INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

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

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
The GLORI strategy for multiprocessors: Integrating optics into the interconnect architecture

Pinkston, Timothy Mark, Ph.D.
Stanford University, 1993

Copyright ©1992 by Pinkston, Timothy Mark. All rights reserved.
THE GLORI STRATEGY FOR MULTIPROCESSORS:
INTEGRATING OPTICS INTO THE INTERCONNECT
ARCHITECTURE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Timothy Mark Pinkston
December 1992
I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Michael Flynn
(Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Joseph W. Goodman

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Martin Hellman

Approved for the University Committee on Graduate Studies:

Judith J. MacIntyre
Abstract

A major portion of multiprocessor communication latency is associated with the interconnect network. A fixed, low-bandwidth network that is unable to adapt to memory reference traffic patterns of processors to memory can result in high latency. We propose a high performance interconnect strategy that exploits reference locality to lower network traffic. It does so even in cases where such exploitation by caches results in poor performance. The GLORI strategy, which stands for Global-Local Optical Reconfigurable Interconnect, reduces network latency and lowers interconnect cost.

GLORI is based on free-space holographic diffractive optics. Its organization is based on a 2-tier shared bus-hypercube topology. A processor-memory node directly communicates to neighboring nodes sharing its optical bus; these are a subset of all nodes in the system. Proper spatial positioning allows optical bit-lines of each node to use the same hologram facet linking the nodes. Reconfigurable optics can establish the interconnection of various permutations of nodes onto local buses.

Communication to non-bus nodes requires multiple hops across remote bus clusters arranged in an optical hypercube. Global network hops have higher latency than local network hops. Hop count is minimized by reconfiguring system interconnections to exploit reference locality. Hence, an advantage of GLORI is its ability to dynamically cluster communicating nodes to capture the majority of referencing activity locally.

Analysis shows GLORI's optical routing unit is feasibly implemented in $\approx 1cm^3$ volume (static 64 node system). GLORI's high-bandwidth optical links ($1GBps$) can transport 128-bit messages in an estimated $20nsec$. Simulations show that GLORI's reconfigurability (100$\mu$sec switch time) provides significant speed-up in applications with high cluster locality, but minimal gains in those with low cluster locality.
Acknowledgements

I am much indebted to Professor Michael J. Flynn and Professor Joseph W. Goodman for their inspiring guidance, steadfast assurance, and unequalled technical expertise. More than thesis advisors, these men are my "personal mentors." I also thank Professor Martin Hellman, Professor Martin Morf, and Professor Steven Lundstrom for their technical advice and enlightening discussions.

The contributions to this work by my fellow colleagues in Professor Flynn's and Professor Goodman's research groups are acknowledged. Special thanks is extended to Dr. Derek Wong, whose insightful suggestions have helped to refine this work.

Collaborations with researchers at IBM T. J. Watson Research Center proved instrumental in acquiring simulation results. The valuable efforts of Dr. Sandra J. Baylor, Dr. Deepu Rathi, and Dr. Christos Georgiou which made possible portions of this work is especially appreciated. This research was further supported by Hughes Research Laboratories, General Electric, GTE, and NASA under contract NAGW 419.

Heartfelt gratitude is extended toward my family, - especially my loving parents, Dr. Harold E. Pinkston, Sr., and Dr. Margaret C. Pinkston. Their continued encouragement and answered prayers have allowed me to maintain resilience through all life's challenges. This dissertation is dedicated to them - I love you all!

Above all, I thank GOD our Father through the Lord Jesus Christ, who is the Light of the world.
Let your Light so shine before men,
that they may see your good works,
and GLORI-fy your Father which is in heaven.

- Matthew 5:16
Contents

Abstract iv

Acknowledgements v

1 Introduction 1
  1.1 The Problem .......................................................... 3
  1.2 Related Work .......................................................... 5
  1.3 Methodology ............................................................ 8
  1.4 Research Contributions .......................................... 10
  1.5 Dissertation Outline ............................................. 11

2 Optics For Computers—A Survey 12
  2.1 Optics for General-purpose Processing .................. 12
  2.2 Optics for Special-purpose Processing .................. 14
  2.3 Optics for Computer Interconnects ....................... 15
    2.3.1 Electronic Interconnect Limitations ................. 16
    2.3.2 Advantages of Optical Interconnects ............... 18
  2.4 Multiprocessor Interconnect Networks
    and Proposed Optical Implementations ................... 19
    2.4.1 A Classification of Multiprocessor Interconnects .. 20
    2.4.2 Crossbar Network ....................................... 22
    2.4.3 Multistage Networks ................................... 25
    2.4.4 Multi-hop Networks .................................... 30
    2.4.5 Single-hop Networks .................................. 34
5.1.2 Reconstructing the Recorded Interconnect Patterns ................ 100
5.1.3 Hologram Encoding .............................................................................. 104
5.2 Optical Bus and Hypercube Implementation ............................... 106
5.3 Feasibility of the Proposed Implementation ......................... 114
  5.3.1 Optical Power Budget .............................................................. 115
  5.3.2 Optical Geometry Constraints ................................................ 117
5.4 Summary ......................................................................................................... 124

6 Performance Analysis .......................................................... 126
  6.1 Trace Collection ............................................................................... 126
  6.2 The GLORI Simulator ....................................................................... 129
  6.3 Performance Results ........................................................................... 130
    6.3.1 Measuring Performance .............................................................. 132
    6.3.2 FFT Benchmark Results .......................................................... 133
    6.3.3 EIG Benchmark Results .......................................................... 143
  6.4 Interpretation of the Results .......................................................... 152
  6.5 Summary ......................................................................................................... 157

7 Summary, Conclusions, and Future Research ................................ 159
  7.1 Summary ............................................................................................... 159
  7.2 Conclusions ............................................................................................ 163
  7.3 Future Research ....................................................................................... 163

Bibliography ................................................................................................. 165

Bibliography ................................................................................................. 177

ix
List of Tables

2.1 Comparison of Interconnect Network Parameters ........................................ 36
4.1 Performance Comparison of Communication Models .................................. 86
4.2 Imaging Optics Required of Communication Models .................................. 93
5.1 Optical Component Parameters ............................................................... 115
6.1 FFT Network Loading Statistics for Static GLORI ................................. 134
6.2 FFT Network Loading Statistics for Dynamic GLORI ................................. 134
6.3 Average Number of Cycles Processors Stall per Global Network Read Ref .......................................................... 139
6.4 Average Global Network Read Stalls per Processor (cycles) ................. 139
6.5 Average Number of Cycles Processors Stalled for Bus Grant per Ref .... 140
6.6 Total Bus Grant Stalls per Processor (cycles) ........................................ 140
6.7 Average Total Latency per Network Reference (cycles) ...................... 141
6.8 Execution Time in Thousands of Cycles ................................................ 143
6.9 Estimated Speed-Up (1 Processor = 2,850,000 cycles) ......................... 143
6.10 EIG Referencing Statistics for Static GLORI (thousands of references) 146
6.11 EIG Referencing Statistics Dynamic GLORI (thousands of references) 146
6.12 Average Number of Cycles Processors Stall per Global Network Read Ref .......................................................... 148
6.13 Average Global Network Read Stalls per Processor (thousands of cycles) 148
6.14 Average Number of Cycles Processors Stalled for Bus Grant per Ref .... 149
6.15 Total Bus Grant Stalls per Processor (thousands of cycles) ................. 149
6.16 Average Total Latency per Network Reference (cycles) ...................... 149

x

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
6.17 Execution Time in Thousands of Cycles .............................................. 151
6.18 Estimated Speed-Up (1 Processor = 16,800,000) ................................. 151
List of Figures

1.1 Illustration of speed-up as a function of system size. ............... 3
1.2 Critical problem: communication complexity and latency. .......... 4
1.3 Reducing communication latency with cached architecture. ........ 5
1.4 Reducing communication latency with combining network. .......... 6
1.5 Reducing communication latency with hierarchical network. ....... 7
1.6 System level interconnect design. ........................................... 8
1.7 The Global-Local Optical Reconfigurable Interconnect Strategy. ... 9

2.1 Comparison of electronic bus and processor performances. ......... 16
2.2 Conceptual illustration of a crossbar network. .......................... 22
2.3 Optical crossbar from matrix-vector multiplication. .................. 24
2.4 Omega network. .............................................................. 27
2.5 Baseline network. ............................................................. 28
2.6 Benes network. ............................................................... 28
2.7 SW-Banyan network. ......................................................... 29
2.8 Hypercube topology. ......................................................... 31
2.9 Tree topology. ............................................................... 32
2.10 Mesh topology. ............................................................. 33
2.11 Utilization of the third dimension by optical multi-hop networks. .. 34
2.12 Shared bus network. ....................................................... 35
2.13 Fully-connected Network. .................................................. 35

3.1 Circumstances under which multiprocessor caching is effective. .... 41
3.2 Spatial and temporal locality at the processor level. .................. 43
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3</td>
<td>Hop Latency as a function of the number of hops through the network</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td>for various FFT network loads with and without combining.</td>
<td></td>
</tr>
<tr>
<td>6.4</td>
<td>Processor Stalls (for Read Refs) as a function of the number of hops</td>
<td>137</td>
</tr>
<tr>
<td></td>
<td>through the network for various FFT network loads with and without</td>
<td></td>
</tr>
<tr>
<td></td>
<td>combining.</td>
<td></td>
</tr>
<tr>
<td>6.5</td>
<td>Processor Stalls (for Read Refs) verses FFT network load (global refs)</td>
<td>138</td>
</tr>
<tr>
<td></td>
<td>for average hop distance of 1.7 with no combining.</td>
<td></td>
</tr>
<tr>
<td>6.6</td>
<td>(a) Average Latency per Ref and (b) Execution Time verses FFT</td>
<td>142</td>
</tr>
<tr>
<td></td>
<td>network load for average hop distance of 1.7 with no combining.</td>
<td></td>
</tr>
<tr>
<td>6.7</td>
<td>Hop Latency as a function of the number of hops through the network</td>
<td>144</td>
</tr>
<tr>
<td></td>
<td>for various EIG network loads with and without combining.</td>
<td></td>
</tr>
<tr>
<td>6.8</td>
<td>Processor Stalls (for Read Refs) as a function of the number of hops</td>
<td>145</td>
</tr>
<tr>
<td></td>
<td>through the network for various EIG network loads with and without</td>
<td></td>
</tr>
<tr>
<td></td>
<td>combining.</td>
<td></td>
</tr>
<tr>
<td>6.9</td>
<td>Processor Stalls (for Read Refs) verses EIG network load (global refs)</td>
<td>147</td>
</tr>
<tr>
<td></td>
<td>for average hop distance of 1.7 with no combining.</td>
<td></td>
</tr>
<tr>
<td>6.10</td>
<td>(a) Average Latency per Ref and (b) Execution Time verses EIG network</td>
<td>150</td>
</tr>
<tr>
<td></td>
<td>load for average hop distance of 1.7 with no combining.</td>
<td></td>
</tr>
<tr>
<td>6.11</td>
<td>Estimated Speed-Up for (a) SData only and (b) SData &amp; 3.3% Instr</td>
<td>153</td>
</tr>
<tr>
<td></td>
<td>Refs as global network refs (FFT).</td>
<td></td>
</tr>
<tr>
<td>6.12</td>
<td>Execution Time for (a) SData only and (b) SData &amp; 3.3% Instr Refs</td>
<td>154</td>
</tr>
<tr>
<td></td>
<td>as global network refs (FFT).</td>
<td></td>
</tr>
<tr>
<td>6.13</td>
<td>Estimated Speed-Up for (a) SData only and (b) SData &amp; 3.3% Instr</td>
<td>156</td>
</tr>
<tr>
<td></td>
<td>Refs as global network refs (EIG).</td>
<td></td>
</tr>
<tr>
<td>6.14</td>
<td>Execution Time for (a) SData only and (b) SData &amp; 3.3% Instr Refs</td>
<td>158</td>
</tr>
<tr>
<td></td>
<td>as global network refs (EIG).</td>
<td></td>
</tr>
<tr>
<td>7.1</td>
<td>Optical interconnect reconfiguration.</td>
<td>164</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Multiprocessing, or parallel processing is the technique of partitioning a process into a number of independent subtasks\(^1\) and executing them concurrently across a number of cooperating processor units. This concept has worked its way into both the architectural and programming "dimensions" of computing, thus causing the advent of parallel systems capable of simultaneous execution of many processes. Parallel processing, therefore, paves the way for large and complex algorithms to be systematically reduced to smaller, simpler forms where, ideally, the execution time decreases in proportion to the number of additional processing resources integrated into the system.

This idea of performing computations in parallel has evolved over the years. It was first incorporated into uniprocessors in the form of pipelining. Pipelining consists of parsing the processing of machine instructions into relatively independent micro-operations, each of which is executed separately by functional units (e.g., instruction fetch unit, instruction decode unit, execute unit, etc.) in discrete amounts of time called processor cycles. By skewing commencement of consecutive instructions by a cycle, micro-operations for several instructions overlapped in time are executed in parallel. As a result, some operation on every instruction in the pipeline is performed concurrently every cycle, and, ideally, one instruction is executed after each processor

\(^1\)Independence between subtasks is achieved when mutual exclusion exists between their critical sections such that the subtasks' results are not influenced by execution order.
cycle once the pipeline is full. Although pipelining can significantly improve machine performance by exploiting parallelism at the micro-operation level, this technique is limited in its potential to exploit parallelism at the higher program level.

Parallelism at the program level is available in many applications. Scientific/engineering (S/E) applications, for instance, generally have algorithms with high parallel content wherein hundreds of program threads can be executed concurrently [1, 2]. These highly parallel algorithms execute on pipelined uniprocessors as non-overlapping sequential processes in a single-instruction stream, single-data stream (SISD) manner. In uniprocessors, parallelism is not exploited at the higher system level. However, multiprocessor computer architectures offer multiple-instruction stream, multiple-data stream (MIMD) processing capabilities that exploit parallelism at the system level.\(^2\) This serves to increase the overall potential computing power and speed of the system.

With proper parallelization of algorithms, significant multiprocessor speed-up is possible relative to a uniprocessor. Speed-up is a metric used to gauge system performance (see Figure 1.1). Because most algorithms contain serial sections in which only one process exists instead of \(N\) processes, speed-up is generally sub-linear. Hence, efficient parallelization of the algorithm is essential to achieving good algorithmic speed-up [4] that is near the ideal speed-up which is linear.

Contestation for shared resources in a multiprocessor system, such as main memory and network links, can also limit the effective amount of computation that can proceed in parallel. This is measured by attainable speed-up as illustrated in Figure 1.1. Processors generate reference streams or traffic to memory. Traffic patterns may vary independently from processor to processor. Regardless of technological improvements in processor performance, the overall system speed is limited by the latency of the communication mechanisms. Efficient communication management ensures minimal traffic contention for the interconnect network connecting the various processors and memory units so that attainable speed-up approaches algorithmic speed-up.

\(^2\)According to Flynn's classification [3], multiprocessing paradigms also include MISD and SIMD in addition to MIMD processing.
Hence, the concurrent processing capability of multiprocessors is limited not only by inefficient parallelization of algorithms but also by the inability of the underlying hardware to efficiently manage the communication patterns of the parallel algorithms. Challenges facing multiprocessor designers can therefore be summarized as the facilitation of \textit{parallelism} and \textit{high performance communication} by the software and the hardware architecture.

\section{The Problem}

A critical problem in multiprocessor systems is the communication complexity and latency arising from multiple processors accessing a shared memory through a common interconnect fabric (see Figure 1.2).

A major portion of communication latency is associated with the interconnection network where, for large systems, routing latency can be orders of magnitude larger than memory access latency. Many routing problems are attributed to the network being fixed and not able to adapt to the memory reference patterns of the processors. Moreover, the network is often bandwidth or throughput limited and quickly saturates as processor references increase.
CHAPTER 1. INTRODUCTION

Figure 1.2: Critical problem: communication complexity and latency.

With advancements in processor design and technology, processors are becoming faster, requiring lower processor to memory latency time for computation-limited processing. Electronic bus technology lags behind processor technology and is becoming a bottleneck for realizing increased multiprocessor performance [5]. To keep stride with processor advancements, the interconnect network must incorporate more advanced designs and technologies.

Computer designers are striving towards an ideal interconnect; one composed of low cost, independent, robust processors and memory units wherein absolute minimal unit delay time (for constant length messages) is incurred by processors when accessing random memory units. Some desirable attributes include:

- minimal traffic latencies
  - high signal propagation speed
  - high bandwidth
  - large degree of parallelism (channel width and number of channels)
  - low switching time

- large degree of interconnect freedom

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 1.3: Reducing communication latency with cached architecture.

- high degree of configurability
- high degree of connectability
- low complexity and routing difficulty

• minimal implementation cost
  - low area (size)
  - low power
  - dense functionality
  - high fault tolerance.

The interconnect strategy proposed in this thesis has many of these attributes. Our strategy is a scalable means of reducing network latency while also lowering the interconnect area/volume.

1.2 Related Work

Many methods have been suggested for reducing communication latency in multiprocessors. One method proposed in the Stanford Dash [6] and MIT Alewife [7] machines
is to cache references such that once a memory block is accessed by a processor, it is buffered locally at the processor so that subsequent references to that block "hit" in the faster cache memory, thus circumventing the network and its associated latencies. Figure 1.3 illustrates a cached multiprocessor architecture.

In order for caching to work well, the memory blocks provided to processors' caches should be the most up-to-date versions. If a block is modified by any of the processors, other copies in main memory and other processor caches must be kept updated or coherent. A problem with multiprocessor caching is that applying this technique to all reference types, particularly shared data references, may provide only limited performance gains because of a phenomenon unique to multiprocessors called cluster locality, explained in Chapter 3.

Another method of reducing communication latency proposed in the NYU Ultracomputer [8] and IBM RP3 [9] machines is to combine or merge references together within the interconnect network that are destined to the same memory location. Figure 1.4 illustrates a combining network architecture. Combining decreases network congestion and "hot-spotting" [10, 11] activity to reduce the overall latency. A problem with this method is that inadequate buffering at each switch node within each
stage of the network can result in tree-saturation and network congestion. Unfortunately, this buffering grows in proportion to the system size, resulting in poor scaling.

Yet another latency reduction method is to statically cluster together processors in a hierarchical interconnect topology [12, 13, 14, 15, 16]. This enables the capture of static referencing locality behavior within local communication spaces to reduce the global network congestion and latency. Figure 1.5 illustrates a hierarchical network architecture. Though useful for parallel algorithms exhibiting static locality, this method is ineffective in capturing dynamically varying memory reference patterns unless used in conjunction with page and process migration.

It is believed by many that optically implemented interconnection networks have significant advantages over electronics. There is a trend to replace slow and bulky electrical wires with faster, denser fiber and free-space optical links. For instance, Thinking Machines and Honeywell have collaborated to demonstrate a 512 to 1 increase in interconnect density in a 16K Connection Machine module with 1.1 Gbps fiber optic links [17]. NTT has demonstrated a 32 processor system employing manually reconfigurable board-to-board bulk free-space optical interconnects with COSINE-1 – essentially, optical plug-boards [18]. COSINE-2 [19] is a more recent version which
uses liquid crystal switch elements for reconfiguration. However, the benefits of optics are not well used with bottom-up approaches; instead, a *top-down* or system level design which equally emphasizes multiprocessor program behavior, architecture, and technology can better incorporate the features of optics into the interconnect architecture.

### 1.3 Methodology

Researchers have proposed optical interconnects ranging from shared buses to hypercubes to multistage to crossbar topologies (see Chapter 2). An important factor often neglected in designing and evaluating the performance of these networks is program behavior's system-level influence on interconnect architecture: though a topology may be implementable in optics in lieu of electronics, it does not necessarily follow that it provides superior performance.

In our approach, behavioral, topological, and technological aspects of multiprocessors are used to arrive at an interconnect *strategy* — more than a network — that
selectively takes full advantage of each aspect. In some way, each aspect affects the others; for instance, the communication behavior of parallel programs determines the type and amount of connectivity, bandwidth requirements, etc., of the interconnect network which, in turn, affects the choice of technology that is best used for its implementation. Key characteristics and features of each aspect are shown in Figure 1.6. Our strategy reduces interconnect latency by dynamically localizing referencing activity at the cluster level. The GLORI strategy developed in this dissertation stands for Global-Local Optical Reconfigurable Interconnect, and it capitalizes on behavior that makes shared data caching problematic — specifically, cluster locality. A conceptual schematic of GLORI is shown in Figure 1.7.

With GLORI's interconnect hierarchy and reconfiguration, referencing activity can be captured locally. Frequently communicating processors and memory blocks can be associated into clusters, where the majority of communication traffic is processed locally. Upon large fluctuations in the memory referencing pattern of an algorithm, processor-memory association can change dynamically and form new clusters by optically reconfiguring the interconnect to re-localize memory references. Furthermore, clustering enables combining at the local interconnect of references destined to
remote locations. Hence, GLORI's architecture is designed to augment already existing latency reduction methods, whether they be caching, combining, interconnect hierarchy, page or process migration, wormhole routing, etc.

1.4 Research Contributions

An optical interconnect network for general purpose multiprocessing systems with low latency communication is designed, its feasibility evaluated, and its performance assessed via simulation. Rather than a bottom-up approach, focusing primarily on low-level concerns such as optical versus electronic technology only or topological issues only (i.e., crossbar versus bus), a top-down approach to interconnect design is taken. The importance of technological, topological, and behavioral considerations of multiprocessor systems are all considered in a strategy for interconnect design.

In our top-down procedure, observed parallel program behavior is used in the development of the interconnect architecture. Among other things, features of the interconnect include a hierarchical topology, clustering, limited combining, minimal delay routing protocols, high bandwidth links, free-space broadcastability, and reconfigurability. The result is a low latency interconnect strategy which exploits low-level features of the technology and high-level characteristics of parallel programs.

The major contributions of this work are summarized below:

1. An interconnect strategy is proposed that takes into account multiprocessor communication behavior, interconnect architecture, and optical technology;

2. Multiprocessor spatial, temporal, and cluster locality are analyzed and characterized;

3. Locality behavior is verified as being exploitable by a reconfigurable system;

4. Key features of optics useful for multiprocessor interconnection are identified and integrated into an interconnect architecture.

5. A high performance reconfigurable shared bus-hypercube interconnect is designed using guided and free-space holographic optical components;
CHAPTER 1. INTRODUCTION

6. A static GLORI implementation of the above is shown to be feasible;

7. Through simulation techniques, GLORI is shown to be effective in reducing network communication latency; and

8. GLORI is shown to be attractive on a cost/performance basis.

1.5 Dissertation Outline

The organization of this dissertation includes five main parts. Chapter 2 is a survey which lays the groundwork for integrating optics into multiprocessor computers.

In Chapter 3, multiprocessor communication behavior is analyzed and characterized. This chapter discusses behavior that can be exploited by the interconnect architecture to lower communication latency.

The architecture of the GLORI strategy is described in Chapter 4, and a feasibility analysis of a static GLORI implementation based on holographic beam steering is presented in Chapter 5.

The fifth part of this dissertation, Chapter 6, evaluates the performance of GLORI in various operation modes and compares this with an ideal optical shared bus network. Various cost/performance tradeoffs are considered in this chapter.

Chapter 7 summarizes key results of this dissertation. It also provides conclusions and future research directions for this work.
Chapter 2

Optics For Computers—A Survey

There has been much recent effort directed towards understanding the feasibility of using optical technology in computer systems [20, 21]. Researchers are investigating possible areas of application for this relatively new computer technology ranging from multiprocessor interconnection to special-purpose processing to general-purpose processing. This chapter discusses the various ways optics is being considered for use in computer systems. Emphasis is placed on the ways in which optics is proposed for multiprocessor interconnection.

2.1 Optics for General-purpose Processing

While some researchers have been working towards all-optical digital computers [22, 23, 24, 25, 26], others are not optimistic about the practically of optics for general-purpose computation [27, 28, 29]. Traditionally, three major factors have contributed to the difficulty in achieving all-optical digital computers: power consumption/speed tradeoffs, device area, and signal compatibility problems. Each of these factors are briefly considered below.

Computers fundamentally consist of memory units, arithmetic-logic units, and input/output units. Each are composed of nonlinearly operating components that serve as logic gate devices and linearly operating components that serve as lines or channels for interconnecting the devices. Optical devices must effectively perform
both nonlinear and linear operations in order for computers to be entirely based on optical technology.

Photons naturally do not easily interact with one another. This characteristic gives optics strength as a communication (interconnect) technology for performing linear operations, but gives it weakness as a logic (switching) technology for performing nonlinear operations. In recent years, discoveries in the area of optical bistability have contributed to considerably better optical nonlinear components [30, 31, 32, 33]. However, the advantage of increased speed by optical switching devices over modern electronic devices is provided at the expense of increased power consumption that can result in thermal problems. Although a digital optical computer has the potential to keep up and possibly be faster than an electronic computer, it would have to use "power-saving strategies," as suggested by Huang [22] in order to operate at comparable power levels. This could decrease its performance to the extent that there may be little advantage in an all-optical approach.

On the basis of area and device integration, there is no advantage in an all-optical approach as well. According to Smith [28], the minimum size of an optical switching element that obtains the largest light intensity for a given input power cannot be reduced below a volume of about \((\lambda/n)^3\), where \(n\) is the index of refraction of the material, unless lossy metallic structures are used. The reason for this limitation is that light cannot be focused to a cross-sectional area smaller than the square of its wavelength due to diffraction effects. Hence, a diffraction-limited optical device would be large, e.g., \(\approx (1\mu m)^3\), compared to electronic devices, e.g., \(<(.5\mu m)^3\).

Another problem with many optical logic gates is that the inputs and outputs of the gates do not share the same phase-intensity and/or wavelength relationship. This results in the inability to directly cascade optical gates by using the output of one as the input of another. Many other factors render the realization of an all-optical digital computer speculative. It has been concluded by some that no optical logic component with satisfactory speed, size, energy consumption, and heat production is feasible, mainly because every known optical nonlinearity is a very weak effect [29].
2.2 Optics for Special-purpose Processing

The strong nonlinear effects required by digital optical processing is not needed with analog optical processing. Thus, hybrid electronic-optical computers that appropriately use digital and analog processing resources are more promising. Those operations that execute orders of magnitude faster using analog optics can be assigned to the optical processor unit. Alternatively, those operations that are inherently more suited to digital electronic processing can be executed by an electronic unit. Accordingly, optical and electronic components within a hybrid system can be integrated in such a way that their individual performances enhance rather than degrade the overall system performance.

Computer vision is a special-purpose processing application particularly attractive for optics. Image input data for vision systems is usually in analog optical form so that the pre-processing step of image transformation is not needed by an optical processor. Computations are usually carried out in at least two dimensions (2-D) which is easily supported by optics. Many vision tasks are concerned with recognizing known objects in a scene cluttered with other objects. This is simply done with optics by performing a 2-D correlation using matched-filtering. Computationally intensive operations such as these requiring minimal logic evaluation are the tasks at which optical processing excels. Moreover, optics inherently has the speed and parallelism advantage that allows many millions of analog computations to proceed simultaneously at the speed of light.

Studies performed at Hughes Research Laboratories [34, 35, 36, 37] have assessed the performance of a realizable optical processor for executing computationally intensive image/vision operations. The assessment was made relative to the performances of digital signal processors (DSPs) and multiprocessor systems which operate in the realm of electronic processing. Four major conclusions were realized.

1. By virtue of optics’ high throughput capabilities, i.e., $O(10^5) \text{ MOPS}$, significant speed performance enhancement can be gained using optics to process computationally complex operations such as 3-D matrix multiplication and correlations.

2. Because of its analog nature, optics suffers a distinct performance disadvantage
in terms of the precision with which it can manipulate data.

3. Optics in comparison to electronics supports a small repertoire of processing functions with which to efficiently solve image/vision operations (small degree of programmability and versatility).

4. Electronic DSP technological advances are shrinking the niche of possible applications which would benefit most from optical processing and will continue to do so unless advances in optical technology — specifically, spatial light modulators — catch up and maintain parity.

These conclusions suggest that the realization of hybrid electronic-optical computers that perform not only functions oriented toward machine-vision but also more generic computationally intensive processing functions depends on the rate of advancements made in maturing optical technology. It is projected that, in the near term, optics will play a more prominent role in computer communication (interconnects) than in processing.

2.3 Optics for Computer Interconnects

The computing power of processor chips is increasing dramatically. The communication bandwidth provided by the interconnect must increase accordingly to not limit system performance.

The interconnect’s technology and topology have direct impact on the degree to which high performance communication is supported. Interconnects based on optical technology not only differ from those based on electronic technology in their higher bandwidth potential but also in their ability to support novel, more flexible interconnect networks. Hence, optical interconnects can directly influence system architecture as well as system performance.

Limits are being reached with electrical interconnects. Electronic interconnect technology at the system level is not advancing as rapidly as processor technology, using bus clock frequency as the figure of merit [38] as illustrated in Figure 2.1. The
following subsections examine more closely the limitations of electronic interconnects and the features of optical interconnects.

### 2.3.1 Electronic Interconnect Limitations

Due to advances in device scaling, computation speed is no longer limited by device switching time – which decreases with scaling – but by interconnect delay, $\tau$:

$$\tau = \max[R \times C, \frac{c \times l}{\sqrt{\mu_r \varepsilon_r}}],$$  \hspace{1cm} (2.1)$$

where $R$ is resistance, $C$ is capacitance, $c$ is the speed of light, $l$ is length of the line, $\mu_r$ is permeability of the line, and $\varepsilon_r$ is permittivity of the line. Charging delay (the first term) is constant through scaling since the product of the interconnect resistance and capacitance remains the same. Signal propagation delay (the second term) is approximately 50% the speed of light, given VLSI parameters. Because electronic interconnects are essentially planar, very long path lengths which lead to significant propagation delays can result from complications in embedding wires crossing a given bisection. If the maximum interconnect delay were less than or equal
to the device switching delay, only then could electronic computers fully exploit the speed of gallium-arsenide (GaAs) and emitter-coupled logic (ECL) components.

Although scaling of VLSI circuits is good for increasing switching speed, it produces a negative effect on signal integrity, namely electromigration. Electromigration produces a movement of conductor atoms under the influence of electron bombardment, resulting ultimately in the breaking of conductor lines [27]. The current density, \( J = \sigma \times E \), in metal-oxide semiconductor (MOS) devices increases in proportion to scaling, which leads to greater electromigration effects. Undesired traits of electronic communication such as crosstalk, electromagnetic interference (EMI), and inductance also have a detrimental effect on signal bandwidth. When electronic signal media are spaced too closely together, electrons interact through their electric charges and magnetic fields. This property of electrons gives them strength as building blocks for use as a basis for switching currents and voltages. Unfortunately, however, it becomes a weakness for electronics for communication over distances through metallic media.

Another potential problem with electronic interconnects is signal skew. This problem arises when a signal originating at a source is distributed globally over varying distances to multiple destinations such that it does not arrive at all destinations simultaneously, e.g., clock skew. Signal skew also arises when signals originating at multiple sources are channeled over varying distances to one destination such that not all arrive at the same time, e.g., circuit races. Skew exists because of the previously mentioned interconnect delay created by resistance and capacitance in the communication medium. This skew of signal arrival can make an integrated circuit gate deliver an erroneous output. Further, clock skew places a constraint on the frequency of clock pulses; the faster the cycle time in the electronic computer, the less clock skew can be tolerated.

Another limitation imposed by electronic interconnects is external chip I/O bandwidth. Depending on the level of gate complexity, a chip can be wiring dominated and have a severe "pin-out" problem [39]. Since chips typically have 100\( \mu \)m center-to-center spacing for bond pads at the periphery, the interconnects for chip-to-chip communications may themselves place a constraint on the total chip size. It is reported that 10% to 20% of chip area typically is used for external communication.
[27]. If these constraints were not imposed, a chip could have more gates available for logic circuitry in the same amount of area, thereby giving rise to more powerful, robust chips. Silicon circuit boards [40] can provide only limited relief, since electronic interconnects require ground lines which further consume chip area.

In summary, cost/performance factors affected by electronic interconnects include propagation speed, signal bandwidth, signal/data skew, and chip area and bandwidth limitations. These factors limit the processing power of computer systems. These problems are not unique to electronic interconnects; however, in many respects, optical interconnects are not limited in the same way.

### 2.3.2 Advantages of Optical Interconnects

There are many advantages in using optics as an interconnect technology. Photons, which do not carry charge, do not naturally interact. Optical signals can propagate through one another in separate or parallel channels free from EMI and crosstalk effects. This allows space-invariant imaging and wavelength multiplexing for bit-parallel transmission over a common medium. Moreover, photons have a much greater information-carrying capacity (bandwidth) than electrons, and ground lines are not needed to reference signal levels.

Photons need not be confined to a waveguiding structure but can travel freely through space. This gives rise to 3-dimensional broadcast systems which may result in parallel processing architectures that can manipulate data in 2-D bit planes as well as 1-D bit streams [20]. Another advantage lies in optics' potential to achieve reconfigurability of interconnect patterns. Ferroelectric liquid crystal spatial light modulators can dynamically reconfigure thousands of resolvable spots in parallel in tens of microseconds [41, 42]. Photorefractive materials allow free carriers to be mobilized when the material is exposed to light. Hence, holograms made out of photorefractive material are reprogrammable [43]. Gallium-arsenide photorefractive materials can maintain patterns on the order of nanoseconds so that real-time programmability is theoretically possible, though power consuming.

These features of optical interconnects are particularly useful for multiprocessor interconnects. The high space-bandwidth product (SBWP) of optics facilitates the
transport of large amounts of parallel data quickly, a property useful for operations such as cache block transfers, page migration, process migration, context switching, and I/O. Because of its high-speed broadcast ability, optics is useful for operations such as cache coherence snooping and invalidations, MIMD synchronization locks, SIMD instruction broadcasting, and clock distribution. Finally, optics' reconfigurability is useful in exploiting locality at several processing levels.

At the job level, for instance in a multi-user environment, a multiprocessor system configuration can be selectively partitioned into optimally interconnected subsystems as the user mix changes. At the task level, static cluster locality in such applications as circuit layout and simulation can be captured for multiple simulation runs of each new simulation. At the program level, optical reconfiguration allows the network to adapt to dynamically varying cluster locality after block-level shifts in reference locality during execution of an algorithm. Because of optic's switching delay, optical reconfiguration extended to the reference level of a system may not be feasible.

In summary, optics shows promise for interconnecting functional units where large amounts of data requiring high bandwidth are communicated at the inter-chip, inter-board, and inter-machine level. Not all interconnect problems are associated with implementation technology. It is important to separate the technology limitations from topology limitations. Some limitations may not be easily distinguishable. The next section examines interconnect limitations as constrained by various multiprocessor interconnect topologies.

2.4 Multiprocessor Interconnect Networks and Proposed Optical Implementations

The next section gives a general classification of interconnect topologies and defines parameters useful in characterizing them. This is followed by sections which discuss four major classes of multiprocessor interconnects and proposed optical implementations. From this study, we narrow down candidate network topologies for our GLORI system based on potential performance merit.
2.4.1 A Classification of Multiprocessor Interconnects

Network latency is the time it takes for a complete message to transverse the interconnect from input node to output node. Network latency, $t_{\text{network}}$, includes but is not limited to the following latency components:

$$t_{\text{network}} = t_{\text{prop}} + t_{\text{switch}} + t_{\text{buffer}} + (t_{\text{transceive}} \times \frac{L_{\text{message}}}{B}).$$  

(2.2)

In the above equation, $t_{\text{prop}}$ is the time to propagate a message across a network link, $t_{\text{switch}}$ is the time to establish (switch into) the proper network configuration, $t_{\text{buffer}}$ is the time to gain access to the desired network link (the time the message spends buffered before transmission), $t_{\text{transceive}}$ is the time delay required between each bit's transmission, $L_{\text{message}}$ is the length of a message, and $B$ is the link width (number of bits per channel).

Channels are defined to be the physical communication links interconnecting nodes of a network. The distance between two nodes is the smallest number of link traversals incurred by a message that is routed from the source node to the destination node. The diameter of a network is the largest distance between any two nodes along the shortest path. The degree of a network is the maximum number of edges incident on the vertices of the interconnect graph or fan-out of a node.

Multiprocessor interconnects are generally classified as direct or indirect networks. Direct networks are static: only one configuration of the topological structure supports all communication requests through address decoding and routing mechanisms. The main disadvantage of direct networks is that they do not scale well either because of too little bandwidth (e.g., shared bus) or too high demand of interconnect resources (e.g., fully-connected network). Their advantage, however, is that messages incur single-hop latency once network access is granted.

Indirect networks can be static or dynamic. In dynamic form, these networks are composed of fixed communication links which fan into and out of stages of switch elements that switch their inputs to proper output links, thereby setting up message paths dynamically. In static form, these networks propagate messages through intermediate nodes that broadcast messages to neighboring nodes which independently
determine routing status. A disadvantage of indirect networks is the nonunit number of network hops messages encounter. An advantage of these networks is the reduced number of interconnect resources required. Multistage networks (e.g., Omega, Baseline, Banyan and Benes) are classified as dynamic-indirect. Multi-hop networks which use switch elements at intermediate nodes to direct messages are classified as dynamic-indirect networks. Those that employ broadcasting are classified as static-indirect networks. The hypercube and mesh are examples of multi-hop networks that can be implemented in static or dynamic forms.

In addition to direct and indirect, networks can also be classified as reconfigurable. Reconfigurable networks are inherently dynamic, capable of translating one functional unit interconnect configuration into another. Processor and memory units in an interconnect graph can be relocated either logically or physically so as to assume various positions within the graph and share a direct communication link between them. Reconfiguration allows the network to have a great amount of flexibility and adaptability. However, present implementations of reconfigurable networks do not scale well because of a high demand on dynamic interconnect resources needed to provide flexibility.

The frequency of network reconfiguration is determined by the processing level at which reconfiguration is applied in the system. We develop terms to distinguish between the two types of network reconfiguration that can occur at the program level. Block-frequent reconfiguration allows a network to switch at the rate of change in execution of process blocks, e.g., after each loop termination of a process. Reference-frequent reconfiguration allows a network to switch at the rate of processor requests, e.g., after each network transaction by a processor. Block-frequent reconfiguration does not rely on high-speed switching capability since network configurations remain static for many execution cycles. However, fast switching capability is required by a network based on reference-frequent reconfiguration.
2.4.2 Crossbar Network

The crossbar network\(^1\) shown conceptually in Figure 2.2 is perhaps the most general-purpose network topology. It is a circuit switched network capable of establishing paths between any two row and column nodes via reference-frequent reconfiguration. A nonblocking network, the crossbar can realize any permutation of inputs to outputs while not interfering with or disconnecting other established paths. Broadcasting of one or more inputs to several outputs can be done but, if not well coordinated, contention at the output nodes by other competing input nodes can occur.

Routing can be in the form of distributed control or centralized control. Distributed routing requires each switch site to independently translate itself into the proper switch setting based on message header information. Centralized routing relies on a central controller to set switch sites appropriately to establish paths based on knowledge of the state of switches throughout the system. This can be implemented by reconfigurable transform matrices or table look-up for various input-output permutations.

As shown in the Figure 2.2, signals propagate through an interconnect composed

\(^1\)Example systems using electronic crossbars are the S-1 (16 x 16) by Livermore Labs, C.mmp (50 x 50) by Carnegie Mellon, IBM's LSM (64 x 64) and IBM's YSE (256 x 256).
of a number of switch elements (SEs). Switch element latency includes control logic
gate delays, switch response time, transmission time, propagation delay through the
path medium, etc. These delays can be lumped into one switch latency parameter,
\( t_{\text{switch}} \), which, for electronic technology, is on the order of nanoseconds for \( 2 \times 2 \)
switch elements. Decreasing \( t_{\text{switch}} \) along with increasing the number of bits per line
has a direct influence on increasing the interconnect's bandwidth. Advancements
in electronics have led to the development of low-power monolithic GaAs crossbar
switching devices ranging in size from \( 4 \times 4 \) to \( 16 \times 8 \) which operate above \( 1 GHz \)
[44, 45, 46].

Advantages of the crossbar network are that it has unit diameter, unit degree,
and is conservative in the number of channels and input ports per node needed for
implementation (\( N \) and 1 respectively). A major drawback of crossbar networks
is the cost of a large number of switch elements, which makes the realization of
such electronic crossbars impractical. Crossbars require on the order of \( N^2 \) switch
elements, all of which must be capable of switching at speeds which match the request
rates of the sender and receiver (reference-frequent reconfiguration). Owing to the
large number of switch sites, the routing control of large crossbars can add to chip
complexity. For example, a \( 32 \times 32 \) 8 bit crossbar chip without routing arbitration
circuitry requires \( \approx 1000 \) pins [47]. If control lines were to be integrated on-chip along
with data lines to accomplish routing, chip I/O pin count would be prohibitively large.

Even with the possibility of fabricating many switch elements onto chips with
high I/O bandwidth, the integration of these chips into a crossbar switch is difficult
with planar technology. Cascading techniques for making \( N \times N \times B \) crossbars out
of \( n \times n \times b \) pin-out limited chips require \( N' \) chips, where

\[
N' = \frac{N \times N \times B}{n \times n \times b} \quad (2.3)
\]

and \( B \) and \( b \) are the data path widths in number of bits per channel. For instance,
for \( N = 1024 \), \( B = 64 \), \( n = 32 \), and \( b = 8 \), the total number of crossbar switch chips
required is \( 8K \). The larger the number of chips, the larger the number of external
wires and, hence, the more difficult to implement (and the more likely the performance

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Many optical crossbar network implementations have been proposed [48, 49, 50, 51, 52]. For example, matrix-vector and matrix-matrix multiplication techniques can be used to implement optical crossbars [53, 54, 55]. An illustration is shown in Figure 2.3. The control functions to set up processor to memory connections are represented by matrix $C$, the processor nodes are represented by vector $P$, and the memory nodes are represented by vector $M$. Multiplication of the control matrix and processor vector yields the resultant memory vector, $M = C \times P$. Broadcasting from one processor to many memory modules is indicated by multiple ones in each column. The example in Figure 2.3 demonstrates matrix values for nonbroadcast one-to-one interconnection of functional units.

There are various means of transforming the matrix dynamically just as there are multiple ways to transmit, expand, and receive the data signals. For instance, a
laser diode array can transmit vector processor data via intensity modulation. These signals can be optically expanded with microlenses. An optical spatial light modulator (SLM) with programmable opaque and transparent regions can implement the dynamic crossbar matrix multiplier mask. The optical signals can again be expanded by microlenses and detected with photodiode optical detectors.

The problem with electronic crossbars is not necessarily the switching speed requirements but the integration and control costs for implementation. Many optical crossbars, although better at avoiding the integration problem, suffer in reference-frequent switching ability. Optical SLMs are relatively slow compared to electronics by approximately a factor of 100 to 1000 (assuming comparative power consumption). Thus, network latency for optical crossbars is often dominated by switch latency:

\[ t_{lat} \approx t_{switch}. \quad (2.4) \]

Moreover, low efficiencies can often result from large optical crossbar implementations based on matrix-vector multiplication. Maximally, only \(1/N^{th}\) of the input light intensity propagates through the SLM and other imaging devices to the output of the interconnect, resulting in limited system scalability.

### 2.4.3 Multistage Networks

Multistage networks\(^2\) are like the crossbar in terms of the number of channels and degree of input ports required, which are \(N\) and 1 respectively. They also rely on reference-frequent reconfigurability. However, they are more cost-effective in terms of the number of switch elements needed.

Any arbitrary permutation of network inputs to outputs can be attained by \(3 \log_2 N - 1\) stages of switch elements [56]. Thus, \((3N/2) \log_2 N - N/2\) switch elements (2 \(\times\) 2) are needed to totally interconnect \(N\) processors in a multistage network with 100% permutability, rather than \(N^2\) elements. This rearrangeable network has the drawback, however, of inevitably disconnecting, re-routing and re-connecting

---

\(^2\)Example systems using electronic multistage networks are IBM RP3x (64 nodes), Burroughs FMP (512 nodes), BBN Butterfly (256 nodes), etc.
existing paths in order to establish all possible configurations of the network. This drawback may not be severe if phased communication is used; here, communication among processors is allowed only during communication intervals, separate from path establishing intervals.

The routing control for multistage networks can be centralized or distributed as in the crossbar. In centralized control, paths are established via a controller which knows a priori the most efficient switch settings for given permutations. In distributed routing, paths are established by propagating message header information through the network which, at every stage, sorts destination addresses. The path over which the binary encoded destination values traverse in the sorting process becomes the processor-to-memory connection link for that particular reference. An alternative distributed routing control makes use of destination tags rather than addresses [56].

Specific types of multistage networks implemented electronically are briefly presented below. This is followed by a brief discussion of multistage networks proposed for optical implementation.

**Omega Network**

Not all systems require 100% interconnectivity between processors during every communication phase. Accordingly, the number of stages in those systems' interconnects can be reduced to as few as \( \log_2 N \), thus creating a blocking network in which not all inputs can directly connect to network outputs. The Omega network is of this type.

In the Omega network [57] (Figure 2.4), at least one input can communicate to any output, but paths from some inputs are blocked due to contention at the switch elements. The Burroughs FMP 512-processor machine [58], for example, used an Omega network with nine cascaded PSE stages wherein, on average, only 28% of all possible input to output permutations could be made successfully. Without the excess redundancy of \( 2 \log_2 N - 1 \) stages, the non-rearrangeable network blocked connections

---

3It should be noted, however, that this may not solve the problem if processors communicate at random times and are thus asynchronous.

4Conflicts can be resolved by allowing one reference to proceed while queuing or discarding the other reference for later retry.
so that not all permutations could be established during the communication phase.

**Baseline Network**

Functional differences between the Baseline network (Figure 2.5) and the Omega network are few. Both are blocking networks which use stages of the *perfect shuffle exchange* [59]. However, at each successive stage in the Baseline, the number of perfect shuffle networks double and their physical connection distances decrease by a factor of two (see figures). This localization of interconnect links eases the fabrication of the interconnect network.

**Benes Topology**

The Benes network [60] is an extension of the Baseline. It employs redundancy totalling \( \log_2 N - 1 \) additional stages of switches in order to achieve rearrangeability (Figure 2.6). Sequential routing and "loop" routing algorithms [56] are attractive alternative routing strategies used to establish circuit switched permutation paths within this network.
Figure 2.5: Baseline network.

Figure 2.6: Benes network.
Banyan Network

The Banyan network [61] (alias Butterfly or Crossover network) is a stand-out from the previous mentioned multistage networks based on the perfect shuffle exchange. The Banyan's topology is adapted from the butterfly concept of the fast Fourier transform (FFT) computation rather than the shuffle concept. As shown in Figure 2.7, the number of stages grows as $\log_2 N$ just as the other blocking networks.

Optical Multistage Networks

The topologies of multistage networks build upon regular structures that are duplicated at each stage in either reduced or nonreduced form. In addition, the connection graph of blocking networks having no redundant stages is a partial ordering in which there exists only one path from any input to any output [62]. With this path-uniqueness property, these networks can be reduced to simple tree structures which connect processors and memories at the root and leaf positions through a hierarchical tier of switch elements.

These characteristics of multistage networks, namely regularity and simplicity, make them amenable to optical implementation. In fact, many optical multistage networks have been proposed [63, 64, 65, 66, 67, 68, 69, 70, 71]. However, since multistage networks are based on reference-frequent reconfiguration, network performance
is limited by optical switching delay – which, presently, is significantly higher than electronic switching delay. Signals must minimally traverse a round-trip number of switch stages equal to $(3 \log_2 N - 1) \times 2$ in a rearrangeable multistage network. Because switch elements in each stage must switch in real-time, multistage networks, like crossbar networks, are relatively intolerant of optics’ slower switching time. Therefore, the interconnect latency is dominated by switch latency and also consist of propagation delay and buffering delay encountered at each stage for the round trip:

$$t_{lat} \approx t_{switch} + t_{prop} + t_{buffer}.$$ (2.5)

### 2.4.4 Multi-hop Networks

Multi-hop networks\(^5\) can establish communication paths without the need for active switch elements. In lieu of a switch network, processors and memory units are integrated within the topology of the interconnect such that references hop from node to node until the final destination is reached. By enabling nodes to broadcast to neighboring nodes (a variation of “flooding” distributed routing), no switch elements are needed at all. An added feature of multi-hop networks is that communication to local nodes in short range distance results in smaller latency than global communication to remote nodes over longer distances. Hence, hop penalty – which can be small if references are localized – replaces reference-frequent switching penalty, which is technology dependent.

Routing is easily accomplished by prefacing messages with header information containing the source and destination node values. Source node bits in the header are complemented appropriately at each intervening node until the header and destination node address match, thus signifying that routing is completed. Multiple paths exist through which a message can traverse to reach a particular destination. As a result, heavily loaded systems are not hampered by bottlenecks as much as unique-path, circuit switched networks previously considered.

\(^5\)Example systems using electronic multi-hop networks are Thinking Machines Connection Machine (64K nodes), NCube/10, iPSC Hypercube, Caltech Cosmic Cube, DAP, MPP, Stanford Dash, etc.
The following subsections briefly present generally used multi-hop networks and discusses alternative optical implementations.

**Hypercube Network**

A widely-used multi-hop network is the the hypercube (alias k-cube) [72]. It has $N$ nodes each requiring $\log_2 N = k$ connections as shown in Figure 2.8. Nodes can be processor-memory elements or routing elements that coordinate communication among clusters of processor-memory elements (as is the case with GLORI). The hypercube topology can graphically be reduced to a tree homomorphic image of the Banyan and Omega networks, where each node in the tree is a processor-memory element instead of a switch element [62].

In contrast to the topologies mentioned previously, where a unit degree of input ports is required, the hypercube experiences a nodal input port degree growth of $O(k)$ for $N = 2^k$ nodes. Thus, for systems where $k$ is large, the problem of connecting together $2^k$ processors in this configuration becomes formidable with planar electronics. Moreover, coordination and support of $O(2^k \times k)$ communication channels required by this topology is difficult. The round trip distance that messages incur varies from 2, for a nearest neighbor reference, to $2 \times k$, the network diameter, for a reference...
to the most remote hypercube node. This differs from multistage networks which minimally have a constant distance of $2 \times k$, for blocking non-rearrangeable networks, and maximally have distances of $2 \times (3 \times k - 1)$, for blocking rearrangeable networks. Thus, the hypercube has advantage over multistage networks in its ability to exploit locality of communication and, thereby, reduce network traffic.

**Tree Network**

Instead of graphically reducing the hypercube to tree form for certain algorithms, nodes can be arranged in a tree structure at the outset to lessen interconnect complexity [73]. In so doing, the number of links per node (degree) remains constant for a particular tree structure no matter how many nodes are in the network. This yields better scalability. As shown in Figure 2.9, a regular and fixed structure results wherein the same amount of distance separates all processors at a given tree level from ancestors of the same generation. This topological feature allows signal skew to be minimized in broadcast situations. However, locality in processing is needed for tree structures to work well, otherwise communication between remotes will inevitably result in bottlenecks toward the top of the tree.

**Mesh Network**

The mesh network is also a regular structure as Figure 2.10 illustrates. Applications that feature nearest-neighbor computations for local interprocessor communication map well to this network. Global communication requires $O(N)$ hops across the
diameter of the network, which can result in large routing latency. The degree of locality on the mesh typically is constant at four (for north, south, east and west connections to nodal neighbors). Hence, the mesh generally takes the form of a fixed grid pattern with wrap-around.

**Optical Multi-hop Networks**

The symmetric and regular structure of multi-hop networks simplifies their optical implementation. With the exception of the hypercube, planar electronic implementations of multi-hop networks which make use of the networks' inherent modularity are not difficult to build. The large bisection width of network links in the hypercube, however, is difficult to implement electronically. As alternatives, optical hypercube networks have been proposed [74, 75, 76, 77]. These make use of the third spatial dimension to establish hypercube links as illustrated in Figure 2.11. Bisection width problems are reduced in these implementations since optical paths can criss-cross and overlap while not interfering.

In modelling network latency, switch latency is not critical since optical multi-hop networks need not employ switches. Instead, interconnect latency is dominated by
Figure 2.11: Utilization of the third dimension by optical multi-hop networks.

the number of hops that messages must incur through the network:

\[ t_{\text{lat}} \approx t_{\text{prop}} + t_{\text{buffer}}. \]  

(2.6)

To reduce hop latency, nodes need not be connected in pure multi-hop form. For instance, hypercube based interconnects can combine with one or more other networks to form hybrid topologies. This idea is incorporated in the GLORI strategy.

### 2.4.5 Single-hop Networks

Single-hop networks\(^6\) also operate without the need for active switch elements. There are two major single-hop networks: the shared bus network and the fully-connected network.

The shared bus topology connects all \( N \) nodes in a system but allows only one node to access it at a time (Figure 2.12). It blocks all \( N - 1 \) other nodes vying for its use. Nodes arbitrate for bus access in some time-division multiplexing scheme. Attractive features of shared bus networks are simplicity, low implementation cost,

---

\(^6\) Example machines using electronic single-hop (bus) networks are Sequent, Encore Multimax, Alliant FX-8, etc.
and broadcastability – nodes can simultaneously receive the same information by snooping on the bus, i.e., for cache invalidation. A disadvantage of shared buses is low throughput. As a result of limited bus bandwidth, access latency can be large once the bus saturates under heavy network loading.

Fully-connected networks provide as much as $N$ times more throughput at the expense of $O(N^2)$ times more dedicated links as shown in Figure 2.13. The resulting lower access latency of this network comes at a significant cost. As in the hypercube network, the bisection width of this network makes electronic implementation infeasible for large networks.

A fully-connected network's large number of channels makes its optical implementation costly in terms of source and detector interconnect resources (although free-space interconnect links come "for free"). Hence, the only viable single-hop network thus far proposed for optical implementation is the shared bus [78, 79, 80, 81].
Optic's high bandwidth lessens the transceiving component of latency; however, because of the low network throughput, bus access time (buffering delay) still dominates interconnect latency:

\[ t_{\text{lat}} \approx t_{\text{buffer}}. \]  

(2.7)

### 2.5 Summary

Weak optical nonlinear effects make the realization of all-optical digital computers unlikely. Analog optical processing is useful for some special-purpose computing problems, but is not applicable to general-purpose computing. Optics has physical properties that are beneficial when used as an interconnect technology. Many features of optics are superior to electronics and are particularly useful in enhancing the performance of multiprocessor interconnects.

Table 2.1 summarizes the various costs associated with the optically implementable multiprocessor interconnects surveyed in this chapter. The more telling cost parameters are the number of switch elements and the number of bit-lines, which determine the number of sources, detectors, and concomitant optical links needed per node. The latency component which limits performance of the interconnects is also provided in the table.

<table>
<thead>
<tr>
<th>Cost Parameter</th>
<th>Direct Network</th>
<th>Reconfig</th>
<th>Indirect Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Full-connect</td>
<td>Bus</td>
<td>Crossbar</td>
</tr>
<tr>
<td>Channels</td>
<td>$1/2(N^2 - N)$</td>
<td>$1$</td>
<td>$N$</td>
</tr>
<tr>
<td>Bit-lines</td>
<td>$B/2(N^2 - N)$</td>
<td>$B$</td>
<td>$BN$</td>
</tr>
<tr>
<td>I/O Ports per Node</td>
<td>$B(N - 1)$</td>
<td>$B$</td>
<td>$B$</td>
</tr>
<tr>
<td>Switch Elements</td>
<td>$0$</td>
<td>$0$</td>
<td>$BN^2$</td>
</tr>
<tr>
<td>Network Latency</td>
<td>$t_{\text{transceive}}$</td>
<td>$t_{\text{buffer}}$</td>
<td>$t_{\text{switch}}$</td>
</tr>
</tbody>
</table>

The latency of fully-connected networks is the lowest, but implementation costs are highest – perhaps prohibitive for large systems. The cost of implementing optical
crossbar and multistage networks is lower than fully-connected networks, but higher than multi-hop and shared-bus networks. The performance of crossbar and multistage networks is limited by the switch latency of the optical technology used to implement switch elements, e.g., SLMs, directional couplers, etc. This is due to these networks' reliance on reference-frequent reconfiguration. Optical switching latency is orders of magnitude larger than typical cycle times of processors and memory units, making the use of these networks unattractive for multiprocessors.

The latency of multi-hop and shared bus networks is not a function of switch delay but, rather, link buffering delay. Unlike switch delay, which is constant, buffering delay decreases with network loading. The performance of optical multi-hop and shared bus networks is not as bound to technological limitations. Instead, the performance of these networks is affected by the traffic patterns of an algorithm.

Based on the above findings, single-hop and multi-hop networks are attractive topologies on which to build the GLORI strategy. The shared bus and hypercube networks are particularly attractive. The benefits of optical technology enhance the performance of these networks, and shortcomings of optics do little to degrade performance. Moreover, these networks have inherent qualities that can be used to exploit communication locality. Hence, understanding memory reference behavior of parallel programs executed on multiprocessors based on these networks is essential in order to minimize interconnect latency.
Chapter 3

Multiprocessor Communication Behavior

Analyzing the locality of memory reference patterns is a difficult task since not all applications behave alike. While extensive empirical data has been obtained concerning sequential reference patterns [82, 83], much less information is available about parallel program memory reference behavior.

In this chapter, multiprocessor locality principles are examined to determine shared data clustering potential in parallel programs. In addition to providing a basis for the feasibility of reconfigurable-hierarchical interconnect architectures such as GLORI, this work may give insight into how algorithms should be parallelized to maximize locality and clusterability.

As a case study, the communication behavior of three parallelized benchmark S/E applications written in FORTRAN were analyzed: a 2-dimensional fast Fourier transform (FFT) computation, a hydrodynamics (HYD) fluid flow problem, and a generalized symmetric eigenvalue (EIG) problem. These applications were chosen because they are widely used in the S/E community. They were executed in a simulated parallel processing environment consisting of $N = 64$ processor elements and the same number of memory elements each storing several memory blocks or pages. An ideal interconnect network served as the logical connection graph between arbitrary
processors and memory elements to make our memory reference behavior results independent of interconnect topology. As the primary interests of this study lie in the analysis of the traffic patterns of parallel S/E applications and not their performance, a simulation tool was developed that monitors and logs interconnect traffic. For each reference cycle of program execution, comprehensive reference information (such as to where and by whom references are made, etc.) was maintained and updated by the trace analyzer tool. Inferences on locality and cluster potential of these benchmark programs are based on the spatial and temporal reference information.\textsuperscript{1}

Although the scope of this study was limited to analyzing benchmark applications, some general insights into parallel S/E program behavior can be learned. This work is an initial step towards exploring many of the issues pertaining to multiprocessor locality and clustering. Some of the issues addressed include the following:

1. How correct are our intuitions about parallel program (multiprocessor) spatial and temporal locality phenomenon? How does this differ from sequential program (uniprocessor) locality?

2. How significant are the differences in memory reference behavior between various reference types (e.g., shared data, private data, instruction, fetch & add)? What impact do these differences have on system architecture (e.g., interconnect, cache, etc.)?

3. How is memory reference behavior affected by altering the application problem size or the algorithm?

4. Is there significant shared data clustering potential in parallel S/E applications? (Is dynamic clustering feasible? Static clustering?) How is the performance of hierarchical interconnects affected by degree of clustering potential?

5. How does clustering potential vary with problem size, page size, and/or chunk size? Does it depend on process scheduling?

\textsuperscript{1}A more detailed explanation of the S/E applications, multiprocessing environment, and method of analysis is given in chapter 5.
6. How should parallel algorithms be formulated to increase locality and cluster potential?

7. What is needed on the part of the user or optimizing compiler to fully exploit clustering potential?

The results indicate that shared data clustering potential is a function not only of the amount of locality inherent in the memory reference behavior of parallel programs but also of the problem size, page size, chunk size\(^2\), and process scheduling of the algorithm and architecture. The results also suggest that shared data cluster potential is largely determined by spatial locality. We have also confirmed that non-shared data references generally exhibit greater temporal locality than spatial, with the possible exception of instruction references. These results substantiate the notion that multiprocessor locality capturing mechanisms, whether caches or clustering via hierarchy, must be designed to exploit the spatial component of referencing locality behavior as well as the temporal.

The next section discusses further motivations for looking into these issues. In Section 3.2, spatial, temporal and cluster locality are defined, and means of measuring the same are presented. In Section 3.3, the results of the locality studies are presented and discussed. Finally, the significance of these results to issues related to multiprocessor architecture and, in particular, the interconnect strategy is discussed in Section 3.4.

3.1 Background

Uniprocessors exploit locality by using caches to lower communication latency. Qualitatively, caching is effective in multiprocessors when the ratio of the total number of read and write memory references to the number of write references is much greater than the number of processors sharing blocks due to cluster locality. The rationale for this is that every write reference made by a processor causes coherence traffic over

\(^2\)The chunk size is essentially the number of consecutive iterations assigned to processors executing a parallel do-loop.
the global network to main memory as well as the caches of all processors sharing that block throughout the system. Moreover, every write reference causes a global network access by all sharing processors the next time each performs a transaction to that block. This is illustrated by the example shown in Figure 3.1, where memory block “J” is valid only for processor 1 (which write-modified a location), and memory block “K” is valid only for processor 2. Hence, uniprocessor caching techniques may not be completely sufficient for capturing locality in multiprocessing environments.

Several hierarchical interconnect organizations for multiprocessors have the capacity to capitalize on memory reference locality [15, 12, 13, 6]. The means by which these networks exploit locality to reduce average memory latency is through clustering. Clustering is the grouping together of processors and memory blocks having high affinity one with another into local spaces having high bandwidth capacity. Memory transactions within these local spaces or clusters incur minimal latency, but transactions destined to more remote locations further up the interconnect hierarchy incur increasing latency. Clustering also enables combining of remote messages by cluster processors destined to memory blocks not associated with the cluster. Thus, clustering can capture most of the referencing activity, significantly reducing global network traffic and contention, which decreases the overall communication latency.

Figure 3.1: Circumstances under which multiprocessor caching is effective.
The GLORI strategy is designed to include such capability.

The effectiveness of hierarchical interconnection networks for reducing latency depends on the amount of clustering potential inherent in parallel programs executed on these systems. Attributes of a given interconnect architecture such as its cluster size and the connectivity between and throughout cluster tiers (topology/organization) must be in accordance with cluster potential. Hence, studies of this sort that characterize communication locality behavior of multiprocessors are important.

Intuition suggests that shared data in parallel S/E algorithms have significant potential for clustering. However, few quantitative studies confirming this tenet have been conducted. Recent studies of multiprocessor memory reference behavior conducted at IBM address this issue. Kumar and So [84] studied the non-uniformity of accesses to memory modules, substantiating the occurrence of hot-spots in shared data. Darema-Rogers, et al., [85] studied the frequency of shared data usage as a fraction of the usage of all reference types. Baylor and Rathi [86] studied how shared data is accessed, e.g., read-only by several processors, read-write by single processors, etc. In related work, Agarwal and Gupta [87] studied how shared data locality behavior affects cache coherence strategies. This research studies the effect that spatial and temporal locality have on shared data clustering potential.

3.2 Locality Metrics

3.2.1 Definitions

In characterizing reference patterns of individual processors to memory locations, two metrics have traditionally been used: spatial and temporal locality. These can operate at the processor level or cluster level. Figure 3.2 illustrates locality operating at the processor level. Spatial locality specifies the relationship of references to memory locations based on distance of occurrence over a given time span. Temporal locality specifies this relationship based on frequency of occurrence over a given time span. Hence, high spatial locality means that there is a high probability that a reference will made to a memory location near to a previously referenced location in the near
CHAPTER 3. MULTIPROCESSOR COMMUNICATION BEHAVIOR

Temporal Locality
Processor $I$, $I + 1$, $I + 2$, $I + 3$, $I + 4$, $I + 5$

Program Execution (Cycles)

Spatial Locality
Processor $I$, $I + 1$, $I + 2$, $I + 3$, $I + 4$, $I + 5$

Figure 3.2: Spatial and temporal locality at the processor level.

future. Likewise, high temporal locality means that there is a high probability that a processor will a memory location previously referenced will be referenced again in the near future.

Many of the highlighted terms are somewhat ambiguous since their values traditionally have depended on various cache parameters. For example, memory locations traditionally signify word or byte addresses. Locations less than or equal to the distance of a cache line (e.g., $2 - 16$ consecutive addresses) away from the current address qualify as near to. Near future generally implies the period of time between a line's placement and its replacement into and out of the cache which is at least the time to reference once all lines within the cache (e.g., for least-recently-used replacement). Locations used greater than or equal to the number of times the average address is accessed are understood to be used frequently.

For the purposes of defining locality in the context of multiprocessors, memory locations signify word or byte addresses. However, near to is redefined to be less than or equal to the distance of a page of consecutive addresses. Near future is redefined to be the period of time between a page's placement and its replacement in memory. We also redefine locations that are referenced heavily (above some threshold)
and consecutively by the same processor to be used frequently (by individual processors). Agarwal and Gupta [87] calls this activity clinging and the resulting behavior processor locality rather than temporal locality.

In addition to spatial and temporal locality, we define a third metric to aid in the characterization of multiprocessor reference behavior: cluster locality. Cluster locality specifies the relationship of groups of processors’ references to memory locations based on both distance and frequency of occurrence over a given time span. Figure 3.3 illustrates locality operating at the cluster level. High cluster locality means that there is a high probability that distinct groups of processors will reference frequently distinct pages previously referenced in the near future; thus, there is high potential for clustering. A necessary condition for cluster locality is that either one or both spatial and temporal locality exist.

3.2.2 Intuitive Behavior

Spatial and temporal locality on multiprocessors (MPs), as is the case on uniprocessors (UPs), are expected to be high for instruction (Instr) references due to the
CHAPTER 3. MULTIPROCESSOR COMMUNICATION BEHAVIOR

sequential and iterative nature of programs. Spatial is expected to be slightly higher on MPs because parallel sections do not generally have branches to remote locations, i.e., subroutine calls, and do-loops are usually parallelized. Other branch constructs (i.e., if-then-else) tend to decrease spatial locality. Conversely, temporal locality is expected to be slightly higher on UPs because loops remains intact. Separate iterations of large loops are assigned to individual processors, in effect, scaling down temporal locality at this loop level. This is a function of the chunk size and problem size.

Fetch-&-Add (F&A) references are references to loop indices and synchronization variables. Low spatial locality is expected in these references in UPs and MPs for several reasons. Since UPs do not require synchronization and loop scheduling, F&A loop index references are expected to have high temporal locality on UPs. However, temporal locality is expected to be lower for MPs since F&A data are referenced only once for each parallel do-loop iteration to maintain process synchronization. Their values are atomically copied to private variables, incremented, and written back the number of times each processor is assigned a unique loop iteration for parallel execution. This is a function of the chunk size and problem size as well.

Data references to non-globally defined data that are not to loop indices are private data (PData) references. These variables are stored in memory local to processors. Many private data items seldom share the same memory block and, therefore, are expected to have low spatial locality. Private data references on MPs are expected to have high temporal locality – but slightly less than UP – for reasons similar to instruction references.

Shared data (SData) references are to locations that are explicitly defined as global – all processors reference global space for these variables. Moderate to high spatial locality is expected on MPs assuming references to shared data structure elements occur semi-consecutively. Low temporal locality is expected since shared variables are often passed by reference to subroutine kernels (which may be parallel tasks) for initial read and final write access by private data variables. This is particularly true for MPs; a fundamental objective of parallel programming is to limit

---

3Parallel do-loops may be nested within larger sequential loops, resulting in moderate to high temporal locality. Otherwise, low temporal locality is expected.
CHAPTER 3. MULTIPROCESSOR COMMUNICATION BEHAVIOR

<table>
<thead>
<tr>
<th>REF TYPE</th>
<th>UP SPATIAL</th>
<th>UP TEMPORAL</th>
<th>MP SPATIAL</th>
<th>MP TEMPORAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTR</td>
<td>MODERATE TO HIGH</td>
<td>HIGH</td>
<td>HIGH</td>
<td>MODERATE TO HIGH</td>
</tr>
<tr>
<td>F&amp;A</td>
<td>LOW</td>
<td>MODERATE TO HIGH</td>
<td>LOW</td>
<td>MODERATE</td>
</tr>
<tr>
<td>PDATA</td>
<td>LOW</td>
<td>HIGH</td>
<td>LOW</td>
<td>MODERATE TO HIGH</td>
</tr>
<tr>
<td>SDATA</td>
<td>MODERATE</td>
<td>LOW</td>
<td>MODERATE TO HIGH</td>
<td>LOW</td>
</tr>
</tbody>
</table>

Figure 3.4: Summary of anticipated MP locality behavior

SData accesses which can cause contention/hot-spot problems for parallel processes. Depending on the iterative nature of the program, however, temporal locality may increase.

In comparing UP and MP locality using the projections given in Figure 3.4, one can arrive at some interesting conclusions. Temporal locality is more prevalent in UPs than in MPs for non-SData references. Spatial locality is not as prevalent in UPs and is relatively lower in UPs than in MPs. Focusing on MP behavior, SData has relatively greater spatial locality potential than temporal. Non-SData references (with the possible exception of Instr references) have greater temporal locality potential than spatial. These notions indicate that traditional cache designs (those tuned primarily to capture temporal locality, e.g., have small line sizes) perform better on UPs than on MPs even when coherence degradations are factored out. S/E algorithms, many of which have relatively high percentages of SData references, are expected to perform less optimally on MPs which do not use this locality capturing technique more fully to exploit spatial locality (e.g., large line sizes).
3.2.3 Measuring Locality

A processor's working set is the collection of locations actively referenced over an interval of time called a working set window. The composition and sizes of working sets and the working set window over which they occur may vary from processor to processor and throughout execution as illustrated in Figure 3.5. The way in which working sets are managed among processors throughout execution determines the existence of locality and cluster potential. For example, if nearly all elements of a given working set reside in a relatively few number of pages (working set elements span a small distance), then high spatial locality exists. If the ratio of the size of a working set and the size of the working set window over which the working set occurs is small (working set elements are used frequently), then high temporal locality exists. Last, if several sets of processors have in common many of the same elements in their respective working sets during large time intervals and/or share many of the same pages storing their respective working sets\(^4\), then high cluster locality exists.

In this study, spatial, temporal, and cluster locality for various page sizes are

\(^4\)When the actual values used by distinct processors are different, this behavior is termed false sharing\([88]\)
quantitatively measured. One basis of measurement is that of determining the static behavior of the three locality metrics. This is obtained by averaging the locality behavior occurring throughout execution over all processors in intervals we call time windows (TWs), thus giving the mean or static locality behavior. Time window sizes are selected large enough to include references to most elements of a working set and sufficiently overlap so as to effectively approximate a sliding window.

Spatial locality is measured by observing, on a per processor basis, the average number of locations referenced within pages used to capture processor working sets for specified TW sizes. This effectively quantifies the sparseness of active pages in TWs; the denser the pages, the higher the spatial locality. Temporal locality is measured by observing on a per processor basis the average number of clinging references made to locations storing processor working set elements for specified TW sizes. This effectively quantifies the usage of active locations in TWs; the more locations are used, the higher the temporal locality. Cumulative spatial and temporal locality is measured by observing the average number of times pages are referenced per processor for specified TW sizes. This measure is useful in calculating the magnitude of cluster locality. By comparing the average number of times pages are referenced by processor clusters (for specified cluster and TW sizes) with results obtained from cumulative locality of individual processors, cluster locality can be determined.

The page size assumed is pivotal to how well locality behavior is measured and exploited. Empirical determination of an optimal page size can be done by noting the size which yields the best overall locality behavior. This will be the minimum page size which on average best captures not only the working sets of individual processors but also the working sets of clusterable processors for all working set windows of a parallel execution.

3.3 Results of Locality Measurements

Figures 3.6 and 3.7 show the relative percentages of memory use verses reference frequency and average frequency of line use for all three applications. This allows us to compare the amount and type of locality potential of each reference type for all
applications. The HYD application is a two-dimensional model of the hydrodynamics and thermal behavior of fluids [89]. The table in Figure 3.8 presents HYD's memory reference behavior. The number of cycles needed to execute this program total 660,524 reference cycles. A low degree of locality is seen in this application. This may stem from its relatively small problem size, its algorithm or chunk size. These characteristics greatly influence the resulting locality behavior of parallel programs. Further exposition on HYD's locality behavior is given in [90, 91, 92].

FFT and EIG applications have extensive use of shared data stored in matrix form. A few comments concerning the policy of storing these matrices into virtual and real memory are in order. FORTRAN, the language in which the applications were written, requires that matrix elements be stored in column-major order (e.g., A(1,1), A(2,1), ..., A(column-N, row-1) in the virtual memory space. Virtual memory is not necessarily mapped to contiguous real memory space. Only those locations within page boundaries remain contiguous through the memory translation process. Thus, spatial consistency is maintained through the translation process in memory.

---

5Mapping heuristics depend on the underlying architecture.
**CHAPTER 3. MULTIPROCESSOR COMMUNICATION BEHAVIOR**

### FREQUENCY OF LINE USE

<table>
<thead>
<tr>
<th>REF TYPE</th>
<th>LINES USED</th>
<th>% TOTAL</th>
<th>REFS</th>
<th>% TOTAL</th>
<th>REFS/LINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTR</td>
<td>3566</td>
<td>33.3%</td>
<td>2,023,951</td>
<td>63.4%</td>
<td>567.6</td>
</tr>
<tr>
<td>F&amp;A</td>
<td>41</td>
<td>0.4%</td>
<td>43,768</td>
<td>1.4%</td>
<td>1067.5</td>
</tr>
<tr>
<td>PDATA</td>
<td>1992</td>
<td>18.0%</td>
<td>1,016,980</td>
<td>31.8%</td>
<td>529.1</td>
</tr>
<tr>
<td>SDATA</td>
<td>5169</td>
<td>48.3%</td>
<td>107,410</td>
<td>3.4%</td>
<td>20.8</td>
</tr>
<tr>
<td>TOTAL</td>
<td>10698</td>
<td>100%</td>
<td>3,192,109</td>
<td>100%</td>
<td>N/A</td>
</tr>
</tbody>
</table>

*Figure 3.7: References to lines of memory for applications.*

*Figure 3.8: HYD's memory reference profile.*
block units of pages.

3.3.1 FFT Memory Reference Behavior

Figure 3.9 profiles the 2-dimensional MVS FFT application's memory reference behavior. Only 45,425 reference cycles were needed to complete execution. The shared data consists entirely of the $64 \times 64$ double-precision complex numbers over which the 2-D FFT computation is performed. These complex numbers are stored in two $64 \times 64$ matrices (one for each of the real and imaginary parts), each of which use up $64\text{KBytes}$ of contiguous virtual memory ($(16 \text{ bytes/element}) \times (64 \text{ elements/column} = 1\text{KByte or 256 words per column}) \times (64 \text{ columns})$). This accounts for SData's relatively large percentage of active memory use (approximately 80% of the total). Interestingly, however, only 8% of the total references were made to SData locations. The overwhelming majority of references were to instructions which comprised only approximately 15% of active memory.

The relatively small percentage of F&A references indicate that the parallel loop iterations of the algorithm were large grained tasks. These tasks were FFT computations across one dimension of the complex number matrices. With a chunk size of

---

6These double-precision numbers are stored as quad-words or 16 bytes, thus allowing the potential for quad-precision storage. They are stored on line boundaries; line size is 4 words.
one, tasks were assigned to processors executing parallel sections. For the first half of execution (Figure 3.10), FFT computations were executed over row elements of each column, resulting in column-major oriented processing. The second half had row-major oriented processing where FFT computations were performed in parallel in the X dimension of the matrices. Column-major oriented processing results in computations proceeding on successive locations within pages whereas row-major oriented processing results in computations proceeding on various locations across pages.

Static behavior of the FFT application is shown graphically in Figures 3.11 thru 3.17. Figure 3.11 indicates that pages storing SData are densely populated with the same. For smaller page sizes, Instr pages are also densely populated. Distance ranges over which working sets may occur can be inferred from this figure. Concentrating on page sizes of 1024 words, Figure 3.12 shows that locality is evident for Instr, SData, and PData references for TW sizes of 3000 reference cycles. TWs less than this number fail to capture on average most elements composing working sets, while greater TW sizes poorly approximate a sliding window. Characteristics of the spatial and temporal locality components occurring for each reference type are shown in Figures 3.13 and 3.14.

As shown in Figure 3.13, spatial locality is high for SData, moderate for Instr,
but low for PData and F&A references. It is captured in TWs of 3000 cycles for SData; smaller TWs of 500 are sufficient for Instr, which is intuitive since branches may occur after a relatively few number of instruction cycles. Temporal locality, on the other hand, is high for PData, moderate for Instr, and low for SData and F&A references. It is captured in TWs of 3000 cycles which also is intuitive since loop traversal takes a relatively larger number of instruction cycles.

Figures 3.15, 3.16, and 3.17 are graphs of locality behavior for the smaller page size of 256 words. Since its measure is independent of page considerations, temporal locality behavior is the same for smaller and larger page sizes. Because 256 elements is the working set size for each processor (1-D FFT computation with chunk size of 1), it would seem that SData spatial locality for 256 word page size should be the same as for 1024 word page size; only for page sizes smaller than 256 words should the magnitude of the spatial locality measure decrease accordingly. This is actually the case. Since a total of four processors make reference to pages of size 1024 words whereas only 1 processor references pages of size 256 words, a linear decrease is seen in the number of references.

The static measures given in the figures average the spatial locality behavior of the two halves of execution, resulting in an overall decrease in spatial locality magnitude.
CHAPTER 3. MULTIPROCESSOR COMMUNICATION BEHAVIOR

Figure 3.12: FFT locality measurement (pagesize = 1024 words).

Figure 3.13: FFT spatial locality (pagesize = 1024 words).
During the first half of execution, no SData pages other than the two used to store the column elements of the complex number (real and imaginary values) are used by each processor. The row-major oriented half of execution results in low spatial locality since locations across pages (instead of within pages) are used in each processor's 1-D FFT computation. Thus, processor working sets may span a large distance range of real memory. Hence, processor working sets span a relatively small distance range (e.g., 256 locations), resulting in high spatial locality (magnitude comparable to 1024 word page size) for the first half of execution. Graphs detailing the dynamic behavior show this contrast and, thus, give a more accurate picture of locality behavior.

**FFT Cluster Locality**

For cluster locality to exist, several sets of processors must have many of the same elements in their respective working sets and/or they share many of the same pages storing their respective working sets. Since data are uniquely assigned to processors for each phase of 1-D FFT computation, working sets are not shared within working set windows by one or more processors. However, for the first half of execution (approximately 22,500 reference cycles), sharing of pages by processors may occur.
CHAPTER 3. MULTIPROCESSOR COMMUNICATION BEHAVIOR

Figure 3.15: FFT locality measurement (pagesize = 256 words).

Figure 3.16: FFT spatial locality (pagesize = 256 words).
since their working sets occupy 256 consecutive words of active memory for chunk size of 1. This chunk size was used by all applications. Cluster locality of this nature (the number of clusters and amount of page sharing within each cluster) depends on the page size, chunk size, and problem size.

For example, page size of 1024 words, chunk size of 1, and problem size of $64 \times 64$ elements for a 64 node system results in cluster locality among 16 sets of 4 processors sharing 2 pages (one for each real and imaginary part). With the same problem size, page size of 1024 words and chunk size of 2 results in cluster locality among 16 sets of 2 processors sharing 2 pages; page size of 512 words and chunk size of 1 results in cluster locality among 32 sets of 2 processors sharing 2 pages. With an increased problem size of $128 \times 128$, the two previous scenarios result in cluster locality among 64 sets of 2 processors sharing 4 pages and 64 sets of 1 processor exclusively using 2 pages (essentially, processor locality is the sole source of cluster locality).

Since for the second half of execution working sets were dispersed across all pages instead of separate sets of pages, no cluster locality resulted; neither of the two conditions were fulfilled. In effect, all processors, each of which had unique working sets, shared a common set of pages, none of which were used by any one set of processors exclusively. The low amount of SData spatial locality plays a prominent

![Temporal Locality Diagram](image-url)

Figure 3.17: FFT temporal locality (pagesize = 256 words).
The EIG application is a parallelized version of the EISPACK algorithm customized for an RP3 like multiprocessor [93]. The solution of the problem requires four major steps: 1) reduction to the standard form, 2) reduction to a symmetric tridiagonal matrix, 3) repeated interval-bisection to locate individual eigenvalues to the desired precision, and 4) computation of the corresponding eigenvectors by inverse iteration. The EIG application's memory reference behavior is displayed in the tables in figures 3.18 and 3.19 for problem sizes of 64 x 64 and 10 x 10 respectively.

Figures 3.20 and 3.21 show how the two problem sizes differ in their relative percentages of memory use, reference frequency, and line use for each reference type. The most startling difference is in SData’s characteristics. Figure 3.22 further shows the difference in page use for SData between the two problem sizes. Since interests lie primarily in SData behavior, EIG (64 x 64) is analyzed in this section. Comparisons between the two problem sizes’ behavior is discussed in the discussion section.

Unlike the FFT application, the EIG (64 x 64) application reference traces are many: 49,809,919 references as opposed to 2,793,901. There are a higher percentage of serial sections in EIG than in FFT as is evident by the higher percentage of F&As.
CHAPTER 3. MULTIPROCESSOR COMMUNICATION BEHAVIOR

<table>
<thead>
<tr>
<th>REF TYPE</th>
<th>LINES USED</th>
<th>% TOTAL</th>
<th>REFS</th>
<th>% TOTAL</th>
<th>REFS/LINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTR</td>
<td>8175</td>
<td>73.8%</td>
<td>2,033,975</td>
<td>54.4%</td>
<td>248.8</td>
</tr>
<tr>
<td>F&amp;A</td>
<td>102</td>
<td>0.9%</td>
<td>67,788</td>
<td>1.8%</td>
<td>664.6</td>
</tr>
<tr>
<td>PDATA</td>
<td>2285</td>
<td>20.6%</td>
<td>933,712</td>
<td>25.0%</td>
<td>408.6</td>
</tr>
<tr>
<td>SDATA</td>
<td>518</td>
<td>4.7%</td>
<td>701,931</td>
<td>18.8%</td>
<td>1,355.1</td>
</tr>
<tr>
<td>TOTAL</td>
<td>11080</td>
<td>100%</td>
<td>3,737,406</td>
<td>100%</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Figure 3.19: EIG (10 × 10) reference profile.

(although both applications have high parallel content). Lengthy execution times of 17,395,486 reference cycles are experienced for each simulation run. Because of these long simulation times, efforts were concentrated on analyzing one subroutine that approximately represents many of the computations performed by the EIG application. The subroutine chosen was REDUCE — the first step of the solution where the generalized eigenvalue problem, \( Ax = \lambda Bx \), is reduced to the standard form, \( Ay = \lambda y \) (see [93, 94]).

Plotting data retrieved from REDUCE only, Figure 3.23 indicates that SData pages are not as densely populated with active locations as FFT’s SData pages. Some reasons for this stem from the storage management of shared data structures; some of the one-dimensional arrays may be too small to completely occupy larger sized pages, and only half of the A and B symmetric matrices are used actively although all elements are stored. The general shape of the non-SData reference page density curves resemble FFT’s and follow intuition.

Figures 3.24 and 3.25 show the reference patterns to the A matrix elements for REDUCE (each arrow indicates a loop iteration assigned to a processor). The pseudocode for REDUCE is shown in Figure 3.26. Two large grained sequential loops having inner parallel do-loop reference streams shown in Figure 3.24 are followed by another large grained sequential loop having an inner parallel do-loop reference stream shown.
Figure 3.20: Comparison of EIG $64 \times 64$ and $10 \times 10$.

Figure 3.21: EIG ($64 \times 64$) frequency of line use.
in Figure 3.25. As shown in the figures, the inner parallel loop used by the first two sequential loops features column-major oriented processing. The resulting potential for SData spatial locality ranges from high to low, depending on the do-loop iteration: smaller valued iterations have larger working sets which densely populate pages whereas larger valued iterations have smaller working sets, sparsely populating pages. Overall, moderate SData spatial locality (lower than the spatial locality of the first half of the FFT application’s execution) is expected for this inner loop. The inner parallel loop associated with the third sequential loop features row-oriented parallel processing. As in the case for the FFT application, low SData spatial locality results from this type of processing.

The inner parallel loops embedded within the sequential do-loops are iterated on about the same number of times as the matrix dimension. SData temporal locality occurs over individual sequential do-loop kernels if each processor is assigned the same inner parallel loop index (task) for all iterations of the sequential do-loops. This ensures that working set elements associated with the tasks of inner parallel loops remain in unique one-to-one relationship with processors for the entire sequential do-loop kernel. Without temporal consistency across outer loop iterations, the potential for temporal locality lies solely on the temporal characteristics within the inner loops,
Figure 3.23: Density of pages for EIG (64 x 64).

Figure 3.24: REDUCE's first two loop reference streams.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
leading to low temporal locality. Parallel constructs (e.g., EPEX's\textsuperscript{7} \textit{barrier} signal) and compiler techniques can synchronize processes, allowing them to be scheduled in accordance with the above scheduling algorithm.

The implementation of REDUCE used process scheduling that facilitated temporal locality only for the first sequential do-loop kernel. The other two kernels essentially had random process scheduling. Moderate temporal and spatial locality occurs in the first kernel. The second kernel has only moderate spatial locality, and the last kernel has neither spatial nor temporal locality. Poor static behavior of REDUCE shown in Figures 3.27, 3.28, and 3.29 result from this process scheduling and programming algorithm. Once again, the data points are the averages of the high and low locality behavior occurring and are therefore conservative. More exact results are obtained with dynamic measurements being applied to REDUCE.

\textsuperscript{7}See Chapter 6 for further explanation of the EPEX parallel programming environment.
CHAPTER 3. MULTIPROCESSOR COMMUNICATION BEHAVIOR

Figure 3.26: REDUCE pseudo-code

<table>
<thead>
<tr>
<th>Procedure REDUCE(n, a, b, d)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>n</strong> is the order of the matrices.</td>
</tr>
<tr>
<td><strong>a</strong> contains the real symmetric input matrix, and on output, in its lower triangle, the symmetric matrix derived from the reduction to standard form.</td>
</tr>
<tr>
<td><strong>b</strong> contains the real symmetric input matrix, and on output, in its strict lower triangle, the strict lower triangle of its Choleski factor.</td>
</tr>
<tr>
<td><strong>d</strong> contains the diagonal elements of the Choleski factor of <strong>b</strong>.</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHAPTER 3. MULTIPROCESSOR COMMUNICATION BEHAVIOR

LOCALITY MEASUREMENT
AVERAGE NUMBER OF REFERENCES TO PAGE VS TIME WINDOW SIZE
EIG (64x64), PAGE SIZE = 1024

![Graph](image_url)

Figure 3.27: EIG (64 x 64) locality measurements.

SPATIAL LOCALITY
AVERAGE NUMBER OF LINES ACCESSED PER PAGE VS TIME WINDOW SIZE
EIG (64x64), PAGE SIZE = 1024

![Graph](image_url)

Figure 3.28: EIG (64 x 64) spatial locality.
EIG Cluster Locality

Cluster locality in EIG is attributed to the same phenomenon which caused it in FFT: page storage of multiple working sets (false sharing). The basic spatial locality and temporal locality conditions necessary for cluster locality to exist are satisfied by the first sequential do-loop kernel. During the 135,000 reference cycles needed to execute this kernel, pages are uniquely shared by groups of 3 processors. Because some processor's working sets straddle 2 pages, groups of 8 processors sharing 4 pages was also used for this working set window size. The moderate spatial locality inherent in the second kernel is not sufficient to support cluster locality. If its process scheduling were tuned to facilitate temporal locality, the second kernel would have cluster locality behavior similar to the first. The working set window size of this kernel is 102,500 reference cycles. The third kernel, also requiring on the order of 100,000 reference cycles to execute, fails to meet the conditions necessary for cluster locality even with proper process scheduling. Its low spatial locality requires that its temporal locality (which at best is only moderate) be the only basis for affinity among sets of processors and pages.
3.4 Discussion

To recapitulate, the memory reference behavior results are summarized. By virtue of the memory mapping of matrix elements to pages and the parallel algorithm, FFT SData has high spatial locality during the first half of execution. Its spatial locality is the only basis for cluster locality since it has low temporal locality. As page sizes are increased to include more than one processor's working set, cluster locality increases.

Two out of three EIG (64 x 64) REDUCE subroutine large grained kernels exhibit potential for moderate SData spatial locality and temporal locality. The realization of the temporal locality potential depends upon process scheduling. Cluster locality is realized for EIG when both spatial and temporal locality are present.

Little locality exists in HYD and EIG (10 x 10) SData reference behavior [90]. EIG's small problem size and HYD's non-optimal processing orientation (row-major verses column-major) are major factors contributing to poor locality. Increasing the problem size (e.g., from 10 x 10 to 64 x 64 for EIG) increases cluster potential: the potential number and size of sets of processors sharing pages as well as the number of pages shared by these sets are increased.

The chunk size parameter affects cluster locality in a way similar to problem size; it specifies the size of tasks (working set and working set window size) executed by processors and the overall number of tasks available for parallel execution. Proper alignment of the algorithm (processing orientation) to the underlying memory mapping function also increases spatial locality and cluster potential as can be seen from FFT. For instance, if a transposition of row elements to column elements took place before commencement of the second half of FFT's execution, high spatial locality and cluster locality would again occur during the working set window period of 22,500 reference cycles.

3.4.1 Meaning of the Results

The working set window sizes over which cluster locality occurred for FFT and EIG were approximately 22,500 and 135,000 reference cycles respectively. Assuming the use of a hierarchical interconnect capable of block-frequent reconfiguration, 100% of
all SData references (both read and write references) made by cluster processors can be made local to pages migrated to those clusters for the FFT application. Thus, ideally, interconnect traffic caused by SData references consists entirely of initial page misses by processors, assuming that clusters are instantaneously formed or pages are migrated to clusters once referenced. This can occur over two periods of 22,500 cycles, separated by a few hundred cycles for matrix transposition. For EIG, approximately 32% of all SData references made by cluster processors can be to cluster pages, and 37% of all SData references to cluster pages can be by cluster processors. Non-cluster processors would make approximately 53% of all SData references to cluster pages. These are predominantly read references (all of which would made after pages are updated – see Figure 3.26) since 95% of all SData write references made to cluster pages are by cluster processors. Therefore, ideally, if SData pages are marked as cacheable after finished being modified by cluster processors, then nearly 100% of all SData references could be accommodated within clusters. Very little network traffic is incurred over two periods of over 140,000 cycles for the approximately 450,000 cycle REDUCE subroutine.

The above ideal arrangements assume a lot on the part of the architecture. The dynamic exploitation of cluster locality depends on whether an architecture that detects and captures it can be designed and implemented. The larger the working set window size, the more probable is its dynamic exploitation with block-frequent reconfiguration. Reconfiguration with optics, for instance, requires large grained cluster locality on the order of 5,000 cycles (see Chapter 6). There are many ways an optimizing compiler can enlarge the working set window size and generate cluster locality. Instead of implementing parallelism at the root level of an algorithm only, implementing parallelism at the branch level increases cluster potential. The inner parallel loops of REDUCE shown in Figure 3.26 are parallel sections available for such multi-level parallel execution.
3.5 Summary

The differences between SData and non-SData reference types have been highlighted; their behavior, to a large extent, conform to intuitive notions. Depending on the iterative nature of the parallel algorithm and the problem size, spatial locality occurs more in SData. With the possible exception of Instr references, temporal locality is more dominant in non-SData references.

SData cluster locality is present in parallel S/E algorithms to varying degrees. Clustering potential is a function of both spatial and temporal locality. It arises principally in our benchmarks from page sharing due to storage of multiple mutually exclusive working sets, and not from page storage of shared working sets. Parallel program and architecture parameters such as problem size, page size, chunk size, and process scheduling affect the realization of cluster locality.

Multiprocessor performance is projected to substantially increase if locality capturing mechanisms (i.e., cache, interconnect hierarchy, etc.) are designed to capitalize on cluster locality behavior. The exploitation of cluster locality depends on the sophistication of the architecture, interconnect, and optimizing compiler.
Chapter 4

GLORI Strategy — Architectural Description

Memory reference locality can be used to lower communication complexity and latency by minimizing global communication traffic, maximizing local traffic, decreasing network congestion and bottlenecks, and reducing the effect of limited interconnect bandwidth. In this chapter, the architecture of an optical interconnect strategy for multiprocessors is functionally described. The imaging capability required of the optical router to function as the interconnect switch is also described.

In defining the interconnect architecture, known benefits of optics, hierarchical networks, and parallel program behavior are used to develop an interconnect strategy optimized for low latency communication. This interconnect strategy, called GLORI for global-local optical reconfigurable interconnect, combines the advantages of direct, indirect, and reconfigurable networks (see Chapter 2) into one hybrid topology. It is envisioned as an inter-board level or backplane interconnect for medium-to-large scaled, tightly coupled multiprocessing systems, but is not limited to this.

An overview of the goals and system organization of the GLORI strategy are given in the next section. The local and global interconnect architecture and interface between them is functionally described in Section 4.2. The imaging optics required by GLORI's optical routing unit is specified in Section 4.3, and Section 4.4 summarizes this work.
4.1 Overview of the GLORI Strategy

4.1.1 Goals

The GLORI strategy meets two basic goals. The first is to reduce interconnect latency so that completion of each network link access approaches a processor cycle time ($t_{\text{network}} \rightarrow t_{\text{processor}}$). The second goal is to reduce interconnect cost (line count, length, etc.) significantly below that of electronics. The GLORI strategy effectively reduces latency components as well as interconnect area by exploiting physical advantages of optics, topological features of single-hop and multi-hop networks, and referencing characteristics of multiprocessors.

Consider first the physical benefits of optics. The high bandwidth of optical links ($BW_{\text{optical}} \geq 1\, \text{GHz}$) provides orders of magnitude increased throughput over electrical coaxial lines ($BW_{\text{electrical}} < 100\, \text{MHz}$) [95]. The increase in throughput acts to significantly lower the transceiving component of latency for time-multiplexed transfer of large data packets such as cache lines. The length of GLORI’s optical board-to-board links is designed to be nominal, $O(1 \text{ meter})$.\footnote{A significant portion of the entire system volume can be consumed by an electrical interconnect network as evidenced by IBM’s RP3 machine.} This reduces the propagation component of latency and makes for a more compact system, reducing interconnect line length and area (see Chapter 5).

The GLORI strategy further reduces interconnect latency with its provision for localizing references, a pseudo-caching effect by the interconnect. Its topology is based on interconnect hierarchy, which enables clustering of reference activity. Frequently communicating processors and memory blocks can be clustered to process locally the majority of communication traffic. Clustering also enables combining of remote messages by cluster processors destined to memory blocks not associated with the cluster. With combining, READ references by cluster processors to identical memory locations are not duplicated in the network during the time span needed to complete the first READ transaction to that location but, rather, are buffered at the originating cluster. Global communication between clusters occurs at a higher level in the interconnect hierarchy.
Interconnect reconfiguration is an alternative way of implementing page migration using hardware to decrease the amount of global network traffic. Upon large grained fluctuations in the referencing pattern of an algorithm, processor-memory association can change dynamically with the formation of new clusters by reconfiguring the interconnect to re-localize memory references. This benefits latency reduction in two ways. First, it reduces the number of network accesses (number of hops) which, in turn, reduces the composite propagation component of latency. Second, it reduces global network contention and, therefore, the buffering component of latency. The GLORI strategy supports this block level type of reconfiguration referred to as block-frequent reconfiguration (see Chapter 2). Radical changes in process traffic patterns are infrequent relative to individual reference cycles so that the cost to reconfigure the system is amortized over the time span over which communication traffic remains local. This time can be significant (see Chapters 3 and 6) so that optics’ relatively slow switching time can be tolerated.

Other architectural features of the GLORI strategy reduce line count and area. Narrow (e.g., 8 bit) line widths are used as opposed to wide (e.g., 64 bit) channels, the address and data lines are unified, and sharing of optical links by multiple numbers of nodes are possible. Using the high bandwidth of optics, bytes of messages are multiplexed onto narrow channels that yield performance comparable to wide electronic lines. Unifying address and data lines improves the utility of data lines used only for write and read response references. As seen in Chapter 3, less than 50% of shared data traffic are write references; the majority are read references. Data accompanies the address field of write messages throughout transmission, using separate address and data lines 100% of the time. However, read references make use of data lines only for the return trip, resulting in 50% use of separate data channels. The overall utilization can therefore be increased by multiplexing the address and data fields of messages onto common high bandwidth optical lines while not affecting performance².

²An example of an electrical shared-bus interconnect with a large line count poorly utilized in a multiprocessor environment is the Nanobus. It has a large datapath width of 64 bits and an address width of 32 bits clocked at 12.5MHz for a throughput of 100MBytes/sec. Compare this with an 8 bit wide unified optical bus clocked at 1GHz for a throughput of 1000MBytes/sec or 1GBytes/sec.
4.1.2 System Organization

The strategy behind GLORI does not necessitate the use of any particular interconnect topology nor any specific number of hierarchical levels, etc. It does, however, require a three-way cooperation between implementation technology (e.g., optics), topological structure (e.g., hierarchical), and program behavior (e.g., locality). For medium scaled systems and applications of interest in this thesis, GLORI's assumed organization is a 2-tier hierarchical shared bus-hypercube configuration illustrated in Figure 4.1. Reconfiguration of processor-memory element connectivity occurs system-wide but the baseline topology remains the same. Ideally, any arbitrary group of nodes can be clustered together in the system.

Several researchers have recognized the advantages of hierarchical interconnects for multiprocessors [15, 96, 97, 98, 99, 100]. Hierarchy enables GLORI to abstract local and global references into distinct communication spaces for simultaneous routing on separate dedicated interconnect resources. As the number of hierarchical levels increase, so does the potential for communication parallelism.
At the lowest level in the hierarchy, the local interconnect, \( M \) processor-memory nodes which make up a cluster are interconnected in a shared bus topology. Least expensive of the direct networks to implement is the optical shared bus which, for moderately sized clusters, does not saturate on account of limited bandwidth (see Chapter 6). Shared bus clusters are arranged in a hypercube topology at the higher hierarchical level or global interconnect. This multi-hop network eliminates the need for dynamic optical switch elements, making GLORI tolerant of optics' major drawback: slow switching time. GLORI's hypercube consists of \( N/M \) clusters or supernodes and has degree \( k = \log_2(N/M) \), where \( M \) is the number of processor-memory nodes within each cluster and \( N \) is the total number of processor-memory nodes in the system. The architecture of both interconnects are discussed in greater detail in Section 4.2.

Global and local interconnect links are passive through optical fiber and free-space channels. As stated previously, fibers act only to guide optical signals to the optical routing unit which performs the actual interconnect routing. The optical routing unit or ORU establishes the interconnect links in free-space by appropriately directing incident optical signal beams emitted by source fibers to receiving detector fibers via a beam steering element as explained in Chapter 5. Reprogramming the beam steering element's input-to-output (processor-to-memory) mapping while maintaining its basic hybrid global-local interconnect topology results in a reconfigured system. Reconfigurability is a fundamental advantage of free-space optics. The optical imaging capability needed by the ORU is described in the Section 4.3.

Interfaces between each cluster's global and local optical interconnects are intercluster routing controllers or IRCs. These modules process routing transactions between the interconnects electronically, actively converting signals from optical to electrical and back to optical forms. The electronic decision-making circuitry within each IRC includes transaction buffers which queue messages for global routing in order by type, a local directory that decides whether cluster messages are local or remote, and a global routing directory which specifies where remote messages are to be routed. Section 4.2 discusses the functionality of the IRC in greater detail.
For the shared bus-hypercube GLORI, physical memory is assumed to be distributed throughout the system at each of the memory nodes. The processors are assumed to be coarse-grained, high performance processors that cache private data and instruction references. Shared data is assumed to be not cached; hence, network traffic is primarily that associated with shared data. Though provisions for implementing shared data cache consistency models do exist in its interconnect architecture, the GLORI strategy is an independent solution to the problem of shared data traffic. In Chapter 6, performance is analyzed irrespective of various cache consistency models by assuming that shared data is not cached by the processors. Moreover, latency of the interconnect is kept independent of the various latencies associated with alternative memory interleaving methods by assuming that memory module access time is transparent to the overall latency of the interconnect. This assumption is valid, for instance, when memory is prefetched.

The 2-tier GLORI strategy supports a hierarchical memory system illustrated in Figure 4.2. Not including those that are cached, references to the distributed shared address space can be destined to one of three places: the memory node associated with the requesting processor, one of the cluster memory nodes local to the requesting processor, or one of the remote cluster memory nodes. If the location being referenced does not reside at the lowest level of the memory hierarchy, the request is sent to the next level, and so on.

A message can be in either one of two forms: read format or write/response.
CHAPTER 4. GLORI STRATEGY – ARCHITECTURAL DESCRIPTION

format. In both, header information is followed by the address being referenced. For writes and responses to reads, the address referenced is followed by data (see Figure 4.3a). The header, shown in Figure 4.3b, contains pertinent routing information. The read/write (R/W) and request/response (RQ/RSP) status bits designate the type of message transaction – read, write or response to read. The originator and destination cluster’s logical addresses, ORIG_ADDR and DEST_ADDR, follow the status bits. These logical addresses each take up $k$ bits of the header. Processor ID and other status bits follow. Status bits can include: a processor or IRC message bit, single or double word request bit, synchronization broadcast request bit, etc. For response messages, the information contained in these fields are reversed for proper return trip routing; the originator cluster’s address becomes the destination cluster address and vice versa. Figures 4.4 and 4.5 show the message protocol.

The message flow of a read request is illustrated in Figure 4.4. The read request is issued on the local interconnect by a master processor. If it is not satisfied by the associated memory node or any of the cluster memory nodes and if a request previously made to its location by another processor is not pending, the message is queued for the global interconnect. When access to the global interconnect is granted, the request hops to neighboring cluster(s) who check its header to determine whether to propagate, ignore, or receive the message locally. The message hops from cluster to cluster until its destination is reached.

Once it is received locally by the remote destination cluster and local interconnect access has been granted, the request is issued, acted on by one of the remote cluster’s memory nodes, packaged and queued for the global interconnect as a response
CHAPTER 4. GLORI STRATEGY – ARCHITECTURAL DESCRIPTION

message. The response message again hops through the global interconnect until it arrives at the originating cluster. The response message is issued onto the local interconnect after access is granted, received by all processors who requested the data, and finally removed from the network. The flow of write messages is illustrated in Figure 4.5 and is similar to read requests except that there is no message combining and no return trip through the network.

4.2 Functional Description of GLORI Architecture

In this section, GLORI's architecture is described. This includes interconnect link arbitration, functionality, and transactions.
CHAPTER 4. GLORI STRATEGY – ARCHITECTURAL DESCRIPTION

In the Glori cluster, a cluster memory node receives data on bus. If remote, it is put in the IRC Local Buffer Unit.

Remote Cluster

• IRC Local Buffer Unit issues WRITE message on bus.
• Cluster memory node receives data on bus.

Intermediate Cluster

• IRC Routing Directory Unit examines WRITE message header and takes appropriate action:
  • propagate: It is put into the IRC WRITE Buffer Unit. Request is sent to hypercube links.
  • receive locally: It is put into the IRC Local Buffer Unit.
  • ignore: Message is discarded.

Local Cluster

• Processor issues WRITE on bus.
• Its destination is determined by IRC Local Directory Unit:
  • If local: Cluster memory node receives data on bus.
  • If remote: It is put in the IRC Write Buffer Unit.

Figure 4.5: The message flow of a write request.

4.2.1 Local Interconnect

The local interconnect is a synchronous B-bit wide optical shared bus (e.g., B = 8 bits) onto which bytes of a message are multiplexed at a minimal clock rate of 1GHz. As previously stated, M processor-memory nodes are assumed to share each bus for a total of N/M separate shared bus clusters system-wide. The maximum physical distance separating entities sharing a bus is assumed to be 2 meters for a roundtrip path length ≤ 4 meters. Considering point-to-point interconnect latency only, a typical message transaction of 64 bits (double word) can be performed within 20 nsec on an 8-bit wide bus (8 nsec for transceiving and up to 12 nsec roundtrip propagation delay). Thus, a bus cycle – defined to be the time required to complete a full length message transaction on the bus – approaches the cycle time of a 50MHz processor. The overall throughput of the optical bus is 1GByte/sec for an 8-bit wide bus.

Arbitration on the bus is performed by a distributed controlled polling strategy similar to the carrier-sense-multiple-access (CSMA) bit-map protocol [101]. One optical control link is dedicated to bus arbitration. An arbitration phase in which bus
requestors (processors and IRC) are polled and an ordering scheme for bus ownership is determined precedes a message transfer phase as shown in Figure 4.6. For \( M \) processor-memory nodes sharing the optical bus, there are \((M + 1)\) 1-bit slots in a polling cycle. Each node has a unique slot in which to broadcast a "1" to request the bus or a "0" to be idle on the bus. Control is distributed — after receiving the polling cycle requests, each node independently determines who will own the bus and when. Synchronous message transmission by bus users occur in the order determined after arbitration.

Arbitration for bus ownership in the next message transfer phase occurs in parallel with the last message of the present message transfer phase. Overlapping control and communication enables full utilization of the optical bus shared resource. The arbitration time is small (e.g., \( \approx 20 \text{ nsec} \) for \( M = 8 \), assuming control logic takes approximately \( 10 \text{ nsec} \)) and constant, independent of bus load. Therefore, the arbitration overhead is very low for a high occupancy bus.

Transactions occur on the optical bus in the following manner. Messages encoded optically are broadcast from the bus owner, or master, (local cluster processor nodes or IRC) to slave entities (local cluster memory nodes or IRC) via the ORU. All slaves receive the transmitted message's address but respond only if its address matches that of the incoming message. For reads local to the cluster, the data from the associated slave memory node is broadcast to all processors on the bus but is accepted only by the bus master.

If the location being read is not within the address space of the cluster memory, the IRC accepts the message. It proceeds by either queuing the message for transmission onto the global interconnect or merging the request with other outstanding cluster requests to the same block. In either event, it sends an acknowledgement.
back to the master processor, instructing it to stall while the request is pending. This acknowledgement is sent during the data transfer portion of the message cycle. When the response message from the remote cluster that serviced the request is received by the IRC and the IRC has ownership of the bus, it is broadcast across the bus and accepted by all processors whose references were merged by the IRC. Thus, the local interconnect behaves like a split transaction bus only for remote references: the master maintains ownership of the bus until the request is satisfied locally or acknowledged as being global, in which case the master releases the bus since the pending request is unable to complete within the allotted unit cycle time for local bus ownership.

4.2.2 Global Interconnect

Like the local interconnect, the global interconnect has B-bit wide channels on which header, address and data are multiplexed. It operates synchronously at a clock rate of $1\text{GHz}$ for a throughput of $1\text{GB/second}$ on each optical hypercube link. A total of $N/M$ shared bus clusters, or supernodes, are interconnected in the global interconnect's hypercube topology. The maximum physical distance separating hypercube supernodes is the same as that for the shared bus. Hence, single-hop transactions on hypercube links also approach processor cycle completion time.

A distributed routing protocol is used to transfer messages over the global interconnect. Routing is controlled by IRCs, the initiators and retrievers of messages by supernodes to/from the hypercube. Messages encoded optically are sent from the IRC of the global link owner, or master supernode, to the IRC of the link listeners, or slave IRCs, via the ORU. A slave IRC examines the transmitted message's header and accepts the message if the location is within its cluster's memory address space or if it determines that it is a supernode along the static routing path of the message. The slave IRC discards the message otherwise.

To resolve global interconnect contention, IRCs arbitrate to gain access to the hypercube links. Arbitration depends on the communication model adopted. Those
CHAPTER 4. GLORI STRATEGY – ARCHITECTURAL DESCRIPTION

considered for the GLORI strategy are the shared-link and the dedicated-link communication models\(^3\) discussed in the next subsections. Four forms of these communication models (defined below) exist: \textit{k-way broadcast/1-way receive} (or \textit{k-B/1-R}), \textit{k-way transmit/1-way receive} (or \textit{k-T/1-R}), \textit{k-way broadcast/k-way receive} (or \textit{k-B/k-R}), and \textit{k-way transmit/k-way receive} (or \textit{k-T/k-R}). Further explanation of these is given below. The first two fall under the shared-link model, denoted as \(*/1-R\). The latter two are forms of the dedicated-link model, denoted as \(*/k-R\). A \textit{k-way broadcast} system (shared or dedicated) can be denoted as \(k-B/**\) and a \textit{k-way transmit} system as \(k-T/**\).

**Shared-Link Model**

In the shared-link model, hypercube supernodes share link usage through time-division-multiplexing. The necessity to share links arises from the design criteria that supernodes have only one sink with which to receive messages from \(k\) neighboring hypercube links (\textit{1-way receive}). Since supernodes can receive from only one neighbor at a time, the usage of links by neighboring supernodes is time-shared to fully utilize the links which, in this model, are \textit{half-duplex}. A simple arbitration scheme such as round-robin access resolves link contention. Supernodes are assigned slots in which to become the sole owner of all hypercube links to which they are directly connected. Slots last a unit duration, i.e., one bus cycle. Sole ownership of all neighboring links is preferred over partial ownership for reasons of simplicity, ease of implementation, and higher link utilization.

One or more supernodes may be link owners in a given time slot so long as their overlap does not result in slaves listening to more than one master at a time. Therefore, contention at the supernodes from incoming messages on hypercube links is automatically resolved in this model. For example, Figure 4.7 shows that logically opposite nodes (i.e., 000 and 111) in a three dimensional hypercube are linked to a mutually exclusive set of nodes (i.e., [001,010,100] and [101,110,011]), so they can be

\(^3\)Feldman differs in his definitions of shared and dedicated link networks. In reference [102], the two networks are distinguished by the number of transmitters at each node whereas here, the distinction is made by the number of receivers at each node.
assigned the same master slot.

Once the owner of its links, a supernode can either broadcast the same message to its $k$ neighboring slaves who are, by default, in listening mode or transmit $k$ distinct messages, one to each of the $k$ neighboring slave supernodes. The former is the $k$-way broadcast/1-way receive, or $k$-B/1-R, system (Figure 4.7) whereas the latter is the $k$-way transmit/1-way receive, or $k$-T/1-R, system (Figure 4.8). An advantage of using the $k$-B/1-R system is its lower line count (i.e., optical sources, fibers, etc.) An advantage of using the $k$-T/1-R is its more efficient routing capability through $k$ degrees of communication parallelism during slot ownership.

Both forms of the shared-link model suffer from limited global communication
parallelism. The bottleneck arises from the one sink per supernode stipulation which ensures contention-free time-sharing of links. As a result, supernodes become slaves during valuable communication cycles rather than always being masters. Moreover, since slaves can listen to only one master at a time, only a few supernodes can be masters simultaneously. For a $k$-dimensional hypercube, the number of supernodes allowed to be link masters in parallel is equal to

$$Masters = 2^{\log_2(2^k)}.$$  

**Proof:** Each node in the system has $k$ neighboring nodes along the $k$ dimensions of a $2^k$ node hypercube. Consider two master nodes $a$ and $b$. These nodes cannot be neighbors since it would require that they be slaves to one another. Neighbors of $a$ cannot be neighbors of $b$ since slaves can listen to one and only one master at any given time. Thus, master nodes must neighbor mutually exclusive sets of slave nodes in isolated neighborhoods. This implies that master nodes, which occupy the central location of these neighborhoods, are minimally separated by a Hamming distance of 3. Since isolated neighborhoods are composed of $k + 1$ nodes, the maximum number of master nodes in the hypercube equals $2^k / (k + 1)$ rounded down to the nearest power of 2. Rounding down is necessary since the number of nodes in a hypercube is always a power of 2, where nodes in this case are isolated neighborhoods.

The number of cycles required for all hypercube supernodes to become link masters is

$$\frac{2^k}{Masters} = 2^{\log_2(k+1)},$$  

which is simply $k + 1$ rounded up to the nearest power of 2. This as opposed to a constant unit number of cycles if the links are not shared. The effect of limited communication parallelism on performance is analyzed further in Chapter 6.

---

4For systems wherein $k = 2^l - 1$, and $l = 0,1,2,...$, masters can serve as I/O supernodes for perfect embedding [103] within the GLORI system.
**Dedicated-Link Model**

Hypercube supernodes fully own neighboring hypercube links in the dedicated-link model. It is as if the supernodes have neighboring links dedicated to them for sole usage 100% of the time. In this model, each supernode has \( k \) sinks, one for each neighboring link (\( k \)-way receive). As a result, hypercube links operate in full-duplex mode, able to receive and transmit messages at the same time. Potential parallelism is high as there is no notion of dividing communication cycles into separate master and slave slots. All \( 2^k \) supernodes are link masters simultaneously every communication cycle, irrespective of the number of supernodes in the system. No arbitration of hypercube links is necessary. This comes at a cost of \( 2^k \times (k-1) \) additional sinks in the interconnect line count. Thus, the number of supernodes allowed to be link masters in parallel is equal to

\[
Masters = 2^k, \quad (4.3)
\]

and the number of cycles between mastership is one.

As in the shared-link model, a supernode can either broadcast the same message to its \( k \) neighboring supernodes (\( k \)-way broadcast/\( k \)-way receive or \( k \)-B/\( k \)-R) as in Figure 4.9 or transmit \( k \) distinct messages, one to each of the \( k \) neighboring supernodes (\( k \)-way transmit/\( k \)-way receive or \( k \)-T/\( k \)-R) as in Figure 4.10. The advantages and disadvantages of each are the same as those discussed previously for the shared-link model.
Figure 4.10: \textit{k-way transmit} dedicated-link model


equation 4.2


equation 4.2


\textbf{Cost/Performance Comparison of Communication Models}

Table 4.1 summarizes how the global interconnect communication models compare in terms of parallelism potential (a measure of performance) as system size increases.

The least costly model to implement is the $k$-B/1-R shared link model as it requires only one byte-wide source/receiver pair per node (a total of $2 \times B$ sources and receivers per node). It also has the worst communication parallelism in the number of nodes that can simultaneously be slot owners and also in the number of distinct messages that can be sent during slot ownership as shown in the table. The most costly model, which has the highest degree of parallelism, is the $k$-T/1-R dedicated link model. It requires $k$ byte-wide source/receiver pairs per node (a total of $2 \times B \times k$ per node) but allows $k$ distinct messages to be sent by all nodes in the system simultaneously.

The parallelism of the $k$-B/1-R dedicated link model is $k$ degrees less than $k$-T/1-R during slot ownership but several times more than the $k$-T/1-R shared link model (as given by equation 4.2) since links are owned every cycle in the dedicated-link model. The implementation cost of $k$-B/k-R is actually less than $k$-T/1-R even though the interconnect resource count is the same ($B \times (k + 1)$ sources and receivers per node). The difference lies in the fact that optical sources generally consume more area than optical receivers. Thus, the additional parallelism of $k$-B/k-R over $k$-T/1-R actually comes at a lower implementation cost.

Chapter 6 further examines the cost/performance tradeoff between shared link
Table 4.1: Performance Comparison of Communication Models

<table>
<thead>
<tr>
<th>N/M = 2^k</th>
<th>Network Throughput (# of messages/cycle)</th>
<th>Link Access Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Shared-Link (SL)</td>
<td>Dedicated-Link (DL)</td>
</tr>
<tr>
<td></td>
<td>k-B/1-R = (2^{\log_2(\frac{N}{M})})</td>
<td>k-B/k-R = (2^k)</td>
</tr>
<tr>
<td></td>
<td>k-T/1-R = (k \times 2^{\log_2(\frac{N}{M})})</td>
<td>k-T/k-R = (k \times 2^k)</td>
</tr>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>64</td>
<td>8</td>
<td>64</td>
</tr>
<tr>
<td>128</td>
<td>16</td>
<td>128</td>
</tr>
<tr>
<td>256</td>
<td>16</td>
<td>256</td>
</tr>
<tr>
<td>512</td>
<td>32</td>
<td>512</td>
</tr>
<tr>
<td>1024</td>
<td>64</td>
<td>1024</td>
</tr>
</tbody>
</table>

and dedicated link GLORI networks.

### 4.2.3 Intercluster Routing Controller

The intercluster routing controller, or IRC, serves as the interface between the local interconnect of a cluster and the global interconnect. Its primary function is to control the routing of intercluster messages. External to the optical routing element, this unit converts optical signals to electrical form and back to optical after making a routing decision. It acts as a pseudo dual ported processor-memory element between the interconnects, issuing and receiving requests to/from both simultaneously. Each IRC consists of six major component blocks: the Local Directory Unit, the Local Cluster Buffer Unit, the Combining Buffer Unit, Read Buffer Unit, the Write/Response Buffer Unit, and the Global Routing Directory Unit. A block diagram of the IRC is shown in Figure 4.11.

The Local Directory Unit is responsible for translating physical memory addresses...
to logical cluster addresses used to decide whether references on the local interconnect are local or remote. As the membership of clusters change upon reconfiguration, it associates clusters’ logical addresses with the physical locations of processors and memories making up the clusters. This feature is needed to support system reconfiguration. A simple physical-to-logical address mapping table consisting of only $2^k$ entries (number of clusters) can be used in the translation process. The Local Directory Unit also packages remote references into proper message format, queues them in the appropriate out-bound transaction buffer unit, and sends a processor stall signal across the local interconnect acknowledging that the request is pending. It is also responsible for updating the Combining Buffer Unit for remote reads and appending this information to read response messages sent to the local interconnect from the Local Cluster Buffer Unit.

The Local Cluster Buffer Unit is responsible for issuing in-bound global interconnect requests onto the local interconnect. The Combining Buffer Unit merges read
references made to the same block by cluster processors. It reduces contention and hot-spotting in the network by filtering out subsequent reads to a block during the interval of time between the initiation and completion of the first read request to the block. Transaction buffer units responsible for initiating out-bound global interconnect requests are the Read Buffer Unit and the Write/Response Buffer Unit. Depending on the communication model used, the IRC will either have one set (for \(k\)-B/*) or \(k\) sets (for \(k\)-T/*) of these buffers. Response messages from reads are queued in the same buffer as write messages since they have similar formats (see Section 4.1.2). Buffers are used alternatively on every ownership cycle unless one buffer is empty. Transaction buffers are separated since read messages are half the size of write and response messages. Therefore two read buffer messages are transmitted when the Read Buffer Unit is accessed. The transaction buffers receive messages from the Local Directory Unit as well as from the Global Routing Directory Unit.

The Global Routing Directory Unit is responsible for determining whether incoming messages from the global interconnect are to be accepted by the cluster and where to queue them. It does so by using the logical address of the cluster stored in this unit, the header portion of the message, and a routing directory table which grows as \(O(2^k)\). Hence, routing is static within each new configuration. If the IRC owns its links and the buffer to which the message would go is empty, the message by-passes the buffer and goes directly to the global interconnect after the header is received. Thus, the IRC has virtual cut-through routing capability as well as store-and-forward \([104]\).

Messages are accepted if the cluster is directly enroute to the destination or if it is the destination. Message selection arises only in the \(k\)-B/1-R and \(k\)-B/\(k\)-R models which broadcast to neighboring links rather than transmitting messages only to links directly in the destination path. In these models, the Global Routing Directory Unit performs a routing directory table look-up which specifies the set of supernodes designated as message propagators for all originator/destination cluster tuples. This is done even for Fetch&Add (synchronization) references. If found to be in this set of supernodes, the cluster receives the message.

In the \(k\)-T/1-R and \(k\)-T/\(k\)-R models, the IRC’s Global Routing Directory Unit
automatically accepts the message and decides which set of $k$ transaction buffers (which hypercube dimension) to queue the message if not destined for the cluster. The logical address of the cluster is compared with the cluster destination address found in the message header to determine whether the cluster is the message's destination. If so, the Global Routing Directory Unit forwards the message to the Local Cluster Buffer Unit.

4.3 Optical Routing Unit

In this section, the imaging optics required by GLORI's optical routing unit to function as the physical interconnect switch fabric is described. As the following analysis shows, the beam steering element of GLORI's optical routing unit must have the capacity to image $O(N \times M \times B)$ unique source-to-detector optical beams. The number of source and detector lines required is $O(N \times B)$. The local interconnect requires $O(M^2/\log(N/M))$ times the beam steering element imaging requirements of the global interconnect. It also requires as much as $O(M/\log(N/M))$ times the source and detector lines of the global interconnect. Chapter 5 suggests a beam steering element and optical system configuration that can implement the imaging optics required of the ORU.

4.3.1 Local Interconnect Imaging Optics

Functionally, GLORI's local interconnect imaging optics is required to image (project, in focus, a representation of) the source beams of $N$ processors and $N/M$ local IRCs onto respective detectors of $N$ memory and $N/M$ local IRCs in accordance with the spatial interconnect pattern of $N/M$ separate shared buses. In doing so, the ORU's beam steering element is required to broadcast image each of the $N/M$ processor or IRC bus master's signals to sets of $M$ slave nodes. Sets for processor bus masters include $M - 1$ memory slave nodes plus the local IRC slave node (see Figure 4.12); sets for IRC bus masters contain $M$ cluster processor slave nodes (see Figure 4.13).

Sets contain some permutation of the nodes within a cluster minus the master
CHAPTER 4. GLORI STRATEGY – ARCHITECTURAL DESCRIPTION

Local Bus Processor Ownership

- Local Bus Implementation (Processor Master).

Local Bus IRC Ownership

- Local Bus Implementation (IRC Master).

Figure 4.12: Local Bus Implementation (Processor Master).

Figure 4.13: Local Bus Implementation (IRC Master).
CHAPTER 4. GLORI STRATEGY – ARCHITECTURAL DESCRIPTION

node. Imaging to nodes within the sets occurs sequentially according to the time-multiplexing bus access protocol previously described. Hence, the beam steering element is required to simultaneously image only \( N/M \) master nodes (one in each cluster) onto \( (N/M) \times M = N \) slave nodes at any given time. However, system-wide it must be able to image all \( N + N/M \) sets of \( M \) optical beams as, potentially, all \( N \) processors and \( N/M \) IRCs can become bus masters. As a static bus router—i.e., one that is not reconfigured after each bus ownership slot—the beam steering element’s imaging capacity increases from \( (N/M) \times M \) to \( (N + N/M) \times M \).

Under the assumption that local buses have widths of \( B \) bits, master and slave nodes have \( B \) distinct sources and detectors, respectively (1 per bit). The beam steering element must image each of these sources to their respective detectors, increasing its imaging capability by a factor of \( B \). Moreover, to implement the bit-mapped bus arbitration scheme, \( N + N/M \) additional sources must be imaged to sets of \( M \) receivers. Thus, for the local bus, the ORU’s beam steering element must have a minimum imaging capacity described in terms of space-bandwidth-product, \( SBWP_{local} \), of

\[
SBWP_{local} = [B \times (N + N/M)] \times M + (N + N/M) \times M. \tag{4.4}
\]

The space-bandwidth-product\(^5\) specifies the number of unique source-to-detector imaging channels implementable by the beam steering element. This is a lower bound on \( SBWP_{local} \) for GLORI operating in static mode (no reconfiguration). It may be increased by as much as a factor of \( C \), the number of shared bus-hypercube configurations, depending on the method of reconfiguration used in dynamic operation of GLORI. The total number of source lines, \( S_{local} \), and detector lines, \( D_{local} \), required by the local interconnect irrespective of GLORI’s operating mode is

\[
S_{local} = D_{local} = [(B + 1) \times (N + N/M)] \tag{4.5}
\]

For example, assuming \( N = 64, M = 8, \) and \( B = 8 \) results in \( SBWP_{local} = 4,680, \)

\(^5\)This definition of SBWP is not to be confused with that used to describe resolution requirements of an optical recording device or an optical component, e.g., number of pixels in a hologram [114]. Rather, it is used to estimate, for instance, the number of independent channels of the interconnect.
$S_{local} = 648$, and $D_{local} = 648$. The significance of these expressions as applied to implementation and performance issues is discussed in Chapters 5 and 6.

### 4.3.2 Global Interconnect Imaging Optics

GLORI's global interconnect imaging optics are required to image the source beams of $N/M$ global IRCs onto the detectors of $N/M$ global IRCs arranged in a hypercube. Master supernodes image to sets of $k = \log_2(N/M)$ slave supernodes, one in each hypercube dimension. These sets overlap. Supernodes have as few as $1 \times B$ but as many as $k \times B$ source lines and detector lines, depending on the communication model. The model also determines whether imaging is done in parallel or sequentially by supernodes. (Multiple numbers of sources and receivers at the nodes enable parallel imaging.)

In a $k-B/k-R$ dedicated link system, each master supernode with only $B$ sources is required to broadcast image to $k$ slave supernodes. This is as opposed to a $k-T/k-R$ dedicated link system wherein each master supernode has $k \times B$ sources, $B$ in each of the $k$ outgoing hypercube dimensions, which separately image to the $k$ slave supernodes. Each slave supernode in both dedicated link systems have $k \times B$ detectors, $B$ in each of the $k$ incoming hypercube dimensions which receive in parallel. Hence, the beam steering element of a $k-B/k-R$ dedicated link system is required to image $B \times (N/M)$ master supernode sources onto $B \times [(N/M) \times k]$ slave supernode detectors simultaneously. Thus, for a $k-B/k-R$ system, it must have a minimum space-bandwidth-product, $SBW P_{global}$, of

$$SBW P_{global} = B \times [(N/M) \times k].$$

(4.6)

The total number of source lines, $S_{global}$, required by a $k-B/k-R$ global interconnect is

$$S_{global} = B \times [(N/M) \times k]$$

(4.7)

and the total number of detector lines, $D_{global}$, is

$$D_{global} = B \times [(N/M) \times k].$$

(4.8)
CHAPTER 4. GLORI STRATEGY - ARCHITECTURAL DESCRIPTION

Assuming $N = 64$, $M = 8$, and $B = 8$ in a $k$-$B/k$-$R$ system results in $SBWP_{global} = 192$, $S_{global} = 64$, and $D_{global} = 192$.

The required imaging optics (capacity of beam steering element and number of sources and detectors) for the other communication models are evaluated in a similar manner. Table 4.2 summarizes the results. It is noted that $SBWP_{global}$ is the same for all models whereas $S_{global}$ and $D_{global}$ differ. $SWBP_{global}$ is a function of interconnect topology only. The number of lines is a function not only of the topology used, but the model used as well.

Table 4.2: Imaging Optics Required of Communication Models

<table>
<thead>
<tr>
<th>Imaging Optics</th>
<th>Communication Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$k$-$T/k$-$R$</td>
</tr>
<tr>
<td>$SBWP$</td>
<td>$B \times [(N/M) \times k]$</td>
</tr>
<tr>
<td>$S_{global}$</td>
<td>$B \times [(N/M) \times k]$</td>
</tr>
<tr>
<td>$D_{global}$</td>
<td>$B \times [(N/M) \times k]$</td>
</tr>
</tbody>
</table>

4.4 Summary

The architecture of the global-local optical reconfigurable interconnect strategy is described in this chapter. The GLORI strategy provides the means to align the communication patterns of parallel algorithms with the interconnect architecture of multiprocessor systems. It does so by exploiting low-level features of optical technology as well as high-level characteristics of parallel programs. Its features include a hierarchical interconnect topology, a minimal delay routing protocol, high bandwidth optical links, limited combining, clusterability, broadcastability, and reconfigurability. This interconnect is designed to reduced communication latency and lower interconnect cost.

The proposed GLORI organization is a 2-tier hierarchical shared bus-hypercube configuration. The local interconnect is composed of clusters of shared buses, each
of which connect a subset of the total number of processor-memory nodes in the system. The global interconnect is a hypercube which connects the shared bus clusters for remote communication throughout the system. Global links can be shared or dedicated, depending on the interconnect resources allocated. The optical router must have the capacity to image $O(N \times M \times B)$ unique source-to-detector optical beams in order to function as the physical interconnect switch fabric. The optical routing unit must also be reprogrammable to allow block-frequent reconfiguration of the interconnect.
Chapter 5

GLORI Implementation

The previous chapter describes architectural features of the GLORI strategy. In this chapter, the feasibility of its implementation is evaluated. A single-pass static optical geometry composed of elementary optical components is presented for analysis. The analysis of a multi-pass reconfigurable optical geometry is much more complex and beyond the scope of this work.

An optical interconnect such as GLORI can be implemented a number of ways using a variety of components. The source of optical beams which encode signals with high intensity (a bright illumination) to represent a “1” and low intensity (a dark illumination) to represent a “0” can be electro-optical transmitters that operate at rates in excess of $1\text{GHz}$. Highly sensitive optical receivers can collect optical beams and convert them back to electronic signals also at rates above $1\text{GHz}$. Single-mode and multi-mode graded-index fibers sustain over $1\text{GHz}$ optical bandwidth for distances less than $1\text{Km}$, guiding optical beams with very low loss and low pulse broadening. Beam steering elements based on diffractive optics (gratings, holograms), reflective optics (deformable mirrors), or refractive optics (directional couplers, prisms, lenses) can efficiently re-orient the direction of incident light in free-space as shown in Figure 5.1.

In this work, the active optical components assumed for beam creation and collection are laser diodes and photodiodes. The passive optical components assumed for propagation and steering are single- and multi-mode graded-index fibers, free-space,
CHAPTER 5. GLORI IMPLEMENTATION

and holographic diffractive elements. The potential advantages of using these specific devices are discussed in the literature [106, 107, 108, 109]. The proposed optical system is not intended to be optimal; rather, it serves to show feasibility of the GLORI strategy from an implementation standpoint. We show that such a system can feasibly interconnect medium-scaled multiprocessors (on the order of 100s of processors) in a static GLORI configuration.

The principal component within the proposed optical system is the beam steering element. By it, the interconnect pattern is established, maintained, and reconfigured. The feasibility analysis performed in this chapter therefore centers around this aspect of the system. To simplify an otherwise complicated analysis, we examine the feasibility of a static, non-reconfigurable GLORI system which employs holographic techniques for beam steering. Results of this analysis can be generalized and extended to dynamic GLORI implementations which employ holographic as well as non-holographic beam steering techniques.

In the next section, holography as a technique for building optical interconnect
networks is introduced. In Section 5.2, the implementation of optical bus and hypercube networks using multi-element, multi-faceted holograms is presented. The feasibility of the proposed implementation is discussed in Section 5.3.

5.1 Holographic Beam Steering Element

Principles of holography are introduced in this section. The theory and practice of recording interference patterns, reconstructing light waves, and encoding holograms for the purpose of optical interconnects is presented.

5.1.1 Recording Interconnect Patterns

Holography is a method of recording the interference pattern resulting from the intersection of coherent light waves such that the original waves can be reconstructed upon proper hologram illumination. Interference patterns take the form of bright and dark fringes which represent the intensity distribution of the interfering light waves. Intensity\(^1\), \(I\), is equal to the squared magnitude of the complex amplitude of light, \(a = ae^{i\phi}\), where \(a\) is the amplitude and \(\phi\) is the phase of the light wave which is dependent upon position for light of a certain wavelength, e.g., \(\phi(x, y, z)\). For two light waves \(a_1\) and \(a_2\) from a common source intersecting in a plane, the combined intensity is given by

\[
I = a \cdot a^* = (a_1 e^{i\phi_1} + a_2 e^{i\phi_2})(a_1 e^{-i\phi_1} + a_2 e^{-i\phi_2})
= a_1^2 + a_2^2 + a_1 a_2(e^{i(\phi_2 - \phi_1)} + e^{-i(\phi_2 - \phi_1)})
= I_1 + I_2 + 2a_1 a_2 \cos(\phi_2 - \phi_1).
\]

The intensity of two intersecting light waves is the sum of the individual intensities plus an interference term which contains relative phase information as given by the cosine term. Holograms record this relative phase information along with amplitude.

\(^1\)Intensity is understood to be the time average of the magnitude of energy flow per unit time (optical power) per unit area normal to the flow [110].
information of interfering light waves which allows the reconstruction of the original object waves via an encoding reference wave.

A record of an interference pattern, i.e., for a particular interconnect topology, is preserved by the hologram as alterations in its optical transmission. Holograms encode interference patterns as amplitude or phase gratings which define a transmittance function. Hologram transmittance of an amplitude grating, $t$, is defined as the ratio of light amplitude transmitted by the hologram to light amplitude incident on the hologram. It is proportional to the spatially varying intensity function of the interfering light waves. For example, for a reference wave, $r = r \exp(i\phi_r)$, and an object wave, $o = o \exp(i\phi_o)$, the transmittance of the recorded hologram diffracts incident light amplitude according to the following:

$$t \propto (r + o)(r + o)^*$$
$$\propto rr^* + oo^* + ro^* + r^*o$$
$$\propto I_r + I_o + ro^* + r^*o$$
$$\propto I_r + I_o + 2ro \cos(\phi_r - \phi_o). \quad (5.2)$$

For interconnection purposes, the object and conjugate reference beams are generated by point sources which produce diverging spherical waves. The phase of a spherical wave (approximated to first order and ignoring a constant phase factor) is
given by the following for the coordinate system shown in Figure 5.2:

\[
\phi = -\frac{K}{2z_r} [(x' - x)^2 + (y' - y)^2],
\]

where the wave propagation constant, \(K\), is equal to \(2\pi/\lambda\), and \(\lambda\) is the wavelength of the light. For clock-wise rotating phasor representations, the sign of the phase is negative for a converging spherical wave and positive for a diverging spherical wave. The object-reference phase difference at coordinates \((x', y', 0)\) in the hologram plane from interfering diverging object and converging reference waves located at coordinates \((x_o, y_o, z_o)\) and \((x_r, y_r, z_r)\) in the input plane (where \(z_o = z_r\) is

\[
\phi_r - \phi_o = -\frac{K}{2|z_r|} \left[ \frac{1}{(x' - x_r)^2 + (y' - y_r)^2} \right]^2 + \left( \frac{1}{(x' - x_o)^2 + (y' - y_o)^2} \right]^2.
\]

In order for a hologram to record the spatially varying intensity distribution of a disturbance, its medium must have a resolution at least equal to the finest interference fringe produced. The fringe frequency a distance \(x'\) from the center of a hologram in the \(x\) direction, \(f_{x'}\), is governed by the relative phase difference given in Equation (5.4) according to the following:

\[
2\pi f_{x'} = \frac{\partial(\phi_r - \phi_o)}{\partial x'}. \tag{5.5}
\]

For an axial converging reference wave \((x_r = y_r = 0, z_r)\), the expression for fringe frequency evaluates to

\[
f_{x'} = \frac{x' - x_o}{\lambda z_r}. \tag{5.6}
\]

A similar fringe frequency, \(f_{y'}\), is observed in the \(y\) direction.

One component of Equation (5.6) is the minimum fringe frequency required of the hologram for on-axis or paraxial imaging as determined by the width of its aperture, e.g., \(|x'| \leq D_h/2\) where \(D_h\) is the hologram diameter or width. Another component is that required for off-axis imaging of light deflected at an angle of \(\theta_h\) from the hologram normal. A hologram that images on-axis is also called a Gabor hologram.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
and one that images off-axis is alternatively called a Leith-Upatnieks hologram [111].

For thin holograms operating in the Raman-Nath regime, the deflection angle is a function of the hologram fringe frequency and the wavelength of light:

$$f_{x'=0} = \frac{\sin \theta_h}{\lambda}, \quad (5.7)$$

therefore,

$$\theta_h = \arcsin(\lambda f_{x'=0}). \quad (5.8)$$

The higher the grating frequency, the larger the deflection angle and the larger the possible interconnect area.

### 5.1.2 Reconstructing the Recorded Interconnect Patterns

The grating pattern of a hologram as defined by the transmittance function determines how light is diffracted into various diffraction orders upon hologram illumination. The object beams for desired interconnections are typically a particular diffraction order, usually the positive 1st order. Diffraction efficiency is maximized when the hologram transmittance minimizes the optical power transferred to all other orders.

Reconstruction of the original object wavefront(s) occurs when a hologram with transmittance $t$ of Equation (5.2) is illuminated by the conjugate of the reference wave, $r^* = r \exp(-i \phi_r)$. Under these circumstances, the complex field amplitude of the wave transmitted through the hologram, $w$, is the product of the conjugate reference wave and the hologram transmittance:

$$w = tr^* \propto r^*(I_r + I_o) + \overline{(rr^*)o^*} + (r^*r^*)o. \quad (5.9)$$

The three terms in Equation (5.9) represent the 0th, +1st, and −1st order diffraction patterns of the illuminated hologram, respectively. The second term, the +1st order diffraction pattern, is the reconstructed object beam in conjugate form, $o^*$, which produces a real image scaled in amplitude by a factor, $I_r$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Space-invariant Imaging

A system is said to be space-invariant if, given that an input $a(x, y)$ produces output $b(x, y)$, then $a(x - u, y - v)$ results in the output $b(x - u, y - v)$, where $u$ and $v$ are constants [110]. For space-invariant holographic imaging systems, the read-out reference beam need not be the conjugate of the recording reference beam exactly in order to reconstruct the object beam. The conjugate reference wave can be located at various positions within the input plane and still yield a real image of the object beam converging to a different point. The area over which the optical system is space-invariant is called an isoplanatic patch. Shown in Figure 5.3 is an isoplanatic patch of a transmission hologram.

For a translated read-out reference beam, $q^* = r \exp(i \phi_q)$, where the diverging spherical wave's phase is modulated in position according to

$$\phi_q = \frac{K}{2z_r} \left[ (x' - (x_r + \delta_x))^2 + (y' - (y_r + \delta_y))^2 \right], \quad (5.10)$$

the complex field amplitude exiting the hologram is

$$w = i q^* \propto q^* (I_r + I_o) + (r q^*) o^* + (r^* q^*) o. \quad (5.11)$$

Ignoring the finite extent of the hologram aperture, the reconstructed object beam (+1st order diffraction pattern) emanating from the paraxially illuminated hologram
(x_r = y_r = 0, z = z_r) has a field given by

\[ w_{+1}(x', y') = (rq^*)o^* = I_o \exp(i[\phi_r + \phi - \phi_o]) = I_o \exp(-i\frac{K}{2z_r}[(x' - (x_o - \delta_x))^2 + (y' - (y_o - \delta_y))^2]). \]  (5.12)

Hence, the reconstructed real image of the object is found in the (x, y) plane translated by an amount -\delta_x and -\delta_y away from its original position. The above result holds when object and reference waves originate at the same distance normal to the hologram (z_r = z_o) and thin hologram regime diffraction is assumed.

**Diffraction-limited Spot Size**

In the preceding analysis, an infinite hologram aperture was assumed. The intensity distribution of the diffraction patterns actually is a function of the finite aperture of the hologram. Using Fresnel diffraction analysis, one can find the field amplitude and the resulting intensity distribution of the diffracted light in the (x, y) plane a distance z_r normal to a hologram of a given aperture \cite{112}. The +1st order diffraction term has a field amplitude in the image plane described by

\[ w_{+1}(x, y) = A \int_{-\infty}^{\infty} w_{+1}(x', y') \exp(i\frac{K}{2z_r}[(x - x')^2 + (y - y')^2]) dx' dy', \]  (5.13)

where A is a constant complex amplitude and the normalized field amplitude just beyond a square apertured hologram of width D_h is

\[ w_{+1}(x', y') = \exp(-i\frac{K}{2z_r}[(x' - x_o)^2 + (y' - y_o)^2]) \text{rect}\left(\frac{x'}{D_h}\right)\text{rect}\left(\frac{y'}{D_h}\right). \]  (5.14)

It can be shown that the diffracted first order field amplitude is simply a Fourier transform of the aperture evaluated at f_x and f_y spatial frequencies which depend on

\footnote{Space-invariant imaging reduces the diffraction efficiency of volume holograms as the Bragg condition is not satisfied.}
The intensity distribution of the $+1^{st}$ order diffraction is the product of the field amplitude and its conjugate, which yields a real valued $\text{sinc}^2$ function shown in Figure 5.4 and given by the following expression:

$$I_{+1^{st} \text{Order}} = (I_{+1}) \text{sinc}^2(D_h f_x) \text{sinc}^2(D_h f_y),$$

where $I_{+1} = A^2$ which is a real valued intensity. The diffraction-limited spot size, $w_{\text{spot}}$, of the positive first order diffraction can be found from the intensity distribution. This is a measure of the focusing power or output resolution of the hologram and is defined to be the width of the main lobe between the first two zeros of the intensity distribution:

$$w_{\text{spot}} = \frac{2\lambda z_r}{(D_h) \cos^2 \theta_h}.$$ 

The spot size is enlarged by a $\cos^{-2} \theta_h$ factor for off-axis imaging to account for the angular projection onto a plane parallel to the hologram ($0^\circ \leq \theta_h < 90^\circ$) [108].
shown in Figure 5.4, the diffraction-limited spot size contains nearly all of the optical power transmitted to the $+1^{st}$ diffraction order.

### 5.1.3 Hologram Encoding

Hologram encoding can be done by an optical recording system or by a computer-based plotting system, e.g., electron-beam lithography scanning system, which produces a computer-generated hologram or CGH [108, 113, 114, 115]. Various tradeoffs are associated with each. Optically recorded holograms are constructed by exposing photosensitive material to a reference beam and one or more object beams. A record of the resulting interference pattern that defines the hologram transmittance function is stored once the material is developed. Fabrication of multi-element holograms for interconnects can be quite costly in terms of time since many complicated exposure steps are necessary to record all the connection patterns. Advantages of optically recorded holograms are the high diffraction efficiencies (up to 90% for volume phase holograms made with dichromated gelatin [110, 116]) and the small emulsion grain size of $O(10nm)$ which enables large deflection angles ($\approx 65^\circ$ [113]) and low $f$-number holograms.

The transmittance function of a CGH is generated by a computer (CAD station) and then either photographically transferred to a mask which is subsequently photoreduced to the actual size of the hologram or plotted to scale on a mask using a scanning system with sub-micron resolution such as E-beam lithography techniques. Masks can then be made into transmission or surface-relief reflection holograms. A significant advantage of this hologram encoding method is that the entire process is compatible with standard VLSI integrated circuit fabrication procedures, which reduces its cost. A potential drawback, however, is that the minimum feature size of the recording device may limit the maximum deflection angle of the hologram and limit how low the $f$-number of the hologram can be, which limits the resolution of the image spot size. Because of the small feature sizes of E-beam lithographic systems (> 2000 line pairs/mm or $10^8$ pixels per $1cm^2$ hologram), CGHs perform commensurate

---

3The $f$-number is defined as the ratio of the hologram focal length to the hologram aperture width.
to optically recorded holograms in most respects except in diffraction efficiency, which is typically lower for thin CGHs \((\approx 10-30\% [114, 115])\).

An example grating structure with a straightforward computer-based encoding is a Fresnel Zone Pattern (FZP) [115, 117, 118, 119, 120]. A FZP, shown in Figure 5.5 is composed of a set of concentric circular fringes with radii, \(R_m = \sqrt{z_m \lambda m}\), where \(m\) is the zone number. With this encoding, which is very efficient for paraxial imaging, the path difference from the image point to each consecutive fringe is \(\lambda/2\). Thus, blocking alternate zones gives a large irradiance at the image point for imaging over relatively small angles. Encodings other than FZPs that approximate lens transmittances for non-paraxial imaging are also possible with computer-based systems using Fresnel diffraction analysis, though not as simple [114, 115].

![Figure 5.5: Fresnel Zone Pattern.](image-url)
5.2 Optical Bus and Hypercube Implementation

A conceptual overview of an optical system that sufficiently implements GLORI's shared bus-hypercube interconnect is provided in figures 5.6 and 5.7. Resident on processor-memory boards are electro-optical multiplexer and opto-electronic demultiplexer drive circuits (see Figure 5.6). The multiplexer driver accepts $B'$ electronic signals in parallel and transmits them optically as a serial bit-stream at $B'$ times the input rate. The reverse is true of the demultiplexer driver. Driver circuits sufficient to sustain GLORI's bandwidth and bit-line parallelism has been demonstrated by Rockwell [121], which built a $B' = 8$ bit, $3GHz$ multiplexer/demultiplexer chip set.

Butt-coupled to optical sources and detectors on processor-memory boards (using V-grooves [122] or other techniques) are source fibers and detector fibers, which connect to the optical routing unit. Housed in the optical routing unit are the imaging optics for performing GLORI's interconnect routing via free-space beam steering (see Figure 5.7). Light beams emerging from source fiber ends are individually imaged onto the beam steering element by an array of microlenses. The beam steering element diffracts light at various angles onto an array of detector fibers at the output of the unit. The dimensions of the imaging optics are on the order of centimeters for the one-way optical path length which spans on the order of meters, as proved at the end of this section.

An elementary beam steering element capable of performing the necessary imaging

---

4A transmissive imaging system rather than a reflective system is assumed because it is simpler to describe.
Figure 5.7: Optical components along the interconnect path, where circles represent portions of the hologram illuminated.

Figure 5.8: Hologram with four elements each with four facets.
for both the local and global interconnects is a multi-element, multi-faceted hologram [108] illustrated in Figure 5.8. The complete hologram consists of multiple subhologram elements which have multiple hologram facets. Space-variant imaging, wherein the relative interconnect pattern mapping source points to detector locations varies over the image space, is used to configure the nodes in the shared-bus or hypercube topology. Space-invariant imaging, wherein the relative interconnect pattern remains constant over the image space, is used to image the bit lines of each node in the same configuration.

The hologram substrate is made up of two separate sections: one for the local buses and the other for the global hypercube. The bus interconnect portion of the hologram substrate is divided into a two-dimensional array of \( N + N/M \) subhologram elements, each with \( M \) facets for \( M \)-fold fan-out upon illumination (see Figure 5.9). Recall that \( N \) is the number of processor-memory nodes, \( N/M \) is the number of intercluster routing controllers (IRCs), and \( M \) is the cluster size. Each processor node illuminates
a different subhologram for space-variant imaging. Each subhologram element has a transmittance function which approximates the transmittances of $M$ lenses, each with a different point-spread-function (PSF). The PSF defines the unique spatial interconnect pattern of each transmitter/receiver node pair. The portions of the subholograms which approximate each lens element are individual hologram facets.

All $B + 1$ sources of a transmitter node ($B$ bit lines and $1$ arbitration line) illuminate the same subhologram designated for that node. Since each facet of the subhologram has a unique PSF, an identical spatial interconnect pattern for each transmitter/receiver node pair is applied to all $B + 1$ optical beams, however translated laterally in opposite directions a distance equal to the bit line separation for space-invariant imaging. Thus, each of the $(N + N/M)$ subhologram elements is a multiple-image hologram that images each source beam to $M$ receivers and up to $B + 1$ source beams to $(B + 1) \times M$ receivers in parallel.

The hypercube interconnect portion of the hologram aperture is similarly divided into a two-dimensional array of subhologram elements as illustrated in figures 5.10 and 5.11 for $k$-way broadcast and $k$-way transmit systems. Each IRC transmitter node illuminates a different subhologram for space-variant imaging. The lens-like facets each have a unique PSF, and all bit-line sources of an IRC transmitter node illuminate the same subhologram for space-invariant imaging. The number of subholograms in the array depends on the communication model used. For $k$-way transmit systems (either shared link or dedicated link), a total number of $(N/M) \times k$ single-faceted subholograms make up the array. For $k$-way broadcast systems (either shared link or dedicated link), a total number of $(N/M)$ multi-faceted subholograms each with $k$ facets for $k^\text{th}$ fan-out are required.

Parallel space-invariant imaging is shown to be possible following the analysis of Section 5.1.2. For $B + 1$ read-out reference beams with phases modulated in position by $\delta_x$ and $\delta_y$, the complex field amplitudes are given by $q^* = r \sum_{j=1}^{B+1} \exp(i \phi_{y_j})$. The complex field amplitude exiting a hologram with transmittance $t$ is

$$w = tq^* \propto q^*(I_r + I_o) + (rq^*)o^* + (r^*q^*)o. \quad (5.18)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Beam Steering Element: Global Hyp (k-B/\*)

Virtual Source Plane

Holographic Imaging Plane (N/M subholograms each with k facets)

Virtual Detector Plane

Figure 5.10: Holographic optical k-way broadcast hypercube.

The +1\textsuperscript{st} order diffracted waves leaving the paraxially illuminated hologram are

\[ w_{+1}(x', y') = I_{r0} \sum_{j=1}^{B+1} \exp(-i(K/2z_{r}))(x' - (x_{o} - \delta_{x_{j}}))^{2} + (y' - (y_{o} - \delta_{y_{j}}))^{2}(3.19) \]

Hence, multiple read-out beams can simultaneously illuminate an isoplanatic patch to produce multiple reconstructed object beams without destructively interfering in the hologram plane. Along with the original object beams are generated B + 1 undesired 0\textsuperscript{th} and -1\textsuperscript{st} diffraction orders. These light waves result in background noise in the detector fiber plane which potentially could interfere with the detection of the signal beam. This crosstalk issue is further explored in Section 5.3.2.

In Chapter 4, Section 4.3, hologram imaging capacity was described in terms of space-bandwidth-product (SBWP). It was found that the number of unique source-to-detector imaging channels required to implement the local and global interconnects...
are

\[ SBW_{local} = (B + 1)(N + N/M)M \] and
\[ SBW_{global} = B(N/M)k. \] (5.20)

In space-invariant imaging, the bit lines imaging channels are multiplexed through free-space in parallel. This saves a factor of \( B + 1 \) and \( B \) facets used to implement the imaging channels. Hence, the total number of hologram facets needed to implement the local interconnect is

\[ \text{Facets}_{local} = (N + N/M) \times M, \] (5.21)

and the number required to implement the global interconnect is

\[ \text{Facets}_{global} = (N/M) \times k \] (5.22)
hologram facets, where \( k = \log(2N/M) \).

For efficient coupling of light into detector fibers, the diffraction-limited output spot size of the hologram given by Equation (5.17) must be smaller than the fiber core diameter, \( d_{\text{core}} \):

\[
d_{\text{core}} > w_{\text{spot}} = \frac{2\lambda z_r}{D_h \cos^2 \theta_h},
\]

where \( d_{\text{facet}} \) represents the aperture width of each hologram facet instead of the entire hologram aperture as expressed by \( D_h \) in equation (5.17). As the facet aperture size decreases, the output spot size increases for a given imaging distance (\( z_r \)) due to diffraction effects. Thus, there is a lower bound on hologram facet width if a desired output resolution spot size is to be maintained. Figure 5.12 plots spot size verses imaging distance for various hologram facet aperture sizes. Assuming \( d_{\text{core}} = 100\mu m, \lambda = .85\mu m, z_r = 2cm, \) and \( \theta_h = 30^\circ \), the hologram facet width must be greater than 450\( \mu m \). The conservative value of 500\( \mu m \) is assumed for \( d_{\text{facet}} \) in the remainder of this work unless otherwise specified.

The area consumed by the hologram as well as the source and detector fiber planes can be modeled and estimated. For the bus interconnect portion of the hologram
substrate, each multi-faceted subhologram consists of $M$ hologram facets of width $d_{\text{facet}}$. If the facets are arranged in a 2-D array, the width of each subhologram, $d_{\text{subhologram}}$, is

$$d_{\text{subhologram}} \geq d_{\text{facet}} \times \lceil \sqrt{M} \rceil.$$ (5.24)

For the hypercube interconnect, subhologram width depends on the communication model used. It is equal to $d_{\text{facet}}$ for $k$-$T$/* systems and is equal to $d_{\text{facet}} \times \lceil \sqrt{k} \rceil$ for $k$-$B$/* systems. The entire multi-element, multi-faceted hologram has a minimal aperture size, $d_{\text{hologram}}$ (assuming zero spacing between subholograms arranged in a 2-D array on the hologram substrate) of

$$d_{\text{hologram}} \geq d_{\text{subhologram}} \times \lceil \sqrt{N + N/M} \rceil + \left\{ \begin{array}{ll} d_{\text{facet}} \times \lceil \sqrt{k} \rceil \times \lceil \sqrt{N/M} \rceil & \text{for } k$-$B$/* \\ d_{\text{facet}} \times \lceil \sqrt{k(N/M)} \rceil & \text{for } k$-$T$/* \end{array} \right\}.$$ (5.25)

The first term of Equation (5.25) represents the bus interconnect portion of the hologram; the second term represents the hypercube interconnect portion.

Fibers are mounted in a 2-D array with center-of-core to center-of-core spacing of $x_{\text{core}}$ to account for fiber core, cladding, outer jacket, etc. The source and detector fiber planes, therefore, have a cross-section of lateral dimensions, $d_{\text{source}}$ and $d_{\text{detector}}$, equal to

$$d_{\text{source}} = x_{\text{core}} \times \sqrt{S_{\text{local}} + S_{\text{global}}}.$$ (5.26)

and

$$d_{\text{detector}} = x_{\text{core}} \times \sqrt{D_{\text{local}} + D_{\text{global}}},$$ (5.27)

where $S_{\text{local}}$, $S_{\text{global}}$, $D_{\text{local}}$, and $D_{\text{global}}$ are the number of sources and detectors required by the local and global interconnects given in Chapter 4, Section 4.3. Typically, $x_{\text{core}} < d_{\text{facet}}$ for $d_{\text{core}} = 100\mu m$ so that if $\lceil \sqrt{B + 1} \rceil < \lceil \sqrt{M} \rceil$, the cross-section of the fiber plane is much less than that of the hologram plane. The unused area in the fiber plane can be used to trap background optical noise and mitigate optical crosstalk as discussed in Section 5.3.2.

For a $k$-$B$/$k$-$R$ GLORI system of $N = 64$ processor-memory nodes with cluster size of $M = 8$ and $B = 8$ bit-lines, the area of the source and detector fiber planes with
\( x_{\text{core}} = 300 \mu m \) is approximately equal to \( .8 cm \times .8 cm \). The entire hologram substrate width is minimally \( d_{\text{hologram}} = (33)d_{\text{facet}} \), and for a facet width \( d_{\text{facet}} = 500 \mu m \), the area of the hologram is \( \approx 1.65 cm \times 1.65 cm \). For a hologram-to-fiber plane orthogonal distance, \( z_r \), of \( O(1cm) \), the volume of the optical routing unit is \( O(1cm^3) \), far less than a comparable electronic shared bus-hypercube interconnect. Further comparisons of electronic and optical parameters in reference to the GLORI strategy are made in Chapter 6.

### 5.3 Feasibility of the Proposed Implementation

Thus far in this chapter, we considered the fundamentals of hologram formation, image reconstruction, and optical component integration for building GLORI's shared bus-hypercube. An optical system geometry was proposed that employs fiber optics and a multi-element, multi-faceted hologram for beam propagation and steering. In this section, the feasibility of the proposed design is examined and its maximum system size is evaluated.

The maximum fan-out for one-to-many interconnection and/or the maximum fan-in for many-to-one interconnection determines system size in a broadcast based interconnect such as GLORI. This is parameterized by the variable, \( M_{\text{max}} \). For example, the maximum number of nodes interconnected by a bus is \( M_{\text{max}} \) and the maximum number of supernodes in a hypercube is \( 2^{K_{\text{max}}} \), where \( K_{\text{max}} = \lfloor \log_2 M_{\text{max}} \rfloor \). Hence, the maximum number of processor-memory nodes in a GLORI multiprocessor system is

\[
N_{\text{max}} = M_{\text{max}} \times 2^{\lfloor \log_2 M_{\text{max}} \rfloor}.
\] (5.28)

As many as \( N_{\text{max}} = 256 \) nodes can be interconnected by a static GLORI system with restricted (non-arbitrary) cluster compositions. System size is shown to be fundamentally limited by optical splitting loss and practically limited by system geometrical constraints such as source fiber area of illumination, hologram facet field.
of view, fiber numerical aperture, and detector fiber placement. Table 5.1 summarizes values assumed for various optical component parameters of importance to this discussion.

Table 5.1: Optical Component Parameters.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>Assumed Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>laser diode diameter ($d_{LD}$)</td>
<td>60μm</td>
</tr>
<tr>
<td>laser diode power ($P_{in}$)</td>
<td>0dBm</td>
</tr>
<tr>
<td>laser diode divergence angle ($\theta_d$)</td>
<td>30°</td>
</tr>
<tr>
<td>laser diode wavelength ($\lambda$)</td>
<td>.85μm</td>
</tr>
<tr>
<td>receiver diameter ($d_{PD}$)</td>
<td>100μm</td>
</tr>
<tr>
<td>receiver sensitivity ($P_{out}$)</td>
<td>-30dBm</td>
</tr>
<tr>
<td>fiber core diameter ($d_{core}$)</td>
<td>100μm</td>
</tr>
<tr>
<td>fiber core separation ($x_{core}$)</td>
<td>300μm</td>
</tr>
<tr>
<td>fiber acceptance angle ($\theta_a$)</td>
<td>30°</td>
</tr>
<tr>
<td>microlens focal length ($f_l$)</td>
<td>1mm</td>
</tr>
<tr>
<td>hologram imaging distance ($z_r$)</td>
<td>2cm</td>
</tr>
<tr>
<td>hologram deflection angle ($\theta_h$)</td>
<td>30°</td>
</tr>
</tbody>
</table>

5.3.1 Optical Power Budget

Along the passive optical path, several factors including source-to-fiber, fiber-to-microlens, air-to-fiber and fiber-to-detector coupling loss, fiber transmission loss, hologram diffraction loss, and hologram splitting loss reduce output power received by the photodiodes. To sustain a given bit-error rate (e.g., $10^{-15} BER$), the optical link must operate within a minimal optical power budget

$$P_{in} - P_{out} = P_{loss} = 10 \log_{10} \frac{P_{in}}{P_{out}},$$

(5.29)

where non-primed terms are in decibels (dB) and primed terms are in mW. Optical power loss represents a fundamental limit restricting the system size of a broadcast-based optical interconnect as it provides an upper limit on the broadcastability of the
system.

Considering coupling loss, multimode fibers must have a large enough acceptance angle and core width to collect input optical power. An important parameter that relates the acceptance angle to the refractive indices of free-space (air) and the fiber core and cladding is the fiber numerical aperture (NA):

\[ \text{NA} = n_0 \sin \theta_a = \sqrt{n_1^2 - n_2^2}, \]  

(5.30)

where the index of refraction of the air, fiber core and fiber cladding are \( n_0, n_1 \) and \( n_2 \), respectively (for graded-index fibers, \( n_1 \) varies with core radius). The NA is a useful measure of the light-catching ability of a fiber as it defines the angular range over which inserted light propagates down the length of the fiber through internal reflection without being attenuated in the cladding. For GLORI's short interconnect length, a NA value of .5 can be assumed which yields \( \theta_a = 30^\circ \) as given in Table 5.1.

Coupling efficiency is maximized when light is inserted at an angle less than or equal to \( \theta_a \) and its beam width is less than the fiber core diameter, \( d_{\text{core}} \). For the LD-to-fiber interface, for example, \( \theta_d \leq \theta_a \) and \( d_{LD} < d_{\text{core}} \), where \( \theta_d \) is the LD divergence angle and \( d_{LD} \) is the laser beam width. Values given in Table 5.1 indicate that these relations are feasible. It is also noted that the fiber acceptance angle constrains the hologram deflection angle \( (\theta_h \leq \theta_a) \). This may limit system size also as explained in Section 5.3.2.

Fiber transmission loss is collectively material absorption (radiation), material scattering, and fiber bend radiation losses which total \( \approx 5dB/Km \) for typical multimode fibers [122]. Thus, it is reasonable to assume that transmission loss over distances on the order of meters is negligible. A conservative estimate for the aggregate fiber coupling and transmission loss is 6dB. This estimate, which also accounts for Fresnel reflection, longitudinal misalignment, lateral misalignment, and angular misalignment losses, agrees with theoretical and empirical results [123, 122].

Hologram diffraction loss is due to material absorption, material scattering, and diffraction into undesirable \( 0^{th} \), \(-1^{st} \), and higher diffraction orders. The maximum
diffraction efficiencies achieved to date are 32.6% for thin phase transmission holograms and 90.0% for volume phase transmission holograms [110]. Therefore, a conservative hologram diffraction loss of $6\, \text{dB}$ is assumed. In addition to this loss, a splitting loss of $3\, \text{dB}$ is incurred for every multiple of 2 optical fan-out ($+1^{\text{st}}$ order diffraction) from the hologram. Thus, hologram splitting loss is $(3\, \text{dB}) \log_2 M_{\text{fan-out}}$, where $M_{\text{fan-out}}$ is the degree of fan-out of the hologram.

For laser diode power $P_{\text{in}} = 0\, \text{dBm}$ and receiver sensitivity $P_{\text{out}} = -30\, \text{dBm}$ (appropriate for $1\, \text{Gbps}$ transmission speed), the optical loss allowed for proper signal detection can be calculated using Equation (5.29).\footnote{The notation $\text{dBm}$ specifies power in decibels referenced to $1\, \text{mW}$, e.g., $-30\, \text{dBm} = 0.001\, \text{mW}$.} Totalling the loss along GLORI’s optical path,

$$30\, \text{dB} \geq P_{\text{loss}} = 12\, \text{dB} \left[ \frac{1}{\text{coupling loss}} + \frac{(3\, \text{dB}) \times \log_2 M_{\text{fan-out}}}{\text{splitting loss}} \right]. \quad (5.31)$$

With this optical power budget, the maximum number of nodes that can share an optical bus is an estimated

$$M_{\text{fan-out}} = 64. \quad (5.32)$$

This is also the maximum number of supernodes in an optical hypercube for a $k$-way broadcast system. The two interconnects integrated into a two-tier hybrid topology makes possible a GLORI system with thousands of nodes. A GLORI system so large can be realized practically only if physical geometrical limits in implementing the optical system are circumvented.

### 5.3.2 Optical Geometry Constraints

**Source Fiber to Hologram Imaging**

One geometrical constraint limiting system size is the hologram area illuminated by the source fibers. This area determines the maximum aperture size of subholograms and therefore limits the number of facets per subhologram that can be used to distribute light to various interconnect points in the output image plane. The
broadcastability or fan-out of the system is thus bounded by the beam spread of the source fiber onto the hologram plane.

The light cone emerging at the output of the source fiber is approximately the same as that entering at its input, which maximally diverges at an angle $\theta_a$. Pseudo-collimating microlenses are used to lower beam divergence as light travels in the $z$ direction. The diameter of the light cone emerging from the fiber-microlens combination a distance $z_r$ away from its end, $d_{beam}$, is

$$d_{beam} = 2 f_i \tan \theta_a + \frac{d_{core} z_r}{f_i} + d_{core}, \quad (5.33)$$

where $f_i$ is the focal length of the microlens. Since each source fiber must illuminate a distinct subhologram as shown two-dimensionally in Figure 5.13, the ideal source fiber beam diameter at the hologram plane is greater than or equal to the subhologram width:

$$d_{subhologram} \leq d_{beam}. \quad (5.34)$$

Combining equations (5.24) and (5.34) gives an upper limit on the number of facets per subhologram or, equivalently, the degree of fan-out of the source fiber,
For the parameters assumed in Table 5.1, the subhologram area of illumination is \( \approx (3.25\,mm)^2 \) and the maximum number of facets within each subhologram for a facet width of \( d_{\text{facet}} = 500\,\mu m \) is \( M_{\text{fan-out}} = 36 \).

**Hologram to Detector Fiber Imaging**

Another geometrical constraint which influences system size is the maximum lateral separation distance connectable between source and detector fibers. This places bounds on the maximum fan-in capability of the optical system’s detector fibers.
As shown one-dimensionally in Figure 5.14, centers of nodes (bit-line detector fiber bundles) are separated in the output array by a minimum distance equal to the sub-hologram width (assuming it to be larger than the fiber bundle width). Only those nodes within the imaging range or field of view of a subhologram facet can be interconnected as determined by the lateral separation distance, \( x_o \), and the space occupied by each of the nodes, \( d_{\text{subhologram}} \). A large field of view where possible fan-in, \( M_{\text{fan-in}} \), equals \( N \) is desired so that any source node in the system can be imaged to any detector node for arbitrary cluster composition. For \( M_{\text{fan-in}} < N \), only a restricted set of nodes can be clustered.

The maximum number of nodes within the imaging range of a hologram facet, excluding the transmitting node itself, is given by the following equation:

\[
M_{\text{fan-in}} \leq \left( \frac{x_o}{d_{\text{subhologram}}} \right)^2 - 1
\]

\[
\leq \left[ \frac{z_r \tan \theta_h}{2f_t \tan \theta_a + \frac{d_{\text{facet}} - d_{\text{core}}}{f_t}} \right]^2 - 1. \tag{5.36}
\]

As shown in Figure 5.15, the lateral separation distance is dependent upon the hologram imaging distance, \( z_r \), and the hologram deflection angle, \( \theta_h \), by the following relation:

\[
x_o = z_r \tan \theta_h. \tag{5.37}
\]

Several factors, however, limit how large the hologram deflection angle and imaging distance can be for practical system geometries. The hologram fringe frequency discussed in Section 5.1.1 is one such limiter.

The hologram deflection angle is a function of the recordable hologram fringe frequency as given by Equation (5.8). Optical recording material and computer generated hologram recording facilities both have sub-micron feature sizes which enable \( \theta_h \) to be as large as \( \approx 65^\circ \). If hologram fringe frequency were the only limiter, separation distances up to 22 mm for \( z_r = 1 \text{ cm} \), \( w_{\text{spot}} = 100 \mu \text{m} \), and \( d_{\text{facet}} = 500 \mu \text{m} \) are possible and have been demonstrated [108]. This results in cluster sizes as large...
as $M_{fan-in} = 35$ nodes. However, when light is to be coupled to fibers instead of photodetectors directly, the hologram deflection angle is limited by fiber NA instead of the hologram grating period which ordinarily limits it.

As mentioned in Section 5.3.1, fibers can efficiently collect light waves that are incident at angles less than or equal to its acceptance angle, $\theta_a$, and have widths less than the core diameter, $d_{core}$. Equation (5.17) of Section 5.1.2 indicates that an direct relationship exists between output spot size and imaging distance so that $w_{spot}$ increases with $z_r$. To maintain $w_{spot} < d_{core} = 100\mu m$, $\theta_h$ must be less than or equal to $36^\circ$ for $z_r = 2cm$ and $d_{facet} = 500\mu m$. The hologram deflection angle is further constrained by the fiber NA. Light incident on the fiber at angles greater than $\theta_a$ is attenuated due to loss in the fiber cladding. Consequently, the hologram deflection angle can be no greater than the acceptance angle of the detector fibers: $\theta_h \leq \theta_a = \arcsin(NA/n_a) = 30^\circ$. This provides a maximum lateral source fiber to detector fiber separation distance of $x_o = 11.5mm$ for $z_r = 2cm$ and, from Equation (5.36), a maximum number of nodes within a facet's field of view of $M_{fan-in} \leq 8$.

**Optimizations**

Optimizing imaging distance, facet width, and wavelength within the limits of the maximum deflection angle and output spot size requirements results in $z_r = 3cm$, 

![Hologram to Detector Fiber Imaging](image)

Figure 5.15: Hologram facet to fiber imaging parameters.
$d_{\text{facet}} = 520 \mu m$, and $\lambda = 0.65 \mu m$. This optimized optical geometry increases the fan-out capability of source fibers and the fan-in capability of detector fibers to an estimated

$$M_{\text{fan-out}} = 64 \text{ and }$$  \hfill (5.38)

$$M_{\text{fan-in}} = 16. \text{ (5.39)}$$

The number of nodes within shared bus and hypercube clusters are bounded by $M_{\text{max}}$, where

$$M_{\text{max}} = \min [M_{\text{fan-out}}, M_{\text{fan-out}}, M_{\text{fan-in}}] = 16. \text{ (5.40)}$$

The area of $M_{\text{max}}$ subholograms with $M_{\text{max}} = 16$ facets of width $d_{\text{facet}} = 520$ is minimally $(.83 cm)^2$. The limited field of view provided by this optical geometry restricts cluster composition to be only those nodes within this azimuthal area. According to Equation (5.28), the total number of nodes in the shared bus-hypercube optical interconnect system is limited to $N_{\text{max}} = (16)(2^4) = 256$. The number of bit lines implementable per node has yet to be calculated. This number is shown to be limited by the amount of optical crosstalk allowable in the system.

**Optical Crosstalk**

A distinguishing feature of optical signals is the absence of EMI crosstalk with neighboring signals throughout propagation. However, other sources of crosstalk owing to the optical system geometry can reduce the observed signal-to-noise ratio (SNR) at the detector. Contributors to optical crosstalk are $0^{th}$, $-1^{st}$, and higher-order diffraction terms from the illuminating reference beam and the $+1^{st}$ order diffraction terms from neighboring bit-line reference beams. In order to maintain maximum

---

6Experimental laser diodes operating in the visible spectrum have yet to be introduced commercially.

---

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
SNR, power from non-desired diffraction orders must be relatively insignificant or be sunk into areas remote to optical detection. The feasibility of these conditions are examined below.

The simplest method of minimizing crosstalk from non-desired diffraction terms is to reduce the power transmitted to those orders to negligible levels. It was previously mentioned that transmission holograms with \( \approx 90\% \) diffraction efficiency into the \(+1^{st}\) order are implementable using thick dichromated gelatin photosensitive material. Space-invariant imaging using \( 0.85 \mu \text{m} \) light may dictate the use of thin phase transmission holograms with only \( \approx 33\% \) diffraction efficiencies. The rest of the optical power is absorbed by the hologram and is also diffracted into the \(0^{th}, -1^{st}, \) and higher orders which, for thin holograms, contributes significantly to background noise at various locations on the detector plane. To ensure no crosstalk by these orders, detector fibers must reside at targeted areas which do not coincide with background noise. Since each fiber occupies some portion of this area, a limited number of bit-lines are allowed per node as dictated by the geometry of the optical system.

Consider the filtering out of background noise generated from power transfered to the \(0^{th}\) diffraction orders. The beam spread at the detector fiber plane is given by Equation (5.33) with \( z \) replace by \( 2z_r \), the total distance traveled. If each node’s detector fiber bundle is placed opposite of its source fiber bundle, the majority of the \(0^{th}\) order power falls on the transmitting node’s inactive detector fibers and, therefore, is inconsequential. The remaining noise is leaked to nodes adjacent to the transmitting node. In order for signal beams to these nodes to not overlap the background noise, the following relation limiting the number of bit-lines must hold:

\[
B \leq \left[ \frac{2d_{subholo} - d_{beam}}{x_{core}} \right]^2. \tag{5.41}
\]

For the parameters assumed in Table 5.1, it is estimated that the number of bit-lines per node for maximum SNR is limited to

\[
B \leq 16. \tag{5.42}
\]
Reducing the +1\textsuperscript{st} order crosstalk of adjacent bit-line diffractions is also essential for high SNR since most of the optical power is distributed to this order. The normalized intensity distribution of the +1\textsuperscript{st} order diffraction pattern of a square aperture was shown to be a \text{sinc}\textsuperscript{2} function in Section 5.1.2, where the majority of power is located within the spot size width. The intensity of the side-lobes dampens to minuscule values tens of microns away from the main-lobe as shown in Figure 5.16. Plotted are the normalized intensity distributions of three adjacent bit-line reference beams of spot size $w_{\text{spot}} = 100\mu m$ and separated by $x_{\text{core}} = 300\mu m$. By spacing fiber cores far enough apart, bit-line crosstalk becomes negligible as shown to scale in the figure.

5.4 Summary

The giga-bytes per second bandwidth assumed by GLORI's architecture is achievable with straight-forward integration of elementary active and passive optical components. A multi-element, multi-faceted hologram that acts as an optical interconnect routing switch is pragmatically feasible for a shared bus-hypercube interconnect system of many hundreds of nodes. Space-invariant imaging saves a factor of $B$ hologram facets so that bit-parallel transmission is essentially “free” in terms of hologram costs.

System size is fundamentally limited by optical loss primarily due to optical beam splitting which provides a fan-out of $M_{\text{fan-out}} = 64$, assuming conservative values for
optical component parameters. System size is practically limited by optical geometry constraints which, conservatively, limit source fiber fan-out to $M_{\text{fan-out}} = 64$ and detector fiber fan-in to $M_{\text{fan-in}} = 16$. The maximum sized shared bus cluster and hypercube degree are bounded by the most severe of these limitations: detector fiber fan-in. Hence, a static GLORI system of $N = 256$ processor-memory nodes is feasibly implementable.

Detector fibers' limited field of view is primarily attributed to fiber numerical aperture which restricts the hologram deflection angle to be no greater than the fiber acceptance angle. If this constraint were lifted (as would be the case if detector fibers were bypassed and the photodiodes were imaged to directly), optimizing the optical geometry would result in $M_{\text{max}} = 36$ and $N_{\text{max}} = 1,152$, which is now bounded by the output resolution spot size. The point at which the system is optimized occurs when $M_{\text{fan-out}} = M_{\text{fan-in}}$ by varying $z_t$, $d_{\text{facet}}$, $\lambda$ and $\theta_h$, e.g., $z_t = 2\text{cm}$, $d_{\text{facet}} = 520\mu\text{m}$, $\lambda = .65\mu\text{m}$, and $\theta_h = 45^\circ$. The dimensions and sensitivity characteristics of the photodiode ultimately determine the required output resolution and the optimal value for $M_{\text{max}}$.

A less restricted system wherein any source node can be imaged to any detector node for arbitrary cluster composition is formed by ensuring that $N = M_{\text{max}}$. In such a system, $N$ is bounded by the fundamental optical fan-out limit, $M_{\text{fan-out}}$. Conservatively, this equals 64 nodes, but $M_{\text{fan-out}}$ can be $O(100)$ by assuming more aggressive values for laser diode output power, optical loss, and photodiode sensitivity. To increase source fiber fan-out and detector fiber fan-in, the geometrical constraints of the optical system must be decreased. This is possible with multi-pass optical geometries [124, 102], which broaden field of view by diffractive-reflective holographic optical elements. A system size of $N = 64$ with non-restrictive clustering is therefore possible.
Chapter 6

Performance Analysis

Simulation tools were developed to evaluate the performance of GLORI. By integrating our trace evaluator and GLORI simulator into existing simulation tools developed at IBM T. J. Watson Research Center, we were able to not only study memory reference behavior discussed in Chapter 3, but also procure performance statistics of various network configurations through experimentation. The trace-driven GLORI simulator executes processor-memory reference traces generated by applications and measures interconnect latency.

In the following section, the method of collecting application traces is explained. An overview of the GLORI simulator is given in Section 6.2. The performance results obtained for a shared bus-hypercube GLORI network and for an ideal optical shared bus are presented in Section 6.3. Results are interpreted in Section 6.4, and conclusions are summarized in Section 6.5.

6.1 Trace Collection

The three S/E applications analyzed in Chapter 3 were used as benchmarks for the evaluation of GLORI’s performance. These applications exhibit various degrees of parallelism and cluster locality and, therefore, represent a wide spectrum of multiprocessor communication behavior.

Traces from these applications were collected using EPEX [125, 126] and PSIMUL
[127] as shown in Figure 6.1. A VM-based IBM 3084 system was used for this trace collection process. EPEX is a preprocessor that emulates a shared memory parallel system via multiprogramming on a host IBM S/370 system [128]. It uses the Single Program Multiple Data (SPMD) model of parallel computation. This model assigns the same application to run on all processors, but each processor can execute different instruction streams within the application while processing different data. Explicit language extensions/constructs are used to declare data as shared or private and to provide for loop parallelization, serialization, and synchronization.

EPEX creates a virtual memory space for the application. The shared common blocks of data (as explicitly declared by the user writing the FORTRAN code) is then placed in the upper portion of this virtual address space. Instructions and private data occupy a lower portion of this address space up to but not including the beginning of the shared data section of the address space. The output of the preprocessed and compiled code is then used to drive PSIMUL, a tool that assumes a 370 instruction set. However, if a good optimizing compiler is assumed that minimizes instruction and data referencing external to the processor-memory element (PME), then differences between external processor references of 370-type and RISC (i.e., RT) architectures are minimized.
PSIMUL generates and collects the memory reference traces of application programs executed in the EPEX environment. It should be noted that PSIMUL essentially interprets the application’s compiled code to generate these traces. Therefore, the traces are free of any S/370 caching and paging effects. The PSIMUL output is a single trace of virtual memory addresses interleaved with markers delineating the various sections of the application’s code. Each address entry specifies the referenced address and the reference type (instructions or data, shared or private and loads or stores). Four types of code sections are marked: parallel loops, replicate sections, serial sections and barrier synchronization. A trace scheduler tool is used to read this trace file, interpret the markers and use them to dynamically control the assignment of work to processors. A first-come-first-served, round-robin processor scheduling algorithm is used for this process, initially starting with processor zero.

For example, if the initial section of the trace is a serial section then processor zero is assigned to execute this section of code. All other processors then wait until the serial section is completed. If the next section of code is a parallel section, then a separate processor (beginning with processor 1 and continuing sequentially, i.e., processors 1, 2, 3, ...) executes $C$ iterations of this parallel loop where $C$, the chunk size, is an input parameter. (In this work, the value of $C$ is 1). If a replicate section is encountered, then all processors are scheduled to execute the code in this section. If a barrier synchronization section is reached, then each processor will, in turn, sleep wait until the last processor has arrived at this section. When this occurs the last processor will broadcast a message to all waiting processors to resume execution. The next processor to be scheduled will be the next sequential processor from the last processor reaching the barrier section.

The trace analyzer tool is used to procure reference behavior statistics like those presented in Chapter 3. It emulates an address-logic analyzer connected to each simulated PME network interface. It also emulates an ideal interconnect network which has no contention and 1 cycle latency between nodes of the system. Hence, the trace analyzer simply monitors interface transactions, logging and updating spatial and temporal referencing statistics every cycle. This information can be used by the GLORI simulator which emulates non-ideal networks as described below. Both the
trace analyzer and GLORI simulation tools are driven by completing a processor cycle for all scheduled processors as discussed earlier.

6.2 The GLORI Simulator

The GLORI simulator can emulate the operation of various electrical, optical, and hybrid electro-optical network configurations including shared buses, hypercubes, and shared bus-hypercube networks. Presented in this dissertation are simulation results of a shared bus-hypercube GLORI network and of an ideal optical shared bus network.

In addition to measuring overall message latency of the network, the simulator measures the average number of cycles processors stall from message initiation to message retrieval, the average bus access latency, the locality of messages, buffer sizes, and more. Flexible enough to analyze several different processing scenarios, the simulator can give performance data assuming various cache hit ratios, hypercube routing algorithms, bus sizes and hypercube dimensionality, and routing protocols. The individual performances of combining and not combining various shared reference types (e.g., SData, F&A refs) can also be assessed by this tool.

The simulated parallel environment created by EPEX consisted of 64 RP3-like PMEs, but system size is extendable to any reasonable number of nodes. A maximum of 1 reference per cycle per PME was assumed in the simulations. It was also assumed that processor activity stalls until the completion of read transactions but not for write transactions (infinite write buffer assumed). Taking into account optic's high bandwidth of 1 Gbps, it was assumed that each network link transaction of a 128 bit message over 8-bit parallel lines takes just 1 cycle, where the cycle time is 20 nsec. With electrical wires, the number of cycles could be about an order of magnitude larger.

For comparison, two networks were simulated:

- a 64-node shared bus-hypercube GLORI network consisting of $N/M = 8$ local bus clusters, each accommodating $M = 8$ PMEs, and

- an ideal optical shared bus network connecting together all 64 PMEs. The
optical bus is ideal in that once a bus access is granted, a reference transaction takes only 1 cycle.

Simulations were made of GLORI operating in shared link (SL), dedicated link (DL), combining enabled (CE), combining not enabled (CN), static, and dynamic modes. Recall from Chapter 4 that SL operation insists that shared bus clusters share neighboring hypercube links through some form of time-division multiplexing. In DL operation, clusters exclusively own neighboring hypercube links and do not have to arbitrate for link access. The higher throughput with DLs come at a cost of more global interconnect resources. With CE, read references to remote locations can be merged at the local cluster. This comes at the cost of comparator logic and $M$ message buffers at each cluster in the system. Static GLORI operation does not support interconnect reconfiguration, whereas dynamic operation does allow reconfiguration of the interconnect. The adaptability of a dynamic GLORI network (which can significantly improve performance as seen in Section 6.3) comes at the cost of a more complex optical routing unit and reconfiguration heuristics (see Chapters 5 and 7). In simulating the dynamic GLORI network, reconfiguration heuristics followed along the lines of post-game analysis [129].

Empirical data was gathered for various network loads. Section 6.3 concentrates on the results for the following network loads: all SData references going to the network (all other references serviced by caches) and all SData references along with 3.3% of Instr references going to the network (all other references hit in the caches). Data for network loading of 100% of SData and Instr references is provided for the purpose of comparing typical loads with the extreme case. This represents an unrealistic scenario, and is useful only as a projection of the asymptotic limit.

### 6.3 Performance Results

The memory reference profiles for the FFT, EIG's REDUCE subroutine, and HYD are shown in Figure 6.2. Simulating the entire EIG application's 50 million references (which actually has a much higher SData percentage than REDUCE) was too time consuming, so its REDUCE subroutine was used instead. Shared data references in
Figure 6.2: Memory reference profiles for FFT, EIG's REDEUCE, and HYD.
FFT and EIG-REDUCE represent almost 10% of the total referencing activity, with a fair amount of that being write references. Hence, in the interest of space, this section presents the results of these applications. The interpretation of HYD's results are included in Section 6.4, and further results are given in [130].

6.3.1 Measuring Performance

A number of metrics can be used to measure performance — among which are average latency per network reference, average number of processor stalls per reference, average number of network hops per reference, message queue length at nodes (i.e., for store-and-forward routing), etc. Not only were these used, but an absolute measure was examined: execution time. For our purposes, it is defined to be the total number of cycles it takes for the execution of an application on the system to complete. By comparing the total simulated execution time of GLORI with that of an ideal, one cycle-per-reference simulated uniprocessor system, performance in terms of attainable speed-up can also be calculated.

As given by the following equation, execution time consists of the cycles spent in performing local referencing activity (first 3 terms), global referencing activity (fourth term), and reconfiguration activity (fifth term):

\[
\]

(6.1)

By gathering statistics on reference behavior and network parameters, it is possible to analytically predict the contribution of each component of execution time given above. For instance, by knowing the average number of cycles processors stall for bus access to be granted and the average number of bus references made by processors, the average total number of execution cycles consumed in waiting for bus access can
be calculated. The following equation expresses how the critical parameters relate to overall execution time:

\[
\text{Execution Time} = \left( \frac{\text{Ave.Number.Cache.Refs}}{\text{Processor}} \right) \times \left( \frac{1}{\text{Cycle Ref}} \right) \\
+ \left( \frac{\text{Ave.Bus.Cycle.Stalls}}{\text{Bus.Ref}} \right) \times \left( \frac{\text{Ave.Number.Bus.Refs}}{\text{Processor}} \right) \\
+ \left( \frac{\text{Ave.Number.Local.Refs}}{\text{Processor}} \right) \times \left( \frac{1}{\text{Cycle Ref}} \right) \\
+ \left( \frac{\text{Ave.Global.Net.Stalls}}{\text{Processor}} \right) \times \left( \frac{\text{Ave.Global.Refs}}{\text{Processor}} \right) \times \left( \frac{1}{\text{Cycle Ref}} \right) \\
+ \left( \frac{\text{Number.Cycles}}{\text{Reconfiguration}} \right) \times (\text{Number.Reconfigurations}), \quad (6.2)
\]

where

\[
\frac{\text{Ave.Global.Net.Stalls}}{\text{Processor}} = \left( \frac{\text{Number.Cycles}}{\text{Hop}} \right) \times \left( \frac{\text{Ave.Number.Hops}}{\text{Global.Read.Ref}} \right) \\
\times \left( \frac{\text{Number.Global.Read.Refs}}{\text{Processor}} \right).
\]

Values for all of the variables in Equation (6.2) are among the outputs of the simulator except for the number of cycles needed per reconfiguration. For this parameter, it is assumed that ferroelectric liquid crystal SLMs can perform interconnect reconfigurations in about 100\(\mu\)secs or, equivalently, 5000 cycles [41].

6.3.2 FFT Benchmark Results

This section presents the statistics gathered for the local and global network parameters of GLORI and the simulated ideal optical shared bus for the FFT benchmark.\(^1\)

Tables 6.1 and 6.2 show the difference in global network loading of static and dynamic GLORIs for the FFT application. The degree to which a network becomes loaded directly affects network congestion and the latency experienced by references. The non-adaptive static GLORI captured only about 10% of all network references

\(^1\)The entries enclosed in brackets are estimated numbers assuming the cluster locality content for Instr references are the same as SData references.
### Table 6.1: FFT Network Loading Statistics for Static GLORI

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Number of Network Refs</th>
<th>Global Network Refs</th>
<th>Global Refs per Processor</th>
<th>Read &amp; Write</th>
<th>Read Refs</th>
</tr>
</thead>
<tbody>
<tr>
<td>SData</td>
<td>227,000</td>
<td>197,000 (87%)</td>
<td>3,100</td>
<td>1,500 (50%)</td>
<td></td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>286,000</td>
<td>260,000 (91%)</td>
<td>4,100</td>
<td>2,500 (60%)</td>
<td></td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>1,998,000</td>
<td>1,758,000 (88%)</td>
<td>27,500</td>
<td>25,800 (94%)</td>
<td></td>
</tr>
</tbody>
</table>

### Table 6.2: FFT Network Loading Statistics for Dynamic GLORI

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Number of Network Refs</th>
<th>Global Network Refs</th>
<th>Global Refs per Processor</th>
<th>Read &amp; Write</th>
<th>Read Refs</th>
</tr>
</thead>
<tbody>
<tr>
<td>SData</td>
<td>227,000</td>
<td>5,300 (2.3%)</td>
<td>80</td>
<td>40 (50%)</td>
<td></td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>286,000</td>
<td>6,600 (2.3%)</td>
<td>100</td>
<td>60 (60%)</td>
<td></td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>1,998,000</td>
<td>36,800 (2.3%)</td>
<td>580</td>
<td>540 (94%)</td>
<td></td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
locally whereas the adaptive dynamic GLORI with block-frequent reconfiguration localized over 95% of the references within clusters. Two interconnect reconfigurations were required. This disparity in clusterability affects the total number of global references made per processor. Of particular interest is the number of global read references per processor. Since read latency has the effect of stalling processor activity, a minimal number of global reads by each processor is desired.

In the following subsections, global network latency and local network latency are first considered separately and then together.

**Global Latency**

Global network latency is larger than local network latency and varies with distance for multi-hop networks — the more remote the reference, the larger the latency. This increase in latency with hop count is shown graphically in Figure 6.3 for two network loads of the FFT benchmark. It is also shown for read references only (write references not included) in Figure 6.4. The vertical dashed lines in the figures indicate the average number of hops processor required by references to arrive at destination clusters (1.7 hops). As shown, the latency difference with hop count is less pronounced with combining enabled GLORI networks and is virtually nonexistent with dedicated link GLORI networks.

The number of cycles references spend latent in the network directly translate into cycles stalled by processors. Table 6.3 gives this information on a per read reference basis; Table 6.4 gives this information on a per processor basis. By obtaining and plotting the data graphed in Figure 6.4 for a number of network loads, it is possible to predict global network read latency (and processor stalls) for any network load. This is done in Figure 6.5. By extrapolating data from this figure, bracketed table entries are estimated for the dynamic GLORI.

Global read references experienced the least amount of average latency with the dynamic GLORI network. The reduced global referencing activity as given in Table 6.2 lowered network congestion and, thus, the resulting latency of each reference. The total number of global network read stalls per processor is very low as well because of dynamic GLORI’s fewer number of global references.
Figure 6.3: Hop Latency as a function of the number of hops through the network for various FFT network loads with and without combining.
Figure 6.4: Processor Stalls (for Read Refs) as a function of the number of hops through the network for various FFT network loads with and without combining.
Figure 6.5: Processor Stalls (for Read Refs) versus FFT network load (global refs) for average hop distance of 1.7 with no combining.
Table 6.3: Average Number of Cycles Processors Stall per Global Network Read Ref

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Static (CN)</th>
<th>Static (CE)</th>
<th>Dynamic (CN)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
<td>SL</td>
</tr>
<tr>
<td>SData</td>
<td>78</td>
<td>14</td>
<td>30</td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>80</td>
<td>15</td>
<td>52</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>105</td>
<td>59</td>
<td>27</td>
</tr>
</tbody>
</table>

Table 6.4: Average Global Network Read Stalls per Processor (cycles)

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Static (CN)</th>
<th>Static (CE)</th>
<th>Dynamic (CN)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
<td>SL</td>
</tr>
<tr>
<td>SData</td>
<td>120,000</td>
<td>21,600</td>
<td>46,200</td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>195,000</td>
<td>36,600</td>
<td>127,000</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>2,710,000</td>
<td>1,520,000</td>
<td>697,000</td>
</tr>
</tbody>
</table>

**Local Latency**

As represented in Equation (6.1), local network latency is determined by three principal factors: 1) cache hit latency, 2) local bus access latency (arbitration cycles for bus ownership to be granted), and 3) local bus transaction latency (after bus permission is granted). Table 6.5 gives the average number of cycles processors stalled waiting for permission to use the local buses on a per reference basis for the FFT application; Table 6.6 provides this information on a per processor basis. Most of the GLORI networks have less than 1 cycle bus access latency. Split-transaction bus operation for global references (see Chapter 4) is the underlying reason for this. Buses that might otherwise be occupied are freed by processors making remote references. Since global references have large latencies, suspended processors do not contend for the local bus, therefore lowering local bus load.

What further reduces bus access latency is the limited number of processors assigned to each bus as a result of interconnect hierarchy (only 8 nodes per bus). Hence, the maximum latency possible even on a saturated bus is only 8 cycles for the GLORI...
Table 6.5: Average Number of Cycles Processors Stalled for Bus Grant per Ref

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Static (CN)</th>
<th>Static (CE)</th>
<th>Dynamic (CN)</th>
<th>Bus Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
<td>SL</td>
<td>DL</td>
</tr>
<tr>
<td>SData</td>
<td>.1</td>
<td>.4</td>
<td>1.1</td>
<td>.8</td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>.2</td>
<td>.3</td>
<td>.2</td>
<td>.5</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>.3</td>
<td>.2</td>
<td>3.1</td>
<td>3.1</td>
</tr>
</tbody>
</table>

Table 6.6: Total Bus Grant Stalls per Processor (cycles)

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Static (CN)</th>
<th>Static (CE)</th>
<th>Dynamic (CN)</th>
<th>Bus Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
<td>SL</td>
<td>DL</td>
</tr>
<tr>
<td>SData</td>
<td>350</td>
<td>1,400</td>
<td>3,800</td>
<td>2,800</td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>900</td>
<td>900</td>
<td>900</td>
<td>2,200</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>9,200</td>
<td>6,200</td>
<td>95,400</td>
<td>95,400</td>
</tr>
</tbody>
</table>

networks. The optical shared bus, which has no second tier global interconnect to direct references so as to reduce its own load (and therefore latency), is loaded by all 64 nodes. When saturated, it has a latency that approaches 64 cycles. This is reflected in the data provided by the tables.

Total Network Latency

The total local and global latency per network reference averaged over all processors for the FFT benchmark is given in Table 6.7. This information is plotted in graphical form in Figure 6.6 (a). This simulated data agrees with analytical estimates of total latency.

The dynamic GLORI provides the lowest average latency per network reference, as shown in the table. Since the majority of latency is associated with the global network, enabling combining significantly reduces latency in the static shared link GLORI, which has the lowest throughput of the GLORI networks. Average latency
Table 6.7: Average Total Latency per Network Reference (cycles)

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Static (CN)</th>
<th>Static (CE)</th>
<th>Dynamic (CN)</th>
<th>Bus Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
<td>SL</td>
<td>DL</td>
</tr>
<tr>
<td>SDATA</td>
<td>46</td>
<td>10</td>
<td>26</td>
<td>9</td>
</tr>
<tr>
<td>SDATA &amp; 3.3% Instr</td>
<td>48</td>
<td>12</td>
<td>37</td>
<td>11</td>
</tr>
<tr>
<td>SDATA &amp; 100% Instr</td>
<td>99</td>
<td>50</td>
<td>26</td>
<td>14</td>
</tr>
</tbody>
</table>

of the ideal optical shared bus is the largest under practical network loads but, in the asymptotic case, is lower than the static SL GLORI (see Figure 6.6 (a)). This is largely due to the fact that average reference latency closely corresponds to median reference latency with the shared bus but does not with GLORI.

Under saturated conditions, the maximum shared bus latency is bound by the number of nodes on the bus and is experienced by all references. The maximum latency of the static SL GLORI under saturated conditions has a larger bound owing to GLORI's higher throughput 2-tier topology. Though local messages may take a maximum of 9 cycles (8 to gain bus access and 1 for message transfer), global messages might have latencies of over a hundred cycles. Hence, the latency of static SL GLORI has a wider variance (standard deviation) than the shared bus. Higher throughput provides for more messages at any given time to be handled by the network at the possible cost of higher network latency. Ideally, a network has high throughput and low latency. This is accomplished when throughput exceeds the minimum threshold needed to mitigate network congestion. As seen in Table 6.7, higher connectivity dedicated link GLORI networks provide sufficient throughput.

It is possible to analytically predict execution time for a given network loading (i.e., various cache hit-ratios) using the data presented in the preceding tables. Figure 6.6 (b) is such a plot. However, simulations provide more accurate estimations for particular network loads. Table 6.8 gives the execution time of the FFT benchmark for various loadings of the simulated networks. Speed-up estimates are given in Table 6.9. Again, simulated data corroborates with analytical estimates to within acceptable limits using Equation (6.2).
Figure 6.6: (a) Average Latency per Ref and (b) Execution Time verses FFT network load for average hop distance of 1.7 with no combining.
CHAPTER 6. PERFORMANCE ANALYSIS

Table 6.8: Execution Time in Thousands of Cycles

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Static (CN)</th>
<th>Static (CE)</th>
<th>Dynamic (CN)</th>
<th>Bus Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
<td>SL</td>
<td>DL</td>
</tr>
<tr>
<td>SData</td>
<td>166</td>
<td>69</td>
<td>96</td>
<td>65</td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>242</td>
<td>83</td>
<td>174</td>
<td>80</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>2,770</td>
<td>1,570</td>
<td>838</td>
<td>451</td>
</tr>
</tbody>
</table>

Table 6.9: Estimated Speed-Up (1 Processor = 2,850,000 cycles)

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Static (CN)</th>
<th>Static (CE)</th>
<th>Dynamic (CN)</th>
<th>Bus Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
<td>SL</td>
<td>DL</td>
</tr>
<tr>
<td>SData</td>
<td>17</td>
<td>41</td>
<td>30</td>
<td>44</td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>12</td>
<td>34</td>
<td>16</td>
<td>36</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

6.3.3 EIG Benchmark Results

This section presents the statistics gathered for the local and global network parameters of GLORI and the simulated ideal optical shared bus for the EIG benchmark (REDUCE subroutine). Tables 6.10 and 6.11 show the difference in global network loading of static and dynamic GLORIs. The non-adaptive static GLORI captured just less than 15% of all network references locally; the adaptive dynamic GLORI with block-frequent reconfiguration localized just over 30% of the references within clusters. Two interconnect reconfigurations were required.

Global Latency

The increase in global network latency with hop count is shown graphically in Figure 6.7 for two network loads of the EIG benchmark. For read references only (write references not included) hop count data is shown in Figure 6.8. As with the FFT benchmark, the latency difference is less pronounced with combining enabled GLORI networks and is virtually nonexistent with dedicated link GLORI networks.
Figure 6.7: Hop Latency as a function of the number of hops through the network for various EIG network loads with and without combining.
CHAPTER 6. PERFORMANCE ANALYSIS

Figure 6.8: Processor Stalls (for Read Refs) as a function of the number of hops through the network for various EIG network loads with and without combining.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Table 6.10: EIG Referencing Statistics for Static GLORI (thousands of references)

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Number of Network Refs</th>
<th>Global Network Refs</th>
<th>Global Refs per Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Read &amp; Write</td>
</tr>
<tr>
<td>SData</td>
<td>880</td>
<td>750 (86%)</td>
<td>12</td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>1,200</td>
<td>1,000 (86%)</td>
<td>16</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>11,000</td>
<td>9,500 (86%)</td>
<td>150</td>
</tr>
</tbody>
</table>

Table 6.11: EIG Referencing Statistics Dynamic GLORI (thousands of references)

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Number of Network Refs</th>
<th>Global Network Refs</th>
<th>Global Refs per Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Read &amp; Write</td>
</tr>
<tr>
<td>SData</td>
<td>880</td>
<td>600 (69%)</td>
<td>9.0</td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>1,200</td>
<td>830 (69%)</td>
<td>13</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>11,000</td>
<td>7,600 (69%)</td>
<td>119</td>
</tr>
</tbody>
</table>

Table 6.12 gives the number of cycles processors stall due to read reference latency on a per read reference basis; Table 6.13 gives this information on a per processor basis. The estimated global network read latency as a function of network load is plotted in Figure 6.9.

The dynamic DL GLORI resulted in lower latency per global read reference as compared to all other networks except the static DL GLORI with combining. However, the dynamic SL GLORI resulted in higher latency as compared to all other networks except the static SL GLORI without combining. This disparity in latency highlights the throughput difference between shared link and dedicated link networks.

Global referencing activity, as given in Table 6.11, is decreased slightly for the dynamic GLORI networks. The latency is therefore only slightly reduced for read references. Combining, which resulted in about 50% of the references getting merged at the local cluster, more effectively reduced network congestion and, hence, latency of each reference.
Figure 6.9: Processor Stalls (for Read Refs) versus EIG network load (global refs) for average hop distance of 1.7 with no combining.
Table 6.12: Average Number of Cycles Processors Stall per Global Network Read Ref

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Static (CN)</th>
<th>Static (CE)</th>
<th>Dynamic (CN)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
<td>SL</td>
</tr>
<tr>
<td>SData</td>
<td>105</td>
<td>22</td>
<td>21</td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>109</td>
<td>24</td>
<td>32</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>162</td>
<td>52</td>
<td>28</td>
</tr>
</tbody>
</table>

Table 6.13: Average Global Network Read Stalls per Processor (thousands of cycles)

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Static (CN)</th>
<th>Static (CE)</th>
<th>Dynamic (CN)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
<td>SL</td>
</tr>
<tr>
<td>SData</td>
<td>900</td>
<td>190</td>
<td>180</td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>1,400</td>
<td>310</td>
<td>410</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>24,000</td>
<td>7,700</td>
<td>4,100</td>
</tr>
</tbody>
</table>

Local Latency

Table 6.14 gives the average number of cycles processors stalled waiting for permission to use the local buses on a per reference basis for the EIG application. This information is provided on a per processor basis in Table 6.15. For the same reasons as with the FFT benchmark, most of the GLORI networks have less than 1 cycle bus access latency. Bus access latency, on occasion, exceeds 1 cycle in cases where global references are satisfied in short order, thus allowing more processors to contend for the bus. Access latency for the optical shared bus network again approaches its maximum since there is no global network to relieve bus contention.

Total Network Latency

The total local and global latency per network reference averaged over all processors for the EIG benchmark is given in Table 6.16. This information is plotted in graphical form in Figure 6.10 (a) and corroborates with analytical estimates of total latency.
Table 6.14: Average Number of Cycles Processors Stalled for Bus Grant per Ref

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Static (CN)</th>
<th>Static (CE)</th>
<th>Dynamic (CN)</th>
<th>Bus Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
<td>SL</td>
<td>DL</td>
</tr>
<tr>
<td>SData</td>
<td>.2</td>
<td>.4</td>
<td>1.6</td>
<td>1.5</td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>.2</td>
<td>.4</td>
<td>.5</td>
<td>.8</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>.3</td>
<td>.5</td>
<td>3.0</td>
<td>2.8</td>
</tr>
</tbody>
</table>

Table 6.15: Total Bus Grant Stalls per Processor (thousands of cycles)

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Static (CN)</th>
<th>Static (CE)</th>
<th>Dynamic (CN)</th>
<th>Bus Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
<td>SL</td>
<td>DL</td>
</tr>
<tr>
<td>SData</td>
<td>2.7</td>
<td>5.4</td>
<td>22</td>
<td>20</td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>3.8</td>
<td>7.5</td>
<td>9.4</td>
<td>15</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>52</td>
<td>86</td>
<td>520</td>
<td>480</td>
</tr>
</tbody>
</table>

Table 6.16: Average Total Latency per Network Reference (cycles)

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Static (CN)</th>
<th>Static (CE)</th>
<th>Dynamic (CN)</th>
<th>Bus Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
<td>SL</td>
<td>DL</td>
</tr>
<tr>
<td>SData</td>
<td>67</td>
<td>19</td>
<td>19</td>
<td>11</td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>71</td>
<td>21</td>
<td>25</td>
<td>10</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>130</td>
<td>44</td>
<td>25</td>
<td>13</td>
</tr>
</tbody>
</table>
Figure 6.10: (a) Average Latency per Ref and (b) Execution Time verses EIG network load for average hop distance of 1.7 with no combining.
CHAPTER 6. PERFORMANCE ANALYSIS

The average latency of dedicated link and combining enabled GLORI networks is significantly lower than that of the ideal optical shared bus network. The average latency of the static GLORI networks under realistic network loads is comparable to that of the shared bus. Dynamic GLORI networks, however, do not result in significantly reduced latency as compared to the static GLORI networks. These latencies are reflected in the number of cycles needed to execute the benchmark.

Table 6.17: Execution Time in Thousands of Cycles

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Static (CN)</th>
<th>Static (CE)</th>
<th>Dynamic (CN)</th>
<th>Bus Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
<td>SL</td>
<td>DL</td>
</tr>
<tr>
<td>SData</td>
<td>1,400</td>
<td>680</td>
<td>690</td>
<td>590</td>
</tr>
<tr>
<td>SData &amp; 3.3% Instr</td>
<td>1,900</td>
<td>800</td>
<td>910</td>
<td>620</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>25,000</td>
<td>8,300</td>
<td>5,000</td>
<td>2,800</td>
</tr>
</tbody>
</table>

Analytical predictions of execution time as a function of network loading is plotted in Figure 6.10 (b). Table 6.17 gives the execution time of the EIG benchmark for various loadings of the simulated networks. Speed-up estimates are given in Table 6.18.

Table 6.18: Estimated Speed-Up (1 Processor = 16,800,000)

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Static (CN)</th>
<th>Static (CE)</th>
<th>Dynamic (CN)</th>
<th>Bus Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SL</td>
<td>DL</td>
<td>SL</td>
<td>DL</td>
</tr>
<tr>
<td>SData</td>
<td>12</td>
<td>25</td>
<td>25</td>
<td>29</td>
</tr>
<tr>
<td>SData &amp; 100% Instr</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>
6.4 Interpretation of the Results

FFT

Figure 6.11 plots the results contained in Table 6.9. In comparing the speed-up attained by the networks, note that the FFT application is highly parallel since its algorithmic speed-up (denoted by the dotted line) is so near the ideal speed-up (denoted by the solid line). Also, the dynamic GLORI network provides much higher speed-up over the optical shared bus, and an unexpected result is that good speed-up is attained with static DL GLORI networks. These results can be better understood by examining the constituent parts of execution time.

Figure 6.12 plots the information contained in Table 6.8 with a break-down of the execution cycles spent performing local referencing activity, global referencing activity, and interconnect reconfiguration. Execution time of an ideally interconnected multiprocessor system that provides algorithmic speed-up is denoted by the dotted line. In this ideal case, each network reference is assumed to take only 1 cycle to complete.

A number of observations can be made upon examining Figure 6.12. GLORI outperforms the ideal optical shared bus even without reconfiguration, combining enabled, and dedicated hypercube links because the shared links are still better able to accommodate FFT's network load than the shared bus. The dominant portion of execution time for the static SL GLORI network is global referencing activity. By enabling combining, which results in 34% of the references getting merged at the clusters, a substantial reduction in the execution time devoted to global referencing activity results in the static SL GLORI network.

The contribution to global referencing activity to execution time is reduced by a factor of 4 by using dedicated links instead of shared links in the static GLORI because of DL's higher network throughput. Very little difference in execution time results with combining in the static DL system which, as shown, is dominated by local referencing activity. The throughput advantage of DLs had a strong effect on lowering network latency, so the minor reduction of global network congestion provided by combining had little effect on lowering latency.
(a) SData Refs only

(b) SData & 3.3% Instr Refs

Figure 6.11: Estimated Speed-Up for (a) SData only and (b) SData & 3.3% Instr Refs as global network refs (FFT).
Figure 6.12: Execution Time for (a) SData only and (b) SData & 3.3% Instr Refs as global network refs (FFT).
A more important observation is that GLORI operating in dynamic mode yields the best performance even with shared links. Because of FFT’s great amount of cluster locality (see Chapter 3), the network is able to be reconfigured to capture over 95% of the references for local processing. This, in effect, reduces global referencing activity to nominal levels handleable even by the lower throughput shared links. Overhead execution cycles devoted to interconnect reconfiguration were much less than the execution cycles saved in performing global referencing activity. Hence, block-frequent interconnect reconfiguration pays off in this case.

**EIG**

Figure 6.13 plots the results contained in Table 6.18. The algorithm provided poor speed-up since the problem size is too small. Nevertheless, a relatively fair amount of speed-up is attained with all GLORI networks except the shared link without combining enabled networks.

Figure 6.14 plots the information contained in Table 6.17 with a break-down of the constituent parts of execution time. The EIG benchmark, unlike FFT, did not parallelize well as seen by its algorithmic speed-up. Moreover, the degree of cluster locality was only moderate as compared to FFT. Just 31% of the SData references made could be localized via block-frequent reconfiguration. The other 69% necessarily encountered the global network.

With so many references going to the global network, the dynamic SL GLORI network, with its low throughput, quickly became saturated, thus losing much of the gains brought be reconfiguring the interconnect. Hence, the dominant portion of its execution time was devoted to global referencing activity. The dynamic DL GLORI network reduced global referencing activity considerably. It marginally outperformed the static SL with CE network and the static DL network while marginally performing poorer than the static DL with CE GLORI network. This is because combining, which resulted in as many as 50% of the references getting merged, did a better job of decreasing global referencing activity than did interconnect reconfiguration.

These findings substantiate an intuitive, yet very important result: a minimum
Figure 6.13: Estimated Speed-Up for (a) SData only and (b) SData & 3.3% Instr Refs as global network refs (EIG).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
amount of cluster locality is needed in order for block-frequent reconfiguration to provide significant performance gains over a static system with and without cluster-level combining. For the applications in this study, cluster locality must be in the neighborhood of 30% for a dynamic GLORI network to outperform a static GLORI network without combining, and it must be about 50% for a dynamic GLORI network to outperform a static GLORI network with combining.

6.5 Summary

For our applications, block-frequent reconfiguration proved effective in reducing communication latency when cluster locality is in sufficient quantity to enable 30% or more localization of references. Combining was found to be effective in reducing latency, but was not as good as interconnect reconfiguration for cluster locality greater than 50%. Even with reconfiguration, insufficient network throughput (as seen with shared links) can limit the attainable speed-up if network load is not sufficiently reduced by reconfiguration.

Dedicated links in GLORI provide enough network throughput to preserve the speed-up gains brought by interconnect reconfiguration. The cost of the additional interconnect resources to implement dedicated global interconnect links is small in comparison to the performance gains attainable (e.g., only 18% more optical sources, detectors, and fibers are needed for a 64 node 8-way 3-cube GLORI system with no increase in optical routing unit volume).

Finally, the overhead cycles spent in physically performing optical reconfiguration of the interconnect is almost insignificant compared to the time span over which a configuration remains static and compared to the cycles saved in performing global referencing activity.
Figure 6.14: Execution Time for (a) SData only and (b) SData & 3.3% Instr Refs as global network refs (EIG).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Chapter 7

Summary, Conclusions, and Future Research

7.1 Summary

The GLORI strategy is a multiprocessor interconnect scheme designed to aid the programmer, compiler, and/or run-time scheduler in facilitating locality, e.g., in partitioning, placement, and relocation of data and processes. It also aids the hardware architecture in performing the functions useful in capturing and exploiting locality, e.g., clustering, combining, and caching. The impetus behind GLORI is to use the interconnect as an additional resource for taking advantage of multiprocessor locality and to use optics to achieve this goal. Among some of GLORI’s non-optical features are hierarchical topology, clustering, combining, and minimal delay routing protocols.

Optics is a promising interconnect technology for multiprocessors. The usefulness of optics’ many interconnect features are evaluated using GLORI as a framework. In the proposed GLORI organization, $M$ processor-memory elements (PMEs) are clustered together onto shared buses at the local interconnect level, and $N/M$ bus clusters are arranged in a hypercube at the global interconnect level. Reconfiguration of PME connectivity occurs system-wide to allow arbitrary groups of PMEs to be clustered, but the baseline shared bus-hypercube topology is assumed to remain

159

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
intact. Interconnect links are established by fiber and free-space holographic diffractive optics. Full-scale implementation of the proposed system is feasible based on reasonable expectations of the development of optical technology in the near-term.

In designing and evaluating GLORI, a number of interesting discoveries are made relating to multiprocessor communication behavior, interconnect organization, and optical implementation. From Chapter 2, we learn that the performance of optical single-hop and multi-hop networks is not bound by switching speed, as it is with crossbar and multi-stage networks; instead, it depends on memory reference patterns of parallel algorithms. These networks operate well within optics' switching limitations, hence they are good candidates on which to build GLORI's topology. Since the latency of these networks increases as reference activity across them decreases, locality capturing mechanisms can boost performance.

From Chapter 3, we learn that reference patterns of parallel algorithms exhibit cluster locality behavior. The degree to which cluster locality manifests depends on problem size, page size, chunk size, process scheduling, and parallelization. This behavior can be exploited by a system able to dynamically reconfigure its interconnects in synchronization with changes in cluster locality. Cluster affinity among PMEs is seen to be maintained over intervals of tens to hundreds of thousands of execution cycles. Interconnect reconfiguration occurring once every millisecond is therefore sufficient to exploit this behavior.

Ferroelectric liquid crystal spatial light modulators and experimental photorefractive holographic devices can operate well within this switch frequency. The GLORI architecture functionally described in Chapter 4, which makes use of block-frequent reconfiguration, is therefore tolerant of optic's relatively slow switching time. Networks which require reference-frequent reconfiguration are not as tolerant of optic's switching limitations. The proposed GLORI communication protocols also take full advantage of the features of optics in managing access to local and global network links.

We learn from Chapter 5 that a number of important issues determine the feasibility of our proposed GLORI system. Some issues indirectly impact system feasibility, such as those relating to the holographic beam steering element. For instance,
• the imaging range over which interconnects can be established is determined by the deflection angle produced by the hologram, which depends on the recordable hologram fringe frequency;

• the focus of light to within the cross-sectional area of fiber cores places restrictions on hologram deflection angle and facet size;

• there are certain conditions under which the same hologram facet can be used to simultaneously image several bit-lines to reduce hologram area; and

• hologram diffraction efficiency and losses associated with other components along the passive optical path limit system size.

Other issues, such as those relating to optical geometry, directly impact how large a system can be feasibly implemented due to fan-out and fan-in limitations. For instance,

• the beam spread of source fibers determines how many hologram facets can be illuminated, which limits nodal fan-out;

• the detector fiber field-of-view determines the maximum number of subholograms that can image to a particular detector fiber, which limits nodal fan-in; and

• crosstalk from non-desired diffracted wave orders limits the number of bit-lines per node.

We also learn that a multiprocessor system of \( N = 256 \) processor-memory nodes can be feasibly implemented with a static optical shared bus-hypercube GLORI network employing elementary guided and free-space holographic optical components. By improving aspects of the optical system which most limit scalability, systems on the order of thousands of nodes can be built. The cubic centimeter volume of GLORI's shared bus-hypercube optical routing unit is far less than comparable electronic hard-wired implementations.
From Chapter 6, we learn the usefulness of GLORI’s cluster-level combining, and we verify the benefits and limits of GLORI’s block-frequent interconnect reconfigurability. GLORI’s cluster-level combining of read references can reduce global network load by up to a factor of $M$ (the cluster size) depending on an algorithm’s referencing and hot-spotting activity. For benchmark applications used in our simulations, a reduction in global network load ranging from approximately 35% to 50% resulted. It can therefore be postulated that combining techniques used at the cluster level have applicability in hierarchically interconnected multiprocessor systems.

Another important finding is that exploiting cluster locality through optical block-frequent reconfiguration can significantly reduce network latency. For benchmark applications used in our simulations, a reduction in global network load ranging from approximately 30% to 95% resulted. Performance is not encumbered by overhead cycles spent in optically switching from one configuration to another. This delay is relatively insignificant compared to the period over which each configuration remains static, and it is small compared to the execution cycles potentially saved by reconfiguring the system. Improved optical switching time would have little effect on the resulting performance of the proposed GLORI system, but it may influence the reconfiguration policies of future generation GLORI systems.

Perhaps somewhat of an unexpected discovery is that GLORI’s block-frequent reconfiguration policy is not a general solution for improving performance; cluster-level combining may significantly reduce global network reference activity in situations where interconnect reconfiguration does not. Global network load was reduced by 50% with cluster-level combining but only by 30% with interconnect reconfiguration for the EIG application. Block-frequent reconfiguration is shown to prove useful when it reduces global reference activity to levels that do not saturate the global network. In other words, if there is not sufficient cluster locality, reconfiguration is unlikely to reduce network latency. The same is true for cluster-level combining: sufficient hot-spotting is needed for combining to significantly reduce global network load and latency.
7.2 Conclusions

A systems approach to multiprocessor interconnect design makes judicious use of optical technology to achieve a low latency, low cost communication network, as evidenced by the GLORI strategy.

Space-invariant imaging, high bandwidth links, and reconfigurability are useful interconnect features provided by optics. Space-invariant imaging is useful in optically transmitting bit-line signals in parallel over a common interconnect medium. Optical links providing in excess of an order of magnitude more bandwidth than modern electronic wires are useful in reducing line width and message latency. Optical reconfiguration is useful in adapting processor connectivity to conform to the available locality inherent in memory references.

This research shows that significant performance gains are achievable with multiprocessor systems based on the GLORI strategy. Cluster-level combining reduces network latency given the existence of sufficient hot-spot activity. Interconnect reconfiguration also reduces network latency with the existence of sufficient cluster locality. The performance advantages gained by these locality capturing techniques can coexist in an interconnect based on optical technology. Maximal benefit from such an interconnect facility is gained when multiprocessor algorithms are highly parallel and constructed so as to maximize cluster locality behavior.

7.3 Future Research

The applications considered in this work primarily feature loop-parallelism. It would be interesting to see how well GLORI performs on applications featuring task-queue parallelism and, especially, multi-level parallelism which has the tendency to increase cluster locality. Finding other techniques to increase cluster locality of algorithms is important to fully utilize interconnect reconfiguration provided by optics.

The development of run-time and/or compile-time reconfiguration heuristics would ease the task of programmers in identifying when interconnect reconfigurations should take place during program execution. It is conceivable that adaptations of heuristics
used in performing load balancing on multiprocessors can be applied to interconnect reconfiguration [129, 131].

There is ongoing research into reconfigurable interconnects based on holographic optical elements [132, 133]. Using multiple-grating holograms and multi-faceted holograms, interconnect patterns can be reconfigured by wavelength tuning techniques which allow $O(100)$ configurations or space-division techniques which allow $O(1000)$ configurations. For instance, in the Holoswitch shown in Figure 7.1, an SLM directs optical beams toward one of many sets of pre-recorded holograms. Each contains a frequently used interconnect pattern, thus enabling interconnect reconfiguration. Further advancements in the area of dynamic holograms is needed before building a fully operational GLORI system.

Scalability issues should also be explored further. Can massively parallel applications be programmed to have cluster locality? Is it feasible to implement a reconfigurable GLORI system of many nodes? Do the performance gains provided by optically reconfiguring a large system out-weigh the associated costs? Many such questions regarding the role of optics for multiprocessor interconnection remain to be resolved by future research.

Figure 7.1: Optical interconnect reconfiguration.
Bibliography


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.