8: **Energy Consumption Savings Assuming Truncated Linear Distribution (1-D Mesh vs. Bus and 2-D Mesh vs. Bus).**



4.5.6 Truncated Exponential Decay

Figure 4.9: **Energy Consumption Savings Assuming Truncated Exponential Distribution (1-D Mesh vs. Bus and 2-D Mesh vs. Bus).**



It is evident that as the number of processors in the system increases the energy savings for two-dimensional systems become overwhelming compared to the energy of bus-based systems. Another significant observation that comes from the previous figures is that for no locality or low locality traffic patterns (uniform and linear decay, respectively) the energy savings of the two-dimensional systems is significant compared to one-dimensional systems, while in the presence of highly-localized communication the energy performance of the two-dimensional system is comparable to the energy performance of the one-dimensional point-to-point systems.

The previous analysis should persuade the reader that point-to-point networks are superior to bus-based networks in terms of energy consumption. Additionally, two-dimensional point-to-point networks are more robust over various traffic patterns compared to one-dimensional networks.

Summary

This chapter extended the one-dimensional model to include any rectangular processor grid arrangement. We presented a two-dimensional point-to-point interconnection model and derived a closed-form equation for the energy dissipation of two-dimensional meshes assuming a uniform distribution of communication among processors.

We showed that the energy savings are $O(\sqrt{N})$ compared to the energy of a one-dimensional point-to-point system and a bus-based system. Additionally, we analyzed the communication costs for localized traffic patterns and presented the additional energy savings compared to the energy expended on bus-based systems.

CHAPTER 5

ENERGY SAVINGS AND MODEL VALIDATION USING NETWORK TRACES

So far in our analysis, we used probability distributions to model different traffic patterns and get the expected energy consumption in the various networks that we examined. Further, we assumed that the same number of words was communicated between processors. In this chapter we incorporate actual network traces from applications running on the MIT Raw multicore processor into our framework and get an estimation of the energy dissipated in the on-chip interconnection network.

Communication locality depends on the algorithm in the application as well as on the partitioning and placement of data on the different tiles of the system, so we examine the different locality characteristics of the traces and their effect on the total energy dissipated. As we did in

the previous chapters, we present energy savings compared to the energy dissipated on a bus-based interconnection network.

5.1 Network Traces

The network traces were obtained running benchmarks on the MIT Raw Microprocessor. The traces for the fourteen applications were produced using "*BTL*," [50] (pronounced *beetle)* a cycle-accurate tiled processor simulator for four different processor configurations: two tiles, four tiles, eight tiles, and sixteen tiles. The benchmarks were auto-parallelized using the Raw compiler [33]. The compiler, scheduling for ILP, arranges sequential C or Fortran programs across the tiles in two steps. First, the compiler distributes the data and code across the tiles balancing locality and parallelism. Then, it schedules the computation and communication to maximize parallelism and minimize communication latency.

Table 1 lists the 14 applications that we examine, the corresponding sources, their type, and the total number of messages communicated per application.

Table 1: List of Applications

Application	Source	Type	Total Messages
1.adpcm	Mediabench	Streaming	100247
2.aes	FIPS-197	Irregular	239987
3.aes_fix	FIPS-197	Irregular	228506
4. btrix	Nasa7: Spec92	Dense Matrix	80389
5.cholesky	Nasa7: Spec92	Dense Matrix	1725120
6.fpppp	Nasa7: Spec92	Irregular	315181
7.jacobi	Raw benchmark suite	Dense Matrix	36719
8.jacobi_big	Raw benchmark suite	Dense Matrix	3317070
9.life	Raw benchmark suite	Dense Matrix	707916
10.mxm	Nasa7: Spec92	Dense Matrix	667084
11.sha	Perl Oasis	Dense Matrix	716323
12.swim	Spec95	Dense Matrix	5579025

Table 1: List of Applications

Application	Source	Type	Total Messages
13.tomcatv	Nasa7: Spec92	Dense Matrix	1092888
14.vpenta	Nasa7: Spec92	Dense Matrix	105322

A short description of the applications follows:

- "*adpcm*" is an algorithm used in audio coding and stands for Adaptive Differential Pulse Code Modulation.
- "*aes*" (and "*aes_fix*") stands for Advanced Encryption Standard and is a cryptographic algorithm that can be used to protect electronic data.
- "*btrix*" is a vectorized block tri-diagonal solver.
- "*cholesky*" is a cholesky decomposition/substitution algorithm.
- "*fpppp*" is a floating point benchmark used in quantum chemistry.
- "*jacobi*" and "*jacobi_big*" are jacobi relaxation algorithms.
- "*life*" is Conway's Game of Life.
- "*mxm*" is a matrix multiply benchmark.
- "*sha*" stands for Secure Hash Algorithm and it is a multimedia benchmark.
- "*swim*" is a shallow water model and is a floating point benchmark with parallel loops.
- "*tomcatv*" is a mesh generation program with Thompson's solver.
- "*vpenta*" is an algorithm that inverts 3 pentadiagonals simultaneously.

The auto-parallelized benchmarks were simulated in "*BTL*" and on each cycle we recorded the communication among the tiles and obtained the network traces for traffic across the tiles. Knowing the communication information between the tiles, we write Eq. 4.2 on page 71 as

$$E_{i,j} = M_{i,j} \cdot E_l \cdot H_{i,j}, \qquad 5.1$$

where $M_{i,j}$ replaces $M \cdot p_{i,j}$ and describes the actual number of data words that processor P_i

sent to processor P_j, and reveals the locality patterns for each benchmark. In the above equation $E_{i,j}$ is the energy dissipated on the path that connects two neighboring tiles and $H_{i,j}$ corresponds to the distance (in number of hops) between processors P_i and P_j.

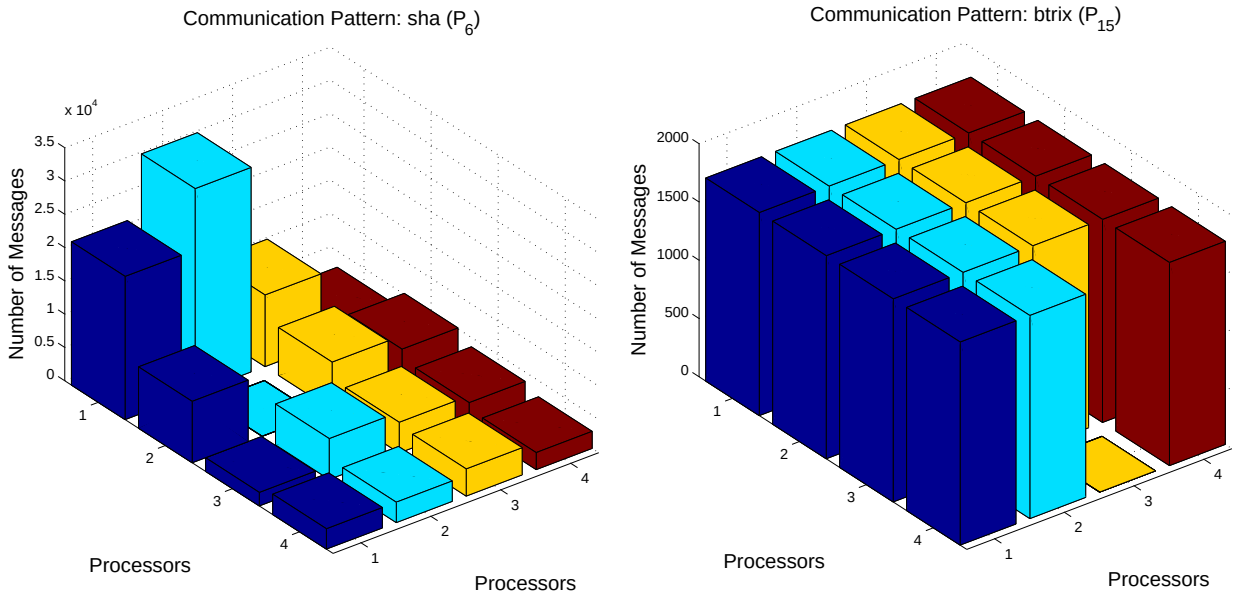
The total energy cost of transmitting the data is

$$E_{P2P} = \sum_{i=1}^{N} \sum_{j=1}^{N} E_{i,j} = \sum_{i=1}^{N} \sum_{j=1}^{N} [M_{i,j} \cdot E_l \cdot H_{i,j}]. \quad 5.2$$

It is worth mentioning at this point that the energy calculated by the above equation will no longer be an expected value, but the actual energy dissipated due to the traffic pattern captured in the matrix M.

As an example of the traffic pattern captured by M, we present the total number of messages sent from one processor on a 16-tile system for the benchmarks *sha* and *fpppp*. Specifically,

Figure 5.1: **Number of Messages Transferred from One Processor to Other Processors in the Network for *sha* and *btrix*.**



(a) Communication Pattern - sha

(b) Communication Pattern - btrix

Fig. 5.1(a) shows the number of data transfers originating from the processor on the second row and the second column for the *sha* benchmark. In this case, the compiler exploits communication locality, since the largest number of data transfers have as their destination a neighboring tile. The pattern resembles both the exponential decay distribution for data directed to the processors on the first row and the linear decay distribution for the other processors. On the other hand, in Fig. 5.1(b) we plot the data transfers originating from the processor on the fourth row and the third column for the *btrix* benchmark. In this case, the communication pattern resembles the uniform distribution.

In Appendix B we show detailed communication patterns between processors for all the applications that we examine.

5.2 Trace Communication Statistics

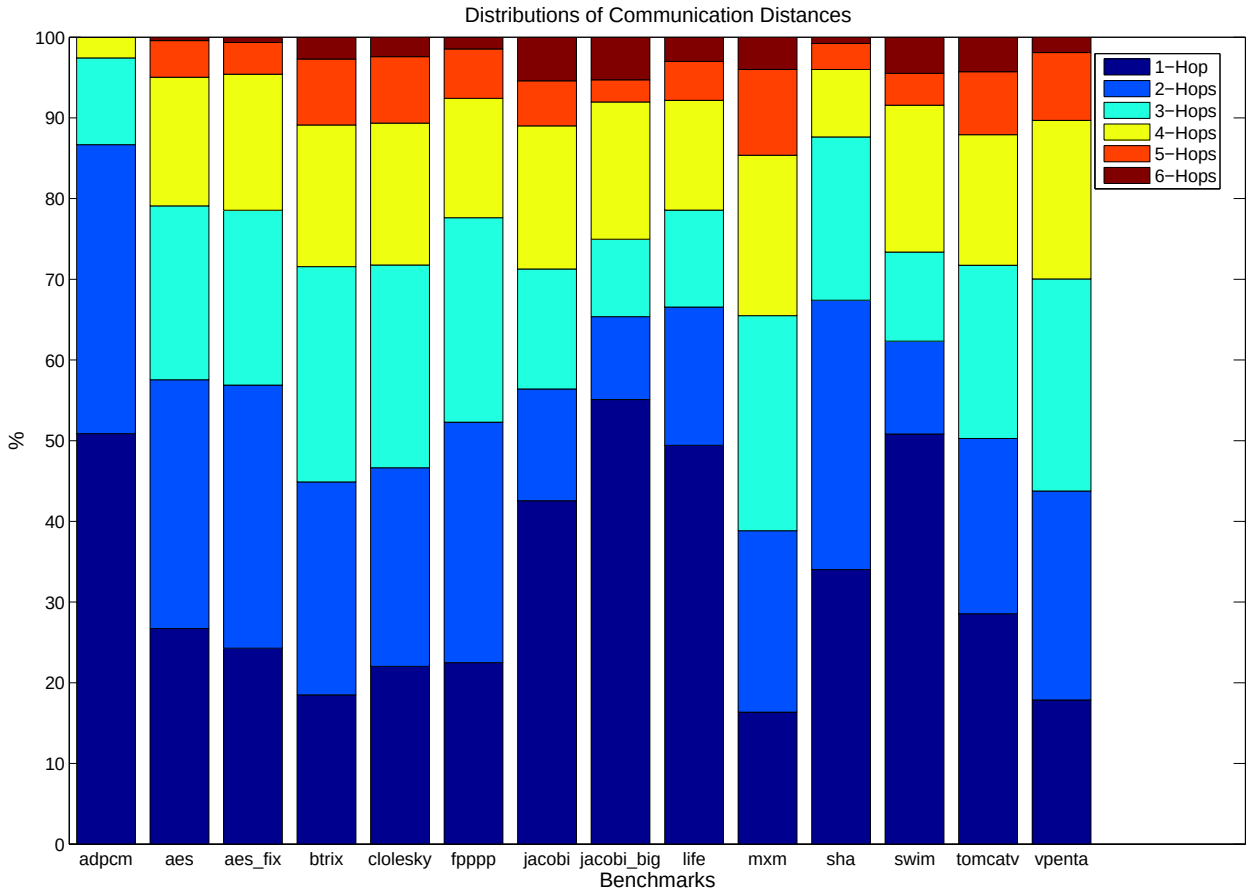
In this section we examine the communication characteristics for each of the benchmarks. On a four-by-four tile configuration the minimum and maximum distances that a message can travel is one and six hops, respectively. Fig. 5.2 shows the various distributions of communication distances for our set of benchmarks.

Communication locality depends both on the particular benchmark and on the partitioning and placement of data on the different tiles of the system. In most of the benchmarks 50% of the total communication consists of messages between next (1 hop) or near (2 hops) neighbors.

Adpcm exhibits the most localized communication characteristics allowing messages between processors that are located no more than 4 hops away. Good communication locality behavior is further evident due to the fact that 1-hop and 2-hop messages comprise more than 85% of total communication among tiles.

On the other end, matrix-multiply (mxm) exhibits the smaller percentage (16%) of next neighbor transfers compared to all examined benchmarks. Additionally, it exhibits the largest percentages for 3-, 4-, and 5-hop transfers, 27%, 20%, and 11%, respectively, and the third largest percentage (after *swim* and *tomcatv*) for 6-hop transfers (4%).

Figure 5.2: **Distributions of Communication Patterns for All Applications.**



5.3 Energy Savings

As explained in the two previous chapters, the total energy for the communication of data among tiles is directly related to the traffic patterns. Using Eq. 5.1 we calculate the total communication energy for each benchmark and compare it to the energy for the same communication pattern assuming the bus-based analytical model and the same number of processors in each

configuration.

Table 2: Percentage Energy Savings (2-D vs. Bus)

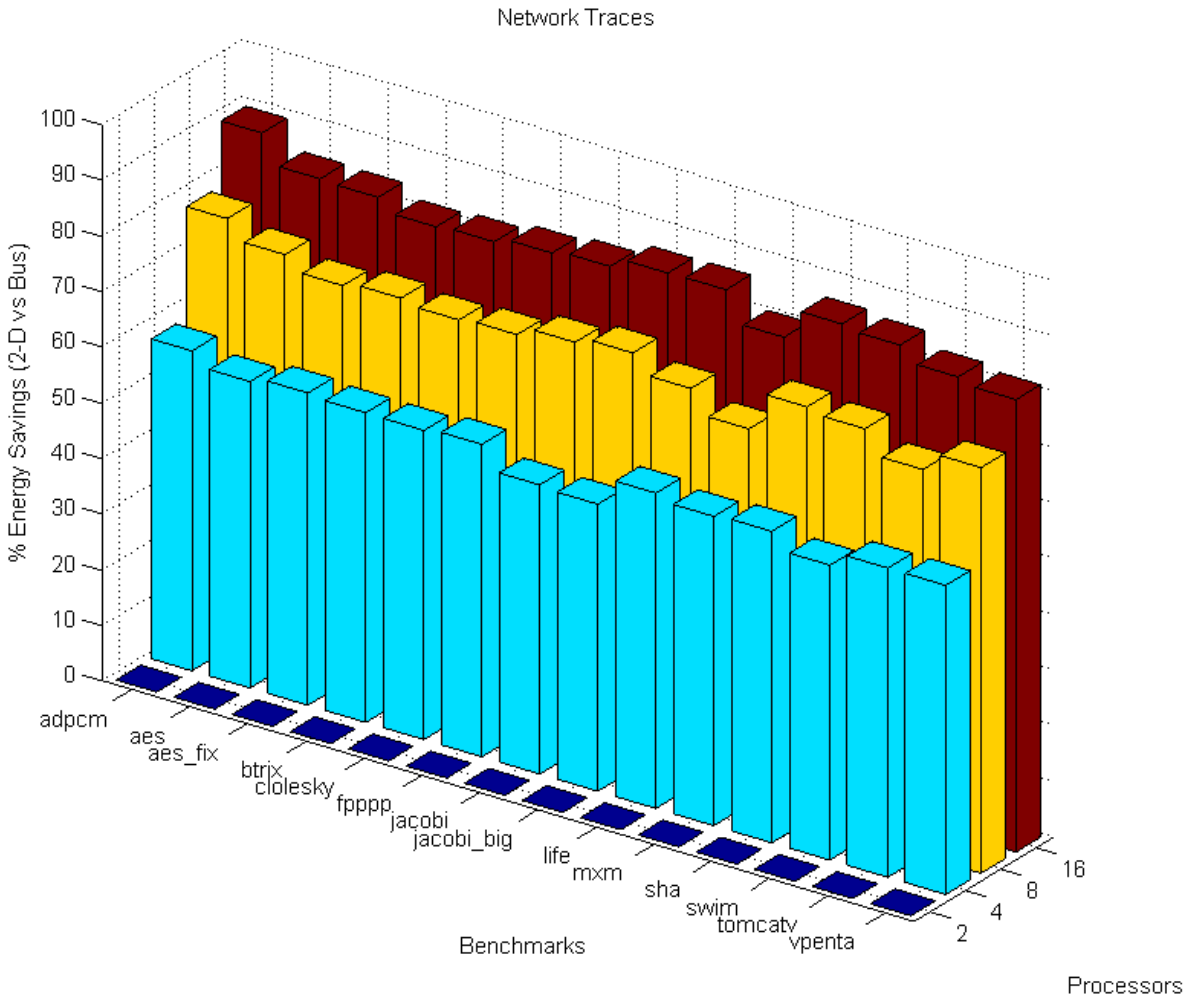
Application	4 Tiles (%)	8 Tiles (%)	16 Tiles (%)
1.adpcm	57.5	77.4	89.0
2.aes	55.1	74.1	83.9
3.aes_fix	56.1	71.6	83.6
4. btrix	55.6	72.4	81.4
5.cholesky	55.6	71.6	81.8
6.fpppp	56.1	72.1	82.9
7.jacobi	51.9	73.8	83.6
8.jacobi_big	51.6	74.9	85.5
9.life	56.7	71.7	85.6
10.mxm	55.6	67.5	80.1
11.sha	56.0	74.5	85.6
12.swim	53.0	73.7	84.9
13.tomcatv	55.6	69.5	82.3
14.vpenta	55.5	72.9	81.3

Fig. 5.3 shows the energy savings for the two-dimensional mesh network over the bus-based network for the fourteen benchmarks that we run on the four different processor configurations (2-tiles, 4-tiles, 8-tiles, and 16-tiles).

On the 4-tile configuration *jacobi_big* shows the least energy savings of 52% compared to the bus-based system. For the 8- and 16-tile configurations *mxm* shows the least energy savings of 68% and 80%, respectively. We will see that this correlates with the locality in the benchmark.

An interesting observation that rises from Fig. 5.3 is the fact that for different tile configurations the benchmarks do not always exhibit proportional energy savings. For example, when we observe the energy savings for *swim* and *tomcatv*, we see that on a 4-tile system the energy savings of *tomcatv* are higher compared to *swim*. However, on the 8-tile system *swim* exhibits higher energy savings.

Figure 5.3: **Savings Over Different Numbers of Tiles for the 14 Applications when No Contention is Assumed in the Network Switches.**



Another similar case appears by the comparison of the energy savings for an 8-tile and 16-tile system for *tomcatv* and *vpenta*. On the 8-tile system the energy savings for *vpenta* are higher; however, this case changes when the two benchmarks run on 16 tiles. The reason for this phenomenon is the way the compiler distributes the data across the tiles and then schedules the computation within each tile and the communication among them. Some applications exhibit more parallelism as the tiles increase and reduce the amount of communication among the tiles.

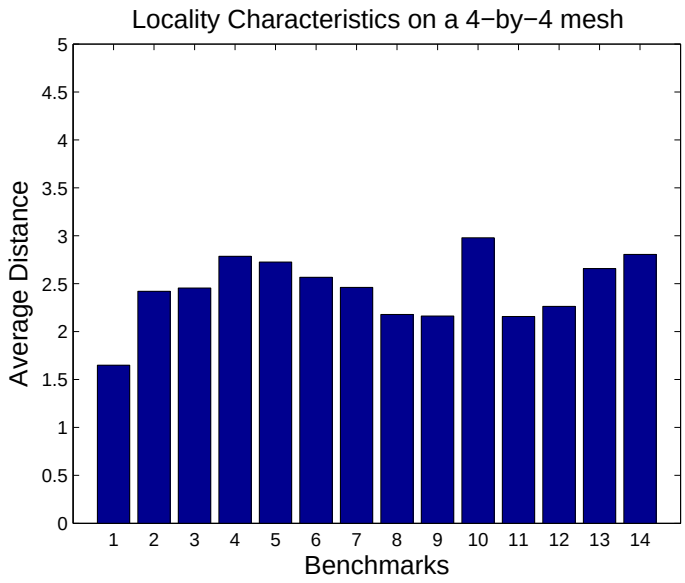
The table in Fig. 5.4 presents the average distance that a message travels in the network for all the benchmarks that we ran on a four-by-four tile configuration. This average distance captures the notion of locality for each benchmark. We showed earlier that the benchmark with the most localized communication pattern is *adpcm* and the communication patterns resembled in most cases a truncated exponential decay and a step distribution. The benchmark with the least localized pattern is *mxm*. In this case we observed permutation traffic patterns [13] directed to non-neighboring tiles, which explains why the average distance was greater than the average distance for the uniform distribution, which is 2.67[1] hops for a four-by-four mesh.

This analysis shows that applications running on tiled architectures exhibit different traffic patterns. However, if we can estimate what the communication among tiles would look like, we can model these communication patterns and have interconnection network energy reports using our framework without running the applications and recording the network traces. Especially since benchmark simulation can be expensive and not extremely useful if the designers are in the phase where they are making architectural decisions.

1.The average distance in an *n*-dimensional network with no end-around connections is given by the product of the dimensions *n* and the radix *k* divided by 3 $\left(\frac{n \cdot k}{3}\right)$ [12].

Figure 5.4: **Benchmark Locality Characteristics. Average Distance in Number of Hops for Each Application.**

Benchmark	Average Distance
1.adpcm	1.65
2.aes	2.42
3.aes_fix	2.46
4.btrix	2.79
5.cholesky	2.72
6.fpppp	2.57
7.jacobi	2.46
8.jacobi_big	2.18
9.life	2.16
10.mxm	2.98
11.sha	2.16
12. swim	2.26
13.tomcatv	2.66
14. vpenta	2.80

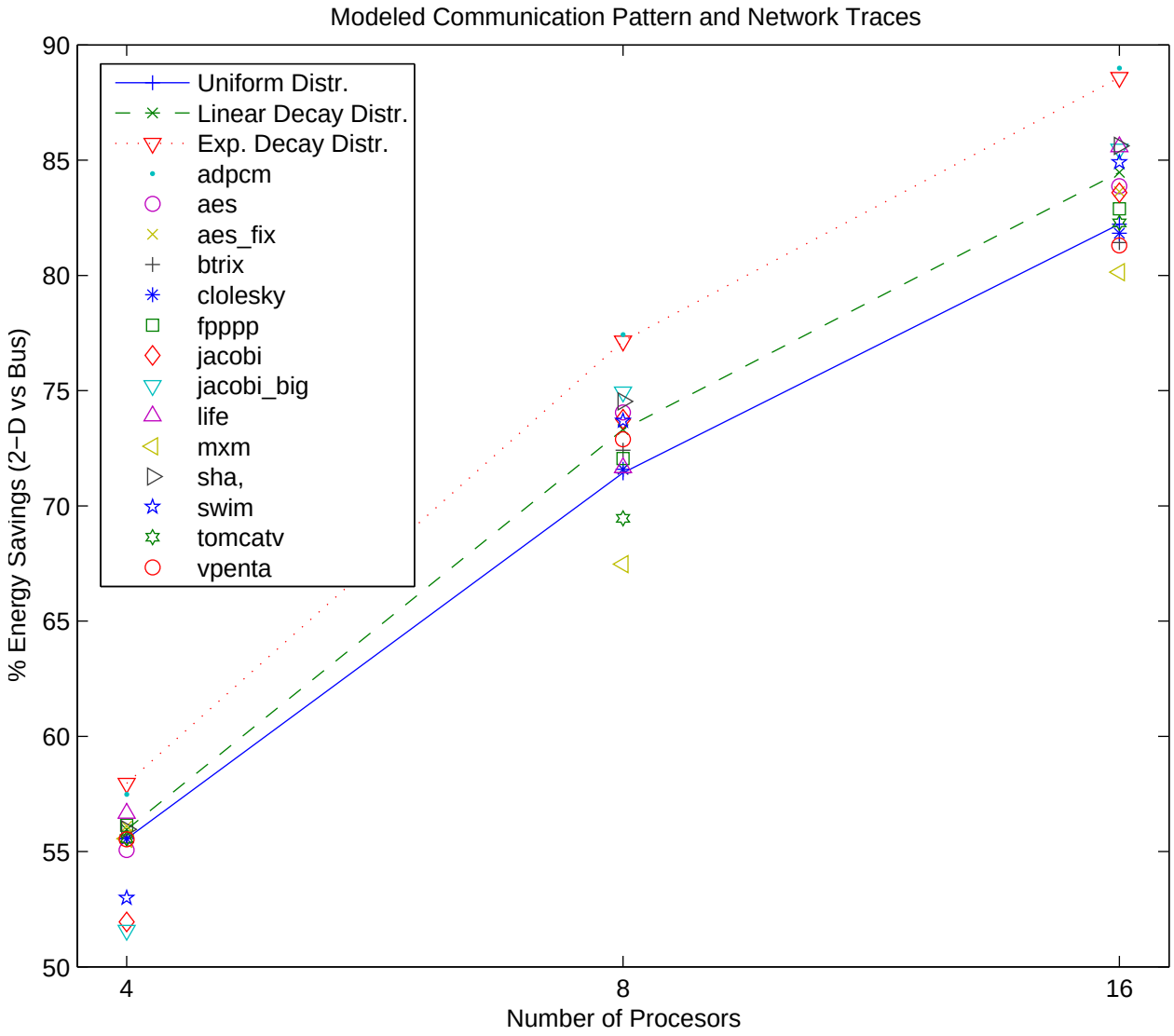


We plot the average distance for the fourteen benchmarks on a four-by-four mesh in Fig. 5.4. It is clear that the average distance correlates with the energy savings of the four-by-four mesh network compared to the bus-based system in Fig. 5.3. The data transfer average distance has a direct relation to the total energy dissipated on the interconnection network for the benchmarks examined. For example, *adpcm* has the minimum average distance compared to the other benchmarks and shows the most energy savings of all benchmarks.

5.4 Model Validation

Fig. 5.5 shows the energy savings for the different benchmarks on a 16-tile system along with the energy savings that we get using our model assuming uniform, linear and exponential decay distribution probabilities. The lines on the graph correspond to the results assuming the modeled communication probabilities (uniform, linear decay, and exponential decay distributions) and the different plot marks correspond to the results for the different benchmarks, as they are explained

Figure 5.5: **Energy Savings (2-D vs. Bus) for Benchmarks and Modeled Communication Patterns.**



in the legend.

The average distance for the linear decay probability (with parameters $b = 14$ and $a = 2$) is 2.32 hops for the sixteen processor system and best matches *life* and *jacobi*. The exponential decay probability distribution (with parameters $b = 5.5$ and $d = 0.5$) is yields greater energy savings with average distance 1.71 hops for the sixteen processor case and matches *adpcm*.

Table 3 shows the percentile error on the estimated energy savings predicted by the modeled

Table 3: Percentile Error of Modeled Savings Compared to Actual Savings

Benchmarks	Probability Distribution	Percentile Error (%) 4Tiles	8Tiles	16 Tiles
1.adpcm	Exponential Decay	1	0.4	0.5
2.aes	Linear Decay	1.6	1	1
3.aes_fix	Uniform	1	0.2	1.7
4.btrix	Uniform	0.1	1.4	1
5.cholesky	Uniform	0.1	0.2	0.5
6.fpppp	Uniform	1	1	1
7.jacobi	Linear Decay	7.7	1	1
8.jacobi_big	Linear Decay	8.5	2.2	1.2
9.life	Linear Decay	1.3	2.3	1.3
10.mxm	Uniform	0.1	5.9	4
11.sha	Linear Decay	0.2	1.6	1.3
12. swim	Linear Decay	5.6	0.5	0.5
13.tomcatv	Uniform	0.1	2.8	0.1
14. vpenta	Uniform	0.1	2	1.1

distributions compared to the actual energy savings we get using the network trace information for the three different configurations (4, 8, and 16 tile). We determine which probability distribution should model each benchmark based on the average distance for each benchmark. For example, adpcm is best modeled by an exponential decay probability distribution because its average communication distance (1.65) can be precisely modeled with this distribution.

Our modeled distributions are very flexible in modeling different traffic patterns. This is the main reason that we chose to use them in our framework. This fact is clear observing the relative errors in Table 3. In most cases we can model the traffic patterns extremely precisely with the probability distributions that represent different locality characteristics; from no locality for uniform

distribution to low locality for the linear decay distribution to high locality for the exponential decay distribution. The maximum relative error is 8.5% and the minimum 0.5% across the different configurations.

Summary

In this chapter we extensively used real network traces from benchmarks running on a tiled microprocessor to compare the energy performance of OCNs, and to validate the analytical model. We examined the communication characteristics of each benchmark, presenting statistical data on the distance (in number of hops) for the messages travelling in the network.

On the 4-tile configuration *jacobi_big* shows the minimum energy savings of 52% compared to the bus-based system. For the 8- and 16-tile configurations *mxm* shows the minimum energy savings of 67% and 80% respectively. The savings for the benchmarks that we examined vary from 80% (*mxm*) to 89% (adpcm) on a 16-tile system.

We validated our model by comparing the energy savings of the benchmarks for different tile configurations, with the expected energy savings predicted by the framework for three different probability distributions that exhibit similar average communication distance compared to the benchmarks. The maximum relative error was 8.5% and the minimum 0.5% across the different configurations. The results revealed that the set of distributions that we developed closely model the communication characteristics of benchmarks.

CHAPTER 6

SWITCH ENERGY AND WIRE LENGTH

The previous chapters ignored the energy dissipated in the switch relative to the energy dissipated in the wires. One of the main reasons for this assumptions is to gain intuition and carefully examine the effect of traffic patterns and communication locality separately.

This chapter factors the switch energy into the analysis and generalizes the model to multi-dimensional networks. We modify the equation of the expected communication energy between a pair of processors in the system to account for the energy overhead of the switch control circuitry. Mapping high-dimensional networks to a two-dimensional substrate requires longer wires than in a two-dimensional mesh network to connect the different communicating processors.

Our analysis incorporates the effect on energy of VLSI realizations of high-dimensional networks under various assumptions of communication locality. In accomplishing that, we

develop and present an algorithm that calculates the logical and physical distances among processors after they have been mapped into two dimensions. At this point in our analysis we ignore the effect of contention on the switch energy, aiming to understand the effects of processor-placing in the two-dimensional plane. However, we introduce the contention effect in our framework and analyze the switch energy components in detail in the next chapter.

6.1 Switch Energy

When we decompose the total energy communication between two processors, P_i and P_j, into the energy dissipated in the control logic in the switches and the energy dissipated on the wires along the path, the expected energy cost $E_{i,j}$ (Eq. 3.3 on page 49) of transferring data from processor P_i to processor P_j, when every link of the network has the same length, becomes

$$E_{i,j} = M \cdot p_{i,j} \cdot (E_C + E_S) \cdot H_{i,j}, \qquad 6.1$$

where E_C is the energy cost of accessing one segment that connects two neighboring processors as before and E_S is the energy dissipated in a switch to transfer a word between two neighboring processors. M is the number of messages transferred by each processor, $p_{i,j}$ is the probability that processor P_i communicates with P_j, and $H_{i,j}$ is the distance in number of hops between processors P_i and P_j.

Eq. 6.1 holds for any two-dimensional interconnection network and Eq. 3.18 is still valid ($E_l = E_C + E_S$). In the next section we will revise the equation to correspond to the energy dissipated in high-dimensional networks (three and greater) after they are mapped to two physical dimensions.

For the networks in the Raw microprocessor the values for the energy costs of the different components in the switch are $E_C = 34.5pJ$ and $E_S = 17pJ$ when there is no contention in the network. Appendix A presents the methodology for estimating the energy costs and Chapter 7 analyzes the energy expended in the switch E_S in the presence of contention in the network.

6.2 Switch Energy and Varying Wire Lengths

When high-dimensional networks are mapped to two dimensions the physical and logical distance can be different for each dimension. In this case, the expected energy cost $E_{i,j}$ of transferring the data from processor P_i [1] to processor P_j is given by

$$E_{i,j} = M \cdot p_{i,j} \cdot (E_C \cdot D_{i,j} + E_S \cdot H_{i,j}), \quad 6.2$$

where $D_{i,j}$ is the physical length of the wire, in integer multiples of l, which is traversed between processors P_i and P_j, after the network is mapped into two dimensions[2]. "l"is the minimum distance that connects two neighboring processors as in the mapping of a two dimensional mesh network. $H_{i,j}$ corresponds to the logical distance on multiple dimensions between communicating processors P_i and P_j [3].

High-dimensional networks are expected to suffer higher switch energy E_S compared to two-dimensional networks. The switch energy increases as the dimensionality increases mainly because of the higher dimensionality of the crossbar within the switch and the arbiter control logic. However, we assume that the switch energy E_S of the networks that we examine is the same.

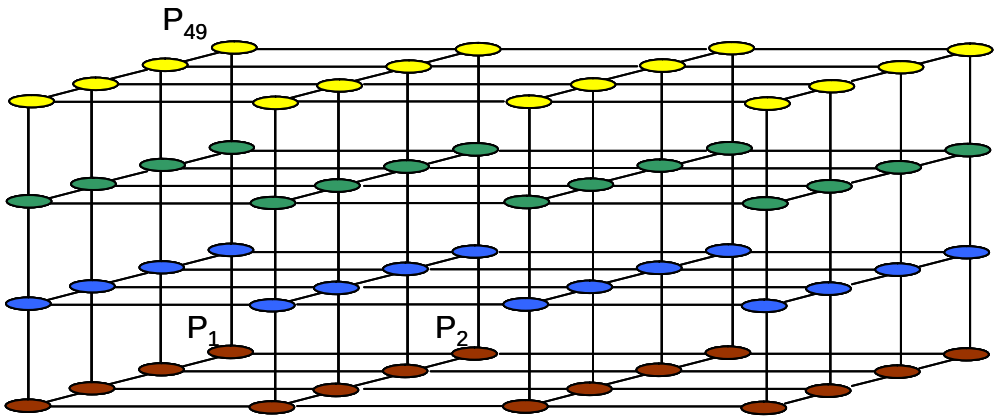
The total expected energy cost of transmitting the data when all higher dimensions are mapped to two dimensions is

$$E_{P2P} = \sum_{i=1}^{N}\sum_{j=1}^{N} E_{i,j} = M \cdot \sum_{i=1}^{N}\sum_{j=1}^{N} [p_{i,j} \cdot (E_c \cdot D_{i,j} + E_s \cdot H_{i,j})]. \quad 6.3$$

In high-dimensional networks, the physical and logical distances between processors located on the same two-dimensional plane are equal by construction of the mapping. However, $D_{i,j} \neq H_{i,j}$

1.Our labeling convention assumes that the processor IDs increase along the first, then second, then third e.t.c. dimensions.
2.Directly connecting two tiles requires manhattan routing because of VLSI constraints.
3.We still assume dimension-order routing for messages travelling in the network. Each message exhausts the path in the first dimension, then switches to the second dimension and higher dimensions.

Figure 6.1: **64-Tile System Implemented in Three Dimensions. There are 4 Processors in Each Dimension.**



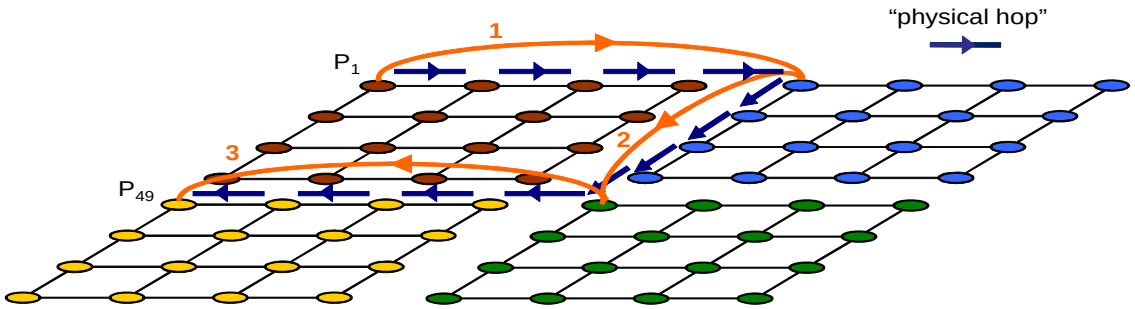
if processors P_i and P_j are not on the same plane. The different values of D and H have different effect on the total energy. The probability distribution of the communication between the processors captured in the communication probability matrix p can weight the effect of D and H differently. The effect of communication locality on the total energy is investigated in Section 6.5.

6.3 High-Order Dimension Mapping

This section depicts an actual visualization of the mapping of high-order dimensions to the physical plane by showing how a three-dimensional network with four processors on each dimension is mapped to two dimensions. This work is related to [12]; [12] used the mapping of wires into two-dimensions to calculate delays. We do so to calculate energies.

Fig. 6.1 shows a three-dimensional system with 64 tiles arranged in a cube with four processors ($k = 4$) in each dimension. We assume that the cores occupy area and that the wires that connect the cores are laid on top. In this arrangement, messages from processor P_1 to P_{49} traverse three hops. When the system is mapped to two dimensions, as shown in Fig. 6.2, the logical distance remains the same and corresponds to the total number of switches the messages have to go through until they reach their final destination (P_{49}). We show the logical distance for this case with the orange arrows in Fig. 6.2.

Figure 6.2: **Mapping of the 64 Tiles Into Two Dimensions. The Logical Distance Between P_1 and P_{49} is 3 Hops and the Physical Distance is 12 "Physical Hops".**



On the other hand, the physical distance between the two processors is different and the equivalent wire length that a message has to travel adds up to the equivalent of twelve "physical hops". We show this with the blue arrows in Fig. 6.2. As "physical hop" we define the minimum physical wire length long enough to connect two neighboring cores. Essentially, it is the distance between the centers of physically-adjacent cores.

6.4 Calculation of Matrices *D* and *H*

We calculate the matrix *"H"* that describes the logical distance between two processors and the matrix *"D"* that describes the normalized physical distance between two processors in a mesh, with the following algorithm implemented in Matlab. We present the case of a four-dimensional network mapped to two dimensions.

```
function [H] = four_dimensions_logical(k1,k2,k3,k4)
% FOUR_DIMENSIONS: Logical distance for a 4-D network.
%     H(i,j) returns the logical distance in hops from processor P_i to
%     processor P_j  for a 4-D mesh.

  N = d1*d2*d3*d4;                                % Total Number of nodes
  [X,Y,Z,W] = ndgrid(1:k1,1:k2,1:k3,1:k4);        % Create N-dimensional grid
  x = repmat(X(:),1,N);                           %
  y = repmat(Y(:),1,N);
  z = repmat(Z(:),1,N);
```

```
  w = repmat(W(:),1,N);
  s = sort([d1 d2]);                              % Assumption:d1,d2 ≥ d3,d4
  H = abs(x-x')+abs(y-y')+abs(z-z')+abs(w-w');


function [D] = four_dimensions_physical(k1,k2,k3,k4)
% FOUR_DIMENSIONS: Physical distance for a 4-D network mapped to plane.
%     D(i,j) returns the physical distance from processor P_i to processor P_j
%     for a 4-D mesh after it is mapped to 2-D normalized to the minimum
%     processor distance in two dimensions.

  N = d1*d2*d3*d4;                                % Total Number of nodes
  [X,Y,Z,W] = ndgrid(1:k1,1:k2,1:k3,1:k4);        % Create N-dimensional grid
  x = repmat(X(:),1,N);                           %
  y = repmat(Y(:),1,N);
  z = repmat(Z(:),1,N);
  w = repmat(W(:),1,N);
  s = sort([d1 d2]);                              % Assumption:d1,d2 ≥ d3,d4
  D = abs(x-x')+abs(y-y')+abs(z-z')*s(1)+abs(w-w')*s(2);
```

Description of the algorithm:

The function "four_dimensions_logical" takes as input arguments the number of processors on each dimension k_i and returns matrix "H". The elements $H(i,j)$ hold the logical distance between processors P_i and P_j respectively.

The function "four_dimensions_physical" takes as input arguments the number of processors on each dimension k_i and returns matrix "D". The elements $D(i,j)$ hold the physical distance between processors P_i and P_j respectively. The distance is normalized to the minimum physical distance in two dimensions. $D(i,j)$ returns the number of "physical hops" that connect processor P_i to P_j.

We describe each command of the algorithm below:

- [X,Y,Z,W] = ndgrid(1:k1,1:k2,1:k3,1:k4);

  This command transforms the domain specified by the vectors 1:k1, 1:k2, 1:k3, and 1:k4 into the 4-dimensional array grids X, Y, Z, W respectively. The i-th dimension of the output array (X, Y, Z, W) are copies of elements of the corresponding vector.

- x = repmat(X(:),1,N);

  The command X(:) transforms the 4-dimensional array into a vector with elements the elements of the array indexed with increasing dimensions. The command `repmat(X(:),1,N);` replicates the elements of the vector X, N times creating an N-by-N array.

- s = sort([d1 d2]);

  The command sort([d1 d2]) sorts the radices of the x and y dimensions (k1 and k2) and stores the incrementing values in vector “s”. We make the assumption that $d1$, $d2 \geq d3$, $d4$. s(1) contains the smallest of the two dimensions $\{d1, d2\}$.

- H = abs(x-x’)+abs(y-y’)+abs(z-z’)+abs(w-w’);

  The absolute value of the difference of the matrix “x” and the transpose matrix “x’ ” gives the equivalent logical number of hops the message travels on that dimension. $H_{i,j}$ is the total count of orange arrows shown in Fig. 6.2.

- D = abs(x-x’)+abs(y-y’)+abs(z-z’)*s(1)+abs(w-w’)*s(2);

  The absolute value of the difference of the matrix “x” and the transpose matrix “x’ ” gives the equivalent number of hops the message travels on that dimension. The algorithm first lays the highest dimension along the largest of the $\{d1, d2\}$ dimension and then lays the third dimension. $D_{i,j}$ is the total count of the blue arrows shown in Fig. 6.2 and corresponds to the total “physical” hops.

6.5 Energy Comparison for High-Dimensional Networks

To illustrate the trade-offs between the energy costs of the wire and the switch in association to the dimensionality of the system, we compare the total energy of three point-to-point networks: a two-dimensional network with processors laid out in an array of sixteen rows and sixteen

columns, a three-dimensional network with twelve processors on the first dimension, seven processors on the second dimension and three processors on the third dimension[1], and a four-dimensional network with four processors on each dimension. Using Eq. 6.3 we calculate the total normalized (for $M = 1$ and $E_C = 1$) energy for the three networks assuming uniform and exponential decay distributions.

The average logical and physical distances for the uniform and exponential decay distributions are shown in Table 1. It is clear that the logical and physical distances are the same for two-dimensional systems.

Table 1: Average Physical and Logical Distance for Uniform and Exponential Decay Distributions

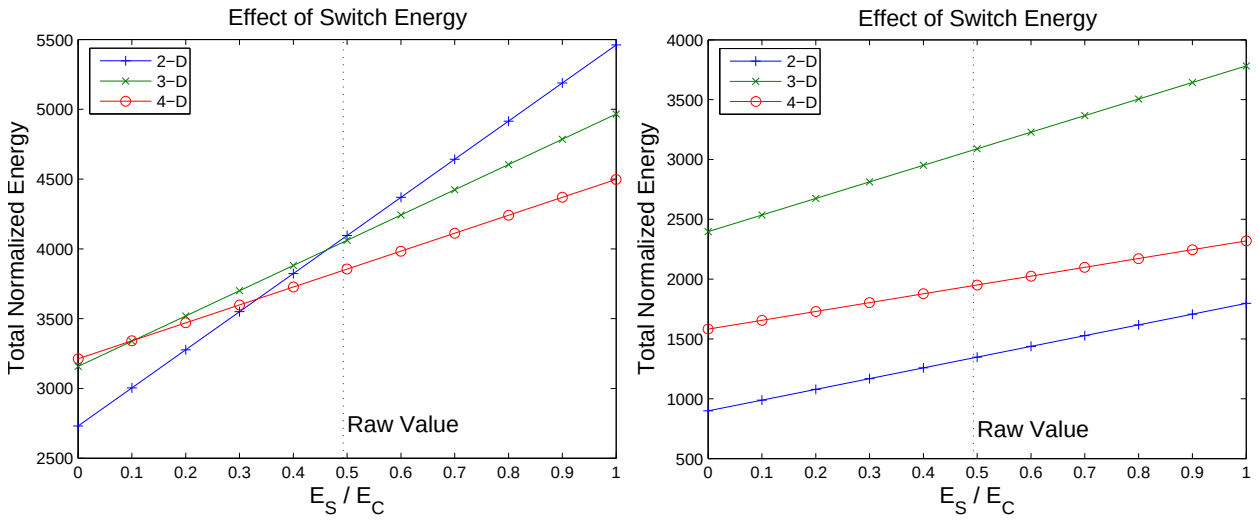
Dimensions	Uniform		Exponential	
	Phys. Dist.	Log. Dist.	Phys. Dist	Log. Dist
2-D	10.67	10.67	3.51	3.51
3-D	12.53	7.18	9.51	5.49
4-D	12.55	5.02	6.18	2.88

Varying the switch energy, E_S, the trade-offs between the physical and the logical distance are evident in the following figure. Fig. 6.3(a) plots the total energy when communicating processors send messages to each other with equal likelihood. Note that we plot the total energy vs. the ratio of the energy cost of the switch, over the energy cost of the wires E_S/E_C.

When the switch energy, E_S, is small relative to the channel energy, E_C, the most energy efficient network topology is the two-dimensional because it has the smallest physical distance compared to the other networks. The high-order networks, on the other hand, show smaller logical distance, so when the switch energy increases, the portion of the total energy dissipated

1.We choose this arrangement although it results in fewer processors than the other two systems (252 vs. 256) because it provides an efficient trade-off between the physical and logical distance when a three-dimensional network is laid out in two dimensions.

Figure 6.3: **Energy Comparison for 2-D, 3-D and 4-D Networks with 256 Nodes.**



(a) Uniform Distribution

(b) Exponential Decay Distribution

on the switch becomes significant. Because of the smaller logical distance of the four-dimensional network, the total energy of the two-dimensional network exceeds the total energy of the four-dimensional network at a smaller value of E_S compared to the three-dimensional network.

However, two-dimensional networks are more energy efficient in the presence of communication locality in the traffic patterns. Fig. 6.3(b) plots the total normalized energy for the three networks assuming an exponential decay probability distribution. The average physical distance for the two-dimensional network is much smaller compared to the average physical distance of the three and four-dimensional networks (3.51, 9.51 and 6.18 hops for the two-, three-, and four-dimensional networks, respectively). The two- and four-dimensional systems have comparable logical distance values (3.51 and 2.88 hops, respectively) due to the high communication locality of the exponential decay distribution, while the logical distance for the three-dimensional system is larger (5.49 hops).

The graphs also depict the actual ratio of the switch energy cost over the channel energy cost $E_S/E_C = 0.493$ for the dynamic networks in the Raw multicore processor. As discussed in the next chapter, the measured values are $E_S = 17pJ$ and $E_C = 34.5pJ$ for one network hop in the dynamic network between two processors.

From this analysis, there is an evident connection between the effect of the switch energy cost on the total energy, and the effect of switch delays on the latency of interconnection network discussed in [12]. In [12] it is shown that two-dimensional networks have the lowest latency when switch delays are ignored, but higher-dimensional networks are favored otherwise. As is the case for the total energy dissipation, two-dimensional networks regain their advantage when communication locality exists.

The graph also reveals that high-order networks with a dimensionality that is not a power of two are not energy efficient when they are mapped into two dimensions. The reason is that there can hardly ever be an arrangement that minimizes both the physical and the logical distances between communicating processors. Minimizing the logical distance requires a number of processors that is as similar as possible on each dimension. Minimizing the physical distance requires the planar realization of the network to have a number of processors on the two dimensions that are as similar as possible.

The previous analysis assumes that the compiler does not change the distribution of data and code or the scheduling of instructions for the topologies that get mapped on the two-dimensional plane. If the compiler changes the distribution and scheduling to exploit the new topology, our framework can still calculate the total communication energy. Eq. 6.3 holds for any traffic pattern or any topology. Matrices p, D, and H need to be adjusted to correspond to the new distribution of data and scheduling of instructions performed by the compiler.

Summary

High-order point-to-point interconnection networks exhibit interesting trade-offs when they are

mapped to two-dimensions. We presented an algorithm that calculates the logical and physical distances when those network are mapped to two dimensions. The energy comparison of a two-, three-, and four dimensional network revealed that communication locality is essential for choosing the most energy efficient network.

We show that when the switch energy is small relative to the channel energy, the most energy efficient network topology is the two-dimensional because of its small physical distance compared to the other networks for a uniform distribution. The four-dimensional network, on the other hand, is the most efficient when the switch energy increases. However, two-dimensional networks are more efficient in the presence of communication locality in the traffic patterns.

CHAPTER 7

CONTENTION ENERGY ANALYSIS FOR MULTI-DIMENSIONAL NETWORKS

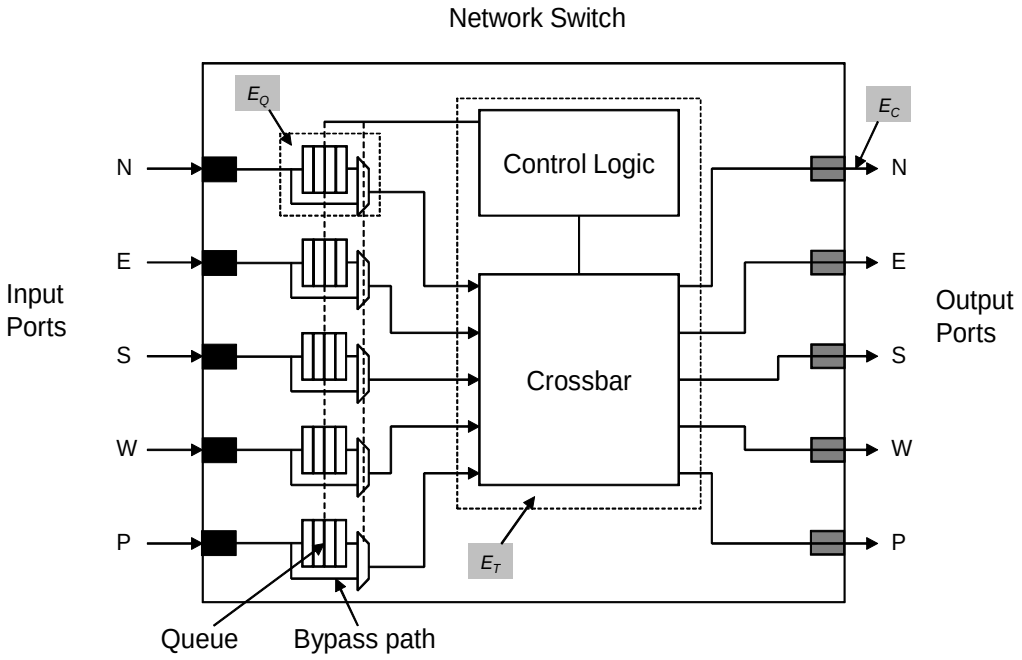
This chapter presents an analysis of the effects of network contention on the total energy dissipation in on-chip interconnection networks. We present a high level schematic of a typical network router and show the major components for energy dissipation. We expand our framework to include the effect of network contention and the resulting energy dissipated on the network queuing buffers. We examine contention in the interconnection network when processors communicate with equal likelihood and present energy estimates for different channel utilization values. We derive a closed-form solution for the energy dissipation on the network for different values of channel utilization and present lower and upper bounds on the energy dissipated using traces from benchmarks running on the Raw microprocessor.

7.1 Switch Model

The model assumes that the switch has a buffer associated with every input port at each network dimension as well as a buffer for messages generated from the processor of the same node. When multiple packets request the same output port in a cycle, the control logic arbitrates among them allowing only one message to be transmitted on the output port and causing the other messages to be queued in their respective input queuing buffers.

Fig. 7.1 shows the high-level microarchitecture of a typical on-chip network switch similar to the one used in the Raw multicore processor. The switch consists of input and output ports, input buffers, a crossbar, and control logic circuitry. The input and output ports can correspond to the processor port and north, east, south, and west directions of the node.

Figure 7.1: **Typical Microarchitecture of An On-Chip Network Switch for 2-D Mesh.**



In the schematic we depict the energy costs that are essential to our analysis. E_C is the energy consumed at the channels that connect two neighboring nodes as before. E_C consists of the energy dissipated on the wires that connect an output port of the switch to the corresponding

input port on the neighboring switch and the energy dissipated on the buffers that are used to drive the signals (for example in Raw there are two buffer stages).

E_T is the energy dissipated on the crossbar and the switch control logic circuitry that determines whether a message is consumed at the node, changes or continues in the same direction, or gets queued at the input buffer. The energy expended in the crossbar is dominated by wiring capacitances that connect the inputs with the outputs through various multiplexors, while the energy of the control logic is expended in the gates that perform the logic functions.

E_Q is the energy expended when writing and reading the message in the input buffer queue (energy lumped together for a FIFO write and subsequent read). In general, depending on the implementation of the FIFO there are trade-offs concerning the energy and performance of the input buffer. A memory circular queue implementation is energy-aware since reading values from different locations does not require any additional energy dissipated for advancing messages in the queue. Input buffers can also be implemented using shift registers.

Depending on the implementation of the buffer, E_Q is not significantly dependent on the total buffer size. In an energy-efficient implementation, the clocks that drive the flip-flops should be clock-gated and the control logic that monitors the writing and reading of data can fairly simply clock-gate the flip-flops that are not accessed that specific cycle. Routing data in and out of the FIFO is going to be more energy consuming as the buffer size increases, however, the energy expanded to write the data to the flip-flops and clock energy are the most significant energy components in a FIFO and these components are not determined by the buffer size.

Input buffers have a bypass path that a message takes in the case when the queue is empty and the output port is not servicing any other requests. The energy E_S in Eq. 6.1 is a function of the two energy costs E_T and E_Q. The input buffer in a Raw switch is a 4-entry, 32-bit wide FIFO.

Figure 7.2: **Physical Placement of the Crossbar, the Control Circuitry, the Input Queue Buffer, the Channels for the Dynamic Network in Raw.**

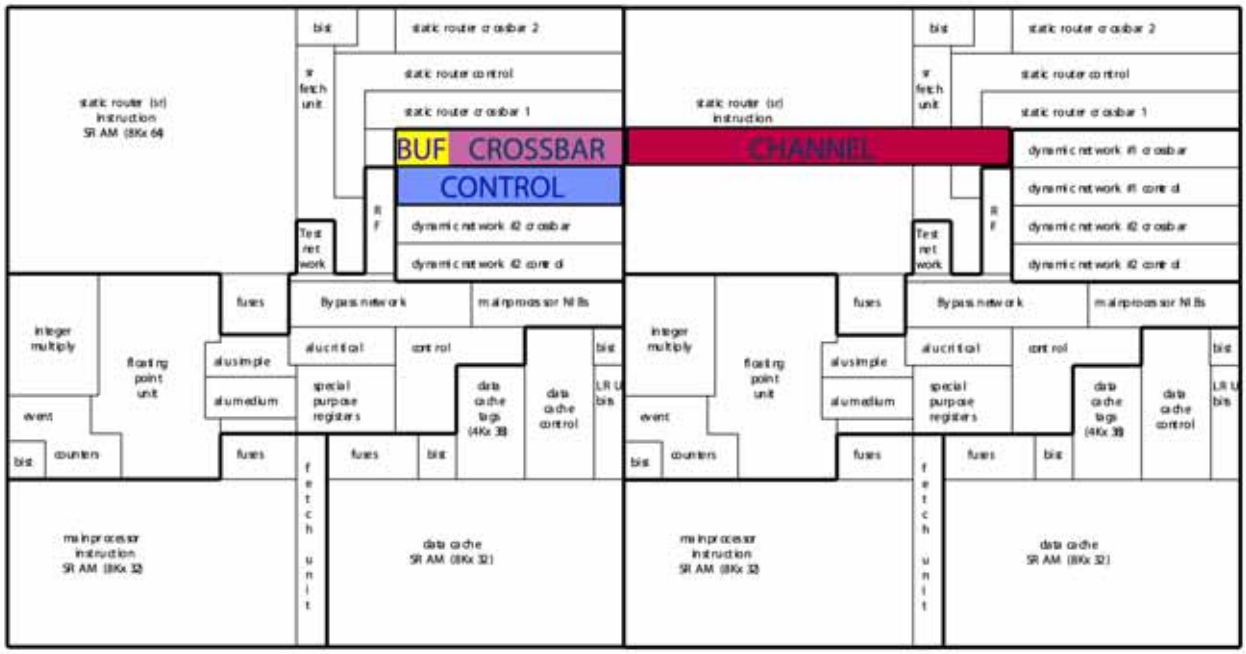


Fig. 7.2 shows the physical placement of the crossbar, the control logic circuitry, the input queue buffer, and the channels for one of the dynamic networks in the Raw multicore processor. For the networks in the Raw microprocessor the values for the energy costs of the different components in the switch are: $E_C = 34.5pJ$, $E_T = 17pJ$, and $E_Q = 12pJ$.

Appendix A presents in detail the methodology for estimating the energy costs. We present here a brief summary of the methodology. We estimated the energy costs based on capacitance values from the Raw microprocessor dynamic networks. For wiring and metal capacitance values, we used extracted capacitance values generated by the IBM ChipEdit capacitance extractor tool for the final layout of the Raw microprocessor. For the cell input and output capacitances we used the values provided in the IBM documentation for their cells in the SA-27E process.

The estimation of E_C involved following a path from the west output of a tile to the east input of the neighboring one and measured the capacitance of the total length of the channel and the input and output capacitance of the inverters that propagate the 32-bit signals. We assumed independent, identical data in our analysis for the energy estimation.

Similarly, the energy cost of the crossbar is due to the propagation of the signals (energy dissipated in the wires) after the input buffer to one of the output ports of the switch and the energy dissipated in the multiplexors and drivers to direct the data to a specific output port. The energy in the control logic is for the generation of the signals that determine the route of the data (from the input to one of the five possible output ports) and the signals that control the input queue buffer.

The major energy components of the input queue buffer are the costs of routing the data to one of the four entries of the FIFO for a write, the cost of routing the data out of the FIFO to the multiplexor after a read, the cost of storing the data in the flip-flops of the FIFO, and the energy dissipated for routing the clock to the flip-flops.

In Section 8.4 we present an analysis for the scaling of the energy dissipated in the switch and the channels. If the scaling factor comparing two process technologies is α, the analysis suggests that these energy values will scale with α.

7.2 Life of a Message in the Network

In the absence of contention on the output port, the message goes through a bypass path[1] within the input queueing buffer to the crossbar. In this case, the energy cost for moving a message between two neighboring processors is

$$E_l = E_C + E_T. \qquad 7.1$$

On the other hand, when the output port is not free, the message is queued into the input buffer

1.We assume that the bypass energy is negligible.

and is read when the output port has finished servicing other messages. Therefore the above equation becomes

$$E_l = E_C + E_T + E_Q. \quad 7.2$$

At any given switch, the energy overhead of queueing the message into the input buffer depends on the probability the message is delayed due to contention. The expected energy cost E_S within the switch is a function of E_T and E_Q described by

$$E_S(E_T, E_Q) = E_T + q \cdot E_Q, \quad 7.3$$

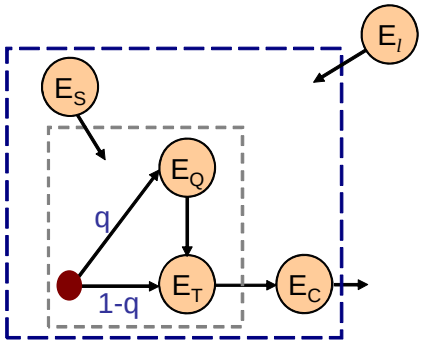
where the probability q describes the likelihood that a message arriving at a switch will suffer contention and will be stored in the queue buffer. For a message that is injected into the network and has to travel some number of hops to its destination, there is a lower and upper bound on the energy expended due to contention.

Summarizing, when there is no contention for the output port, the energy dissipated in the switch is $E_S = E_T$; with contention $E_S = E_T + E_Q$. Thus, we can model the expected energy of a switch as

$$E_l = E_C + E_T + qE_Q. \quad 7.4$$

Fig. 7.3 describes the previously equation graphically.

Figure 7.3: **Expected Energy Consumption of a Message for One Network Hop.**



● : message

q : probability of contention

We can generalize Eq. 7.1 and Eq. 7.2 to express the energy dissipated on the whole message path. In the case of a two-dimensional network, the expected energy $E_{i,j}$ dissipated on the interconnection network for a message originating from processor P_i with destination processor P_j (Fig. 7.4) is given by

$$E_{i,j} = (E_C + E_S) \cdot H_{i,j}, \qquad 7.5$$

where $H_{i,j}$ represents the distance (in number of hops) between the two processors. This is true since all wires are minimum length.

For a message from P_i to P_j the upper energy bound is

$$E_{max} = (E_C + E_T + E_Q) \cdot H_{i,j}, \qquad 7.6$$

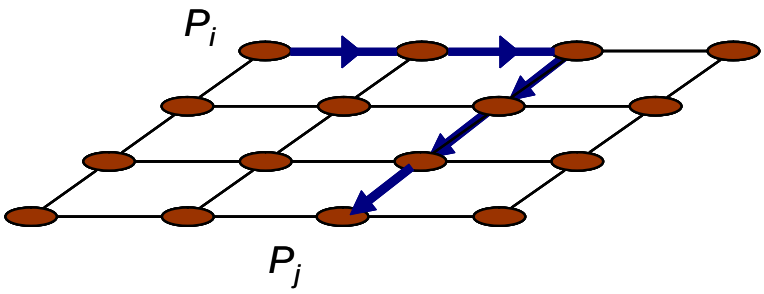
when the message suffers contention at every switch in the path of the message.

The lower bound is given by the following equation

$$E_{min} = (E_C + E_T) \cdot H_{i,j}, \qquad 7.7$$

in the absence of contention in the network. Notice that the energy is related not only to contention, but also the number of times the message is queue in its path.

Figure 7.4: **Example of a Message Route on a 2-D Network.**

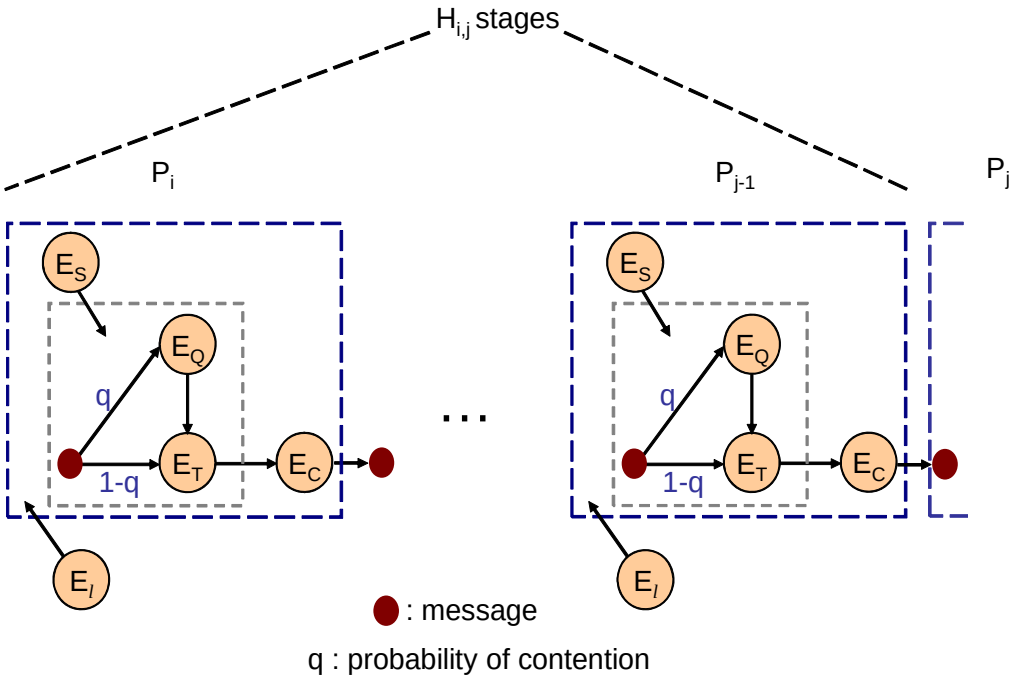


We can graphically represent the above equations with the process shown in Fig. 7.5. A message generated by processor P_i will go through the switch with no delay with probability $1 - q$. In this case, the total energy dissipated for one hop is $E = E_T + E_C$.

The probability of contention is encapsulated by probability q. The energy dissipated at a switch when there is contention is $E = E_Q + E_T + E_C$. This process is repeated $H_{i,j}$ times at each switch, until the message reaches its destination.

The analysis so far reveals that minimizing the energy overhead of storing messages into the input queue buffers requires careful scheduling by the compiler. Specifically, compilers need to schedule message injections into the network after assuring a minimum number of conflicts for resources for the entire message path. From the energy standpoint, a message should wait as many cycles as possible in a buffer until contention is minimized at every stage of the total route.

Figure 7.5: **Process Describing the Total Expected Energy Cost for Moving a Message from P_i to P_j.**



7.3 Calculation of Contention Probability

We can calculate the probability q of contention at the switch in a given cycle in the case of uniform communication patterns among the processors in the system for a message entering the switch requesting an output port. The probability q is a function of the number of messages that request the given output port on any cycle and the state of the queue corresponding to the input port on which the message arrived, whether it is empty or not.

The analysis that describes the calculation of the contention probability for the uniform distribution is performed under the condition that a message is entering the switch through one of the input ports. Thus, the probability q is

$$q \text{ is the probability for a message entering a switch to face contention}, \quad 7.8$$

or

$$q = probability\left(\begin{array}{c} \text{contention for a} \\ \text{given output port} \end{array} \middle| \begin{array}{c} \text{a message entered the switch} \\ \text{requesting that output port} \end{array} \right). \quad 7.9$$

If the queue is not empty, any new message entering the switch will contend and will be stored in the queue. This probability is described by the probability there is *at least* one message in the queue.

If the queue is empty, the probability the message will contend depends on the total number of messages requesting the same output port. We define υ as the random variable that describes the number of messages that request a single output port. If the message is the only one requesting the output and the queue is empty, it will not contend for it. With an empty queue, there is probability of contention when $\upsilon \geq 2$ and that probability is $(\upsilon - 1)/\upsilon$, since every message has equal probability of $1/\upsilon$ to be serviced.

Therefore, we can write the probability of contention q on any given cycle as

$$\left[\begin{array}{c} q \text{ equals the probability that the queue is not empty } and \\ \text{the probability the message is not routed out given the queue is empty} \end{array} \right], \quad 7.10$$

or

$$q = \left[\begin{array}{c} \text{probability queue} \\ \text{is not empty} \end{array} \right] + \left[\begin{array}{c} \text{probability queue} \\ \text{is empty} \end{array} \right] \times \left[\begin{array}{c} \text{probability the message} \\ \text{is not routed out} \end{array} \right]. \quad 7.11$$

Equivalently,

$$q = p_{Q \neq 0} + p_{Q = 0} \cdot \sum_{\upsilon = 2}^{\infty} p(\upsilon) \cdot \frac{\upsilon - 1}{\upsilon}, \quad 7.12$$

where

- $p_{Q \neq 0} = 1 - p_{Q = 0}$ is the probability the queue has *at least* one message.
- $p_{Q = 0}$ is the probability the queue is empty.
- $p(\upsilon)$ is the probability υ messages request that specific port.

For M/M/1, M/G/1, and M/D/1 systems the probability of an empty queue is $p_{Q=0} = 1 - \rho$ ([61], [60]), where ρ is the channel utilization and describes the probability of a message arriving at an incoming channel (and the probability there is *at least* one message in the queue).

Thus, Eq. 7.12 becomes

$$q = \rho + (1-\rho) \cdot \sum_{\upsilon=2}^{\infty} p(\upsilon) \cdot \frac{\upsilon - 1}{\upsilon}. \qquad 7.13$$

As suggested in [12], we can determine ρ as follows. Let m be the message injection rate to the network by the processor[1]. Each message travels k_d hops on average in each dimension, for a total of nk_d hops. Since each switch has $2n$ associated channels, the channel utilization is given by

$$\rho = \frac{m \cdot n \cdot k_d}{2n} = \frac{m \cdot k_d}{2}. \qquad 7.14$$

Next, we need to calculate $p(\upsilon)$.

In [12], Agarwal calculates the probability distribution $p(\upsilon)$ of the random variable υ for uni-directional channels. We extend the analysis performed in [12] to derive the probability mass function of the random variable υ for bi-directional channels and provide a closed-form solution for the probability q (Eq. 7.13).

The average number of hops that a message travels in one dimension in a network with no end-around connections and bi-directional channels is $k_d = (k - 1/k)/3$, where k is the number of processors on each dimension.

Any message arriving with probability ρ at the switch has three options: continue along the same direction, change direction, or be consumed by the processor.

1.Our analysis uses the message injection rate m. This relates to the number of messages M generated by a processor (see Chapter 3) by dividing M to the total communication time.

Figure 7.6: **Channel Utilization at a Switch. The channel utilization ρ is composed of three components: ρ_i (messages generated/consumed by the node processor), ρ_c (messages continuing to this direction through the switch), and ρ_s (messages switching to this direction at the switch). (a) shows how ρ is composed and (b) shows how ρ is decomposed.**

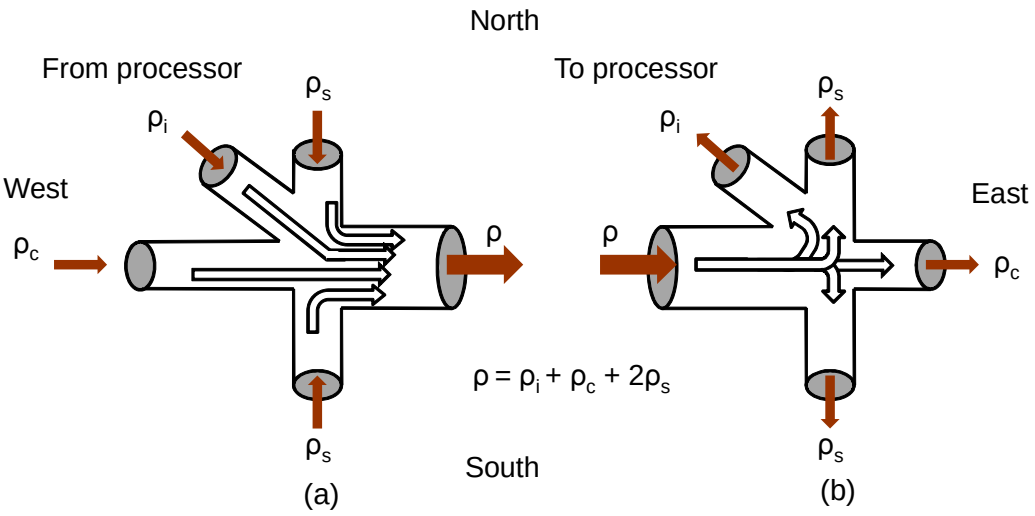


Fig. 7.6 shows these options; Fig. 7.6.a shows how ρ is composed and Fig. 7.6.b shows how ρ is decomposed. The message probability ρ in a channel along a given dimension is composed of three components:

- ρ_i: messages injected (or consumed) into this direction from the node processor.
- ρ_s: messages that switch to this direction in the switch.
- ρ_c: messages continuing along the same direction through the switch.

Those probabilities are computed as follows. The probability a message is generated by the processor at the switch in any cycle is m, and the probability the message routes to any output channel is $1/(2n)$. Therefore, $\rho_i = m/(2n) = \rho/(nk_d)$.

Out of ρ, ρ_i is the probability a message exits the network. Thus, the probability a message remains in the network is $\rho - \rho_i$. A message switches dimensions once every k_d hops; the

probability it changes direction at any given cycle is $\rho_s = (\rho - \rho_i)/(2k_d)$, since there are two directions (North and South in Fig. 7.6).

Similarly, the probability that a message will continue along the same dimension is $\rho_c = (\rho - \rho_i)(1 - 1/k_d)$.

In a mesh with bi-directional channels $\upsilon \in \{1, 2, 3, 4\}$ (Fig. 7.6.a). We can know describe the distribution of υ as

$$p(\upsilon) = \begin{cases} (1-\rho_s)^2(1-\rho_c)(1-\rho_i) & (\upsilon = 0) \\ \rho_c(1-\rho_s)^2(1-\rho_i) + \rho_i(1-\rho_c)(1-\rho_s)^2 + & \\ \quad + 2\rho_s(1-\rho_s)(1-\rho_c)(1-\rho_i) & (\upsilon = 1) \\ 2\rho_c\rho_s(1-\rho_s)(1-\rho_i) + \rho_s^2(1-\rho_c)(1-\rho_i) + & \\ \quad + 2\rho_i\rho_s(1-\rho_s)(1-\rho_c) + \rho_i\rho_c(1-\rho_s)^2 & (\upsilon = 2) \\ 2\rho_i\rho_c\rho_s(1-\rho_s) + \rho_i\rho_s^2(1-\rho_c) + \rho_c\rho_s^2(1-\rho_i) & (\upsilon = 3) \\ \rho_c\rho_i\rho_s^2 & (\upsilon = 4) \\ 0 & (\upsilon > 4) \end{cases} . \qquad 7.15$$

Using Eq. 7.15 we can write Eq. 7.13 as

$$q = \rho + (1-\rho) \cdot \sum_{\upsilon=2}^{4} p(\upsilon) \cdot \frac{\upsilon - 1}{\upsilon} = \rho + (1-\rho) \cdot \left[\frac{p(2)}{2} + \frac{2p(3)}{3} + \frac{3p(4)}{4} \right]. \qquad 7.16$$

The previous equation is a closed-form solution and reveals that the contention probability is a function of the channel utilization, the average distance for message, and the network dimensionality. For a specific network, where n and k_d are set, the probability of contention on the network depends only on the channel utilization ρ.

Figure 7.7: **Probability of Contention for Uniform Distribution.**

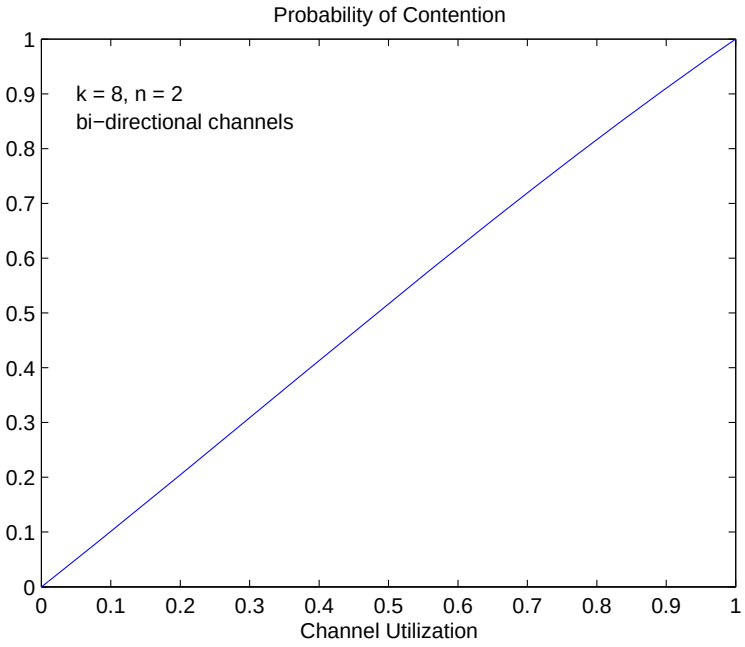


Fig. 7.7 shows the probability of contention at the switch at any given cycle, assuming a uniform communication pattern for different values of the channel utilization ρ and a two-dimensional mesh with $k = 8$ for both networks that we examined.

The graph looks like a linear function with regard to the channel utilization ρ and reveals that when a new message arrives at the switch it will contend mostly due to a non-empty buffer ($p_{Q \neq 0} = \rho$).

Next, we simplify equation (Eq. 7.16) that describes the probability of contention q with an approximation of the probability for contention. We can rewrite Eq. 7.16 as

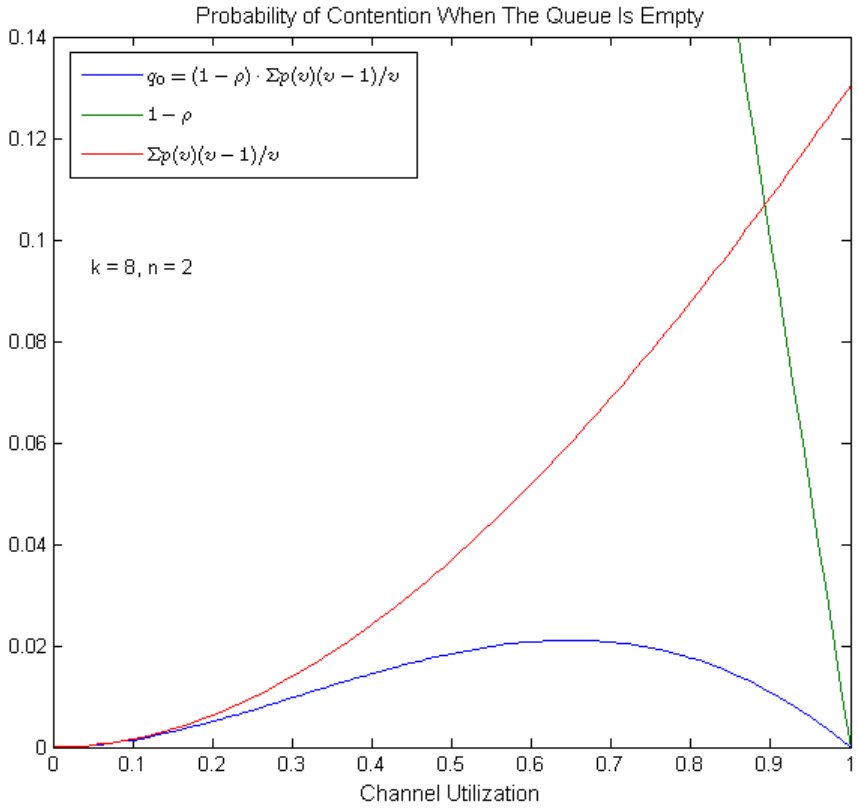
$$q = \rho + q_0, \tag{7.17}$$

where

$$q_0 = (1-\rho) \cdot \sum_{\upsilon = 2}^{4} p(\upsilon) \cdot \frac{\upsilon - 1}{\upsilon}, \tag{7.18}$$

and describes the probability of contention for a message entering the switch when the queue is

Figure 7.8: **Probability of Contention for Uniform Distribution when the Queue is Empty.**



empty. Probability q_0 is plotted in Fig. 7.8.

This graph reveals that when the buffer is empty the probability a message will contend at the switch is very low $q_0 < 0.0211$. An intuitive explanation of the graph comes from the nature of the uniform distribution. Since processors communicate with random patterns, the probability that a large number of messages have as destination a unique processor (or a row or column of processors) is very low. Therefore, the probability that many messages request a specific output port is also low.

Fig. 7.7 and Fig. 7.8 reveal that, at any given cycle, messages that arrive to the switch or generated by the node processor have minimal contribution to the total probability of contention at that node. An incoming message will contend mostly due to an non-empty queue buffer.

Thus, we can approximate the probability of contention q (Eq. 7.13) in the switch with q' as

$$q' \approx \rho = \frac{mk_d}{2}, \tag{7.19}$$

where m is the message injection rate, k_d the average distance per dimension, and not deviate significantly from the exact value of q. In this case, $q_0' = 0$.

For nonunit-sized messages, the previous equation becomes

$$q' \approx \rho = \frac{mBk_d}{2}, \tag{7.20}$$

where B is the average length of the message.

A better approximation for q comes after we expand Eq. 7.18. The expansion of the summation produces a sum of products of probabilities. We discard the products of three probabilities or more. Additionally, in two-dimensional networks with dimension-ordered routing a message continues along one direction and switches dimension after it has exhausted the path on each dimension. Therefore, $\rho_s \ll \rho_c$ and we can underestimate the effect of ρ_s in the summation.

Doing so, we approximate q_0 (Eq. 7.18) with

$$q_0 = (1-\rho) \cdot \sum_{\upsilon=2}^{4} p(\upsilon) \cdot \frac{\upsilon - 1}{\upsilon} \approx (1-\rho) \cdot \rho^2 / 2nk_d. \tag{7.21}$$

In this case, we can approximate the probability of contention with q'' defined as

$$q'' = \rho + q_0'', \tag{7.22}$$

where

$$q_0'' = (1-\rho) \cdot \rho^2 / 2nk_d. \tag{7.23}$$

Figure 7.9: **Approximations of the Probability of Contention when the Queue is Empty for Uniform Distribution.**

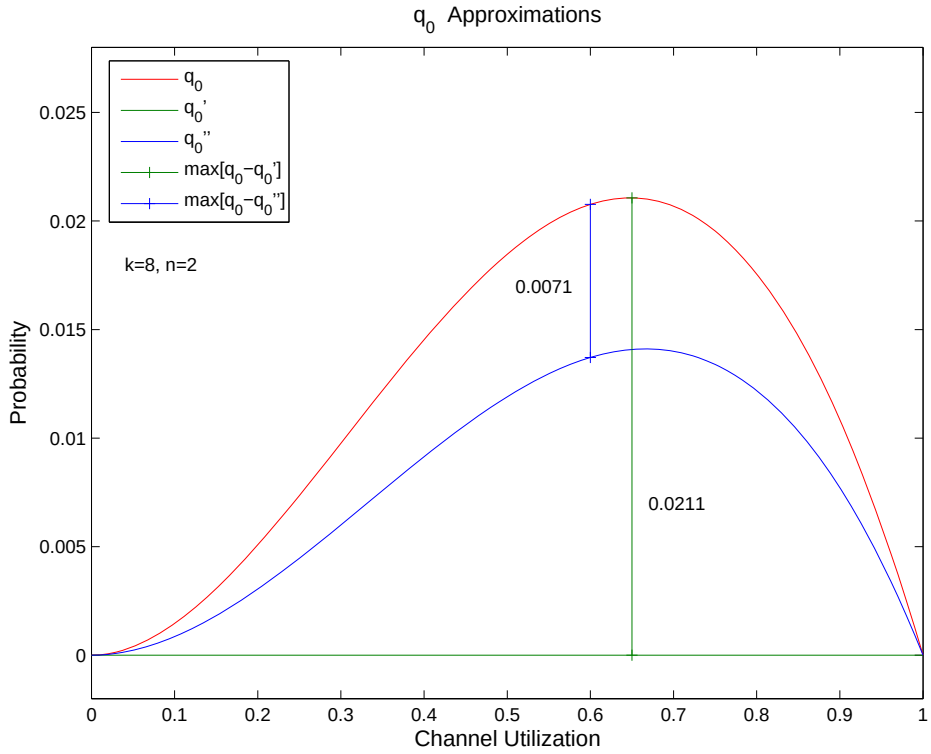


Fig. 7.9 plots the probability of contention for a message entering the switch when the queue is empty q_0 and the two approximations (q_0' and q_0") that we proposed. The red curve shows the exact value of the probability. The green one shows our first approximation ($q_0' = 0$) with maximum deviation of 0.0211 from the exact value. The blue curve shows our second approximation ($q_0'' = (1-\rho)\cdot\rho^2/2nk_d$). In this case, the maximum deviation from the correct value is only 0.0071.

Therefore, the following equation

$$q \approx \rho + (1-\rho)\cdot\rho^2/2nk_d \qquad 7.24$$

is a very good approximation of the exact value of the probability of contention for a message entering the switch in a two-dimensional point-to-point network, when processors communicate

with each other with equal likelihood.

In the following sections we derive the probability of contention in the network in one-dimensional point-to-point networks and bus-based systems.

7.3.1 Contention Probability In One-Dimensional Networks

After the analysis that we performed for the probability of contention in a two-dimensional point-to-point network, the analysis for the probability of contention in one-dimensional network and a bus is very straightforward.

In a one-dimensional network there is no ρ_s component in the channel utilization ρ, since messages cannot arrive at the switch from different dimensions. Therefore, the message probability ρ in a channel has two components:

- ρ_i: messages injected (or consumed) into this direction from the node processor.
- ρ_c: messages continuing along their path through the switch.

Since there are bi-directional channels in the network $\rho_i = m/2 = \rho/k_d$, where m is the processor message injection rate and k_d is the average distance a message travels in the network. Additionally, $\rho = \rho_i + \rho_c$ and $\rho_c = \rho - \rho_i = \rho(1 - 1/k_d)$.

The probability distribution $p(\upsilon)$ of the random variable υ of the number of messages requesting a specific output port in the case of a one-dimensional network is given by

$$p(\upsilon) = \begin{cases} (1-\rho_c)(1-\rho_i) & (\upsilon = 0) \\ \rho_i(1-\rho_c) + \rho_c(1-\rho_i) & (\upsilon = 1) \\ \rho_i\rho_c & (\upsilon = 2) \\ 0 & (\upsilon > 2) \end{cases} . \qquad 7.25$$

In this case, Eq. 7.13 that describes the probability of contention in the switch becomes

$$q_{1D} = \rho + (1-\rho) \cdot \left(p(2) \cdot \frac{2-1}{2} \right) = \rho + (1-\rho)\rho_i\rho_c/2 . \qquad 7.26$$

Substituting, $\rho_i = \rho / k_d$ and $\rho_c = \rho(1 - 1/k_d)$, Eq. 7.26 becomes

$$q_{1D} = \rho + (1-\rho)\rho^2(k_d - 1)/(2k_d^2). \qquad 7.27$$

This is the probability of contention in a one-dimensional point-to-point network, where ρ is the channel utilization and k_d is the average distance for a message in the network.

7.3.2 Contention Probability In Bus-Based Networks

To calculate the probability of contention for a message sent by a processor in a bus based system, we can treat the bus as a queuing server with υ messages joining the queue in a cycle. These υ number of messages are generated by the processors that want to transmit their data to other processors in the system. As was the case in the point-to-point networks, υ is a random variable and in the case of the bus it can take values ranging from 0 through N, where N is the number of processors in the system.

We can calculate the probability distribution of υ as follows. Let the probability that a processor requests the bus on any given cycle be m (m is the processor injection rate for a processor in a bus-based system). If we assume that the requests from the processors in the system are independent, then the distribution of υ is given by

$$p(\upsilon) = \begin{cases} \binom{N}{\upsilon} m^{\upsilon}(1-m)^{N-\upsilon} & 0 > \upsilon \geq N \\ 0 & \upsilon > N \end{cases}. \qquad 7.28$$

Eq. 7.11 that describes the probability for contention in the switch, holds for the bus-based systems, and we are modifying it here to reflect the probability of contention for the bus

$$q_{BUS} = \begin{bmatrix} \text{probability queue of} \\ \text{the bus is not empty} \end{bmatrix} + \begin{bmatrix} \text{probability queue of} \\ \text{the bus is empty} \end{bmatrix} \times \begin{bmatrix} \text{probability the message} \\ \text{is not routed out} \end{bmatrix}, \qquad 7.29$$

or

$$q_{BUS} = p_{Q \neq 0} + p_{Q=0} \cdot \sum_{\upsilon = 2}^{\infty} p(\upsilon) \cdot \frac{\upsilon - 1}{\upsilon} = \rho + (1 - \rho) \sum_{\upsilon = 2}^{N} p(\upsilon) \cdot \frac{\upsilon - 1}{\upsilon}. \quad 7.30$$

Substituting the value of $p(\upsilon)$ from Eq. 7.28 we get

$$q_{BUS} = \rho + (1 - \rho) \sum_{\upsilon = 2}^{N} \left[\binom{N}{\upsilon} m^{\upsilon} (1 - m)^{N - \upsilon} \cdot \frac{\upsilon - 1}{\upsilon} \right]. \quad 7.31$$

This equation reveals that the probability of contention in the bus is also dominated by the channel utilization as was the case for the point-to-point networks. However, in a bus-based system $\rho = N \cdot m$, while in a two-dimensional point-to-point network $\rho = mk_d/2$. Since $N > k_d/2$, the previous observation reveals that for the same injection rate m, the channel utilization, and therefore the contention, in buses is much larger compared to the channel utilization in point-to-point networks.

So far our analysis provided closed-form equations for the probability of contention in two-dimensional networks, one-dimensional networks, and bus-based systems (Eq. 7.16, Eq. 7.16, and Eq. 7.16, respectively). Next we compare these probability values with respect to the message injection rate of the processors in the system.

Figure 7.10: **Probability of Contention for a Two- and One-Dimensional Network, and a Bus-Based System.**

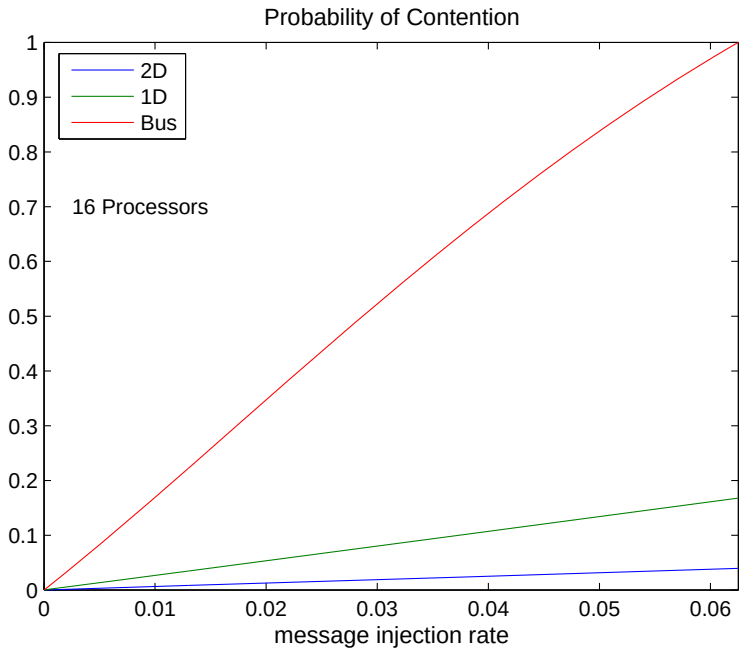


Fig. 7.10 plots the probability of contention for the three systems with 16 processors in each, assuming the same message injection rate. For the same injection rate, the probability of contention in a bus-based system is 25 times greater compared to probability of contention in the two-dimensional network, and almost 6 times greater compared to probability of contention in the one-dimensional network. Comparing the two point-to-point networks, the probability of contention in the one-dimensional network is 4.2 times greater compared to the probability of contention in the two-dimensional network.

7.4 Energy Consumption Assuming Uniform Distribution

We have calculated the energy costs for a message transfer on the point-to-point network (Appendix A). Thus, we can calculate the total expected energy cost after all processors in the system have transmitted their data assuming a uniform distribution.

The total expected energy is given by

$$E_{P2P} = \sum_{i=1}^{N}\sum_{j=1}^{N} E_{i,j} = \sum_{i=1}^{N}\sum_{j=1}^{N} M \cdot p_{i,j} \cdot (E_C + E_S) \cdot H_{i,j}, \quad 7.32$$

where N is the number of processors in the network, M is the total number of messages that each processors sends, $p_{i,j}$ is the probability that processor P_i communicates with processor P_j, E_C and E_S are the energy expended in the channel and the switch, respectively, and $H_{i,j}$ described the distance is hop counts between processors P_i and P_j.

Using Eq. 7.3 we can rewrite the previous equation as

$$E_{P2P} = \sum_{i=1}^{N}\sum_{j=1}^{N} M \cdot p_{i,j} \cdot (E_C + E_T + E_Q \cdot q) \cdot H_{i,j}, \quad 7.33$$

where E_T is the energy expended in the crossbar and the control logic of the switch and E_Q is the energy for a write and read in the input queue buffer of the switch.

The total expected energy expended writing and reading messages in the queue buffer because of contention in a point-to-point network is

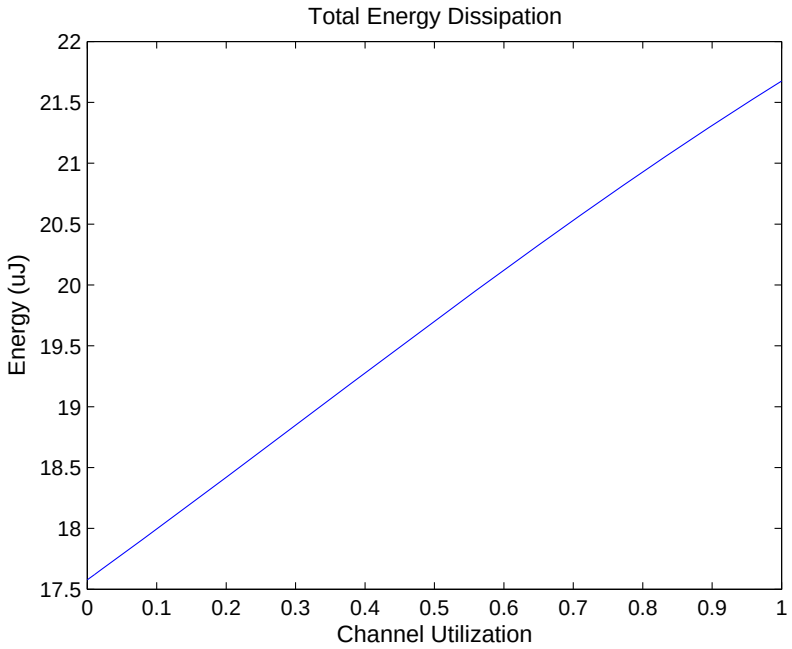
$$E_{P2P\_\text{Contention}} = \sum_{i=1}^{N}\sum_{j=1}^{N} M \cdot p_{i,j} \cdot (E_Q \cdot q) \cdot H_{i,j}. \quad 7.34$$

Next we express the previous equations in the case of a square point-to-point network assuming a uniform distribution of communication between processors.

When processors communicate with each other with equal likelihood the communication probability is $p_{i,j} = 1/(N-1)$. Additionally, we showed in Chapter 4 that in two-dimensional square point-to-point networks with bidirectional channels $\sum_{i=1}^{N}\sum_{j=1}^{N} H_{i,j} = N \cdot (N-1) \cdot 2\sqrt{N}/3$.

Therefore, Eq. 7.33 that describes the total expected energy becomes

Figure 7.11: **Total energy consumption in a 8-by-8 system for different values of channel utilization. Each processors transmits 1000 words and processors communicate with equal likelihood with each other.**



$$E_{2D} = M \cdot \frac{2N\sqrt{N}}{3} \cdot (E_C + E_T + E_Q \cdot q). \tag{7.35}$$

Fig. 7.11 shows the total expected energy consumption in the Raw dynamic network in a system with 64 processors assuming uniform communication patterns. Each processor transmits 1000 data words.

When there is no contention ($q = 0$) in the network the total expected energy becomes

$$E_{2D\_\text{No\_Contention}} = M \cdot \frac{2N\sqrt{N}}{3} \cdot (E_C + E_T). \tag{7.36}$$

In a square point-to-point network assuming uniform communication distribution the energy expended in the queue buffer due to contention (from Eq. 7.34) is

$$E_{2D\_\text{Contention}} = M \cdot \frac{2N\sqrt{N}}{3} \cdot E_Q \cdot q. \tag{7.37}$$

The ratio of the total expected energy when there is contention in the network (E_{2D}), over the energy when we assume there is no contention ($E_{2D\_\text{No\_Contention}}$) from Eq. 7.35 and Eq. 7.37 is

$$\frac{E_{2D}}{E_{2D\_\text{No\_Contention}}} = \frac{M \cdot \frac{2N\sqrt{N}}{3} \cdot (E_C + E_T + E_Q \cdot q)}{M \cdot \frac{2N\sqrt{N}}{3} \cdot (E_C + E_T)} = 1 + \frac{q \cdot E_Q}{E_C + E_T}. \quad 7.38$$

Therefore, the additional energy expended in the switch because of contention in the network, or the energy overhead is given by

$$E_{Overhead} = \frac{q \cdot E_Q}{E_C + E_T}. \quad 7.39$$

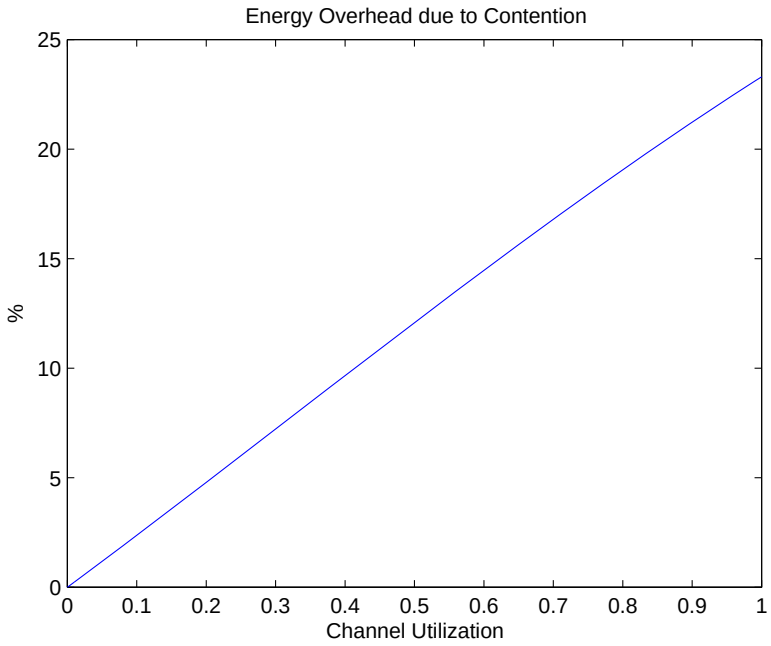
Using the energy costs of E_C, E_T, and E_Q that we calculated for Raw, the previous equation becomes

$$E_{Overhead} = \frac{q \cdot 12}{34.5 + 17} = 0.233 \cdot q. \quad 7.40$$

This is the energy overhead due to contention in the Raw dynamic network and it is a function of the probability of contention q in a switch.

Fig. 7.12 shows the energy overhead due to contention in a network switch for different values of q. The maximum amount of energy overhead that we pay is 23.3% when the channel utilization is $\rho = 1$, and is the case where every message in the network is queued in the input buffer of each switch along its path from the source to the destination processor.

Figure 7.12: **Energy Overhead due to Contention for the Uniform Distribution.**

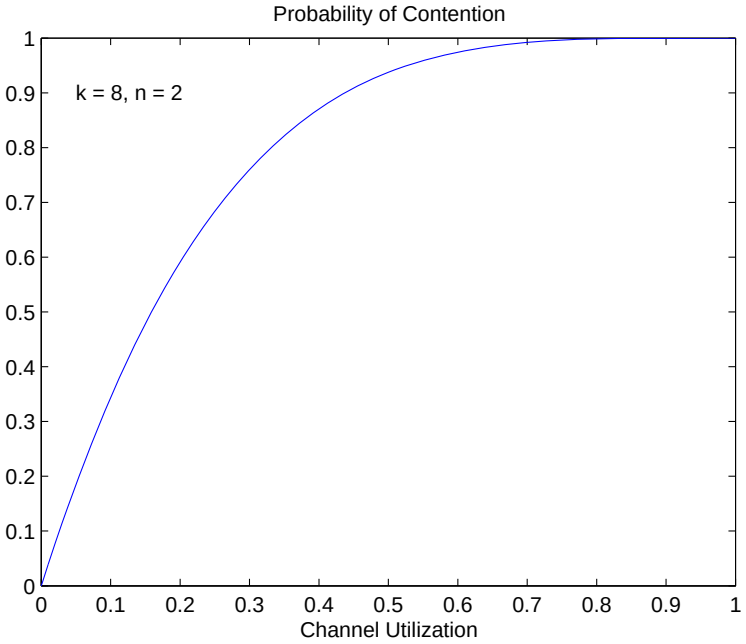


7.5 Energy Consumption Using Network Traces

In the previous section we performed an analysis on the effect of contention on the total energy dissipation on the interconnection network for uniform communication patterns. This section estimates the energy dissipation on the Raw network for various applications running on a 16-tile configuration. The applications communicate scalar operands over the Raw static network. The switch in these networks blocks all messages for all output ports if there is contention in one output port. This is the source of contention in the static network, since the static scheduling eliminates any contention for a specific port for a message in the network. Therefore, in the static network the probability that a message contends for a specific output port depends on the state of the other ports.

We calculate the probability of contention in the Raw static switch for a message requesting a port as follows. We have shown (Eq. 7.19) that the probability there is contention at a single port

Figure 7.13: **Probability of Contention for Uniform Distribution in the Raw Static Switch.**



is $q = \rho$, where ρ is the channel utilization. Thus, the probability there is no contention at a single port is $1 - \rho$; the probability there is no contention at the other four ports is $(1-\rho)^4$.

Therefore, the probability of contention q for a message entering a static switch in Raw is

$$q = 1 - (1-\rho)^4. \qquad 7.41$$

Fig. 7.13 shows the probability of contention at the Raw static switch at any given cycle, assuming a uniform communication pattern for different values of the channel utilization ρ and a two-dimensional mesh with $k = 8$ for both networks that we examined.

Using the network traces we calculate a lower and upper bound for the energy consumption. The traces that were collected do not provide information about the number of times a message was queued into the switches on its path from the origin tile to the destination tile. Each message from processor P_i to P_j is tagged with two time stamps, namely the cycle that the message was inserted into the network and the cycle in which it was consumed. The difference of those time stamps reveals the total delay of the message on the network. If this number is not equal to the

distance between P_i and P_j then the message was queued at least once in one of the switches on its path from P_i to P_j.

In addition, the difference of the time stamps provides information about the maximum number of times a message was queued in the input buffers of the switches. While the minimum number of queuing for the messages that were contented is $d_{min} = 1$, the maximum number can range from $d_{max} = 1$ to $d_{max} = H_{i,j}$, where $H_{i,j}$ is the distance in hop-count between P_i and P_j. Based on this information we can calculate the lower and upper energy bounds for the different benchmarks.

With the information in the network traces, we calculate the rate at which the processors inject messages into the network for each of the benchmarks, m_t, as

$$m_t = \frac{\text{Total Messages Sent}}{\text{Duration of Communication} \times \text{Number of Processors}}. \quad 7.42$$

All fourteen applications that we examine report information for network communication for scalar operands; therefore they are one word messages. If there are non unit-length messages travelling in the network, Eq. 7.42 is modified to

$$m_t = \frac{\text{Total Messages Sent}}{\text{Duration of Communication} \times \text{Number of Processors} \times \text{Average Message Length}}. \quad 7.43$$

Table 1 shows the total number of messages sent, the number of those that suffered delay, the lower and upper energy bounds, the lower and upper bound overhead paid due to contention, and the estimated channel utilization for each benchmark that was calculated from Eq. 7.42 and Eq. 7.14.

Table 1: Lower & Upper Energy Bounds

Bench mark	Total Msgs	Delayed Msgs	Lower Energy Bound	Lower Energy Over-head	Upper Energy Bound	Upper Energy Over-head	Channel Utilization
1.adpcm	100247	26813	8.84uJ	3.8%	9.04uJ	6.2%	0.145
2.aes	239987	86379	29.95uJ	3.6%	31uJ	7.3%	0.094
3.aes_fix	228506	99573	30.1uJ	4.1%	31.3uJ	8.4%	0.093
4. btrix	80389	7439	11.6uJ	0.8%	11.7uJ	2.0%	0.057
5.cholesky	1725120	687946	250.4uJ	3.4%	256.8uJ	6.0%	0.119
6.fpppp	315181	230922	44.4uJ	6.6%	48.3uJ	15.9%	0.092
7.jacobi	36719	18414	4.8uJ	4.7%	5.0uJ	7.6%	0.146
8.jacobi_big	3317070	2413292	401.1uJ	7.8%	420.4uJ	12.9%	0.142
9.life	707916	449420	84.2uJ	6.8%	86.7uJ	9.9%	0.094
10.mxm	667084	443420	107.7uJ	5.2%	118.3uJ	15.6%	0.080
11.sha	716323	336915	83.6uJ	5.1%	86.2uJ	8.4%	0.102
12.swim	5579025	2607419	681uJ	4.8%	720.5uJ	10.8%	0.139
13.tomcatv	1092888	328462	153.5uJ	2.6%	158.6uJ	6.0%	0.122
14.vpenta	105322	13126	15.4uJ	1.0%	15.7uJ	3.3%	0.078

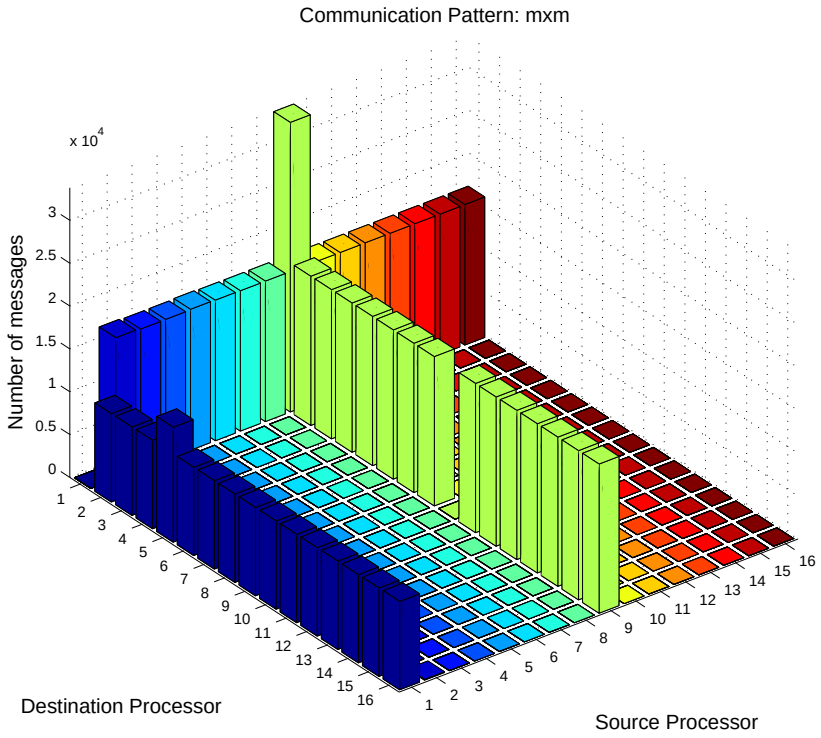
Table 2 shows the mean and standard deviation values for the lower and upper energy overheads.

Table 2: Mean and Standard Deviation Values for the Energy Overhead

	Lower Bound	Upper Bound
Mean	4.87%	9.3%
Standard Deviation	2.9%	4.7%

Section 7.4 presented an analysis for the energy overhead due to contention when processors communicate with equal likelihood. In real applications, however, this is a rare

Figure 7.14: **Communication Pattern: *mxm*. The graphs shows the total number of messages generated by the processors in the system and their destinations.**

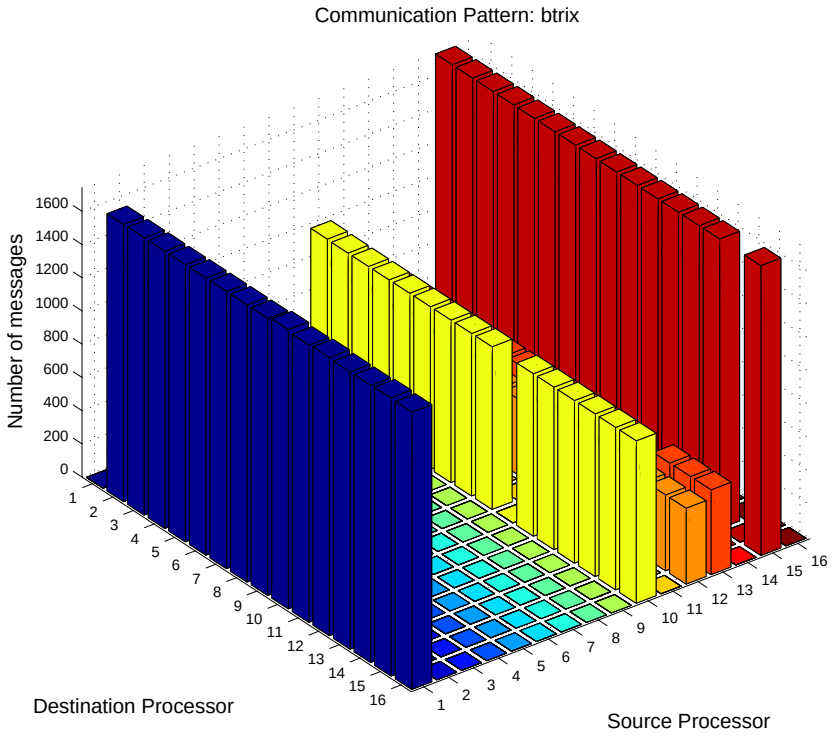


phenomenon, since many times few processors in the system are responsible for a large percentage of the total communication.

Fig. 7.14 shows the communication pattern for *mxm* running on a 16-tile Raw configuration. In this case, processors P_1 and P_9 are responsible for transmitting 65% of the total messages injected into the network. Additionally, the messages generated by all other processors have a unique destination processor P_1.

.

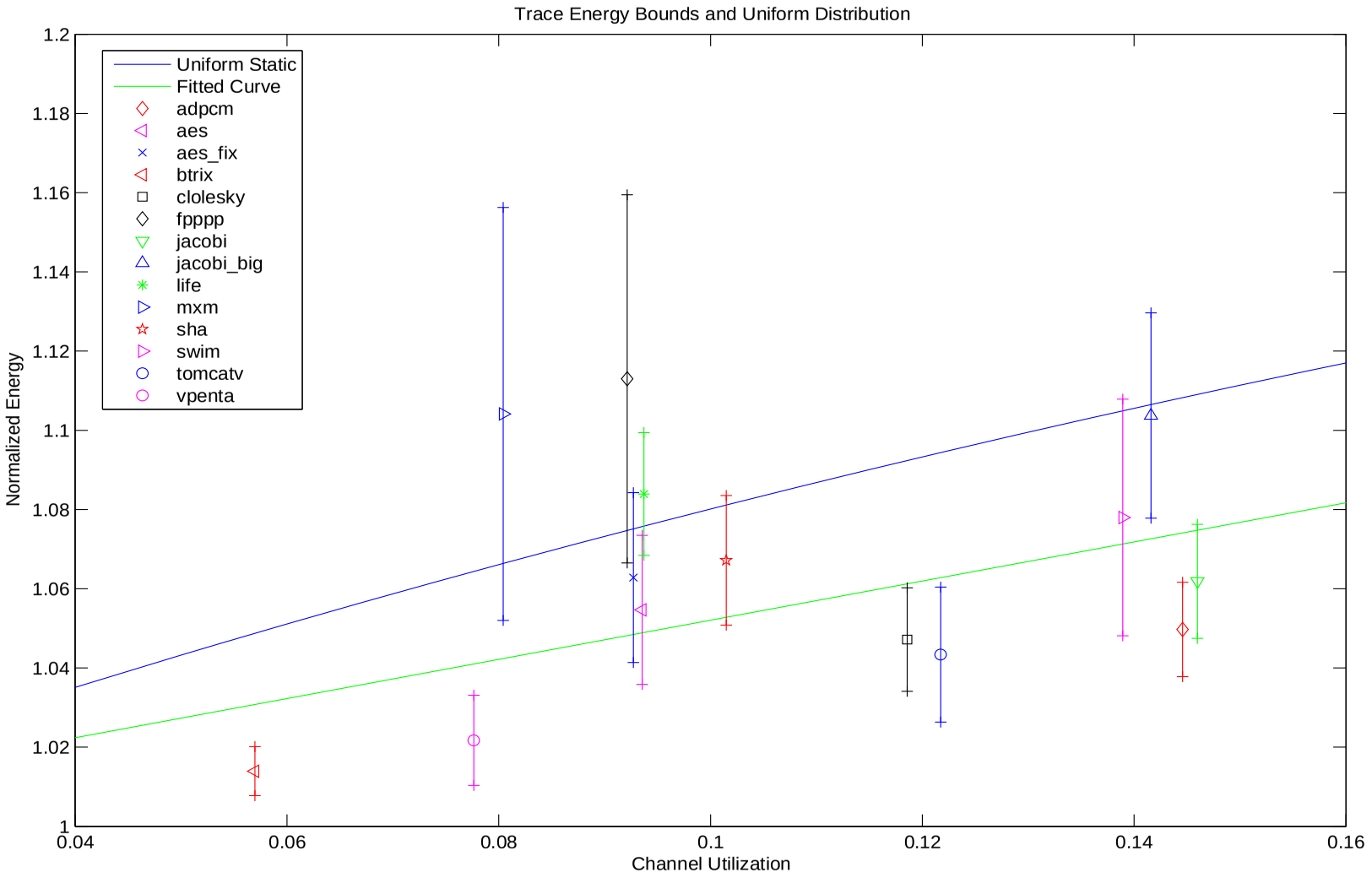
Figure 7.15: **Communication Pattern: *btrix*. The graphs shows the total number of messages generated by the processors in the system and their destinations.**



On the other hand, *btrix* exhibits a uniform communication pattern among the processors that inject messages into the network. Fig. 7.15 shows the source and destination processors for this application. In this case only processors P_1, P_{10}, P_{12}, P_{13}, and P_{15} send messages to other processors. It is clear from this graph that when these processors transmit a word, this word has equal likelihood to have as destination any other processor in the network. Appendix B presents graphs with the communication patterns from all the applications that we examined.

We want to compare the total energy for each benchmark with the total energy predicted by the analytical model for the uniform distribution assuming similar channel utilization values and see the additional energy added by a non-uniform traffic pattern into the network. Fig. 7.16 plots the normalized energy bounds ($E_{Contention}/E_{No\_Contention}$) for each benchmark and the expected normalized energy for the uniform distribution.

Figure 7.16: **Upper and Lower Energy Overhead Bounds for Benchmarks Compared to the Energy Overhead Assuming Uniform Distribution.**



The blue solid line shows the energy overhead predicted by the analytical model for the static network assuming a uniform distribution for the communication among processors. The graph also shows the lower and upper bounds of the overhead expected for every application that we examined based on the trace information. The marks for each application correspond to the average value of the overhead calculated by the upper and lower bounds.

The green solid line is a first order polynomial fit based on the averages of the overhead values. Two applications, *mxm* and *fpppp*, were not included in the polynomial fit, because they exhibit the largest difference between the upper and lower bound overhead values. A first order polynomial fit was chosen, because the theoretical curve looks linear for small values of ρ. When we examined the communication pattern of *mxm* (Fig. 7.14), we noticed that the communication was heavily targeted on two processors. This is the main reason the upper bound and average value of contention is higher compared to other applications.

In the Raw static network most of the contention is suffered within the processor; the static scheduler holds data in the tile before it ensures that the path to destination is as clear as possible. This is the main reason the fitted curve falls below the values of the overhead predicted by the analytical model.

Summary

Network contention of network resources results in message delays and increased energy dissipation in the switch, when messages are written into queueing buffers waiting to be serviced by a specific output port. We examined the effect of contention on the energy dissipated in interconnection networks and derived a closed-form solution for the energy for various channel utilization values, assuming processors communicate to each other with equal likelihood. We performed this analysis for the case of two- and one-dimensional networks, and bus-based interconnects. We quantified the amount of contention for these three cases for similar message injection rates and showed the advantages of point-to-point networks.

We used energy estimates for the energy dissipated in the interconnection networks in the Raw microprocessor to quantify the energy overhead and showed that the maximum amount of additional overhead paid is 23.3%. Additionally, using network traces we estimated the lower and upper bounds for the energy dissipation in the communication network for the benchmarks.

CHAPTER 8

FUTURE WORK

There are numerous opportunities to extend this work. This chapter describes directions that could provide additional insight to the energy characteristics and scalability of point-to-point interconnection networks.

First, we present an enhancement to the model that could evaluate the energy consumption of distributed hardware structures and compare it to the energy consumption of centralized structures (caches and register files, for example). Secondly, an interesting addition to the analysis presented might include the energy characteristics of network topologies in addition to the ones already examined and compare the energy characteristics of other topologies. An opportunity for extension of this work is the examination of additional network traces and their energy characteristics. Another extension of the thesis would be the investigation of the effect of

technology scaling on the energy dissipation of on-chip networks. Ultimately, this model can be expanded to model the energy consumption of tiled processor architectures.

The following sections explain in more detail the potential directions of this work.

8.1 Advantages of Distributed vs. Centralized Hardware Resources

The analysis of the energy dissipation of interconnection networks is useful when examining the energy consumption of systems with distributed structures. The model can be used to evaluate the energy advantages of tiled architectures with distributed structures, over centralized structures of similar characteristics (total size in terms of caches for example).

In modern microprocessor systems large structures, such as multi-ported register files, multi-ported cache memories with many sub-banks require long interconnection buses to achieve their functionality. The result is increased power dissipation for each access on these structures.

First, we describe a typical cache memory architecture and analyze the energy inefficiencies of typical banked multi-ported caches. Then we modify our model to estimate the total expected energy dissipation when data stored in caches communicate into the network in tiled processor designs. We present probability distributions that model various spatial locality characteristics for the data stored into the various caches in the system.

8.1.1 Cache Architecture

The dominant cache organization employed in modern microprocessors is the set-associative cache (the direct-mapped cache and fully associative cache organizations are two extremes of this organization) [37], [38]. In a normal m-way set associative cache, there are m tag and data array pairs.

One of the major sources of power dissipation in a conventionally organized cache can be attributed to transitions in the bit lines of the data and tag arrays. Bit line dissipations occur when the bit lines are precharged or discharged. To achieve savings on the bit line energy, the data

arrays can be subdivided into sub-banks, so that only those sub-banks that contain the desired data can be read [40]. A sub-bank consists of a number of consecutive bit columns of the data array. A data line is thus spread across a number of sub-banks. Since data are read out from one sub-bank at a time, a common set of sense amps can be shared across the sub-banks, cutting down on the cache area. In effect, columns are multiplexed within a sub-bank. Column multiplexing in this manner is routinely used within static RAMs. The size of a sub-bank refers to the physical wordwidth of each sub-bank.

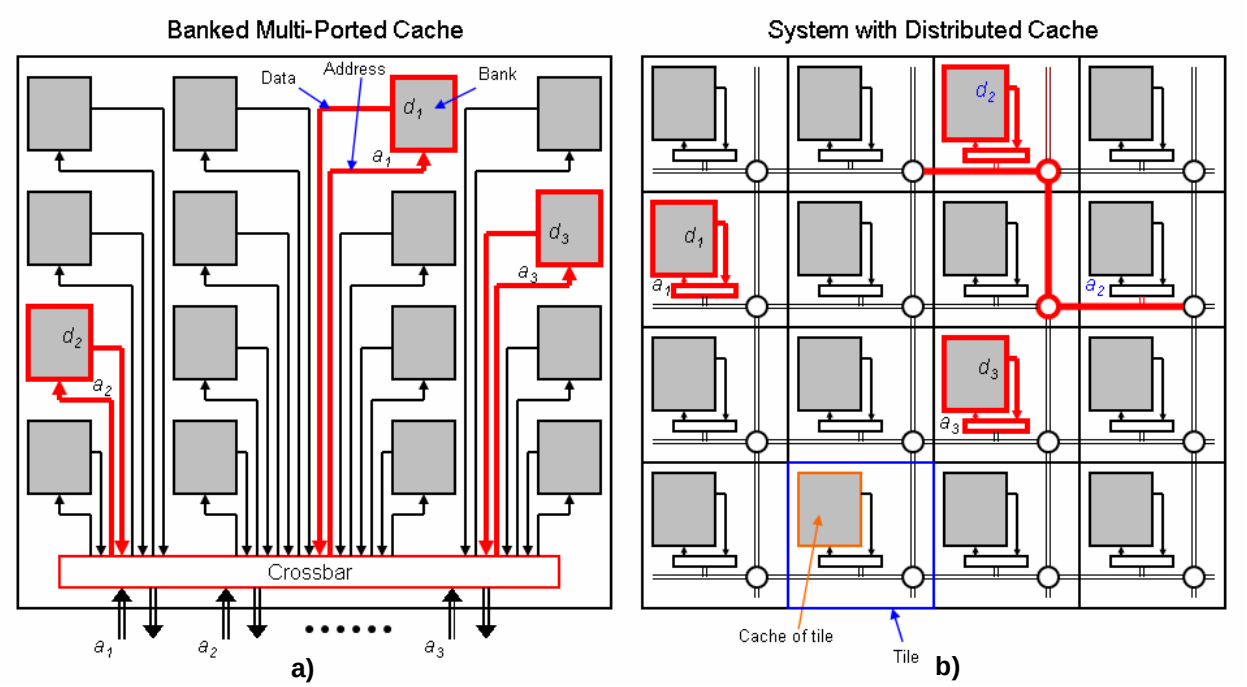
Fig. 8.1.a shows a simple abstract cache structure[1]. In the case where the data are located at a sub-bank placed far from the crossbar and the cache ports (address *a1*, data *d1*) long interconnection wires are required for the address bus, as well as the bus for the data returned from the sub-bank. In addition, the complexity of the crossbar increases; consequently the energy dissipated on the crossbar routing and control logic, also increases with the number of sub-banks.

On the other hand, in tiled architectures the cache memory is distributed across the chip area, resulting in small cache structures saving total access power. Accessing a small cache with few banks decreases the energy dissipated on the address and data busses that connect the cache ports to a specific bank. Additional savings are produced, when the compiler exploits data locality resulting in reduced interconnection power costs. Fig. 8.1.b shows that in two cases the data requested by the processor are located within the local tile cache. In the other case, data *d2* traverses two hops.

We should point out that the two systems shown in Fig. 8.1 are not drawn to scale.

1.There are techniques that propose hierarchical cache arrangement and sub-cache structures within the cache [47]. Those techniques however increase the design complexity of the cache and do not solve the problem of long wires when accessing sub-banks at the far edges of the cache.

Figure 8.1: **a) Systems with centralized cache structures result in long interconnection buses within the cache for accessing the contents of a sub-bank. b) In tiled architectures large structures are distributed across the chip area, saving power when the compiler exploits data locality, compared to centralized structures.**



8.1.2 Energy Savings for Distributed Caches

In tiled architectures, the compiler is responsible for scheduling instructions for the different applications to increase the locality of communication. We could examine three different spatial locality patterns:

- *No locality* - uniform distribution of data (data requested from a tile can reside in the same or any other tile with equal probability).
- *Low locality* - linear decay distribution of data (data requested from a tile have a high probability of residing in the same or a neighboring tile. When the data are not in the

requesting tile cache, the probability they are stored in another tile drops linearly with the distance of the two tiles).

The probability mass function of Eq. 8.1 models low spatial locality:

$$p_{i,j} = \frac{|b - (a \cdot |i - j|)|}{\sum_{j=1}^{N} |b - [a \cdot |i - j|]|}. \quad 8.1$$

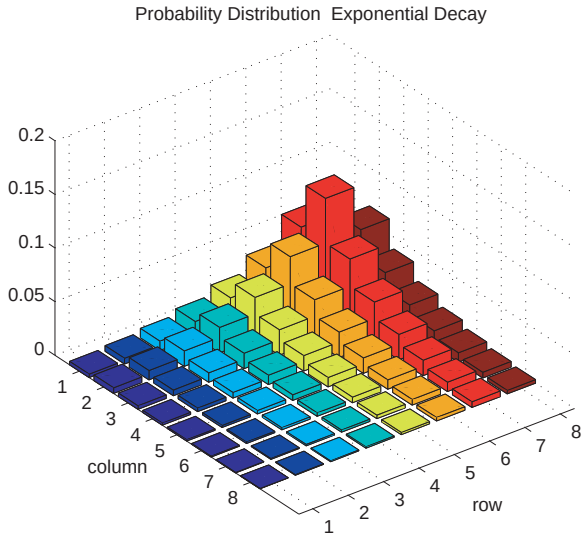
• *High locality* - exponential decay distribution of data (data requested from a tile have a high probability of residing in the same or a neighboring tile. When the data are not in the requesting tile cache, the probability they are stored in another tile drops exponentially with the distance of the two tiles).

An exponential decay distribution is shown in Fig. 8.2 for a system with sixty four processors laid in an array of eight rows and eight columns. The graph shows the probability the processor on the second column and seventh row requests data stored in the distributed caches.

The probability mass function of Eq. 8.2 models high spatial locality. It is different from the exponential decay distribution that we examined when we modeled high localized traffic patterns (Chapter 3 and Chapter 4) because it accounts for cache accesses within the tile ($p_{i,i} \neq 0$):

$$p_{i,j} = \frac{\frac{1}{d} \cdot b^{-\frac{|i-j|}{d}}}{\sum_{j=1}^{N} \frac{1}{d} \cdot b^{-\frac{i-j}{d}}}. \quad 8.2$$

Figure 8.2: **Exponential Decay Probability Distribution.**



The model could compare the total energy dissipated on the centralized cache, with the total expected energy of the distributed system, for the same number of cache accesses, taking into account the energy dissipated on the interconnection network. In the distributed system, the cache access requests from the different tiles of the system follow the distributions described above (uniform, linear decay and exponential decay) and describe different spatial locality patterns.

An interesting challenge, however, is the accurate energy modeling of cache accesses in current technologies. Available cache models[1] tend to over-estimate the energy of cache accesses minimizing the effect of the interconnection network energy. Modeling the energy consumption for caches would be beyond the scope of this thesis.

1.eCACTI [46] and CACTI [4], [5] are analytical models that provide timing, power, and area estimates for caches of various size, associativity, and I/O ports.

Quantifying the energy advantages of distributed resources over centralized ones could involve comparing the total energy of accessing a large centralized cache to the total energy expended for accessing and communicating cache data in tiled processor systems assuming different tile configurations and various locality characteristics.

A simple variation on the model could be used for the energy estimation within Non-Uniform Cache Access [62] (NUCA) structures that could additionally account for hierarchical placement within the NUCA structure.

8.2 Network Topologies

The work in the thesis focuses on the analysis for the energy dissipation of mesh point-to-point networks with bi-directional channels and no end-around connections. Evaluating network topologies such as ring and tori networks could direct future research concerning the energy advantages of point-to-point networks and communication locality.

We should point out that evaluating those topologies just involves modifying the matrices $H_{i,j}$ and $D_{i,j}$ of Eq. 6.2 on page 101 to account for the specific point-to-point logical and physical distances. However, a complete approach for the evaluation of those topologies would involve acquiring network traces and applying the framework in a similar fashion that was described in Chapter 5.

8.3 Traces

As far as we know, this work is the first to extensively use traces from benchmarks in the analysis of the energy consumption in on-chip interconnection networks. The results from this analysis could be augmented by applying our framework to additional network traces than the ones that we examined.

Moreover, we would like to see energy results using traces from applications running on larger tile configurations. Running applications on 32, 64 or greater number of tiles would

definitely provide a more spherical view on the energy scalability of on-chip interconnection networks.

For the analysis of the energy overhead due to contention, we used the upper and lower bounds based on the minimum and maximum delays that a message would suffer through the path from source to destination. We would like to narrow this range by using exact information for the number of times that a message was stored in the input queue buffer in a switch and get a better estimate for the energy overhead because of contention in the network.

Additionally, it would be interesting to acquire network trace information from benchmarks running on other tiled processor architectures. Comparing the energy results for the same benchmarks, running on the same number of tiles would provide extremely interesting insight in how the compiler of each architecture schedules data and handles communication locality and partitions and places data across the different tiles of the system.

8.4 Technology Scaling Effect

The analytical framework that was developed is relatively invariant to process scaling. It models the total energy of communication in interconnection networks, assuming various costs for: (a) the energy expended when data move between neighboring tiles; (b) the energy dissipated for the control logic in the network switches; and (c) the energy for writing and reading from the network input buffer.

The model provides estimates for the energy dissipation in the network independently of those costs. These estimates, however, are going to be sensitive to technology scaling, since the relative values of those costs may vary significantly.

Projections for the scaling of wires appear pessimistic ([14], [32]) and as the costs of communicating data scale up, our framework could be of great importance to architects who could examine trade-offs on the placement and partitioning of data across the system resources.

If the scaling factor comparing two process technologies is α, a first order analysis for the energy consumption reveals that the energy in the interconnect and logic will scale with a range that falls between α and α^2.

In interconnect, the length and width of wires scale by α, so the area scales by α^2. However, the oxide thickness “tox” scales as well. Assuming it scales with α then the total interconnection capacitance of the metal plate to the ground scales with α.

The fringing capacitance, which is the capacitance from the side-walls of the metals to the ground, scales with a range that falls between α and α^2. The length of the wire scales by α; but the height of the wires does not scale accordingly for new process technologies to keep the resistance of metal wires low. For the same reason, the cross-coupling capacitance scales down within that range. Scaling the spacing of the wires increases the capacitance between them, however, in actual implementations the spacing between neighboring wires is set to a distance that does not affect significantly the total capacitance or the integrity of the signal.

In logic, the gate area scales with α^2, because both the gate width and length scale with α. However, the gate oxide thickness scales as well. Another source of parasitic capacitances in CMOS is the junction capacitances. The bottom-plate junction capacitance scales by α^2 and the side-wall junction capacitance scales by α. Therefore the overall capacitance, hence the energy scales with a factor between α and α^2. We suggest the use of α as a scaling factor for energy for both interconnect and logic.

With technology downscaling, the portion of the total energy in microprocessors that is dissipated in leakage has been significantly increasing. If leakage becomes a very significant component of the total energy, the switch architecture most probably would change to enable powering-down of the input queue buffers when they are empty. In this case our model needs to be modified to account for leakage energy.

One approach would be to estimate the total expected leakage energy dissipated by an input buffer for the whole execution time. The probability there is at least a message in the input queue buffer equals to the channel utilization ρ and describes the probability that an input buffer is powered on at any given cycle. The leakage energy cost per cycle for each input buffer depends on the size of the buffer, the operating frequency, and the process technology. The total expected leakage energy for each input buffer is the product of the channel utilization, the average leakage energy cost per cycle, and the execution time of the application.

We address only the leakage energy suffered in the input buffers of the switch, because this is where the leakage is maximized. The transistors in the control logic circuitry and the crossbar are usually active, therefore it is uncommon to power-down these structures. On the other hand, memory structures like the ones used in the input buffer are very easy to turn on and off.

8.5 Energy Performance of Tiled Processor Architectures

We view this work as the first solid step in making a high-level energy framework that could model the energy dissipation of tiled processor architectures. Developing a framework that models the energy of computation within the tiles is an excellent supplement of this work.

A complete framework for the energy of tiled processor systems would enable designers to make architectural decisions about both tile size and the number and size of various architectural structures (such as caches, ALUs, register files within the tile, available communication networks).

CHAPTER 9

RELATED WORK

Power and energy of on-chip networks have become a significant portion of the total power and energy budget [54], so studies that explore these issues are important for processor designers. Recently, there have been studies that address several important issues.

To our knowledge, Orion [36] was the first power/performance interconnection network simulator, capable of providing detailed power and performance reports. The simulator is triggered on every cycle by different activities and calculates the network power based on power models for each on the components of the network. Orion simulates various network topologies and workloads, providing power and performance network characteristics. Orion is constructed within the Liberty Simulation Environment (LSE) [64], a processor and memory simulator, and provides further modeling of network components. The basic network components are message sources and sinks, router buffers, crossbars, arbiters, and links. Orion classifies these modules

into two classes: the message transporting and the message processing class. Links and crossbars fall within the first class, while message sources and sinks, buffers, and arbiters fall within the second.

Orion uses Cacti [5] to compute low level capacitance values, such as gate and diffusion capacitance values of transistors, and capacitance value for metal wires. The simulator uses capacitance equations and switching activity estimation to derive energy consumption values per component. This methodology estimates energy for blocks like FIFO buffers, crossbars, and arbiters.

Our work differs in that our results, derived both from an analytical model and traces of real benchmarks (that are gathered from a single execution run), provide energy estimates for various types of communication traffic and can be used early in the design cycle, with no need for time-consuming architectural simulation. Further, models provide a different level of insight compared to a simulator, particularly as they relate to asymptotics and in many cases can provide an intuitive basis from which energy characteristics may be extrapolated to any benchmark without simulating it.

Eisley and Peh [39] addressed the need for high-level network power analysis by proposing a framework that uses link utilization as the unit of abstraction for network power. The analysis takes as input the message flows across nodes, the specific topology, and the routing protocol to derive a power profile of the network. Specifically, message flows along with their sources and destinations and injection rate, are transformed into link utilization functions, taking into account contention among different flows.

Instead, our framework uses hop count as the unit of abstraction to measure the energy efficiency of the network. Link utilization is a valid unit of abstraction, however (1) hop-counts and (2) probability distributions that model traffic patterns assist intuition on communication locality, especially since the applications that we studied have patterns that roughly match the modeled

distributions. Link utilization cannot infer communication locality since different locality patterns can produce the same link utilization.

Our model takes communication locality into account, communication locality is not addressed in the work of Eisley and Peh. The framework we present can be used to derive link utilizations from locality distributions or real traces. We further extract information about traffic patterns and locality from traces of benchmarks running on a tiled processor and use the results both to validate our model and to assess the impact of communication locality.

Furthermore, the analysis of Eisley and Peh does not take into account the effect of buffering at each individual node. This buffering occurs at the switch when an output port is not free and the message has to wait in the input queue buffer for the port to be available. In this thesis we present a detailed analysis of the effect of contention and the overhead of message queueing to the total energy dissipated in the interconnection network.

Wang et al. [41] present a method to evaluate the energy efficiency of different two-dimensional square network topologies and predict the most energy-efficient network topology based on the network size, technology predictions, and architectural parameters. The methodology uses the *average* flow control unit traversal energy as the network energy metric; this is the energy cost to transmit unit data in the network.

Our model is based on precise distributions. Therefore it can (1) capture arbitrary topologies, (2) incorporate actual network traces, and (3) use arbitrary analytical distributions. We believe our work is the first to address communication locality in analyzing multicore interconnect energy and to use real multicore interconnect traces extensively. Further, the results presented in Wang assume uniform distributions, while our work is not limited to one distribution to model traffic patterns. Rather we present a framework that can incorporate any traffic pattern, including those from real traces, and use many distributions to model various forms of communication locality. Interestingly, for some simple cases, we show that our distribution-based approach

asymptotically yields the same results as an approach using averages. One of our major results is that network energy is heavily related to communication locality, which was not captured in [41]. Our framework also models high-dimensional networks, providing insights on trade-offs between the physical and logical distance between the processors in the system, for the mapping that our algorithm performed. Our framework can be used for the energy estimation for high-dimensional networks for any mapping the compiler chooses, with subsequent changes of the model parameters.

Moreover, the real appeal of our approach lies in the ease of incorporating actual network traces in our framework. Additionally, our work provides energy comparison of several interconnection networks with buses. Buses are currently popular in commercial multicore chips ([57], [58]) and therefore merit special attention.

Kumar et al. [65] present a detailed analysis of the area, power, performance, and design issues for shared-bus fabrics and crossbar interconnection systems. They show that the design choices for the interconnect have significant effect on the rest of the chip. Additionally, designs that treat the interconnection independently would not arrive at the best multicore design.

They present their results examining 3 multicore configurations: 4, 8, and 16 cores. They show that in the shared-bus fabric interconnect the total power due to the interconnect can be significant. Specifically, the interconnect power for the 16-core processor is more than the combined power of two cores and is equal to the power of a single core for the 8-core system. Power increases superlinearly with the number of connected units, thus a shared-bus fabric is not a viable solution for multicores as the number of processors in the system scales up. Their analysis reports power consumption of $7.5W$, $9.5W$, and $22W$ for the system with 4, 8, and 16 cores, respectively.

In the second interconnection scheme examined in [65], a crossbar connects cores (with L1 caches) to a shared L2 cache. The analysis shows that the power overhead due to crossbars is

very significant; the overhead can be more than the power taken by three full cores for a completely shared cache and more than the power of one full core for 4-way sharing. Even for 2-way sharing, the power overhead of the crossbar is more than half the power consumption of a single core. A crossbar implemented in the 1X, 2X, or 4X metal layer dissipates about $31.5W$, $26.5W$, and $25W$, respectively.

Kim et al. [42] present measurements of energy consumption in the Raw microprocessor. This work was an experimental study of a real tiled processor, and as such has provided many of the driving parameters in our modeling and trace-driven study. In their study, they present measurements of energy consumption in the Raw microprocessor and show that on idle state (when the clock is grounded) the chip draws a leakage current of $28mA$ and dissipates $45mW$.

Additionally, they examined the average current of an application with average instruction mix running on a single Raw tile, which showed that the compute power consumes a very small fraction of the power compared to the clock power. Therefore, they concluded that implementing clock-gating at the tile-level is highly desirable for tiled architectures.

The work in [42] continued examining the energy costs of communication over the two types of network in the Raw multicore processor. The measured numbers for the static and dynamic networks are $85pJ$ and $90pJ$, respectively. Both numbers are measured for a maximum toggle-rate per word; consecutive words injected to network would cause the channel lines to alternate on every cycle, so these values correspond to the maximum energy dissipation per cycle.

CHAPTER 10

CONCLUSIONS

Advances in VLSI technology offer increasingly abundant transistor and silicon resources to microprocessor designers. As a result, the complexity of modern microprocessors has increased. Consequently serious power dissipation issues have risen which make power the most important problem that designers are facing.

Power and complexity limitations are moving processor designs toward tiled architectures. Tiled processor architectures attempt to mitigate these issues by organizing the processor resources in a distributed manner. They allow the performance of general purpose microprocessors to scale, as the silicon resources become increasingly available, without making power a limiting constraint.

This work evaluates the energy scalability of on-chip interconnection networks. We presented a detailed analytical model for the energy dissipation in different point-to-point network topologies

and compared the energy performance of point-to-point networks with bus-based topologies. We presented a set of probability distributions to model various traffic patterns that closely match the communication characteristics of various benchmarks. These distributions can be used to represent any level of locality among the processors in the system.

We showed that for the uniform distribution the total expected energy for the point-to-point network is one third of the total energy for the bus-based system. The savings for more localized traffic patterns are even more considerable. For example, in the case where processors communicate with a pattern modeled by an exponential decay probability distribution, the energy dissipation on the bus-based system can be 10 times larger compared to the energy consumption for a one-dimensional point-to-point network.

We expanded the model and our analysis to include any rectangular processor grid arrangement. We derived a closed-form equation for the energy dissipation of two-dimensional meshes and showed that the energy savings are $O(\sqrt{N})$ compared to the energy of a one-dimensional point-to-point system and a bus-based system. Our results show that point-to-point networks are more robust over various traffic patterns compared to one-dimensional networks.

In this work we extensively used real network traces from benchmarks running on a tiled microprocessor to compare the energy performance of OCNs, and to validate the analytical model. We examined the communication characteristics of each benchmark, presenting statistical data on the distance (in number of hops) for the messages travelling in the network. The savings for the benchmarks that we examined vary from 80% (*mxm*) to 89% (adpcm) on a 16-tile system. We validated our model by comparing the energy savings of the benchmarks for different tile configurations, with the expected energy savings predicted by the framework for three different probability distributions that exhibit similar average communication distance compared to the benchmarks. The maximum relative error was 8.5% and the minimum 0.5%

across the different configurations. The results revealed that the set of distributions that we developed closely model the communication characteristics of benchmarks.

High-order point-to-point interconnection networks exhibit interesting trade-offs when they are mapped to two-dimensions. We presented an algorithm that calculates the logical and physical distances when those network are mapped to two dimensions. The energy comparison of a two-, three-, and four dimensional network revealed that communication locality is essential for choosing the most energy efficient network. We show that when the switch energy is small relative to the channel energy, the most energy efficient network topology is the two-dimensional because of its small physical distance compared to the other networks for a uniform distribution. The four-dimensional network, on the other hand, is the most efficient when the switch energy increases. However, two-dimensional networks are more efficient in the presence of communication locality in the traffic patterns.

In the thesis we presented a detailed analysis of the energy costs of a switch. We showed that the estimated values for channel energy, switch control logic energy, and switch queue buffer energy are 34.5pJ, 17pJ, and 12pJ, respectively. As process technologies move from one to the next, these numbers are likely to scale with α, where α is the scaling factor comparing two process technologies.

Network contention of network resources results in message delays and increased energy dissipation in the switch, when messages are written into queueing buffers waiting to be serviced by a specific output port. We examined the effect of contention on the energy dissipated in interconnection networks and derived a closed-form solution for the energy for various channel utilization values assuming processors communicate to each other with equal likelihood. We used energy estimates for the energy dissipated in the interconnection networks in the Raw microprocessor to quantify the energy overhead and showed that the maximum amount of

additional overhead paid is 23.3%. Additionally, using network traces we estimated the lower and upper bounds for the energy dissipation in the communication network for the benchmarks.

NOMENCLATURE

- P_i: Corresponds to the i_{th} processor in the system. *p 45*
- P_j: Corresponds to the j_{th} processor in the system. *p 46*
- N: Total number of processors in the system. *p 45*
- M: Number of data words that each processor transmits. *p 45*
- E_{BUS}: Total energy expended in the bus-based system. *p 46*
- E_L: Energy cost of accessing the full length of the bus. *p 46*
- L: Total length of the bus. *p 45*
- C_L: Total capacitance of the bus. *p 46*
- E_{P2P}: Total expected energy dissipated in the point-to-point interconnection network. *p 47*
- E_l: Energy cost for one network hop in the point-to-point interconnection network. *p 48*
- E_C: Energy cost expended in the channel/wires that connect two neighboring processors. *p 52*
- E_S: Energy cost expended in the switch. *p 52*
- $E_{i,j}$: Expected energy consumption due to transmitting of all data words that have as source processor P_i and destination processor P_j in the point-to-point interconnection network. *p 47*
- l: Length of the segment that connects two neighboring processors in the point-to-point network. *p 46*
- C_l: Capacitance of a channel/wires (for all bits) of length l. *p 46*
- $p_{i,j}$: Communication probability. Probability that processor P_i communicates with processor P_j. *p 48*

- $H_{i,j}$: (Logical) Distance in number of hops that connects processor P_i to processor P_j. *p 49*
- M_i: Total number of words injected in the network by processor P_i in point-to-point networks with unequal loads. *p 49*
- E_{1D}: Total expected energy in a one-dimensional point-to-point network. *p 50*
- X: Number of processors in a row in a two-dimensional point-to-point network. *p 70*
- Y: Number of processors in a column in a two-dimensional point-to-point network. *p 70*
- E_{2D}: Total expected energy in a two-dimensional point-to-point network. *p 73*
- $M_{i,j}$: Total number of data words sent from processor P_i to P_J. *p 87*
- n: Dimensionality of a network. *p 94, 121*
- k: Number of processor in a dimension in a point-to-point network. *p 94, 102, 121*
- k_d: Average distance (in number of hops) per message in one dimension assuming uniform communication patterns. *p 121*
- $D_{i,j}$: Physical distance in number of “physical hops” that connects processor P_i with P_j. *p 101*
- E_Q: Energy expended writing and reading a message in the input buffer queue in a switch. *p 112*
- E_T: Energy expended on the crossbar and the switch control logic circuitry in a switch. *p 112*
- q: Probability of contention when a message enters a switch. *p 116*
- $p_{Q=0}$: Probability that the queue of a specific output port is empty. *p 120*
- $p_{Q\neq 0}$: Probability that the queue of a specific output port has *at least* one message. *p 120*
- υ: Random variable that describes the number of messages that request a single output port. *p 120*
- m: Message injection rate into the network by a processor. *p 121*

- ρ: Channel utilization. *p 121*
- ρ_i: Component of the channel utilization that corresponds to the messages generated/ consumed by the node processor. *p 122*
- ρ_c: Component of the channel utilization that correspond to the messages that continue along the same direction through the switch. *p 122*
- ρ_s: Component of the channel utilization that correspond to the messages that change to the specific direction in the switch. *p 122*
- q_0: Probability of contention for a message entering the switch when the queue is empty. *p 124*
- B: Average length of a message. *p 126*
- q_{1D}: Probability of contention for a message entering the switch in a one-dimensional network. *p 128*
- q_{BUS}: Probability of contention for a message generated by a processor in a bus-based system. *p 129*
- $E_{P2P\_\text{Contention}}$: Total expected energy expended writing and reading messages in the queue buffer because of contention in a point-to-point network. *p 132*
- $E_{2D\_\text{No\_Contention}}$: Total expected energy in a two-dimensional network when there is no contention in the network. *p 133*
- $E_{2D\_\text{Contention}}$: Total expected energy expended in the queue buffer due to contention in a two-dimensional point-to-point network. *p 133*
- $E_{Overhead}$: Energy overhead due to contention. Ratio of $E_{\text{Contention}} / E_{\text{No\_Contention}}$. *p 134*
- m_t: Message injection rate calculated by the network trace information for applications running on the Raw microprocessor. *p 137*

APPENDIX A

CALCULATION OF ENERGY COSTS

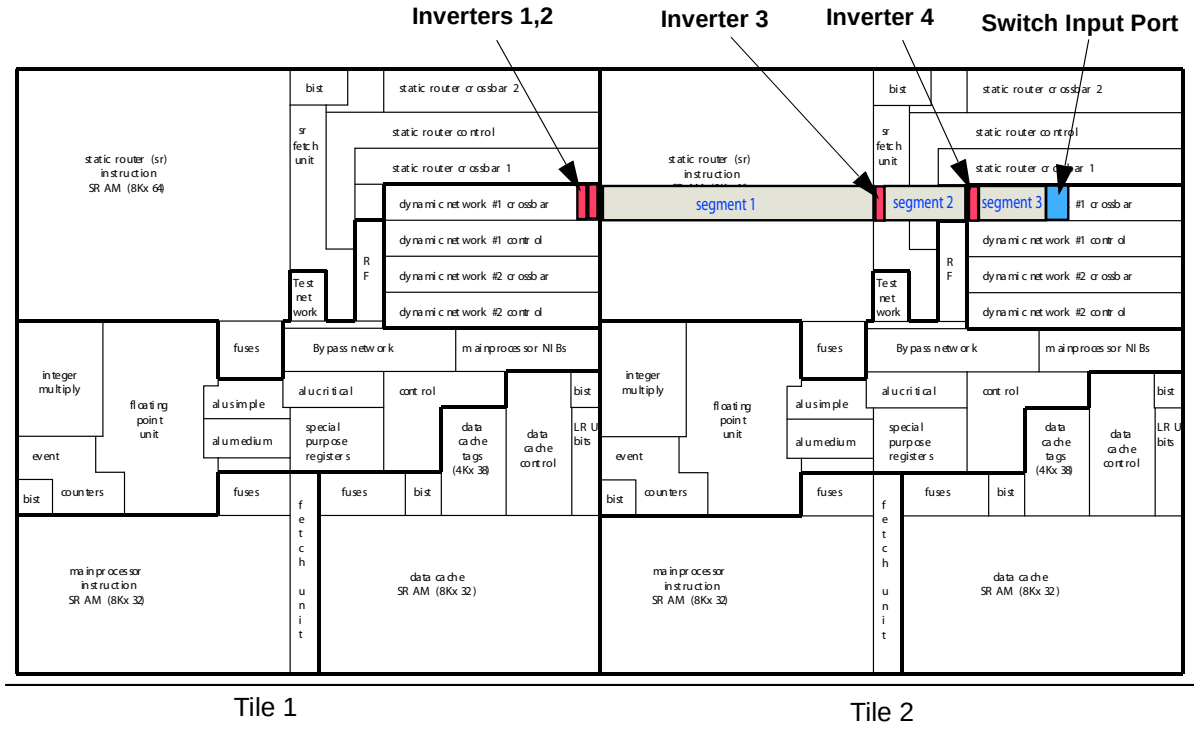
This section describes the methodology for estimating the energy costs E_C, E_Q, and E_T that we use in our analysis.

We follow a low level approach in our methodology based on capacitance values from the Raw microprocessor dynamic networks. For wiring and metal capacitance values, we use the extracted capacitance values generated by the IBM ChipEdit capacitance extractor tool for the final layout of the Raw microprocessor. For the cell input and output capacitances we use the values provided by IBM for their cells in the SA-27E process.

A.1 Link Capacitance Energy Cost Estimation E_C

Fig. A.1 shows the floorplan of two neighboring Raw tiles.We have overlaid the path of a message from Tile 1 to Tile 2.

Figure A.1: **Message Path from the East Output of a Tile to the Switch Input Port of the Neighboring Tile on the East.**



Two inverters (Inverters 1,2) drive the signal to the East direction of Tile 1. They are located at the east edge of the tile. The data path to the Switch input of Tile 2 consists of three segments. The first segment (segment 1) crosses the entire SRAM of Tile 2. The data are regenerated via inversion in “Inverter 3”; before reaching the input port of the Switch in Tile 2, the data are inverted one more time via “Inverter 4”.

The energy contributing capacitances of the data path consist of:

a) The metal capacitance of the three segments. We obtained those values from the extracted capacitance information based on the final layout of Raw. The IBM ChipEdit tool reports for every single node of the chip the best and worst case capacitance values, $25.5pF$ and $16.4pF$ respectively.

b) The input capacitance of the inverters.[1]

c) The internal and output capacitance of the inverters.

Taking into account those contributing components the equivalent capacitance for a 32-bit data transfer is $47.8pF$ and $38.7pF$ for the worst and best case respectively, resulting in an average energy cost of $34.5pJ$ and a maximum cost of $69pJ$ per transfer for an approximate path length of $4mm$ (for a 0.18u Technology and 1.8V power supply).

A.2 Crossbar and Control Logic Energy Cost Estimation E_T

The model for the switch that we show in Fig. 7.1 on page 112 lumps together the energy dissipated on the crossbar and the control logic circuitry, since this energy is always expended when a message enters the switch.

a) Crossbar

The crossbar energy consists of the energy for the propagation of the data signals from the output of the input queueing buffer multiplexor (Fig. 7.1) to one of the output ports of the switch and the energy dissipated on the multiplexors and drivers (input, internal, and output capacitances) that direct the data to the input Inverter 1 (Fig. A.1) of specific port. The average energy expended for a message going through the crossbar from west to east is $10.01pJ$.

b) Control Logic

The control circuitry consists of the input and output control logic. The input control logic provides the logic signals for the selection of the data at the input queue buffer multiplexors, generates request signals for the output control logic, and signals to the output logic the end of the message. The input control has counters and comparators that determine the direction of the message (whether the message has to travel more hops on the X or Y directions, or it has reached its destination).

1.The input and output capacitance values for the IBM standard cells are intentionally not listed, because those values are IBM's proprietary information.

This logic is responsible for requesting service to the output control logic for a specific input port. Some major components include six flip-flops, eight comparators, and a counter. The average energy expended in this block is $3.47pJ$.

The output control logic maintains the control of the destination for the data signals. It controls the output multiplexors for the network port and performs the scheduling for the output. The output logic handles all requests for the output port and is responsible for the logic signals that maintain the route requested by a message until the whole message is transmitted and detects any new message requests based on information from the input control logic.

Additionally, the output control logic includes the logic circuitry that implements a five input random function for the arbitration among the five possible (four input ports and the processor) requests. The output control logic consists mainly of logic gates and four flip-flops that hold the state of the current route of a message (3 bits for choosing among five possible routes) and another state that reveals whether a route is planned or not. The average energy expended in the output control logic block is $3.63pJ$.

A.3 Input Queue Buffer Energy Cost E_Q

The last component of the switch is the input queue buffer. This is the energy cost expended when there is contention for the output port and the message needs to be stored until it can be serviced. The Raw input buffer queue is a 4-entry deep, 32-bit wide FIFO. The total energy cost consists of the energy to write the data in one of the four positions and the energy for reading the stored data.

The flip-flops are standard-cell master-slave implementations with different clocks for the two stages (master and slave). The input data are either propagated through the multiplexer or stored in the flip-flops. The logic signals for the multiplexor selection are provided by the input control logic block.

A significant portion of the total energy is due to the distribution of the two clocks to the two inputs of the flip-flops for the 32 data bits. Moreover, from the available standard-cell implementations in the IBM library, the flip-flops in the Raw network input buffers are the ones that correspond to the highest performance level, resulting in high input, internal, and output cell capacitances. The energy cost to write new data and read the data from the input buffer in Raw is $12pJ$.

The following figures present schematics of the Raw dynamic switch.

Figure A.2: **Schematic of Dynamic Output Block. The block includes the CrossBar block, control logic for the CrossBar and for the control signal to the input block.**

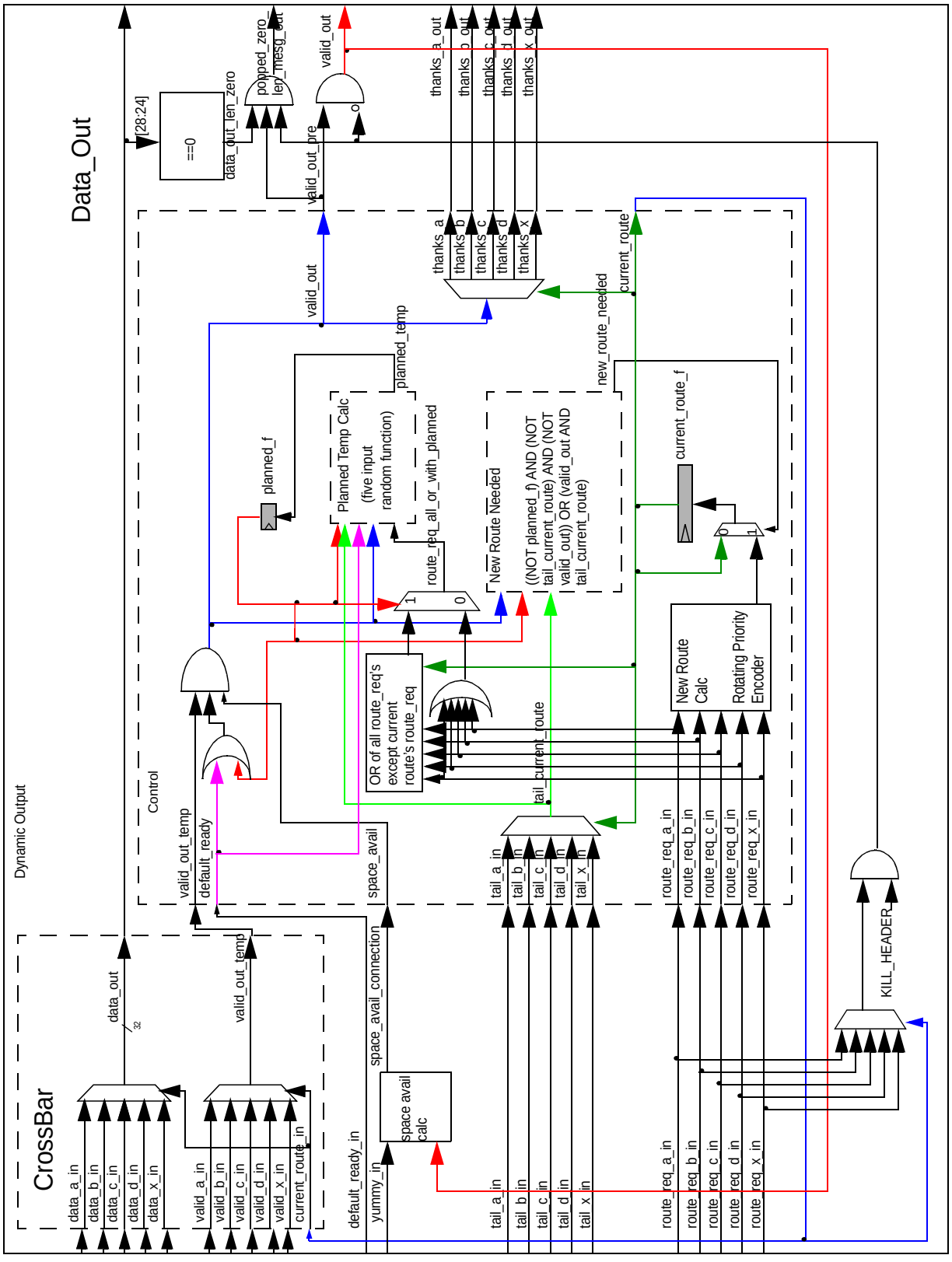
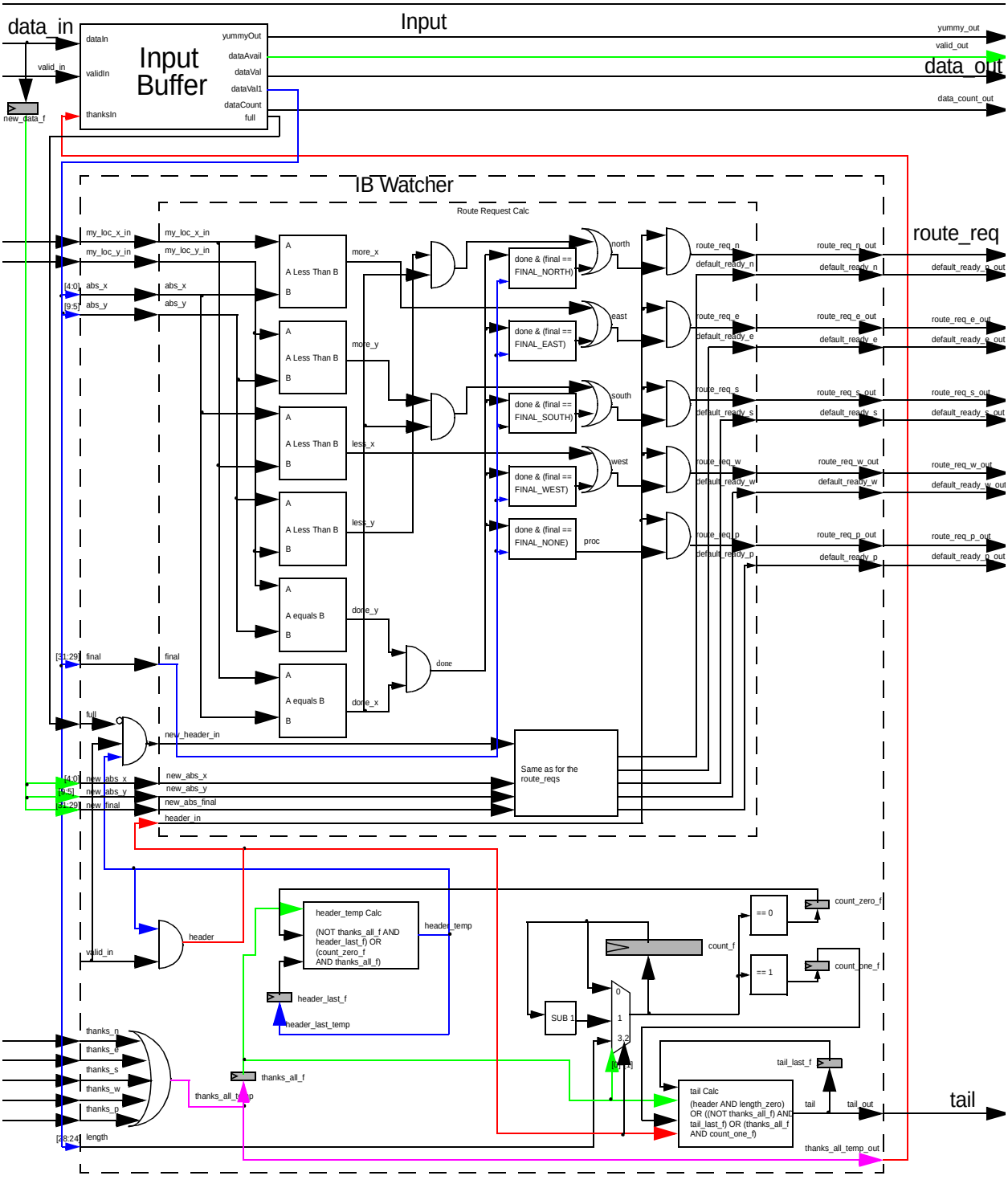


Figure A.3: **Schematic of Dynamic Input Block. The block includes the Input Buffer, control logic for the Input Buffer and the generation of *route_req* and *tail* signals.**



APPENDIX B

COMMUNICATION PATTERNS IN RAW APPLICATIONS

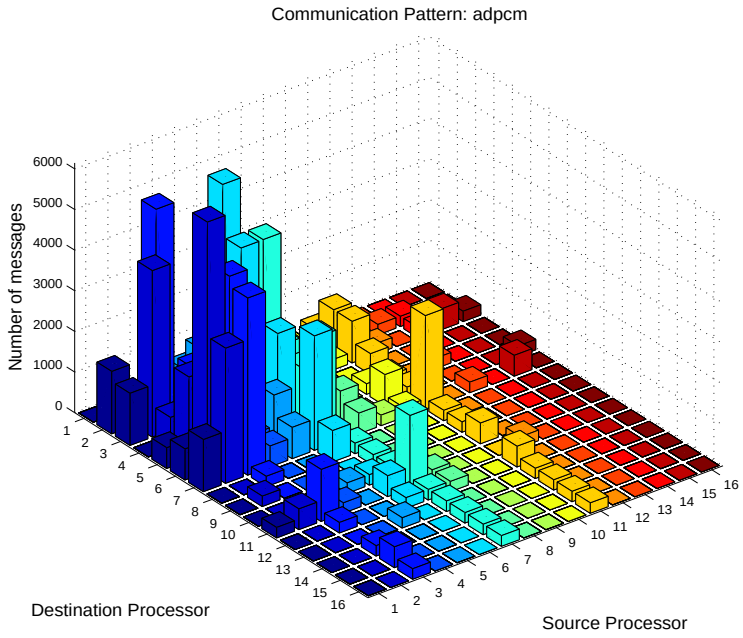
This sections presents graphs with the communication patterns for the applications that we examine in this thesis. We show the total communication between processors as well as communication from selected processors to the other processors in the system for each application. The graphs show the number of messages generated from each the processor and the destination processors of those messages.

The applications and their sources are:

adpcm (Mediabench), aes, aes_fix (FIPS-197), btrix, cholesky, fpppp, mxm, tomcatv, vpenta (Nasa7: Spec92), jacobi, jacobi_big, life (Raw benchmark suite), sha (Perl Oasis), swim (Spec95).

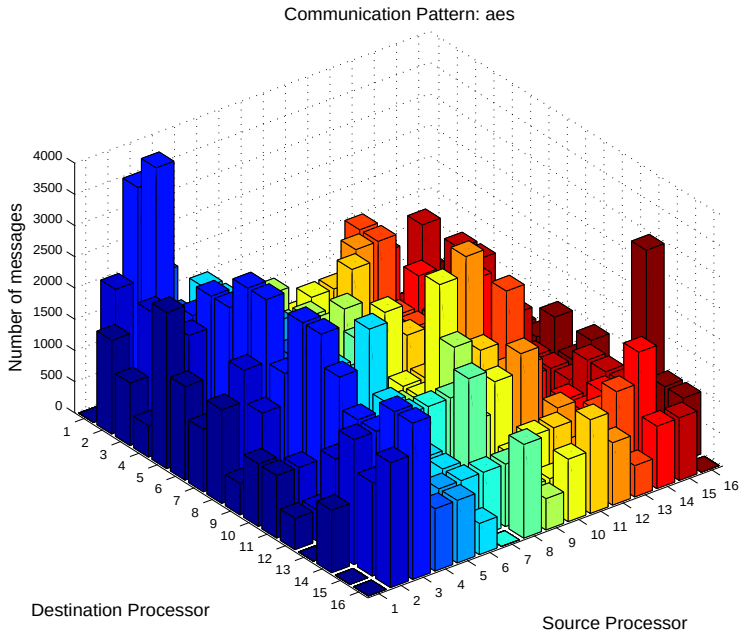
B.1 APDCM

Figure B.1: **Communication Pattern: adpcm.**



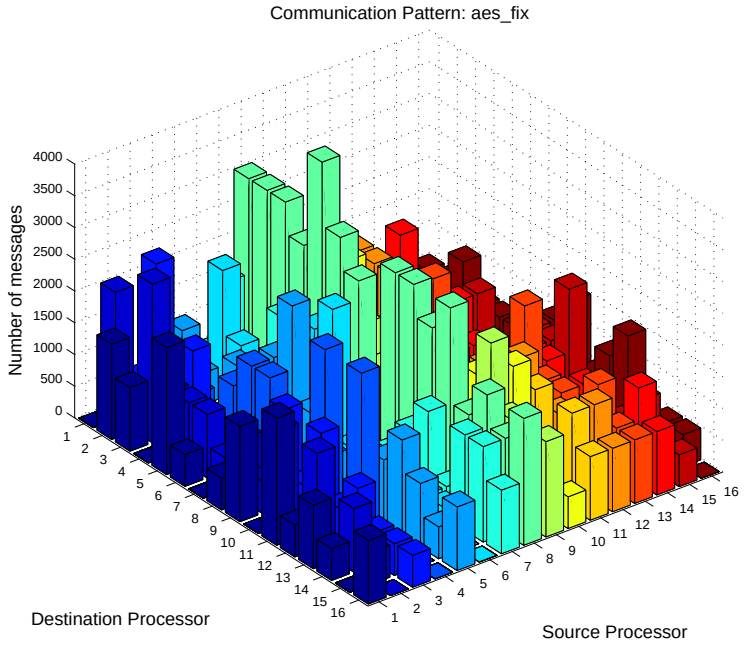
B.2 AES

Figure B.2: **Communication Pattern: aes.**



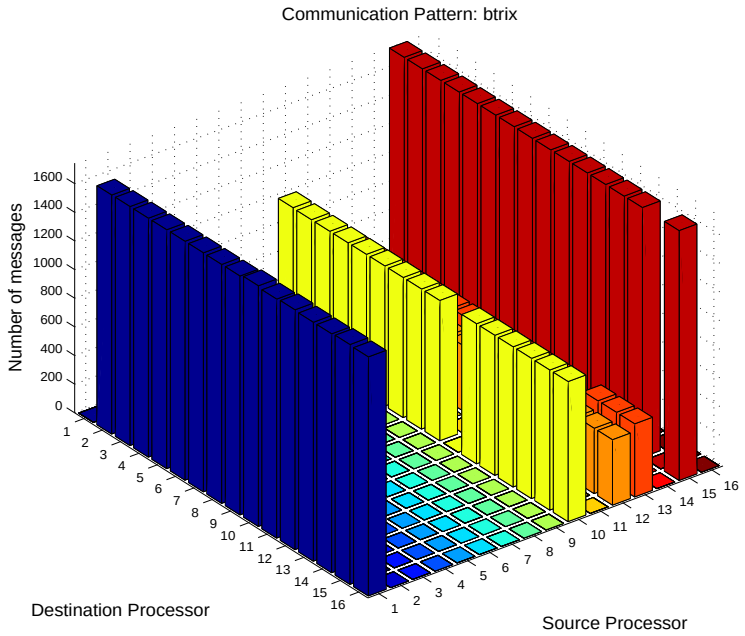
B.3 AES_FIX

Figure B.3: **Communication Pattern: aes_fix.**



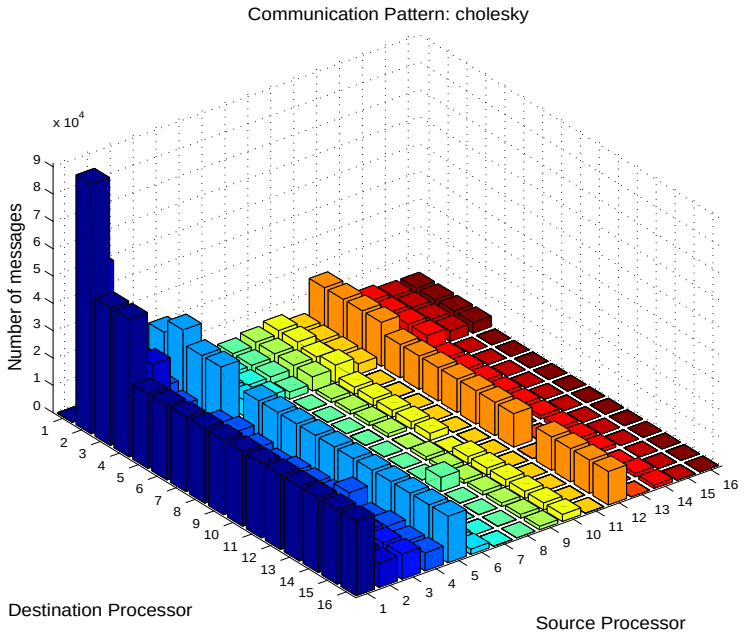
B.4 BTRIX

Figure B.4: **Communication Pattern: btrix.**



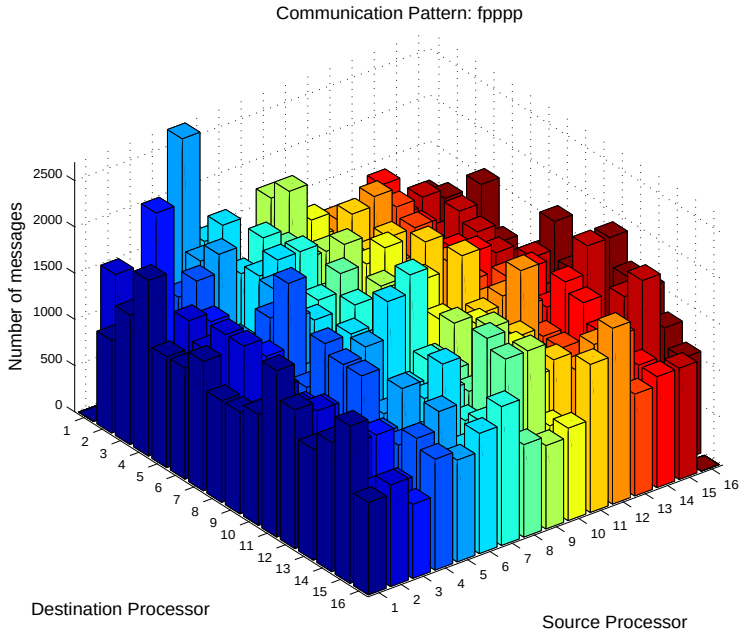
B.5 CHOLESKY

Figure B.5: **Communication Pattern: cholesky.**



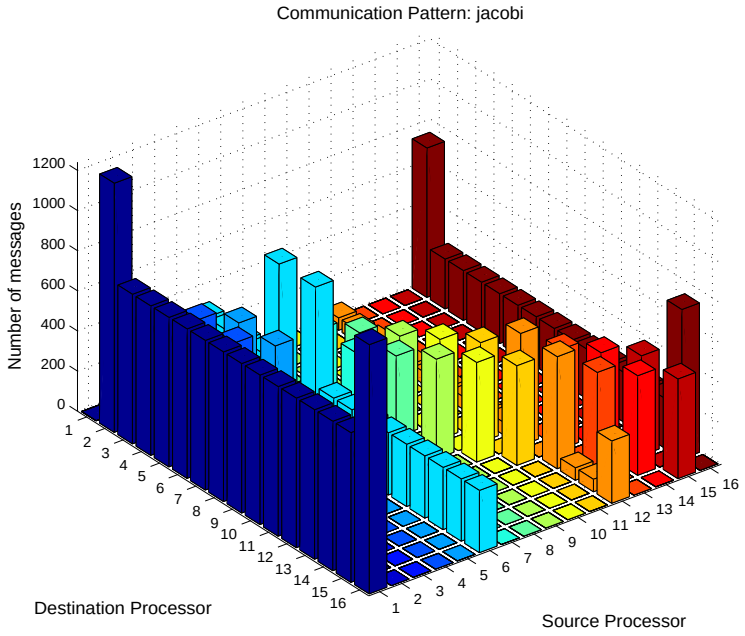
B.6 FPPPP

Figure B.6: **Communication Pattern: fpppp.**



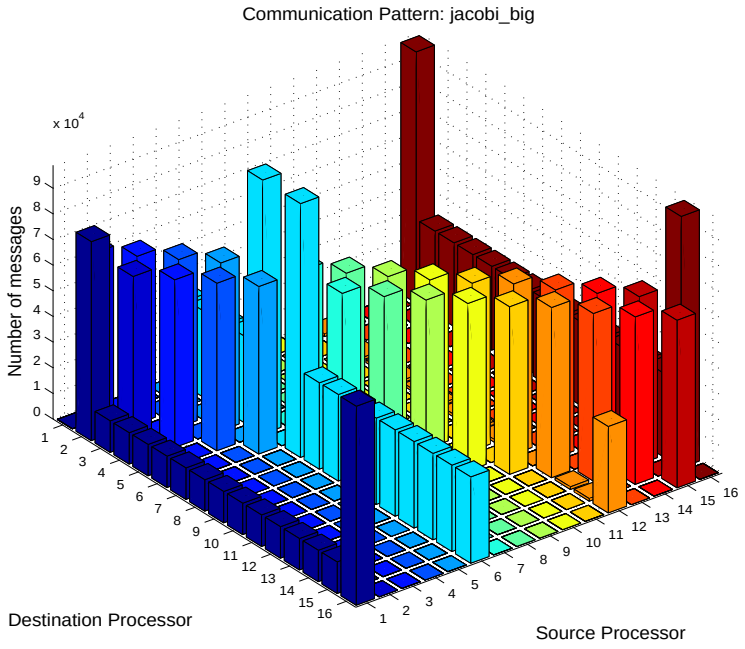
B.7 JACOBI

Figure B.7: **Communication Pattern: jacobi.**



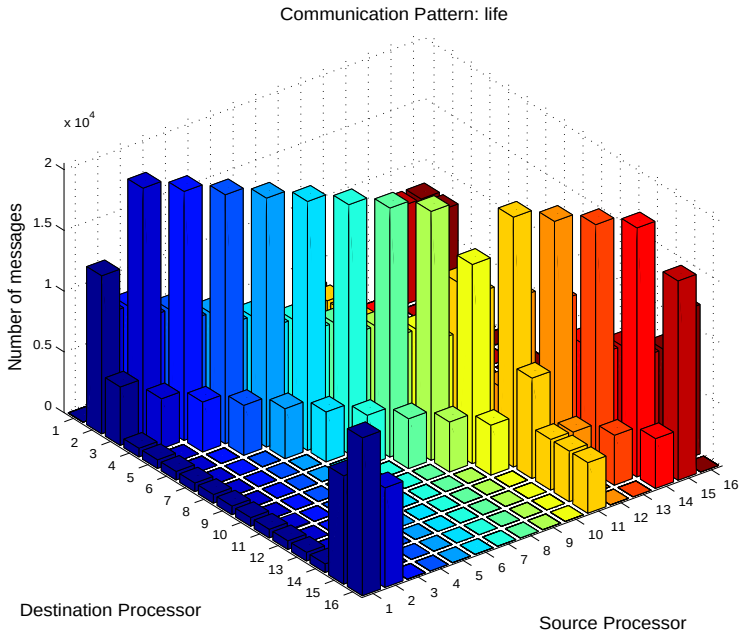
B.8 JACOBI_BIG

Figure B.8: **Communication Pattern: jacobi_big.**



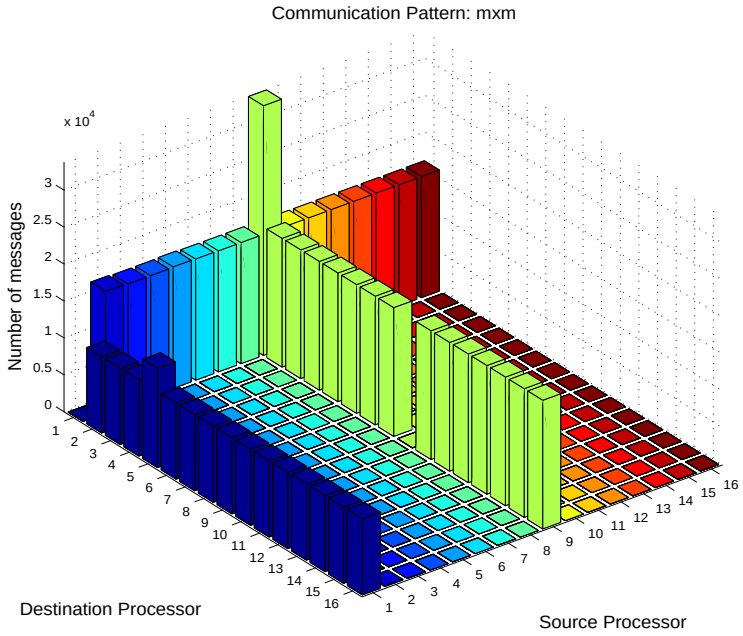
B.9 LIFE

Figure B.9: **Communication Pattern: life.**



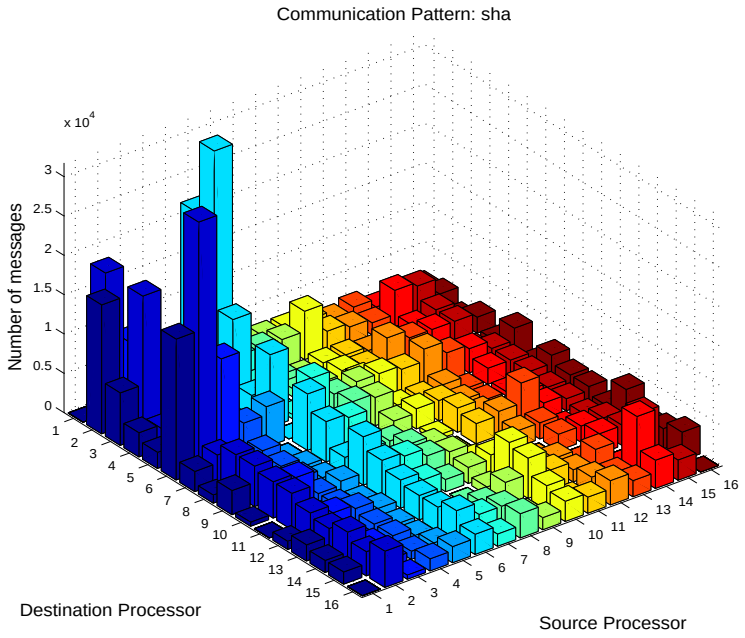
B.10 MXM

Figure B.10: **Communication Pattern: mxm.**



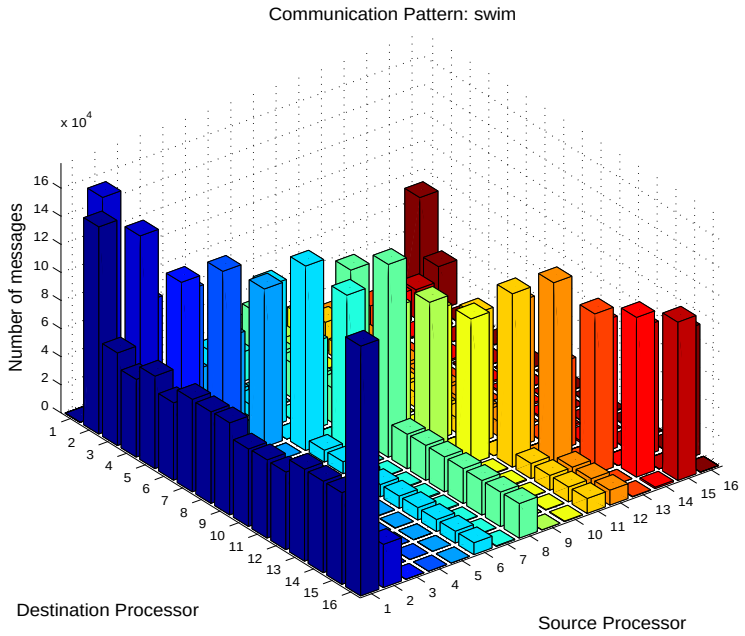
B.11 SHA

Figure B.11: **Communication Pattern: sha.**



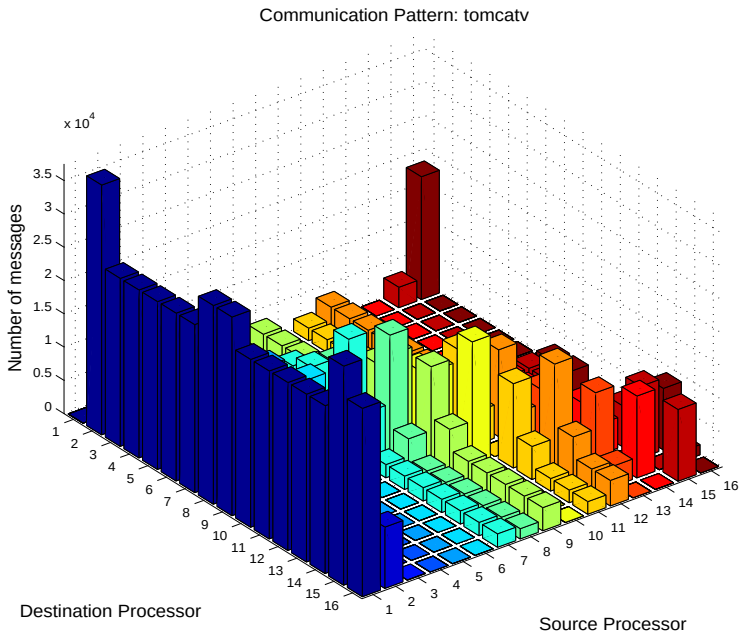
B.12 SWIM

Figure B.12: **Communication Pattern: swim.**



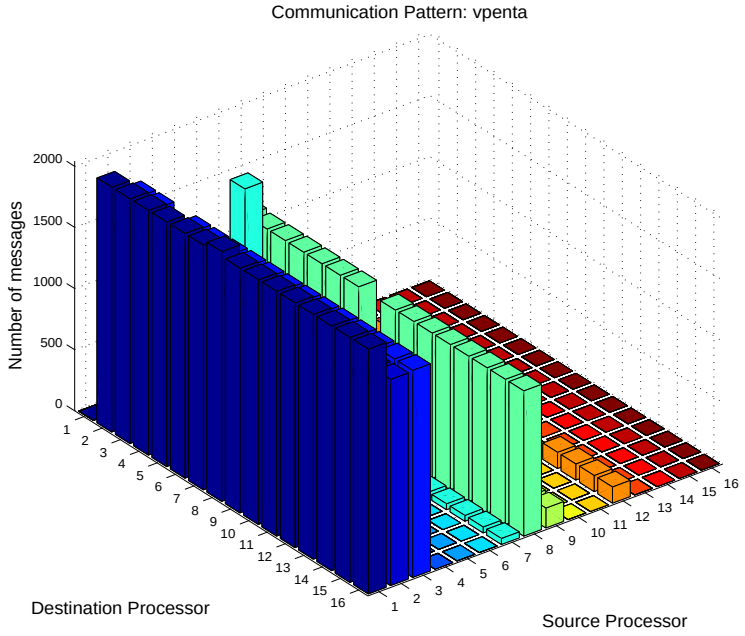
B.13 TOMCATV

Figure B.13: **Communication Pattern: tomcatv.**



B.14 VPENTA

Figure B.14: **Communication Pattern: vpenta.**



APPENDIX C

THREE-DIMENSIONAL INTERCONNECTION NETWORKS

C.1 Interconnection Energy for 3-D Systems

This section presents an analysis of three-dimensional mesh structures. With advances in 3-D VLSI technology the communication between processing elements stacked on multiple planes might be happening soon. We present an analytical approach for the estimation of interconnection energy for three-dimensional systems, assuming that a three-dimensional network can be implemented.

We modify Eq. 4.2 on page 71 to describe the expected energy cost $E_{i,j}$ between two processors P_i and P_j for three-dimensional systems. The expected energy cost $E_{i,j}$ of transferring the data from processor P_i to processor P_j, is given by

$$E_{i,j} = M \cdot p_{i,j} \cdot (E_l \cdot \tilde{H}_{i,j} + E_v \cdot V_{i,j}) . \qquad \text{C.1}$$

Each element of matrix $\tilde{H}$ gives the number of hops between two processors on the two-

dimensional plane. Matrix V describes the distance on the third dimension and E_v is the energy cost of going through a via that connects two planes. The processor I.D.s increase column-wise then row-wise on each plane.

The following analysis assumes that $E_v = E_l$. Therefore, Eq. C.1 can be written as

$$E_{i,j} = M \cdot p_{i,j} \cdot E_l \cdot H_{i,j} \qquad \text{C.2}$$

which is the general equation for the expected energy cost for the communication of the messages originating from processor P_i with destination processor P_j.

In the system realization there are *"r"* number of rows, *"c"* numbers of columns and *"d"* is the system depth. H is an *rcd-by-rcd* symmetrical matrix and looks like:

$$H = \begin{bmatrix}
\begin{matrix} 0 & 1 & \dots & r+c-2 \\ 1 & 0 & \dots & \dots \\ \dots & \dots & 0 & 1 \\ r+c-2 & \dots & 1 & 0 \end{matrix} & \begin{matrix} 1 & 2 & \dots & r+c-1 \\ 2 & 1 & \dots & \dots \\ \dots & \dots & 1 & 2 \\ r+c-1 & \dots & 2 & 1 \end{matrix} & \dots & \begin{matrix} d-1 & d & \dots & r+c+d-3 \\ d & d-1 & \dots & \dots \\ \dots & \dots & d-1 & \dots \\ r+c+d-3 & \dots & \dots & d-1 \end{matrix} \\
\begin{matrix} 1 & 2 & \dots & r+c-1 \\ 2 & 1 & \dots & \dots \\ \dots & \dots & 1 & 2 \\ r+c-1 & \dots & 2 & 1 \end{matrix} & \begin{matrix} 0 & 1 & \dots & r+c-2 \\ 1 & 0 & \dots & \dots \\ \dots & \dots & 0 & 1 \\ r+c-2 & \dots & 1 & 0 \end{matrix} & \dots & \begin{matrix} \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{matrix} \\
\dots & \dots & \dots & \dots \\
\begin{matrix} d-1 & d & \dots & r+c+d-3 \\ d & d-1 & \dots & \dots \\ \dots & \dots & d-1 & \dots \\ r+c+d-3 & \dots & \dots & d-1 \end{matrix} & \dots & \dots & \begin{matrix} 0 & 1 & \dots & r+c-2 \\ 1 & 0 & \dots & \dots \\ \dots & \dots & 0 & 1 \\ r+c-2 & \dots & 1 & 0 \end{matrix}
\end{bmatrix} . \qquad \text{C.3}$$

The *rc-by-rc* matrix H_{rc} (Eq. C.3) gives the distance between two processors in a two dimensional mesh. If we define a *rc-by-rc* matrix with all elements equal to one

$$A = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & 1 \end{bmatrix}, \qquad \text{C.4}$$

then H can be written as

$$H = \begin{bmatrix} H_{rc} & H_{rc}+A & \dots & H_{rc}+A(d-1) \\ H_{rc}+A & H_{rc} & \dots & \dots \\ \dots & \dots & \dots & H_{rc}+A \\ H_{rc}+A(d-1) & \dots & H_{rc}+A & H_{rc} \end{bmatrix} = \quad \text{C.5}$$

$$= \begin{bmatrix} H_{rc} & H_{rc} & \dots & H_{rc} \\ H_{rc} & H_{rc} & \dots & \dots \\ \dots & \dots & \dots & H_{rc} \\ H_{rc} & \dots & H_{rc} & H_{rc} \end{bmatrix} + \begin{bmatrix} 0 & A & \dots & A(d-1) \\ A & 0 & \dots & \dots \\ \dots & \dots & \dots & A \\ A(d-1) & \dots & \dots & 0 \end{bmatrix} = \overline{H} + \overline{A}. \quad \text{C.6}$$

C.2 Uniform Distribution

The uniform communication probability $P_{i,j}$ of Eq. C.1 is equal to $\frac{1}{rcd-1}$. We get the following equation for the expected energy cost for the communication between processors P_i and P_j.

$$E_{i,j} = \frac{M}{rcd-1} \cdot E_l \cdot H_{i,j}. \quad \text{C.1}$$

The total expected energy cost of transmitting the data is

$$E_N = \sum_{k=1}^{N} E_k = \sum_{i=1}^{N} \left(\sum_{j=1}^{N} E_{i,j} \right) = \frac{M}{rcd-1} \cdot E_l \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} H_{i,j}, \quad \text{C.2}$$

where $N = r \cdot c \cdot d$.

We find the sum of the elements of H,

$$\sum_{i=1}^{N} \sum_{j=1}^{N} H_{i,j} = \sum_{i=1}^{N} \sum_{j=1}^{N} \overline{H}_{i,j} + \sum_{i=1}^{N} \sum_{j=1}^{N} \overline{A}_{i,j} = \quad \text{C.3}$$

$$= d^2 \cdot \sum_{i=1}^{rc} \sum_{j=1}^{rc} H_{rc_{i,j}} + \left(\sum_{i=1}^{rc} \sum_{j=1}^{rc} A_{i,j} \right) \cdot sum \begin{bmatrix} 0 & 1 & \dots & d-1 \\ 1 & 0 & \dots & \dots \\ \dots & \dots & \dots & 1 \\ d-1 & \dots & \dots & 0 \end{bmatrix} = \quad \text{C.4}$$

$$= d^2 \cdot \left(\frac{r^2 c(c-1)(c+1)}{3} + \frac{c^2 r(r-1)(r+1)}{3} \right) + (rc)^2 \cdot \frac{d(d-1)(c+1)}{3} = \quad \text{C.5}$$

$$= \left[\frac{r^2 \cdot d^2 \cdot c \cdot (c-1) \cdot (c+1) + c^2 \cdot d^2 \cdot r \cdot (r-1) \cdot (r+1) + r^2 \cdot c^2 \cdot d \cdot (d-1) \cdot (d+1)}{3}\right] = \quad \text{C.6}$$

$$= rcd \cdot \left[\frac{r \cdot d(c^2-1) + c \cdot d(r^2-1) + r \cdot c(d^2-1)}{3}\right] = rcd \cdot \left[\frac{d(rc-1)(r+c) + r \cdot c(d^2-1)}{3}\right]. \quad \text{C.7}$$

So the total expected energy cost of transmitting the data is

$$E_N = \frac{M}{rcd-1} \cdot E_l \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} H_{i,j} = M \cdot E_l \cdot \frac{rcd}{rcd-1} \cdot \frac{d(rc-1)(r+c) + r \cdot c(d^2-1)}{3}. \quad \text{C.8}$$

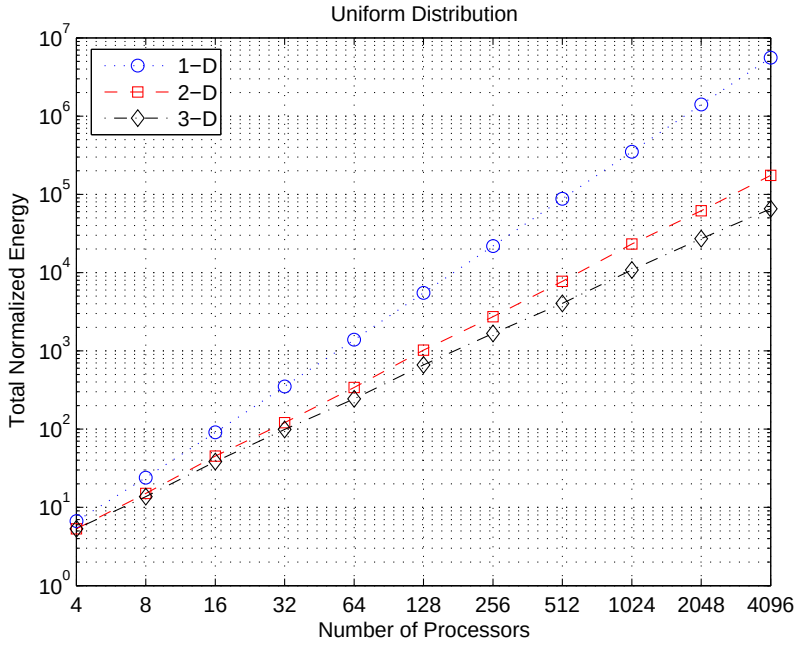
Table C.1 summarizes the total expected energy dissipation of the interconnection system for one, two and three-dimensional mesh networks.

Table C.1: Total expected energy dissipation assuming uniform distribution $c \geq 2, r, d \geq 1$

	1-D	**2-D**	**3-D**
Total Expected Energy E_{P2P}	$ME_l \cdot \frac{c(c+1)}{3}$	$ME_l \cdot \frac{rc(r+c)}{3}$	$ME_l \frac{rcd}{rcd-1} \cdot \frac{d(rc-1)(r+c) + r \cdot c(d^2-1)}{3}$

Fig. B.1 compares the total expected normalized energy ($M = 1, E_l = 1$) assuming uniform distribution for the three systems, and the same number of processors. Both axes are in logarithmic scale (y-axis base 10, x-axis base 2).

Figure C.1: **Normalized Energy - Uniform Distribution for 1-D, 2-D and 3-D mesh networks with same numbers of processors.**



The energy savings increase exponentially as the dimensionality of the network increases. However, when we introduce the switch energy we should expect decreased savings than the ones shown in the graph, since the switch architecture becomes more complex as the dimensionality increases; therefore the switch energy increases.

REFERENCES

[1] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffmann, P. Johnson, J. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe and A. Agarwal. “The Raw Microprocessor: A Computational Fabric for Software Circuits and General Purpose Programs.” *IEEE Micro,* Mar/Apr 2002.

[2] M. B. Taylor, W. Lee, J. Miller, D. Wentzlaff, I. Bratt, B. Greenwald, H. Hoffmann, P. Johnson, J. Kim, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal. “Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams.” *Proceedings of International Symposium on Computer Architecture*, June 2004.

[3] M. B. Taylor, W. Lee, S. Amarasinghe, and A. Agarwal. “Scalar Operand Networks: On-Chip Interconnect for ILP in Partitioned Architectures.” *2003 HPCA*, pp. 341-353.

[4] S. Wilton and N. Jouppi. “An Enhanced Access and Cycle Time Model for On-chip Caches.” In *WRL Research Report 93/5, DEC Western Research Laboratory,* 1994.

[5] P. Shivakumar and N. Jouppi. “CACTI 3.0: An Integrated Cache Timing, Power and Area Model.” In *WRL Research Report 2001/2, Compaq Western Research Laboratory,* 2001.

[6] V. Zyuban and P. Kogge. “Inherently Lower-Power High-Performance Superscalar Architectures.” *IEEE Transactions on Computers,* Vol. 50, Issue 3, March 2001.

[7] T. Mudge. “Power: A first class design constraint,” *Computer Journal*, Vol. 34, (4) pp 52-57, April 2001.

[8] Jason Kim, Michael B Taylor, Jason Miller, David Wentzlaff. “Energy Characterization of a Tiled Architecture Processor with On-Chip Networks.” *Proceedings of the Int’l Symp. Low Power Electronics and Design*, 2003 ISLPED pp 424-427.

[9] G. Hinton, M. Upton, D. Sager, D. Boggs, D. Carmean, P. Roussel, T. Chappell, T. Fletcher, M. Milshtein, Milo Sprague, S. Samaan and R. Murray, “A 0.18-μm CMOS IA-32 Processor With a 4-GHz Integer Execution Unit,” in *IEEE JSSC, vol. 36, No 11,* November 2001.

[10] M. Horowitz and W. Dally, “How scaling will change processor architecture,” in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers,* Feb. 2004, pp 132-133.

[11] J. Rabaey, A. Chandrakasan, B. Nicolic, “Digital Integrated Circuits, A Design Perspective”, 2nd Edition, Prentice Hall, 2003.

[12] A. Agarwal, “Limits on interconnection network performance,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 2, no. 4, pp. 398-412, 1991.

[13] W. J. Dally, B. Towles, “Principles and Practices of Interconnection Networks,” *Morgan Kaufmann*, 2003.

[14] R. Ho, K. Mai, M. Horowitz, “The Future of Wires,” in *Proceedings of the IEEE*, April 2001, pages 490-504.

[15] A. Jain, W. Anderson, T. Benninghoff, 0. Bertucci, M. Braganza, J. Burnette. T. Chang, J. Eble, R. Faber, D. Gowda, J. Grodstein, G. Hess, J. Kowaleski, A. Kumar, B Miller, R. Mueller, P. Paul, J. Pickholtz. S. Russell, M. Shen, T., Truex, A. Vardharajan, D. Xanthopoulos, T. Zou. “A 1.2GHz Alpha Microprocessor with 44.8GEVs Chip Pin Bandwidth”, *Solid-State Circuits Conference, 2001*. Digest of Technical Papers. ISSCC. 2001 IEEE International, 5-7 Feb. 2001, pp 240-241.

[16] Anderson, C.J.; Petrovick, J.; Keaty, J.M.; Warnock, J.; Nussbaum, G.; Tendier, J.M.; Carter, C.; Chu, S.; Clabes, J.; DiLullo, J.; Dudley, P.; Harvey, P.; Krauter, B.; LeBlanc, J.; Pong-Fei Lu; McCredie, B.; Plum, G.; Restle, P.J.; Runyon, S.; Scheuermann, M.; Schmidt, S.; Wagoner, J.; Weiss, R.; Weitzel, S.; Zoric, B., “Physical design of a fourth-generation POWER GHz microprocessor”, *Solid-State Circuits Conference, 2001.* Digest of Technical Papers. ISSCC. 2001 IEEE International, 5-7 Feb. 2001, pp 232 - 233, 451.

[17] Konstadinidis, G.; Normoyle, K.; Wong, S.; Bhutani, S.; Stuimer, H.; Johnson, T.; Smith, A.; Cheung, D.; Romano, F.; Shifeng Yu; Sung-Hun Oh; Melamed, V.; Narayanan, S.; Bunsey, D.; Cong Khieu; Wu, K.J.; Schmitt, R.; Dumlao, A.; Sutera, M.; Jade Chau; Lin, K.J., “Implementation of a third-generation 1.1GHz 64b microprocessor”, *Solid-State Circuits Conference, 2002*. Digest of Technical Papers. ISSCC. 2002 IEEE International ,Volume: 1 , 3-7 Feb. 2002, pp 338 - 472 vol.1.

[18] S. Naffziger, B. Stackhouse, T. Grutkowski, “The Implementation of a 2-core Multi-Threaded Itanium(R)-Family Processor”, *Solid-State Circuits Conference, 2005.* Digest of Technical Papers. ISSCC. 2005 IEEE International, 6-10 Feb. 2005, pp 182 - 183, 592.

[19] M. Taylor, “The Raw Processor Specification”, *Comprehensive specification for the Raw processor*.

[20] T. Xanthopoulos, personal communication, unpublished manuscript 2007. Data source: International Solid State Circuits Conference 1973-2007, Intel Corporation.

[21] J. Schutz and C. Webb, “A scalable X86 CPU design for 90 nm process”, *Solid-State Circuits Conference, 2004.* Digest of Technical Papers. ISSCC. 2004 IEEE International pp 62- 513 Vol.1.

[22] G. E. Moore, “Cramming more Components onto Integrated Circuits”, *Electronics*, vol. 38, no. 8, April 19, 1965.

[23] G. E. Moore, “No exponential is forever: but “Forever” can be delayed!”, *Solid-State Circuits Conference, 2003.* Digest of Technical Papers. ISSCC. 2003 IEEE International pp 20-23 Vol.1.

[24] R. Gonzalez and M. Horowitz. “Energy Dissipation in General Purpose Microprocessors”, *IEEE Journal of Solid-State Circuits, 31(9):1277-84, 1996.*

[25] Krashinsky, R.; Batten, C.; Hampton, M.; Gerding, S.; Pharris, B.; Casper, J.; Asanovic, K.; “The vector-thread architecture”, *Computer Architecture, 2004*. Proceedings. 31st Annual International Symposium on, 19-23 June 2004 Page(s):52 - 63.

[26] K. Mai, T. Paaske, N. Jayasena, R. Ho, W. Dally, M. Horowitz, “Smart Memories: A Modular Reconfigurable Architecture”, *Computer Architecture,* 2000. Proceedings of the 27th Annual International Symposium on. Page(s): 161 - 171.

[27] W. J. Dally, P. Hanrahan, M. Erez, T. J. Knight, F. Labonté, J. Ahn, N. Jayasena, U. J. Kapasi, A. Das, J. Gummaraju, I. Buck, “Merrimac: Supercomputing with Streams”, *SC2003*, November 2003, Phoenix, Arizona.

[28] S. Swanson, K. Michelson, A. Schwerin, M. Oskin, “WaveScalar”, *Microarchitecture, 2003*. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on, 3-5 Dec. 2003 Page(s): 291 - 302.

[29] S.W. Keckler, Doug Burger, C.R. Moore, R. Nagarajan, K. Sankaralingam, V. Agarwal, M.S. Hrishikesh, N. Ranganathan, and P. Shivakumar, “A Wire-Delay Scalable Microprocessor Architecture for High Performance Systems”, *International Solid-State Circuits Conference (ISSCC)*, pp. 1068-1069, February, 2003.

[30] G. S. Sohi, S. Breach, and T. N. Vijaykumar, "Multiscalar Processors," *Proc. ISCA-22*, pp. 414-425, June 1995.

[31] Oliver, J.; Rao, R.; Sultana, P.; Crandall, J.; Czernikowski, E.; Jones, L.W., IV; Franklin, D.; Akella, V.; Chong, F.T.; “Synchroscalar: a multiple clock domain, power-aware, tile-based embedded processor”, *Computer Architecture, 2004*. Proceedings. 31st Annual International Symposium on, 19-23 June 2004 Page(s):150 - 161.

[32] International Technology Roadmap for Semiconductors, 2003 Edition, http://public.itrs.net.

[33] W. Lee, R. Barua, M. Frank, D. Srikrishna, J. Babb, V. Sarkar, and S. Amarasinghe, “Space-time scheduling of instruction-level parallelism on a raw machine,” in *ASPLOS-VIII: Proceedings of the eighth international conference on Architectural support for programming languages and operating systems*. New York, NY, USA: ACM Press, 1998, pp. 46-57.

[34] Moritz, C.A.; Yeung, D.; Agarwal, A.; “Exploring Optimal Cost-Performance Designs for Raw microprocessors”. *IEEE Symposium on FPGAs for Custom Computing Machines*, *FCCM’98*. Napa Valley, CA Page(s): 12-27.

[35] The Device Group at UC Berkeley, “Berkeley Technology Predictive Models”, *BTPM website*, http://www-device.eecs.berkeley.edu/~ptm/introduction.html.

[36] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, “Orion: a power-performance simulator for interconnection networks,” in *MICRO 35: Proceedings of the 35th annual ACM/IEEE International Symposium on Microarchitecture. Los Alamitos, CA USA:* IEEE Computer Society Press, 2002, pp. 294-305.

[37] A. J. Smith, Cache Memories, *ACM Computing Surveys (CSUR)*, v.14 n.3, p.473-530, Sept. 1982.

[38] Jim Handy, The Cache Memory Book, *Academic Press Professional, Inc*. San Diego, CA, 1993.

[39] N. Eisley and L.-S. Peh, “High-level power analysis for on-chip networks,” in *CASES ‘04: Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems*. New York, NY, USA: ACM Press, 2004, pp. 104-115.

[40] C. Su, and A. Despain, “Cache Design Tradeoffs for Power and Performance Optimization: A Case Study”, *in Proc. of the Int’l. Sym. on Low Power Design*, 1995, pp. 63-68.

[41] H. Wang, L.-S. Peh, and S. Malik, “A technology-aware and energy-oriented topology exploration for on-chip networks,” in *Design, Automation and Test in Europe*, March 2005, pp. 1238-1243.

[42] J. Kim, M. B. Taylor, J. Miller, and D. Wentzlaff, “Energy Characterization of a Tiled Architecture Processor with On-Chip Networks,” in *2003 ISLPED*, 2003, pp. 424-427.

[43] A. Aggarwal, M. Franklin. "Energy Efficient Asymmetrically Ported Register Files," *International Conference on Computer Design*, 2003 , vol. 00, pp. 2-7.

[44] V. Zyuban, and P. Kogge, “The energy complexity of register files”. *International Symposium on Low Power Electronics and Design* In Proceedings of the, ISLPED '98, pp. 305 - 310.

[45] J. H. Tseng, and K. Asanovic, “Energy-Efficient Register Access”. In Proceedings of the 13th Symposium on Integrated Circuits and Systems Design (September 18 - 24, 2000). SBCCI. IEEE Computer Society.

[46] M. Mamidipaka and N. Dutt, “eCACTI: An Enhanced Power Estimation Model for On-chip Caches”, *Center for Embedded Computer Systems (CECS) Technical Report* TR-04-28, Sept. 2004.

[47] S. Kim, N. Vijaykrishnan, M. Kandemir, A. Sivasubramaniam, M. J. Irwin, and E. Geethanjali, “Power-aware partitioned cache architectures.” *In Proceedings of the 2001 international Symposium on Low Power Electronics and Design*. ISLPED’01. Huntington Beach, California, United States, pp: 64 - 67.

[48] F. Labonte, P. Mattson, I. Buck, C. Kozyrakis and M. Horowitz, “The Stream Virtual Machine,” *Proceedings of the 13th Intl. Conference on Parallel Architecture and Compilation Techniques (PACT)*, September 2004.

[49] Sriram Vangal, Jason Howard, Gregory Ruhl, Saurabh Dighe, Howard Wilson, James Tschanz, David Finan, Priya Iyer, Arvind Singh, Tiju Jacob, Shailendra Jain, Sriram Venkataraman, Yatin Hoskote and Nitin Borkar. "An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS". *Solid-State Circuits Conference, 2007. Digest of Technical Papers.* ISSCC. 2007 IEEE International pp 98-99.

[50] M. Taylor, “*btl*” Raw Technical Memo #19.

[51] J. Duato, S. Yalamanchili, and L. Ni. “Interconnection Networks”. *Morgan Kaufmann Publishers Inc.*, 2002.

[52] W. Dally. “Performance analysis of k-ary n-cube interconnection networks.” *IEEE Trans. Computers.*, vol. 39, no.6, pp 775-785, 1990.

[53] V. S. Adve and M. K. Vernon, “Performance analysis of mesh interconnection networks with deterministic routing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 3, pp. 225-246, 1994.

[54] V. Raghunathan, M. Srivastava, and R. Gupta, “A survey of techniques for energy efficient on-chip communication,” in *DAC*, 2003, pp. 900-905.

[55] H. Wang, L.-S. Peh, and S. Malik, “A technology-aware and energy-oriented topology exploration for on-chip networks,” in *Design, Automation and Test in Europe*, March 2005, pp. 1238-1243.

[56] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, N. Ranganathan, D. Burger, S. W. Keckler, R. G. McDonald, and C. R. Moore, “Trips: A polymorphous architecture for exploiting ilp, tlp, and dlp,” *ACM Trans. Archit. Code Optim.*, vol. 1, no. 1, pp. 62-93, 2004.

[57] P. Kongetira, K. Aingaran, K. Olukotun, "Niagara: A 32-Way Multithreaded Sparc Processor," *IEEE Micro*, vol. 25, no. 2, pp. 21-29, Mar/Apr, 2005.

[58] D. Pham, S. Asano, M. Bolliger, M. N. Day, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S.

Weitzel, D. Wendel, T. Yamazaki, K. Yazawa “The design and implementation of a first-generation CELL processor,” in *IEEE ISSCC Dig. Tech. Papers* 2005, pp. 184-186.

[59] J. Friedrich, B. McCredie, N. James, B. Huott, B. Curran, E. Fluhr, G. Mittal, E. Chan, Y. Chan, D. Plass, S. Chu, H. Le, L. Clark, J. Ripley, S. Taylor, J. DiLullo, M. Lanzerotti, “The Design of the POWER6 Microprocessor," *Proceedings of the International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, Feb. 11-15, 2007.

[60] E. Gelenbe, and G. Pujolle, “Introduction to Queueing Networks,” p.44 *Publisher John Wiley & Sons, Inc.,* 1998, ISBN 0-471-96294-5.

[61] L. Kleinrock, “Queueing Systems,” *Wiley*, New York, 1975.

[62] Kim, C., Burger, D., and Keckler, S. W. 2002. “An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches.” In *Proceedings of the 10th international Conference on Architectural Support For Programming Languages and Operating Systems* (San Jose, California, October 05 - 09, 2002). ASPLOS-X. ACM Press, New York, NY, 211-222.

[63] E. Waingold, M. Taylor, D. Srikrishna, V. Sarkar, W. Lee, V. Lee, J. Kim, M. Frank, P. Finch, R. Barua, J. Babb, S. Amarasinghe, and A. Agarwal.“Baring it all to Software: Raw Machines” *IEEE Computer*, September 1997, pp 86-93.

[64] M. Vachharajani, N. Vachharajani, D. A. Penry, J. A. Blome, and D. I. August. “Microarchitectural exploration with Liberty.” In *Proc. International Symposium on Microarchitecture*, 2002.

[65] R. Kumar, V. Zyuban, D.Tullsen. “Interconnections in multi-core architectures: Understanding Mechanisms, Overheads and Scaling”. *32nd International Symposium on Computer Architecture*, ISCA-32, Madison, Wisconsin, June 2005.