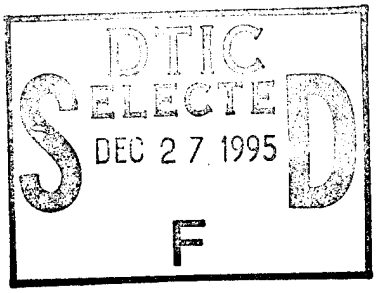
A PHOTONIC, DISTRIBUTED, HIGH-PERFORMANCE COMPUTER INTERCONNECT

U.S. Army Grant Number DASG60-93-C-0148

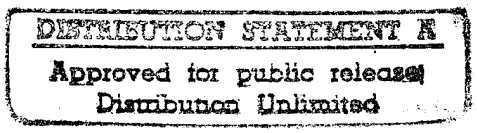
Final Report



December 1, 1995

Principal Investigator

Jon R. Sauer



Optoelectronic Computing Systems Center

University of Colorado at Boulder

Boulder, CO 80309-0525

Contributors:

Alan F. Benner

Daniel J. Blumenthal

John R. Feehrer

Lian Hua Ji

Aruna V. Ramanan

Lars H. Ramfelt

Jon R. Sauer

Michael D. Sprenger

Prepared by:

R. Brian Jenkins

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	12/01/95	Final Technical

4. TITLE AND SUBTITLE

A Photonic, Distributed, High-Performance Computer Interconnect

5. FUNDING NUMBERS

6. AUTHOR(S)

Dr. Jon Sauer, PI

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Office of Contracts and Grants
University of Colorado
CB 19
Boulder, CO 80309-0019

8. PERFORMING ORGANIZATION REPORT NUMBER

1536827

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U. S. Army Strategic Defense Command
PO Box 1500
CSSD-AT-P
Huntsville, AL 35758 ATTN: Max McCurry

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

DASG60-93-C-0148

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

See Attached Report

14. SUBJECT TERMS			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Contents

List of Tables

List of Figures

Chapter 1

Introduction

There are two trends visible in computer systems today. One is the steady increase in microprocessor CPU clock rates due to continual advancements in semiconductor device and lithography technology. For example, Digital Equipment Corporation has announced a 275 MHz 0.5 μm CMOS implementation of its Alpha microprocessor architecture [1]. The second trend is increasing levels of memory hierarchy to offset the ever-widening disparity between memory access times and gate delays. An active area of research is cache designs that reduce the miss penalty for multiprocessors using high-speed RISC computing engines [2], [3].

These trends have had a strong impact on the design of networks for shared-memory multiprocessors. Since processors have generally advanced

in speed by two orders of magnitude over the last decade, in comparison to one for interconnects [4], careful design of the network connecting fast microprocessors and DRAM memory modules together is critical to achieving good performance. Intelligent network design coupled with support for latency hiding [5] and compile-time data structure layout optimization can yield good speedup for parallel programs and can prevent memory latency from becoming a bottleneck. The research results reported here focus on the lower-level details of a packet-switched direct network for shared-memory multiprocessors. A processor-memory pair forms a network node, with a unique contiguous portion of the global address space assigned to memory at each node. A processor has access to physically remote memory as well as to its own local memory. Access to remote memory requires packets to be sent over the network. Every node contains an electronic packet routing processor and a switch which either switches through-going packets destined for other nodes or sinks packets destined for the node, and injects packets from the node into the network. To achieve the high throughput necessary to both reduce and hide memory latency, optical links and optical switching will be necessary, especially so in networks which are widely distributed geographically.

1.1 Research Goals and Results

The advantages to optical transmission media and photonic switches over their electrical counterparts have been cited often (e.g. [6],[7]). Optical waveguides such as glass fiber offer immunity from bandwidth-limiting electromagnetic effects and are capable of higher interconnection density, higher fanout, lower skew, and lower loss. The consensus is that for the foreseeable future, electronics will continue to have no rival for medium or high complexity information processing, while guided-wave optics will likely emerge as the choice for data communication at GHz rates between boards or even between chips. Consequently, efforts are being made to build networks using photonic switches to switch optical signals multiplexed in time or wavelength with switching controlled by electronic processors [8]. Directional couplers [9] are commonly used as switches because they are transparent to bit-rate and can switch a wide optical wavelength band, i.e. 25 nm or more.

We believe the high network bandwidth realizable with optics is an important factor in reducing latency to remote memory locations in multiprocessors. High bandwidth translates into high levels of pipelining, making more effective such widely studied latency-hiding mechanisms as prefetching [5] release consistency models and non-blocking writes [2], and multithreading [10]. Research machines such as DASH [2] and DDM [11], as well as commer-

cial machines (Kendall Square Research KSR-1, Cray T3D, Convex SPP, and others) have advanced understanding of hardware and software approaches to latency-reduction and latency-hiding. The Scalable Coherent Interface (SCI) standard has become a standard model for scalable cache coherency in multiprocessors. The Tera computer [10] exploits heterogeneous parallelism and demands a high-bandwidth network, with its very high clock rate, many multi-threaded processors, and single-cycle context switch. It can operate with hundreds of outstanding memory requests, and includes an electronic packet-switched processor/memory interconnect well-suited to optics.

Our motivation was to learn about practical issues that arise in building an optical interconnection network providing a high performance backbone for multiprocessors that employ caches and latency-hiding methods. The interconnect we have chosen supports the use of multiwavelength fiber-optic links and photonic switches. It has a ShuffleNet topology [12] and resolves contention by *deflection routing* (defined in Section 2.1), which is highly amenable to optical data transmission. We also use optical clock distribution of a global bit clock to minimize skew. Important issues such as reliability, host buffer flow control and overflow, packet re-ordering and other higher-level concerns related to the host interface are ignored to sharpen our focus on lower-level network details. Hardware and software support for memory

consistency is also outside the scope of our work.

As described in greater detail in our initial proposal for research, some requirements of a closely-coupled, distributed computer interconnect include:

- efficient transmission of short messages
- high sustained and burst bandwidth
- scaling to large numbers of users
- scaling to kilometer separations
- incrementally implementable architectures.

Hence, the following goals guided our research decisions for the multiprocessor interconnect:

1. Keep the source-to-destination latency as close as possible to the minimum set by the speed of light by minimizing deflections, and avoid electronic buffers on the critical path from input to output port.
2. Use electronics for relatively complex pipelined logic that processes packet headers and does contention resolution. If cost-effective, use optics for transmission and switching of packets, but leave the architectural path open for future use of optical logic throughout the node.

In any case, the architecture should support the gradual evolution to a fully photonic transmission and switching environment.

3. Keep the packet size small and fixed, in accordance with the majority of transfers in computer traffic. Small packet size lowers the average packet delay in the presence of large data transfers and reduces overhead for small transfers. Fixed packet size simplifies switching logic on the critical path to lower node delay.

4. Provide capacity for scaling to hundreds of nodes, and to several tens of Gbits/s link bit-rates. Network throughput should scale with nodes and with link bit-rate.

The next chapter is a brief review of the ShuffleNet and our previous research in the application of deflection routing in multiprocessor interconnects. The remaining chapters describe the fundamental results of our research. These results can be summarized as follows (references for journal papers and conference papers which have resulted from this research effort are included):

1. Further architectural studies have been completed, as described in Chapter 3, where we propose a multiprocessor interconnect architecture [13] which supports latency-hiding, a shared memory hierarchy, and directory-based cache-coherence.

2. Chapter 4 describes the implementation details of our prototype [14, 15, 16]. The prototype uses bit-parallel electronics, implements various switching protocols, and demonstrates point-to-point communications. System performance and cost tradeoffs have been formulated in order to assess the practicality of different link types and configurations (from fully electronic to fully optical).

3. In addition to our original research goals, we have analyzed packet synchronization in synchronous optical deflection networks [17]. The results are discussed in Chapter 5. A constrained minimization problem was formulated which enables link delays to be optimized during network design.

Chapter 2

Background and Previous Work

This chapter describes the ShuffleNet and highlights previous work in the area of deflection routing, including simulations and experimental research.

2.1 Deflection Routing and ShuffleNet

Deflection routing, first proposed by Baran in 1964 [18], and studied more recently by various groups [19], [20],[21], eliminates the need for electronic buffering of packets when there is contention for a switch output port. If two incoming packets need to use the same output port to minimize delay

to their destinations, then one packet is granted the requested port and the other is *deflected*, i.e. routed immediately, to another port. The payload (data portion of a packet) remains in optical form in a delay line while the header portion containing routing information is converted to electrical form and sent to a routing control processor (RCP), which makes a routing decision and modifies the header. The idea is to optimize network latency for the common case of a packet not being deflected, at the expense of increased latency for a deflected packet. A requirement for deflection routing is that a deflected packet can still reach its destination, implying a network having at least one path between any pair of switching nodes. Any such network with uniform link capacity can use deflection for contention resolution. Networks most suitable are those with low-degree nodes and multiple routes between any two nodes. Two commonly studied topologies with these characteristics are the ShuffleNet [12] and the Manhattan Street Network (MSN) [22]. The ShuffleNet has some advantages over the MSN which are discussed elsewhere [23]. We focus on the ShuffleNet because, at least for uniform load, the ShuffleNet gives lower latency than the MSN. We assume 2x2 nodes, unidirectional links, and link utilization of 80 % or less, to prevent throughput degradation that occurs in very highly utilized deflection networks. Our routing decisions are simple and amenable to pipelined feed-forward logic

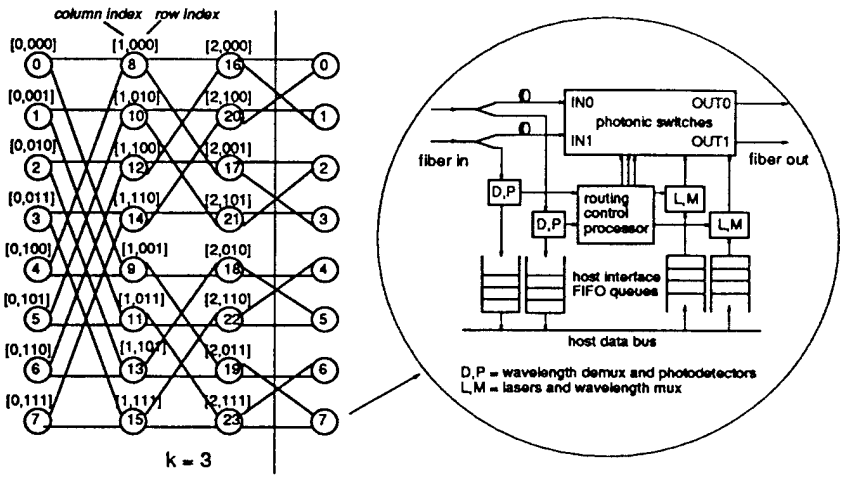


Figure 2.1: A 24-node ShuffleNet and diagram of node.

due to a regular and static network topology.

The ShuffleNet contains k columns of p^k nodes, with nodes in each column connected to nodes in the next by a p-way perfect shuffle. The total number of nodes with $p = 2$ is $N = k2^k$. While it impossible to add (or subtract) a single node to a filled configuration without introducing some inefficiency and asymmetry, this is not a problem for a fixed multiprocessor interconnect. Figure 2.1 shows a 24-node network. Column and row indices are shown in the brackets. The nodes form 2^k different rings, with row indices for nodes in a given ring being cyclic shifts of one another. Node i with column and row indices $[c_i, r_i]$ has its output ports connected to nodes with column index $(c_i + 1)$ mod k. Output port 0 connects to the node with row index equal to

r_i circularly shifted left once; output port 1 connects to the node having row index equal r_i with its most-significant bit complemented, circularly shifted left once. When a packet is deflected, it is sent around the network again, i.e. it must visit each column again. Since the number of columns grows roughly as $log_2 N$, the penalty for deflection also grows at this rate. The *distance* between source and destination nodes is measured in *hops*, or links along the shortest (minimum link count) path connecting the nodes. A property of the direct ShuffleNet is that for any source node, there exist many destination nodes having equal distance from either source output port. Thus, as a packet is routed, it may encounter intermediate nodes for which either output port can be selected without incurring additional hops. As an example, a packet from source node 12 in Figure 2.1 could traverse the network and reach destination node 18 in four hops via a path through nodes 12, 16, 0, 9, 18, or alternatively, through nodes 12, 17, 2, 13, 18. A packet that can be routed out either port is called a "don't-care" packet, while one requiring a specific output port in order to reach its destination in minimum hops is called a "care" packet for the node. If l is the distance between the packet's current node and its destination node, then the packet will care about its output port assignments only during the last $\min(l, k)$ hops [23]. Since the ratio of care hops to don't-care hops decreases with growing network size, deflection

routing is less costly in latency for larger ShuffleNets.

The RCP at each node assigns each through-going or host-generated packet to an output port by generating settings for the photonic switches shown inside the node diagram in Figure 2.1. Each switch can be placed in "cross" or "bar" state. A packet contains priority bits which get updated dynamically. The simplest priority scheme increments a packet's priority each time it is deflected. This associates an "age" with each packet; the older packet wins when two packets contend for an output port. If both have the same age, a winner is randomly chosen. Age priority reduces the variance of the flight latency probability distribution [23]. Only when an output port is not used by through-going packets can the host inject a packet. Because the RCP is pipelined, it can output new switch settings in each packet cycle, even though it may take multiple packet cycles to compute the settings.

2.2 Performance Analysis and Simulations

Extensive simulations have shown performance of different sized networks under varying link utilizations [23]. Simulations permit investigation of analytically intractable parameters such as priority scheme. Results of the simulations are highlighted here; more discussion can be found elsewhere [23], [24]. Using definitions in Ramanan's work [23], *latency* is the delay between the

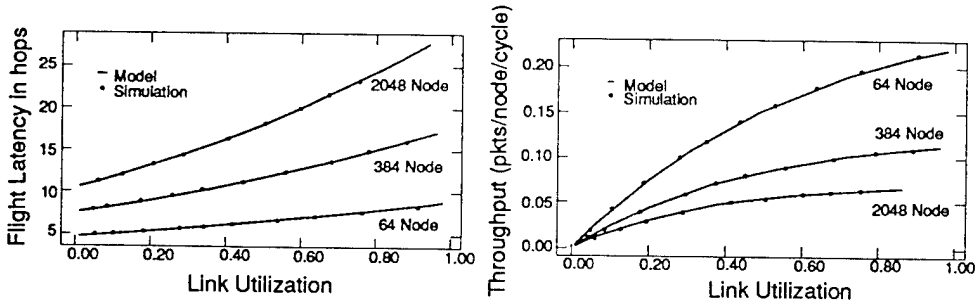


Figure 2.2: Latency and throughput versus link utilization (from [23]).

time a source node places a packet in its network output buffer and the time the packet is received by its destination node. Latency has two components: *flight latency* and *wait buffer time*. Flight latency measures the time between a packet's entry into the network and its reception at the destination node. Wait buffer time is the time a packet waits at the sender for a free output port [1]. *User throughput* is the rate at which a user's packets are injected into and received from the network; *network capacity* is the sum of the throughputs of all users. Packets remaining in the network longer due to deflections waste an increasing amount of bandwidth, and throughput decreases. In all studies, the network node includes 2 input and 2 output packet queues for the host. Figure 2.2 [23] shows how average flight latency and throughput depend on link utilization. The ShuffleNet is scalable, in that the network

[1]There is an additional component of delay called *transmission time*. Transmission time is bits per packet divided by link bit-rate.

capacity measured in users × packets/node/tick (where a tick is the distance occupied by a packet) increases with network size and with link bit-rate.

A variation on deflection routing called "2-Space, 2-Time" (2S2T) switching improves performance and is well-suited to optical switching [24]. In a 2S2T node, there is a space-time permuter which can exchange two incoming packets in time in an effort to reduce deflections. During every packet cycle, the space-time permuter considers 4 packet slots — 2 slots that have just arrived on both input ports and 2 slots immediately ahead of them in time — and permutes the slots as necessary to reduce deflections, then routing them to the node's output ports. The benefit of this algorithm is illustrated in Figure 2.3 [24]. 2S2T switching reduces the tail of the flight latency distribution and yields lower average flight latency compared to spatial switching alone, at the expense of slightly higher delay per hop.

Other groups have presented modifications to support guaranteed circuit services in deflection networks [19] and analyzed deflection behavior for ShuffleNet or similar networks [20]. Acampora et al. [21] compared deflection to store-and-forward on the ShuffleNet, showing that deflection gives poor throughput under the less interesting fully-loaded condition where every host injects a packet whenever a slot at its node is free.

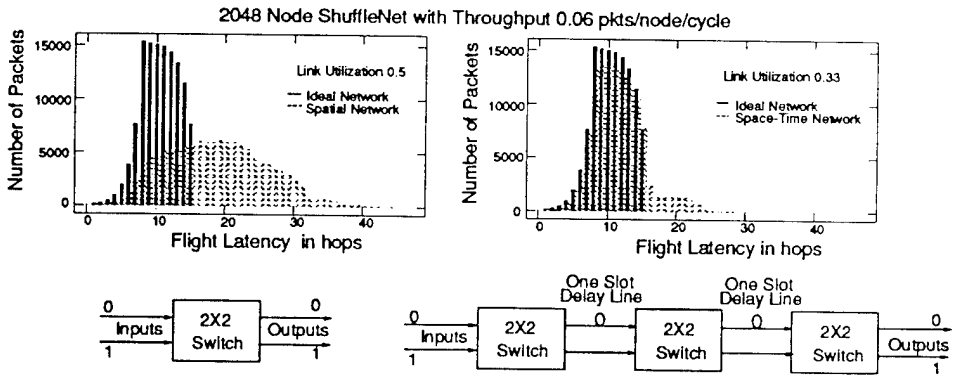


Figure 2.3: Flight latency histograms for 2048-node ShuffleNet, spatial and 2S2T switching.

2.3 Experimental Work

Earlier experiments in packet-switched deflection-routed networks in our group [25] demonstrated multiwavelength communications using a bit-per-wavelength (BPW) packet format, single-mode fiber for links, and lithium niobate ($LiNbO_3$) directional coupler switches. In the most recent experiment, the payload was 4 bits and the header contained 1 address bit and 1 priority bit. There was no connection to a host computer; data was generated from a programmable pattern generator, converted to BPW form using DFB lasers, and directed into the photonic switch. The RCP was implemented in TTL logic, had a single stage, and ran at a 10 MHz clock rate. The header was modified by the RCP electrically, converted to optical form

in a separate waveband, and reinserted with the outgoing optical payload. One output was re-routed back to an input port through a fiber delay line to simulate a network. The experiment showed the limit imposed by power loss: a packet could travel through the switch 3 times and traverse the network twice before its power was too low to be detected. Issues such as wavelength spacing, crosstalk, wavelength demultiplexing, bit-skew, and laser noise were also explored in this testbed [25].

Chapter 3

Multiprocessor System Model

This chapter briefly describes a shared-memory multiprocessor architecture that is compatible with a high-bandwidth deflection-based optical interconnect [13]. Its implementation is described in the next chapter. The architecture provides global memory synchronization, a request/response transaction model, error recovery, and support for multiple outstanding requests per node. The delay to access memory is different for local and remote memory locations so the architecture follows the Non-Uniform Memory Access (NUMA) model. Since our focus is on the interconnect, to simplify our work we designed the network node interface around a standard cache-coherent multiprocessor model. The inter-node transactions use a directory-based scheme similar to that in Stanford's Dash [2].

3.1 Node Architecture

The node architecture, as shown in Figure 3.1, is itself a small multiprocessor with local DRAM memory, a coherent split-transaction memory bus, and one to four processors with second-level caches using 32-byte cache lines. Only one processor is shown in each node of Figure 3.1. A standard snoopy-bus coherency protocol monitors all transactions over the node's local memory bus. The network has two different interfaces to a node's local memory bus. When a local processor wants to read a memory cell in a remote node the transaction passes through the memory interface which behaves exactly as a local memory bank except that it might have much longer latency. The

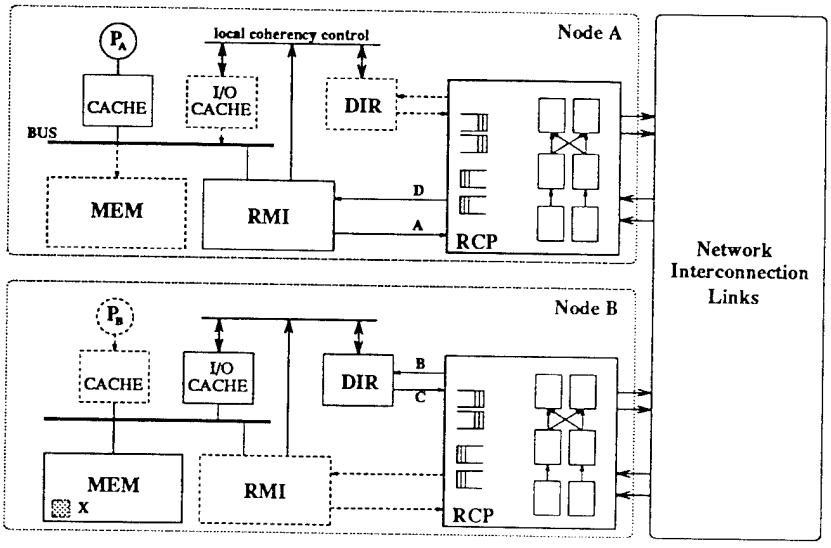


Figure 3.1: The network node architecture.

snoopy protocol on the local memory bus guarantees that no conflicts will occur between local processors. We also include a *coherent I/O cache* [26] in each node, with exactly the same interface to the node's local memory bus as the processor second-level caches. A remote node request to read or write data to the local memory passes through the local I/O cache, behaving exactly as a local processor memory request.

The extension we add to allow remote nodes to cache data belonging to the local node's memory is the Directory (DIR). The Remote Memory Interface (RMI) was also added to store tags and status for locally cached data belonging to remote nodes. The I/O cache is identical to a processor cache and guarantees that the reads and writes from remote nodes always are coherent with respect to local transactions.

There are two types of packets sent over the network: *requests* and *responses*. Requests always originate from a processor issuing memory read or write operations. If the operation is a write, the request packet includes the data to be stored. Figure 3.1 illustrates the operations described here. The path taken by a remote read is: processor P_A issues the read and it goes through the coherent cache, over the local split transaction bus and to the RMI, to the network as a read request packet (A), over the network, and into the remote node DIR (B) and I/O cache. The remote node issues a coherent

read from memory (x) and updates the DIR in accordance with the directory based scheme used for remote coherency. Data exits the remote node via (C) and is passed back to the initiating node (D) as a read response which passes through the RMI, and is finally given back to processor P_A.

Responses originate from the DIR on a node, and include data if the RMI is responding to a read request. The RMI also responds to coherency requests from the DIR in other nodes. The coherency operation may, for example, be an invalidate request for a cache line.

Latency hiding schemes such as third-level cacheing, non-blocking writes, and remote data prefetching may also be implemented in the RMI. The RMI contains a table for matching outstanding requests with responses and keeps a copy of a request in case re-transmission is necessary. Packets carrying payload contain a payload checksum which is checked at the receiver. If there is checksum error, the receiver sends a "negative acknowledgement" response packet back to the sender, and the sender re-transmits the packet. Since the deflection network never drops packets, the re-transmission protocol is greatly simplified; there is no need for re-transmission timers or duplicate packet suppression logic.

3.2 Outstanding Requests and Network Parameters

To guarantee that input packets are synchronized on the packet level at the input ports, we must constrain the network such that the delay in a pass around the network is an integral multiple of the packet cycle. Let T_W be the word cycle in nsec, and let $T_P = wT_W$ be the packet cycle in nsec, where w is an integer greater than or equal to 1. The RCP is pipelined using a global word clock, where r is the number of word cycles in the pipeline. Let l be the link delay in word cycles for each link. The delay in word cycles from the output of one node to the output of the next (one hop delay) is then $T_H = T_W(r + l)$. For example, in the ShuffleNet, with k columns of 2^k nodes, the input port constraint is expressed as:

$$kT_H = mT_P, \quad \text{with } m \in \{1, 2, 3, ...\}. \tag{3.1}$$

If $\langle E \rangle$ is average number of hops taken by a packet, then the average flight delay of a packet is $\langle E \rangle T_W(r + l) + T_W(w - 1)$. The maximum number of outstanding remote memory requests R_{out} that a node should support to make full use of the network bandwidth is related to the average round-trip (request plus response) delay, link bandwidth, and node degree. We estimate R_{out} here based on behavior of a lightly loaded ShuffleNet. The word cycle

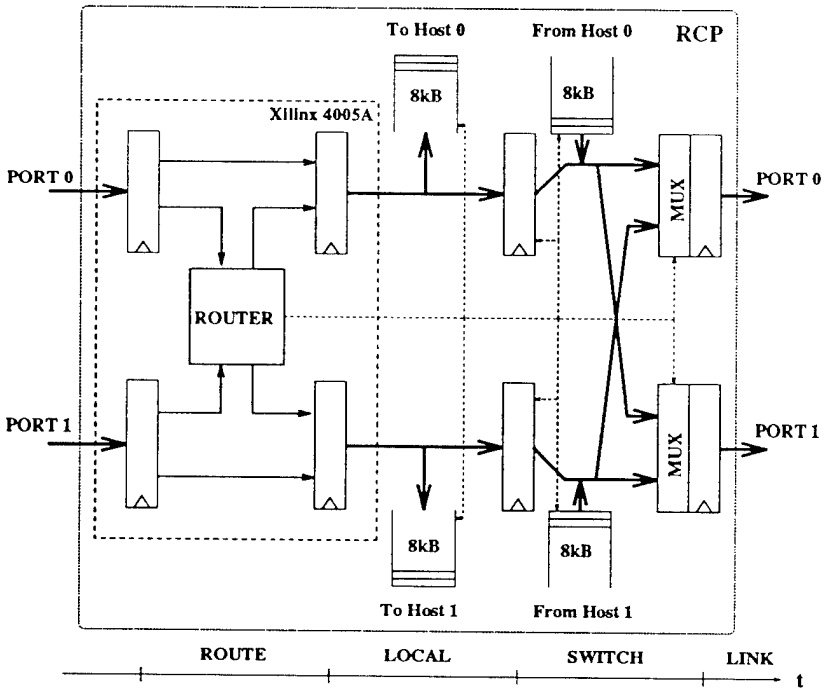


Figure 3.2: The Routing Control Processor (RCP) pipeline.

is about 5 nsec if the RCP is implemented in a CMOS gate array. With a b-bit word, this gives a link bit rate of $b \times 200$ MBit/sec. Assume that a node receiving a write or read request can respond in zero time (i.e. ignore memory latency or processing time in the responding node). Then the average round-trip delay for a transaction, in units of packet cycles, is twice the average hops $\langle E \rangle$ times the packet cycles per hop $(r+l)/w$. In our prototype, with an RCP pipeline as shown in Figure 3.2 (data paths are presently implemented with electronic registers), we have $r = 3$. If we assume a 4 word-cycle

packet ($w = 4$), a link delay of one word cycle ($l = 1$), and a 384 node ShuffleNet ($N = 384$, with $\langle E \rangle = 12$ at a link utilization of 0.8 [23]), then the number of packet slots available in the average round-trip transaction is $2\langle E \rangle (r + l)/w = 2 \times 12 \times (3 + 1)/4 = 24$. The factor of two is necessary since we defined a round-trip as a request and a response. Furthermore, since the node has two output links, $R_{out} = 2 \times 24 = 48$. The hop delay with $T_W = 5$ nsec is $T_H = 20$ nsec, and the round-trip delay is $2\langle E \rangle T_H = 480$ nsec. If there are four processors per node, then each contributes some fraction of the total outstanding requests using latency-hiding methods.

On the receiving side, the worst-case scenario for receiving memory requests is that all processors send request packets to a "hot-spot" node at the same time, meaning that $N \times R_{out} = 18432$ packets need to be held in the receiver's input buffer. For a 42-byte packet carrying a 32-byte cache-line, for example, 774 KB of buffer storage is needed if there is no flow control and deterministic packet switching is used. If we use deflection routing, however, and the buffer is smaller than 774 KB, we have the advantage of being able to deflect an incoming packet to avoid dropping it. The network itself can be used effectively as additional buffer space. We can afford to use much smaller input buffers to cover the common case, while wasting a small amount of network bandwidth for additional storage in the worst case.

Chapter 4

Prototype Implementation

4.1 Overview and Packet Format

To our knowledge, the prototype we describe here is the first implementation of an interconnect with deflection routing and commercial workstation hosts at each node that, in its final version, will feature all-optical data paths [14, 15, 16]. Our main contribution is a complete node implementation including host ports and support for realistic packet sizes with user payload. The service provided by the network to the user is an unreliable, point-to-point, fixed packet-size transportation service with no guarantee of correct ordering. Our network performs error correction only on packet headers. If necessary, the user must do flow control, packet re-ordering, re-transmission, and error

detection/correction on data received from the network.

The node hardware consists of two printed-circuit boards which plug into the host workstation's expansion bus slots, an EISA bus in an HP 715/33 workstation. The first board is called the **RCPB**, shown on the right half of Figure 4.1. It implements the host interface, RCP, and hardware to support electronic packet switching, as in Figure 3.2. The other board is called the **OPTOB**, which contains optoelectronic components including $LiNbO_3$ photonic directional coupler switches, lasers, receivers, wavelength multiplexers and demultiplexers, and polarization controllers, as well as electronic parallel-

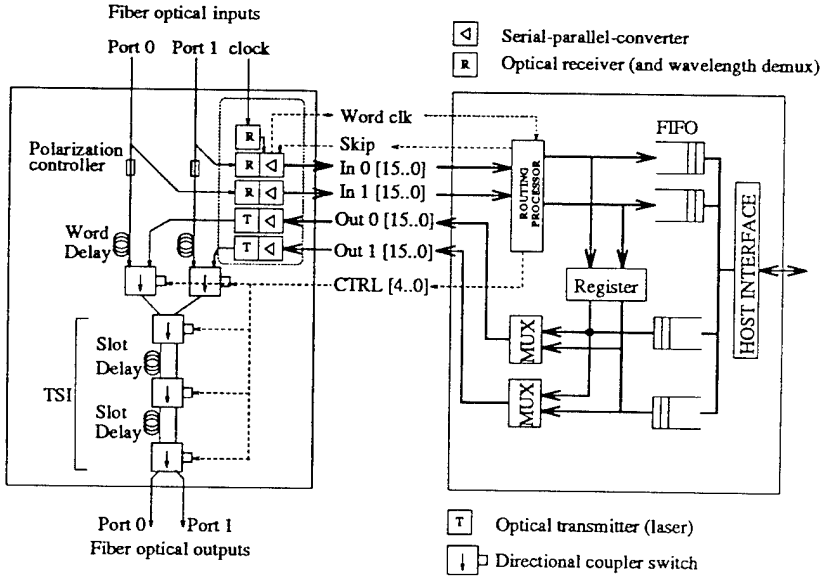


Figure 4.1: Optical board (**OPTOB**) and Routing Processor/Host Interface Board (**RCPB**).

serial converters. The node is configured to perform switching electronically on the **RCPB** when $LiNbO_3$ switches are absent. The **OPTOB** board can drive links having any of the following formats: electronic ribbon cable, "fiber ribbon," serial electronic or fiber, or fiber BPW (with an arbitrary number of parallel wavelengths as supported by the transmission bandwidth of photonic switches or optical amplifiers in a geographically distributed network).

The **RCPB** uses two Xilinx 4005A field-programmable gate arrays (FPGA's) [27] for most of the hardware (see Figure 3.2). Changing the packet format or routing algorithm is done simply by downloading a different FPGA configuration. The parameters that are static in the design are: port width (a 16 bit word for compatibility with the host processor I/O bus), minimum RCP packet header processing time of 25 nsec, and data path layout. A packet comprises a fixed number of 16-bit words; the *number* of words is not hardwired in the prototype. Our goal was to provide an architecture that supports many different packet and header formats.

The hardware is free of electronic buffers on the critical data path between network input and output ports, but electronic FIFOs are used between the **OPTOB** and the EISA bus I/O interface to the host processor. We include no hardware for synchronizing the arrival of packets at the optical input ports, and instead rely on a global bit clock. We restrict link delays to be

uniform and equal to an integral number of word cycles. Framing procedures are used at network startup to synchronize the first bit in a word and the first word in a packet. For small networks, skew in arrival times is negligible, i.e. much less than a bit period. In larger networks a global clock is not feasible, so we would need a dynamic optical input packet synchronization mechanism. This is a topic of current study [28]. Packet synchronization in synchronous networks with non-uniform link delays is discussed in more detail in Chapter 5.

Each network node receives and transmits a continuous stream of *slots*. A slot is either empty or contains a packet. If it contains a packet, the slot is "full" and the packet header is examined by the RCP to assign an output port to the packet. The simplest packet format uses one byte for header and one byte for payload. Figure 4.2 shows a sample occupying multiple word cycles, with 16 bit-parallel wavelengths, λ_0, ... λ_{15}. Presently, the links are implemented in electronic parallel fashion using ribbon cable. The prototype packet header has these fields:

Empty/Full Bit indicates whether or not slot is full.

Address For our initial 8-node network, the address consists of 2 row bits and 1 column bit ($N = 8 = 2 \times 2^2$).

PRI/TTL 4 bits are used for time-to-live stamp and/or priority fields.

These fields may be separately encoded in the 4 bits or treated as the same 4-bit field.

Header ECC Error correction code bits for header (not shown in Figure 4.2).

4.1.1 Routing Processor and Host Interface

The host interface is designed for the industry-standard EISA I/O expansion bus. It includes two incoming and two outgoing 16-bit wide FIFOs which are read and written, respectively, by the HP 715/33 host processor. The FIFOs buffer short bursts of packets (up to 8K words per port, see Figure 3.2) and

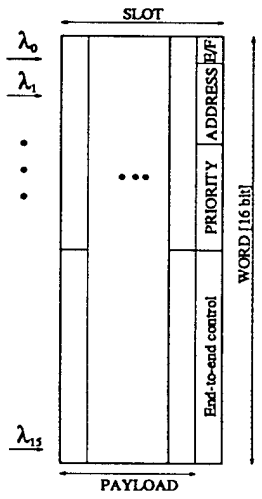


Figure 4.2: Routing processor packet format.

also handle the asynchronous border between the host processor and the network. The devices have independent read and write clocks; the host supplies the clock for reads and writes on the EISA bus. A host packet is only injected if there is an empty slot on its preferred output port. The **RCPB** contains an electronic packet switch in the form of two 16-bit wide 4-to-1 registered multiplexers, one for each output port. This is done in discrete high-speed CMOS. There were three reasons for including this hardware. First, it is possible to do re-generation of through-going packets whose optical power is near the lower limit for detection. While this method violates a basic premise of an all-optical network, it eliminates the need for fiber amplifiers [29], which are currently very costly. Secondly, the electronic switching network supports a broadcast feature whereby the host can inject a packet onto both output ports in the same cycle. In this way, we can artificially increase the prototype network load by duplicating packets. Thirdly, each host queue can be directed to *either* output port. This feature would be more costly if using 2×2 photonic switches.

The critical path (longest logic delay) in the routing processor FPGA limits the word rate and packet rate through the node. For the prototype this path performs the following functions in one word period: map address in header of input packets to preferred port (0 or 1); check for conflicts on

preferred output ports; compare priority fields for contending packets; update priority field of packet to be deflected; and drive output enable signals from the FPGA to a pipeline register on the **RCPB**. This path is relatively simple, i. e. only a few gate levels.

The critical path from input port to output port includes the above path plus optical/electrical (O/E) conversion at the input to **RCPB**, mux-register delay, and electrical/optical (E/O) conversion at the mux-register output (see Figure 4.1). An alternative to our use of logic gates for preferred port computation is routing tables implemented in RAM. Routing tables might be useful to dynamically customize packet routes in a network. Since table size increases linearly with network size and its access cannot be pipelined, however, scalability is poor. A hybrid solution that might work well in our network is to use logic gates to implement the default routing algorithm, in parallel with a small dynamically writable routing table enabled when there is a link or node failure.

4.1.2 Clock Distribution

All nodes operate with a global bit-clock signal which must be distributed with a minimum of skew. High-precision control of propagation delay is difficult in electronics. Therefore we use a global 800 nm laser as the clock

source, distributed to every node on multimode optical fiber constructed in a binary tree using 1-to-2 fiber splitters. Optical clock distribution is simple and relatively cheap: a single commercial 50 mW laser provides enough power to supply the clock to over 1000 nodes. Research has shown it is possible to control dynamic jitter to within several picoseconds in a system using optical clock distribution [30]. Manual adjustment of optical path length to correct for static O/E receiver skew is also easy with optical fiber. Since the payload is never latched along its flight, the timing budget must acccount for uncertainty in payload arrival time at the nodes due to manufacturing and temperature variations. In the prototype, however, it is power loss that limits the maximum hops allowed in the network rather than this timing uncertainty.

4.1.3 Scalability

Our node architecture scales to large numbers of nodes, on the order of several hundred, which is roughly the size of current multiprocessors. Complexity of header processing logic scales logarithmically with the number of nodes in the network. Since the logic is simple and feed-forward, however, it can easily be pipelined. Therefore packet rate need not be limited by this logic delay, though node latency will increase slightly with further pipelining. Ap-

plication of wave pipelining [31] could potentially lower the latency penalty of pipelining. A single-chip CMOS implementation of the RCP could decrease node latency and increase packet rate by an order of magnitude, up to 200-300 MHz. By using fiber optic links and higher bandwidth lasers or more wavelengths instead of the ribbon cable currently used in the prototype, we could increase link bit-rate from 400 Mbit/s to 4 GBit/s, directly impacting throughput by multiplying the available packet slots. These components drive up the cost per node considerably, as will be discussed in Section 4.3, so one would need to weigh the advantage of higher throughput against the higher dollar cost.

4.1.4 Current Status

We have debugged a single-node electronic prototype of the interconnect with a 32 MHz word rate, connected to itself in loop-back fashion. With a 16-bit word size the link bit-rate is 512 Mbit/s. We are currently building an **OPTOB** with bit-serial optical links and optical clock distribution. In parallel, we are building additional **RCPB**s for use in an 8-node network. We are using an HP PA-RISC 715 workstation with EISA expansion slot as the host computer.

4.2 Optical Implementation

4.2.1 Bit-Per-Wavelength (BPW) encoding

The network interface supports multiwavelength encoding. Bits of a packet can be transmitted through several wavelengths in parallel, using BPW encoding [25]. BPW makes it possible to use a single optical switch to switch a parallel data word. For a given aggregate link bit-rate, BPW allows slower, i.e. cheaper, electronics to perform serial-to-parallel conversion than a single wavelength does. We use BPW on header and data in the same waveband centered at 1300 nm to simplify testing, though use of separate wavebands has also been studied [25]. When BPW is used to increase link bandwidth, the benefit is lower transmission time and waiting time. Bit-skew, the difference in arrival times of different wavelengths at a receiving node due to chromatic dispersion, can become a bit-rate limiting factor if links are long or a packet travels many hops in optical form from source to destination. For example, if a packet covering a 50 nm wavelength band has a total flight of 100 meters, the bit-skew assuming dispersion of 16 psec/nm-km is about 80 psec, implying a per-wavelength bit-rate of no more than 3.1 GBit/sec. This skew is small, however, compared with the static manufacturing skew in optical receivers, which will limit bit-rate before anything else.

4.2.2 Optical Component Details

Refer to Figure 4.1 for details of the **OPTOB**. At each optical input port, power splitters are used. One splitter output goes to the fiber delay line before the first stage of $LiNbO_3$ switches and the other output goes to the wavelength demultiplexer and array of optical receivers. The output going to the delay line passes polarization controllers before it reaches the polarization-dependent $LiNbO_3$ switches. Polarization controllers are also needed between each $LiNbO_3$ switch stage. Each input port also is driven by a wavelength multiplexer combining optical power from a set of laser diodes that drive bits of the injected host packet. A breakdown of the optical components is as follows:

Lithium niobate ($LiNbO_3$) **switches** are suitable in systems using BPW encoding due to their wide optical bandwidth window, allowing many wavelengths to be switched simultaneously. Our switches were built by AT&T, have a 25 nm wavelength window centered at 1300 nm,[1] and a 5 V switching voltage. A guard band of 2-3 nsec between packets to cover switching rise/fall time is necessary. If a single node loses electrical power it can still pass through-going packets, with switches

[1]To be compatible with optical amplifiers, switches with center wavelength at 1540 nm are necessary.

defaulting to the "cross" state. All packets will still reach their destinations, except those destined for the faulty node. Fault tolerance is higher than it would be using electronic switches.

Polarization controllers induce mechanical stress to match the electric field polarization at the fiber output to the state needed for proper $LiNbO_3$ switch operation. Polarization-independent switches [32] obviate these bulky devices, with the penalty of higher switching voltage. Another alternative is polarization-maintaining fiber.

PIN diode photodetectors convert the optical packet to electronic form. Since detectors respond to a wide wavelength range, detectors for different wavelengths are identical.

Distributed feedback (DFB) InGaAsP lasers donated by Bell Northern Research convert packets from electronic to optical form in the 1300 nm wavelength range.

Fiber splitters are used in the clock distribution network and also to split power received at each input port. Those in the clock network are 50-50 splitters. The splitter at the input port directs 90 % of the power into the switch and 10 % into the host packet demultiplexer/receiver array.

Single-mode fiber must be used for all links because the $LiNbO_3$ switches do not work with multimode fiber.

Optical demuxing and muxing is done using monolithic guided wave devices. At this time their exact configuration or price is not known because they are not commercially available. Our previous experiments used diffraction gratings and bulk optics [25], but for a multiprocessor interconnect, insertion of lenses and free-space subsystems into the host workstation is infeasible. We are investigating use of an InP spectrometer chip that performs muxing and demuxing of many wavelengths (up to 78) spaced at 1 nm. This device has been demonstrated in the laboratory, is not sensitive to polarization, and could potentially be integrated with arrays of photodetectors or laser diodes to provide compact BPW receivers or transmitters [33]. Other techniques for multiplexing/demultiplexing appear to be promising as well [34].

4.2.3 Power budget

Optical power losses are an important consideration. Regenerating optical signals using detection and re-transmission is undesirable because this process limits bandwidth and increases flight latency. However, some means of restoring power is necessary because with deflection routing we cannot place

an upper bound on the number of hops a packet will take. Our $LiNbO_3$ switches introduce a 5 dB loss due primarily to coupling losses between the fibers and waveguides. This loss severely limits the hops a packet can take can take before its power is too small to maintain a desired SNR. Current research in monolithic semiconductor laser amplifier optical crossbar switch devices with 0 dB insertion loss and 1540 nm center wavelength [35] could be applied to our architecture. Erbium optical fiber amplifiers [29] also amplify power and have an optical bandwidth window that is roughly the same width as that of our $LiNbO_3$ switches [25]. To amortize the high cost of the amplifier pump laser, a future system might have a few centrally located pump lasers distributing optical power to all links containing amplifiers.

4.3 Cost and Performance Comparisons

4.3.1 Interconnection network and links

The physical interconnection network contributes much of the complexity in a large scale multiprocessor. The most important parameter for a multiprocessor node is its aggregate bandwidth to and from the network, i.e. the sum of the bandwidths over all incoming and outgoing links. This is a function of the bit rate per physical channel and the number of connections that is

feasible between boards, limited by the amount of inter-board space or by connector technology. The maximum link bit rate restricts sharing of links and forces designs to use multi-path and multi-stage networks, bit-parallel transmission, optical links and other schemes that deliver high bandwidth service to the nodes.

The cheapest links are parallel electronic wires which can transfer up to a few 100 Mbit/s per wire for the typical inter-node distances in a multiprocessor. State of the art electronic implementations such as the CRAY T3D [36] with 196 bits per processor input and output give each processor module an aggregate link bandwidth of 98×150 MHz $= 14.7$ Gbit/s . The extra cost of optical interconnections would be wasted in such a machine. For higher bit rates, in the 300 MBit/s - 10 Gbit/s region, optics is beneficial or even necessary for inter-board connections [37]. Other advantages to optical waveguide links over electronic technologies such as coaxial cable, differential twisted pair, or flexible microstrip are the isolation it provides between modules, lower power dissipation, and less space taken up by cables. Further, ground pins separating signals and termination devices are not needed, giving potentially much higher PCB edge connection density than available in electronics, though commercial optical connectors are still in their infant stage.

Four different link configurations we have considered are given in Table 4.1. We assume enough pins per node to give the same order of magnitude link bit-rate for each configuration, and a link length on the order of half a meter. The Bw/channel row measures the bit-rate per wire or fiber, or per wavelength in the case of the Fiber BPW system. The Bw/node row measures the total outgoing bit-rate, which is twice the link bit-rate. We also estimate the maximum feasible length for each link for the given channel bandwidth. The last row assesses scalability to larger multiprocessor systems, factoring in both bandwidth scalability and manufacturing complexity. The static bit skew values assume that we manually adjust the optical fiber lengths to compensate for electronic delay variation in receivers. Using this technique, worst-case node-to-node clock skew can be kept to within 20-50 psec, based on skew performance reported by Nordin [37].

Link technology	Electrical BP	Fiber serial	Fiber BP	Fiber BPW
pin count/node	128-256 (50% overhead)	4 fibers	128 fibers	4 fibers
Bw/node [Gb/s]	1.6-12.8	2.0-4.0	32.0	4.0-100.0
Bw/channel [Gb/s]	0.05-0.2	1.0-2.0	0.5	0.5-2.0
max. link length	0.5 m	1 km	100's m	1 km
isolation	no	yes	yes	yes
static bit skew	500-1000 psec	50 psec	50 psec	50 psec
Scalabilty	medium	low	high	very high

Table 4.1: Impact of link technology on system performance and complexity.

Bit-parallel electrical links (Electrical BP) are used in the current version of the prototype. The prototype uses 16-bit wide links, but the table assumes 32-bit wide to allow a fair comparison with the fiber parallel links. Scalability is medium since the increased bit-rate achieved by adding wires is balanced by added complexity and larger connectors. Connector signal pin density is limited by the need to provide an equal number of ground pins if the signals are switching at high speed. Bit-serial fiber transmission requires very high speed serial-to-parallel converters, lasers, and receivers. This scheme has low scalability since bit-rate is limited by electronic speeds and by irreducible skew between clock and the serial data stream. Bit-parallel optical transmission (Fiber BP) assumes a 32-bit wide fiber-ribbon of the type proposed by the DARPA OETC Consortium [38]. Channel bit-rate in this case is limited by bit skew over the channels, but can be increased by adding additional fiber ribbons. Manufacturing complexity is relatively large since each node needs a 128-bit wide fiber connector. Bit-per-wavelength (Fiber BPW) transmission offers the most scalable solution, at the added cost of additional lasers and wider wavelength muxes/demuxes. For a given aggregate link bit-rate, BPW allows slower and cheaper electronics to perform serial-to-parallel conversion versus the bit-serial fiber system. When BPW is used to increase link bandwidth, the benefit is shorter transmission time and user waiting time.

4.3.2 Breakdown of Costs

Table 4.2 shows components costs and link bit-rate for each of six different node configurations, assuming a 64-node network. Configurations vary in the switching technology (electronic or $LiNbO_3$ switches) and in the type of links. Link types include electronic parallel (EP), fiber parallel (FP) of the type proposed by the DARPA OETC Consortium [38], electronic serial (ES), optical serial (OS), and optical bit-per-wavelength (O BPW). Each uses optical

	Optical board implementation					
	Switching	Electronic	Electronic	Electronic	Electronic	Optical	Optical
	Links	E P	F P	E S	O S	O S	O BPW
Component							
Optical fiber		-	500	-	100	100	100
Optical switches		-	-	-	-	9000	9000
Lasers		-	3000	-	4000	4000	64000
Detectors		-	2000	-	2000	2000	32000
λ demux		-	-	-	-	-	2000
λ mux		-	-	-	-	-	60
Electronics		1160	1160	2860	2860	2860	28360
Electronic links		300	0	60	0	0	0
Optical clock		1070	1070	1070	1070	1070	1070
Total node cost		2530	7730	3990	10030	19030	136590
Max. link bitrate [Gbit/s]		3.2	8.0	0.5	2.4	2.4	38.4
cost /[Gbit/s]		791	966	7980	4179	7929	3557

Table 4.2: Breakdown of costs (in dollars) for various network node configurations.

clock distribution so that bit rate can easily scale to larger networks. A node introduces 4 word cycles of delay, including link delay, for all of the configurations. The table assumes spatial switching is used, though the incremental cost of going to 2S2T switching is negligible. The $LiNbO_3$ switches come in "6-pack" modules; the total switch count for 2S2T is five, so no additional $LiNbO_3$ modules are needed. Additional polarization controllers are needed between switches, but they are inexpensive.

Prices for the fiber-parallel transmitters and receivers as well as wavelength muxes and demuxes represent our best estimates, since these devices cannot be purchased yet. Prices of all other components are for commercially available parts. For each configuration, our calculation of link bandwidth assumes the link driver runs at its maximum specified rate, to maximize its cost-effectiveness. Electronic parallel links give cheapest cost per Gbit/s. Fiber parallel links with electronic switching give the second-best price/performance, and have the advantage of supporting longer links (see Table 4.1) due to lower bit-skew and much higher bit-rates. The three configurations with purely serial links are evidently not cost-effective. The BPW system assumes 16 wavelengths to maximize the system value of the expensive $LiNbO_3$ switches. The full 25 nm wavelength window is used with 1.5 nm spacing. The high cost of the BPW node is mainly due to expensive discrete

packaged lasers with narrow line width and single-mode fiber coupling. The optical switches could still function in a BPW system having much higher aggregate bit rate. Using data from Ramanan's work [24], assuming uniform load and 60 % link utilization, the average hop count for $N = 64$ is roughly 7.[2] With a link bit-rate of 38.4 Gbit/s, this gives aggregate user throughput of $\frac{2}{7}(0.6)(38.4)(64) = 420$ Gbit/s, and average user throughput of 6.6 Gbit/s.

[2]With the 25 MHz **RCPB**, 7 hops equals about 1.1 μsec.

Chapter 5

Packet Synchronization

5.1 Introduction

Packet-level synchronization is the process of ensuring that incoming packets are bit-wise aligned to each other before they enter the RCP (see [17] by Feehrer and Ramfelt). Deflection routing works properly only if packets are aligned, because header bits arriving from the input ports must be compared "on-the-fly," with no elastic buffering to correct for mismatched arrivals. Nodes do not have uniform separation if we assume that the nodes are spread across boards, racks, and cabinets. Packets can be synchronized using either a synchronous or an asynchronous method. Synchronous systems have a global clock distributed to every node, providing a timing reference for transmission

of outgoing packet bits and reception of incoming packet bits. Most if not all of the large multiprocessors built for commercial application (e.g. Thinking Machines CM-5 [39], Intel Paragon [40], Tera Computer [41], Cray T3D [36]) implement a synchronous timing philosophy. A weakness in this approach is that clock and data skew place an increasingly severe limit on the link bit-rate, and thus network throughput, as the the number of nodes increases. In high-speed electronic networks, fanout buffers and backplane connections with extremely high delay accuracy and high signal integrity must be used and may require manual adjustment, driving up the cost and complexity of building a synchronous system.

In an asynchronous deflection network, each node has its own local clock. The absence of buffering in each node means that packets move constantly through the network until they reach their destination. This flow-through style of packet switching rules out handshaking schemes, which might be employed in store-and-forward networks. Instead, packet synchronization hardware to perform framing pulse recognition and elastic buffering using tunable delays must be placed before the RCP. This hardware dynamically adjusts packet arrivals and synchronizes them to the local clock [42]. It must be implemented with high-speed electronic or optical logic to supply adequate timing resolution. This hardware increases node cost and latency through

the node. If there are two input ports, a packet synchronizer will on average delay one of the packets half the packet cycle. On average, a packet will suffer an increase in node delay equal to 25% of the packet cycle, relative to the synchronous case. Since the local clocks will vary slightly in frequency, packet loss is also a potential problem. The advantage of improved scalability to larger networks provided by asynchronous timing must be traded off against these shortcomings.

5.1.1 Optical Waveguides

The availability of optical waveguides for signal transmission, however, improves the scalability of synchronous networks, allowing large (up to room-size) systems and making the asynchronous design less attractive. Besides the commonly cited advantages such as high transmission bandwidth, high connection density, low loss, and immunity from electromagnetic interference (e.g. [7]), optical interconnects offer lower skew than their electrical counterparts. Capacitive loading, the source of most delay uncertainty in electronics, is not present along optical paths [37]. This advantage has been demonstrated and quantified by various groups studying optical clock distribution for computers [43, 30, 44]. Nordin [37] showed that static skew can be controlled to within 5 psec per meter in optical fiber, an order of magnitude

better than in teflon PCB traces and two orders of magnitude better than in coaxial cable. Research on how delay varies with temperature has reported 38 psec/km/deg.C [45], translating to about 2 picosecond variation for a 10 meter fiber and a 5 degree temperature drift.

The inherently low skew in optics makes possible synchronous deflection networks with hundreds or thousands of nodes distributed across several cabinets. Optical waveguides are used for both clock distribution and network data links. The length of clock fanout connections does not directly impact the clock rate as it might in equipotential systems, since multiple clock transitions can be pipelined over fanout paths, an idea also proposed for VLSI chips [46]. Total system cost should be lower than that of an asynchronous system running at the same bit-rate, due to the absence of packet synchronizers. This cost difference will become more dramatic as lasers, optical receivers, and optical connectors become cheaper with more widespread commercial use. For these reasons, we believe synchronous timing to be superior to asynchronous timing for room-size deflection networks having maximum link length on the order of 10-15 meters. Here we focus on the problem of packet-level synchronization for these networks and propose a method for *designing* link delays to accomplish this goal. The method capitalizes on the high controllability of delay in optical waveguides and keeps source-to-destination paths as short as

possible. It is less useful in electrical networks, where delays have relatively low precision.

5.1.2 Related Work

To our knowledge, the work we report here is the first study of packet synchronization and link delay optimization for synchronous optical deflection networks. Architectural studies have examined deflection network latency at a relatively high level of abstraction, expressing performance in terms of *hops* and number of deflections [24, 19, 20, 21, 47, 48]. Much of this work has examined telecommunications or metropolitan-area data communication networks with nodes separated by kilometers rather than meters. In that context, there is little concern with optimizing links delays at the clock-period level. One exception is the design of the deflection network for the Horizon architecture [49]. In that design, however, the explicit objective was to *equalize link lengths* throughout the network, and a method for *folding* the 3-D toroidal network into a planar graph to achieve this objective was presented. For large networks, we believe this method may conflict with physical layout restrictions that force the network to be distributed across multiple cabinets or racks. How to decompose a large multistage network into modules so that nodes can be efficiently spread over multiple boards and racks was studied

by Batcher [50], though he did not address link delays and synchronization constraints.

Other previous theoretical and experimental work in our group has explored "time-of-flight" synchronization, a procedure for adjusting path delays to synchronize digital optical circuits [51]. Time-of-flight circuits have no bistable memory devices such as latches or flip-flops. We have successfully implemented a general purpose time-of-flight digital optical computer which implements memory using optical delay lines [52]. The following analysis represents the first application of optical time-of-flight design principles to practical network design.

5.2 Review and Terminology

Figure 2.1 showed a canonical illustration of an optical deflection network node. While degree-2 nodes (2 inputs, 2 outputs) were used, the method described here applies to nodes with any degree. (The interface to the node's host processor that allows the host to inject and receive packets does not affect network synchronization constraints.) As before, each packet contains a *payload* consisting of data bits and a *header* examined by the RCP and used to perform self-routing along the path from source to destination node. The payload remains in optical form in a delay-line while the header is converted

to electrical form and sent to the RCP, which makes a routing decision and modifies the headers. The modified headers are converted back to optical form and re-inserted into the outgoing packets.

In a brief review of Chapters 3 and 4, each node receives and transmits a continuous stream of *slots*. A slot is either empty or contains a packet. All packets have the same size. The width of a slot is the *packet cycle*, denoted T_P. The RCP is pipelined with a *word clock* having period T_W, to decouple packet processing latency from the packet cycle. We assume that T_P is an integral multiple of T_W. The phases of the word and packet clocks are adjusted locally at each node using a framing procedure done once during initialization [14, 16]. The global bit clock is distributed optically to all nodes using a tree of optical splitters. Multiple packets may be pipelined over a link. Alternatively, the bits of a single packet packet may be spread across multiple links and nodes at any given time, depending on the RCP pipeline clock rate, number of stages, bit rate, bits per packet, and link delays. In this sense, deflection routing is similar to wormhole routing [53], except in deflection routing contention for resources is resolved immediately with no blockage of packet flow.

The need for synchronized packet arrivals imposes constraints on delays of the network links. Mechanical packaging details such as board, rack,

or cabinet layout also impose minimum delay constraints on links. Define a *network path* as an alternating sequence of nodes and links forming a directed path, beginning at an input port of a *source node* and ending at an input port of a *destination node.* The *path delay* is the sum of RCP and link delays along the path. Packet-level synchronization is achieved if, for any pair of network paths having a common destination node, the difference in the path delays for the two paths is an integral multiple of the packet cycle.

Time-of-flight packet synchronization relies on static (i.e. done at design time) re-distribution of path delays such that the above condition on path delay differences is not violated. A path delay is altered by altering its link delays. In optics, a link's delay is directly related to its length, so delay adjustment is straightforward. For shared-memory system performance, it is critical to minimize packet flight latency from source to destination. We therefore seek a set of link delays that is optimal rather than a set that merely satisfies all constraints. The network is modeled as a directed graph, with propagation delays of links treated as variables. The re-distribution of delays is similar to the re-distribution of registers that occurs in *retiming* for VLSI [54]. In retiming, registers are shifted forward or backward along logic paths to minimize the clock period while preserving the circuit's timing relationship with the outside world.

Although we use the ShuffleNet [12] for illustration, our method is general enough to work with any topology having at least one directed path between any pair of nodes. The directed graph model for an 8-node ShuffleNet ($k = 2$) is shown in Figure 5.1a. Each network node is represented by a vertex; each optical link is represented by an edge. An edge is directed from its *source* vertex to its *destination* vertex. The graph is denoted $G = (V, E)$. Every edge $e_i \in E$ is assigned a delay. The edge delay has two components: node processing delay D_R, which is the same for all edges, and a variable link

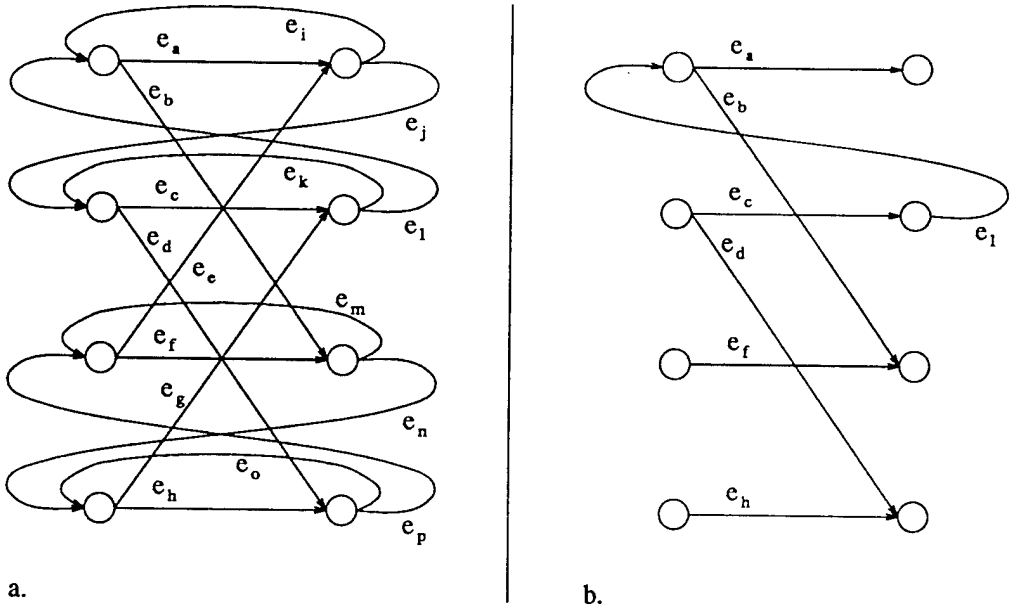


Figure 5.1: (a) Directed graph model for 8-node ShuffleNet. (b) Spanning tree for graph.

delay $D_L(e_i)$. See Figure 5.2. Processing delay introduced by the RCP is the delay from the time the first bit of a packet is sampled at an input port to the time the first bit exits an output port. Note that D_R must be an integral multiple of the word cycle T_W but not necessarily of the packet cycle T_P. The total delay for edge e_i is denoted by the variable $d(e_i)$, with $d(e_i) = D_R + D_L(e_i)$.

5.3 Constrained Minimization Problem

The integer program is constructed as follows:

1) Form a spanning tree $T = (V, E_T)$ for the underlying undirected graph by removing a set of $|E| - |V| + 1$ edges. The edges which are removed make up the *co-tree* [55]. A spanning tree for

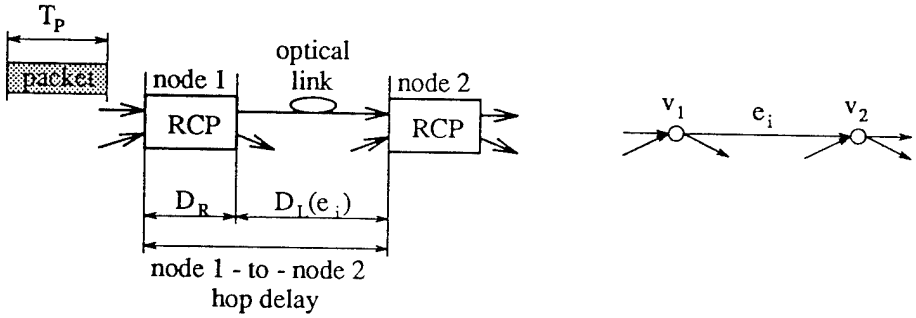


Figure 5.2: Two components of hop delay: processing delay D_R and link delay $D_L(e_i)$.

the 8-node network is shown in Figure 5.1b.

2) For each edge in the set of co-tree edges $E_{CT} = E - E_T$, add the edge back to T. A directed or undirected loop is thus formed for each co-tree edge. The set of loops form a *loop basis* or *circuit basis* [55] for the graph. This set is denoted $L = \{l_1, l_2, ..., l_M\}$, where $M = \mid E \mid - \mid V \mid + 1$. The set of edges making up loop l_j, $j = 1, 2, ..., M$ is denoted $E(l_j)$. Since every loop in the graph can be expressed as a linear combination of basis loops, any loop basis is adequate to represent a circuit's timing constraints, and thus any spanning tree suffices. Note the direct correspondence between equations for loop delays and Kirchoff Voltage Law equations for an electrical network [56].

3) For each basis loop, write an equation stating that the sum of edge delays around the loop is equal to an integral multiple of the packet cycle T_P. For non-directed loops, an arbitrary direction of loop circulation is assigned. If the non-directed loop contains an edge e_i, the function $C(e_i)$ returns 1 if e_i is directed with the assigned loop circulation, and -1 otherwise. For example, given loop l_j with edges $E(l_j) = \{e_A, e_B, e_C, e_D\}$, we have the following equation if l_j is directed (see Figure 5.3a):

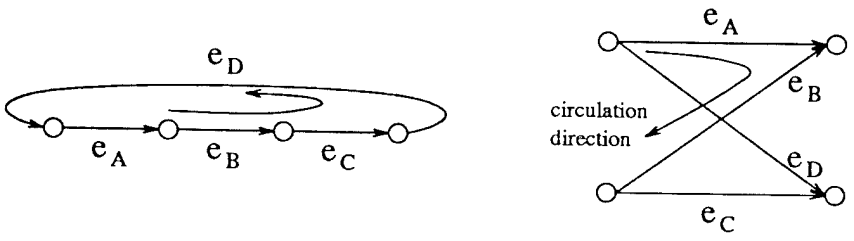


Figure 5.3: Example of basis loops: directed (a) and non-directed (b).

$$\sum_{e_i \in E(l_j)} d(e_i) = n_j T_P, \quad n_j \in \{1, 2, 3, ...\}. \tag{5.1}$$

n_j is positive because of finite processing and link delays around any feedback loop. If l_j is non-directed (Figure 5.3b), the product of $C(e_i)$ and $d(e_i)$ appears in the loop sum:

$$\sum_{e_i \in E(l_j)} C(e_i)d(e_i) = n_j T_P, \quad n_j \in \{..., -2, -1, 0, 1, 2, ...\}. \tag{5.2}$$

4) Each link has a minimum delay. Packaging dimensions can be translated directly to minimum link delays once the network has been partitioned. The minimum link delay for edge e_i is denoted $m(e_i)$.

5) The objective function may depend on the desired network performance. We consider the simplest form here, which is to minimize

the sum of link delays.

The integer program takes on the form:

minimize $\sum_{e_i \in E} D_L(e_i)$,

subject to:

$\forall l_j \in L$:

$\sum_{e_i \in E(l_j)} d(e_i) = n_j T_p, \ \ n_j \in \{1, 2, 3, ...\}$,

if l_j is directed loop, **or** (1.a)

$\sum_{e_i \in E(l_j)} C(e_i) d(e_i) = n_j T_p, \ \ n_j \in \{..., -2, -1, 0, 1, 2, ...\}$,

if l_j is non-directed loop, (1.b)

$d(e_i) = D_R + D_L(e_i), \ \ \forall e_i \in E,$ (2)

$D_L(e_i) \geq m(e_i), \ \ \forall e_i \in E.$ (3)

5.4 Example: 8-node ShuffleNet

The integer program for the 8-node ShuffleNet example is as follows:

minimize $\sum_{e \in E} D_L(e)$,

$$E = \{e_a, e_b, e_c, e_d, e_e, e_f, e_g, e_h, e_i, e_j, e_k, e_l, e_m, e_n, e_o, e_p\}$$

subject to:

$$d(e_a) + d(e_i) = n_1 T_p, \;\; d(e_c) + d(e_k) = n_2 T_p,$$

$$d(e_b) - d(e_f) - d(e_p) - d(e_d) + d(e_c) + d(e_l) = n_3 T_p,$$

$$d(e_b) + d(e_n) + d(e_h) - d(e_d) + d(e_c) + d(e_l) = n_4 T_p,$$

$$d(e_f) + d(e_m) = n_5 T_p, \;\; d(e_h) + d(e_o) = n_6 T_p,$$

$$d(e_a) + d(e_j) + d(e_c) + d(e_l) = n_7 T_p,$$

$$d(e_a) - d(e_e) + d(e_f) - d(e_b) = n_8 T_p,$$

$$d(e_c) - d(e_g) + d(e_h) - d(e_d) = n_9 T_p,$$

$$n_1, n_2, n_5, n_6, n_7 \in \{1, 2, 3, ...\}, \;\; n_3, n_4, n_8, n_9 \in \{..., -2, -1, 0, 1, 2, ...\},$$

$$d(e_i) = D_R + D_L(e_i), \;\; \forall e_i \in E, \;\; D_L(e_i) \geq m(e_i), \;\; \forall e_i \in E.$$

While the network mechanical packaging details of the prototype in Chapter 4 are not available at this time, a realistic set of constraints can still be developed. In a faster state-of-the-art CMOS VLSI implementation of the RCP there would be three or four pipeline stages, depending on the processor implementation, with a word clock rate on the order of 200 MHz (T_W = 5 nsec) [14]. This gives D_R = 15 or 20 nsec, equivalent to approximately 3 or 4 meters of single-mode optical fiber delay. For systems spread across cabinets, inter-cabinet links would have lengths on this order. Therefore it is reasonable to assign minimum link delays that are of the same order of

Edge e	e_a	e_b	e_c	e_d	e_e	e_f	e_g	e_h	e_i	e_j	e_k	e_l	e_m	e_n	e_o	e_p
$m(e)$	17	4	3	2	3	4	8	5	2	2	7	3	1	3	14	6
$D_L(e)$ $(D_R = 4)$	20	4	5	3	3	7	10	8	2	2	7	7	5	3	14	6
$D_L(e)$ $(D_R = 12)$	18	8	9	9	5	5	8	8	8	2	7	3	1	3	18	6

Table 5.1: Optimal solutions for edge delays, for two different RCP delays.

magnitude as D_R. Minimum delays $m(e_i)$ are given in the second row of Table 5.1 for each edge e_i, assuming no particular layout geometry. All delays are expressed as multiples of T_W. A typical packet size is around 160 bits (16 Byte payload plus header), giving $T_P = 10T_W$. The third row of the table gives the solution for $D_R = 4T_W$. The fourth row has the solution assuming $D_R = 12T_W$, reflecting a hypothetical RCP with slower, more deeply pipelined logic running at a 200 MHz word rate.

The first solution (in the third row) was found in 1.4 CPU seconds on a DECstation 5000; total link delay was $106T_W$. Solving the second problem took 2.8 CPU seconds. Total link delay increased slightly to $118T_W$ due to the higher value of D_R. The increase in D_R caused more link delay "padding" to be needed for synchronization.

The solver we used is part of the Cplex linear and mixed integer optimization package [57], and uses a branch-and-bound algorithm. One limitation of the package is that it restricts integers to be non-negative. The values of n_j may be negative for non-directed loops, however. In the example shown

above, all feasible solutions have positive n_j's. This property is not true in general. We have developed a simple heuristic approach to solving the problem. The heuristic determines the direction of loop circulation that will yield positive n_j's for each non-directed loop. The edge set for each such loop is partitioned into two disjoint subsets, where all edges of a subset are directed in the same way with respect to a circulation direction. We then choose the circulation direction such that the direction coincides with the orientation of edges in the subset having *greater total minimum edge delay.* We have verified empirically that this heuristic yields IP's with feasible solutions for ShuffleNet graphs of varying size up to 160 nodes ($k = 5$). A rigorous mathematical analysis showing when the heuristic fails and how its use affects solution quality is needed. For networks with thousands of nodes, our experiments show that branch-and-bound has prohibitive complexity. Alternative methods such as simulated annealing [58] or genetic algorithms [59] should be applied in these situations.

5.5 Two Variations of the Problem

The first variation of the integer program presented in Section 5.3 concerns multiple packet sizes. Shared-memory multiprocessor network traffic can be divided roughly into two classes. Assume that the unit of data transfer

between processors is a cache line, which might be 16 Bytes or larger; the size is not important for this discussion. Every network transaction consists of a *request* packet transmitted by the sender node to the receiver node, and a *response* packet transmitted sometime later from the receiver to the sender [13]. If the request was for a remote memory store, then the request packet contains the data to store at the receiving node's memory. If the request was for a remote memory load, then the response packet contains the data loaded from the receiving node's memory. Transfers that do not carry data can use a smaller packet size than transfers that carry data. This observation suggests two packet sizes, with respective packet cycles denoted T_{PD} (packet carrying data) and T_{PN} (packet carrying no data). Having two sizes yields more efficient use of network bandwidth than having one size with packet cycle T_{PD}. In the latter case, bits would be wasted for read request or write response packets. (A third class of packets used to enforce a directory-based cache coherency protocol [2] could also be defined.)

We must ensure that the slots for the different packet sizes always arrive in the same sequence on the input ports of every node. If this condition is not met, then eventually a routing processor will fail because headers from different input ports will no longer be aligned. We define the *frame delay* T_F as the sum of the packet cycles for the two packet sizes: $T_F = T_{PD}+T_{PN}$. The

new constrained minimization problem is essentially unchanged, except that it constrains incoming *frames* to be aligned across all input ports. Differences in path delays to a common destination must be integral multiples of T_F rather than the packet cycle T_P as in Section 5.3. In the integer program of Section 5.3, T_P is simply replaced with T_F on the right-hand side of each loop constraint.

The second variation of the problem is to modify the objective function to favor latency-critical network paths receiving the most traffic. Traffic behavior is beyond the scope of this paper; it depends on the programming model, the application programs, and the data partitioning algorithms used by the compiler. We can show using the 8-node ShuffleNet example how prior knowledge of traffic patterns can be fed into the optimization problem using a simple weighting function. Assume that data is laid out in memory such that data transfers between nodes in a ring are more common than transfers between nodes in different rings. Assume also that for each ring the delays for packets flowing from column 0 to 1 are more critical to performance than the delays for packets flowing from column 1 to column 0. To optimize the common case, it is most important to minimize delays for the links represented by intra-ring edges e_a, e_c, e_f, and e_h, at the expense of increasing delays over "non-critical" links. This goal translates into the fol-

lowing modified objective function, which weights these critical edges more heavily:

$$\text{minimize} \quad \sum_{e \in E_1} D_L(e) + \sum_{e \in E_2} 2D_L(e),$$
$$E_1 = \{e_b, e_d, e_e, e_g, e_i, e_j, e_k, e_l, e_m, e_n, e_o, e_p\}, E_2 = \{e_a, e_c, e_f, e_h\}. \quad (5.3)$$

The resulting optimal solutions differ from those shown in Table 5.1. Though the sum of all link delays is unchanged, the link delays are better tailored to the traffic profile. For $D_R = 4$, the delays assigned to e_a, e_c, and e_h decrease by 1, 2, and 1, respectively. The delay subtracted from them is "pushed" onto the adjacent links in their respective rings. The delay for e_f actually increases by 1, making the total savings in delay for the critical edges equal to 3.

Chapter 6

Conclusions and Future Work

The initial aspirations for fully optical computing systems have met economic and physical barriers that are unlikely to be completely overcome, and electronic advances continue unabated. Nonetheless, it is commonly believed that the next inroad of photonics with great technological impact, after long-haul telecommunications, will be in intelligent interconnects for distributed processing systems. Conventional electronics is losing the battle in the interconnect arena for closely-coupled, distributed processing systems, as advancements in processor speed have significantly outpaced those in interconnects. Although architectures and algorithms have been adapted to compensate for the disparity in performance between processors and interconnects, such techniques have been stretched to the limit, and are unlikely

to satisfy the requirements of the next generation, teraflop systems.

Current heterogenous computer complexes, with a mixed cluster of supercomputers at the top and a large and varied array of workstations at the bottom, have also outgrown the capabilities of all-electronic systems. The optoelectronic systems being pressed into service poorly satisfy some of the basic needs of computer users. A generic interconnect, well-adapted to the requirements of distributed processing systems, is a growing need. Primarily electronic efforts are attempting to meet the challenge, but must ultimately overcome severe fundamental problems associated with the complexity at switching points and the need for the interconnect point-to-point transmission speed to exceed the expected user word clock rate in bits/sec. Because of rapid advances in the development of compact multi-wavelength devices and wideband optical switches, integrating photonics within computer interconnects is necessary to solve the problems of communications in distributed processing environments.

6.1 Research Summary

This report has discussed an original prototype design of a packet-switched optical interconnect for multiprocessors. The network has a ShuffleNet topology, self-routing packets, simple node design without static buffers, and de-

flection routing for contention resolution. The network is scalable — total user throughput increases with the number of nodes and with the link bit-rate.

The multiprocessor architecture which was implemented uses a directory-based scheme in conjunction with a coherent I/O cache at each node to provide memory coherency and a release consistency model. Packet duplication is avoided through the use of a simple table which tracks acknowledgements of a node's outstanding requests. Recovery from bit-errors in the payload can be handled by retransmission. Simple feed-forward processing of packet headers is done in electronics using FPGAs to provide flexibility for experimenting with routing strategies. Different packet sizes, header formats, or routing algorithms can be easily implemented simply by downloading a different FPGA configuration.

Furthermore, the prototype supports the incremental inclusion of photonics as a means for transmission and switching. The use of BPW encoding, with the vast optical bandwidth and data transparency of photonic switches and optical fiber, is a clear favorite for achieving high bandwidth, scalable, data communications for computer interconnects. However, several problems must be overcome before photonics can be used extensively in commercial multiprocessing applications. First, the high cost of optical components,

expecially packaged lasers for single mode fiber is prohibitive at present. Another problem is the fact that some components, such as wavelength multiplexers and demultiplexers are not yet available in monolithic integrated form. A system built today must include some bulk optics, lowering its practicality. Polarization sensitivity of switches creates another difficulty; polarization controllers greatly increase the complexity and bulkiness of a routing node and the use of polarization maintaining fiber is expensive. Further, power loss through packaged photonic switches is potentially high for deflection networks, where hop count is not deterministic. Integrated low-loss optical switches or fiber amplifiers are needed. Finally, data sheets for off-the-shelf lasers and photodetectors typically do not specify input-to-output latency and latency variations, parameters which are essential to designing high bit-rate synchronous computer networks. As the development of photonic devices advances, most of these problems will be overcome.

A final contribution of the research effort was an integer program formulation of the problem of packet-level synchronization in synchronous optical deflection networks that have non-regular node spacings imposed by packaging of boards, racks, and cabinets. The method is useful for other network topologies and may have applications beyond deflection routing. The underlying principle is "time-of-flight" synchronization, which re-distributes delay

over network paths statically at network design time, so that incoming packets are aligned at every node during operation. Time-of-flight design exploits the highly predictable propagation delay inherent in optical devices. Continuing advances in enabling technologies such as lasers, waveguides, and optical switches should soon make this approach feasible in networks with thousands of nodes.

6.2 Future Work

The phase of research just completed used bit-parallel electrical links and exercised all of the switching protocols and showed end-to-end communication. In the next phase, electrical links must be replaced with bit-serial optical fiber links, running at the same aggregate bit-rate. Finally, we plan to rebuild the **OPTOB** to include $LiNbO_3$ switches to perform optical switching with up to four wavelengths, and either spatial or 2S2T switching. An ultimate goal would be to have an 8-node network with nodes having all-optical data paths and BPW encoding.

To further demonstrate the viability of our approach, simulations of our multiprocessor model using memory traces from the Splash [2] benchmarks are necessary. We also need to address the issues of fault-tolerance and alternatives to global clocking that rely on packet synchronization at the input

ports. Due to the NP-completeness of solving integer programs, future work in packet synchronization needs to examine heurisitic methods for use on larger networks. We have argued that optical path delays are nearly perfectly controllable, ignoring sources of delay uncertainty such as static skew in receivers, noise, jitter, and temperature fluctuations. While the nagnitude of this delay uncertainty is small, it may degrade the bit-rate as the global clock increases beyond the GHz range. Finally, a comparison of synchronous and asynchronous timing disciplines in deflection networks is needed. As network size increases, clock skew and packet arrival skew increase, and asynchronous timing might eventually become more desirable, despite its disadvantages. It would be insightful to determine the practical "break-even" point in terms of node separation, at which the price/performance ratios of synchronous and asynchronous networks become equal.

Simply stated, some fundamental questions that remain to be answered include:

1) How well can a deflection-routed network satisfy memory consistency needs of a shared-memory multiprocessor having latency-hiding support?

2) How can optics be used to increase fault-tolerance?

3) For larger networks, how can we optically synchronize incoming

packets?

4) To what degree is the use of photonics cost-effective in comparison to the performance gains which may be achieved?

Practical problems may still prevent construction of cost-effective and reliable photonic interconnects for multiprocessing environments. However, this research effort is an important step towards the future, when multiprocessor system designers will be able to reap the benefits of optical interconnection networks to reduce the remote memory latency bottleneck currently limiting performance in distributed processing systems.

Bibliography

[1] L. Geppert, "The new contenders," *IEEE Spectrum*, pp. 20–25, December 1993.

[2] D. Lenoski, J. Laudon, K. Gharachorloo, W.-D. Weber, A. Gupta, J. Hennessy, M. Horowitz, and M. Lam, "The Stanford Dash Multiprocessor," *IEEE Computer*, vol. 25, pp. 63–79, March 1992.

[3] M. Dubois *et al.*, "The detection and elimination of useless misses in multiprocessors," The 20th International Symposium on Computer Architecture, May 1993, pp. 88–97.

[4] E. P. Markatos and T. J. LeBlanc, "Shared-memory multiprocessor trends and implications for parallel program performance," Technical Report No. 420, Computer Science Department, University of Rochester, Rochester, NY, May 1992.

[5] T. C. Mowry *et al.*, "Design and evaluation of a compiler algorithm for prefetching," ASPLOS-V, 1992, pp. 62–73.

[6] J. W. Goodman *et al.*, "Optical interconnections for VLSI systems," *Proceedings of the IEEE*, vol. 72, pp. 850–866, July 1984.

[7] J. Midwinter, "Photonics in switching: The next 25 years of optical communications?," *IEE Proceedings*, vol. 139, pp. 1–12, February 1992.

[8] L. Thylen, "Photonics in switching: European systems demonstrators and the long-term perspective," Optical Computing, Optical Society of America, March 1993, pp. 10–14.

[9] R. Schmidt and R. Alferness, "Directional coupler switches, modulators, & filters using alternating Delta- β techniques," *IEEE Trans.*, vol. CAS-26, pp. 1099–1108, 1979.

[10] R. Alverson *et al.*, "The Tera Computer System," in *International Conference on Supercomputing*, June 1990, pp. 1–6.

[11] E. H. et al, "Ddm - a cache-only memory architecture," *IEEE Computer*, vol. 25, pp. 44–54, September 1992.

[12] M. G. Hluchyj and M. Karol, "ShuffleNet: An application of generalized perfect shuffles to multihop lightwave networks," IEEE INFOCOM, March 1988, pp. 379–390.

[13] J. Feehrer, L. Ramfelt, and J. Sauer, "An optical deflection-routed multiprocessor interconnect," 6th European Research Consortium for Informatics and Mathematics Workshop, June 1-3 1994, pp. 103–115.

[14] J. Feehrer, L. Ramfelt, and J. Sauer, "Design and implementation of a prototype optical deflection network," International Conference on Commmunications Architectures, Protocols, and Applications, August 31 to September 2 1994.

[15] J. Feehrer, J. Sauer, and L. Ramfelt, "Design and implementation of a prototype optical deflection network," *Computer Communication Review*, vol. 24, p. 191, October 1994.

[16] J. R. Feehrer, L. H. Ramfelt, and D. Straub, "Implementation details for optical deflection-routed multiprocessor interconnect prototype," Packet Network Laboratory Technical Report, University of Colorado Optoelectronic Computing Systems Center, March 1995.

[17] J. R. Feehrer and L. H. Ramfelt, "Packet synchronization for synchronous optical deflection-routed interconnection networks," to be published in *IEEE Transactions on Parallel and Distributed Systems.*

[18] P. Baran, "On distributed communication networks," *IEEE Transactions on Communication Systems*, vol. 12, pp. 1–9, 1964.

[19] F. Borgonovo, L. Fratta, and F. Tonelli, "Circuit service in deflection networks," Vol. 1, Proceedings of IEEE INFOCOM, 1991, pp. 69–75.

[20] A. Krishna and B. Hajek, "Performance of shuffle-like switching networks with deflection," IEEE INFOCOM, 1990, pp. 473–480.

[21] A. Acampora and S. Shah, "Multihop lightwave networks: A comparison of store-and-forward and hot-potato routing," *IEEE Transactions on Communications*, vol. 40, no. 6, p. 1082, 1992.

[22] N. Maxemchuk, "Regular mesh topologies in local and metropolitan area networks," *AT & T Technical Journal*, vol. 65, pp. 1659–1685, September 1985.

[23] A. V. Ramanan, *Ultrafast Space-Time Networks for Multiprocessors.* PhD thesis, University of Colorado at Boulder, Department of Electrical and Computer Engineering, 1993.

[24] A. Ramanan, H. Jordan, J. Sauer, and D. Blumenthal, "An extended fiber-optic backplane for multiprocessors," Proceedings of Hawaii International Conference on System Sciences, 1994.

[25] D. Blumenthal, R. Feuerstein, and J. Sauer, "First demonstration of multihop all-optical packet switching," *IEEE Photonics Technology Letters*, March 1994.

[26] L. Ramfelt, "A host-network interface based on coherent caches," 4th MultiG Workshop in Stockholm, May 1992.

[27] Xilinx, *The Programmable Logic Data Book.* 1993.

[28] C. E. Love, "Time-of-flight packet synchronizers," (Palm Spring, CA), Optical Computing Topical Meeting, OSA, 1993, pp. 326–329.

[29] E. Desurvire, *Erbium-doped Fiber Amplifiers: Principles and Applications.* John Wiley and Sons, 1994.

[30] P. J. Delfyett *et al.*, "Optical clock distribution using a mode-locked semiconductor laser diode system," *IEEE Journal of Lightwave Technology*, vol. 9, pp. 1646–1649, December 1991.

[31] D. C. Wong *et al.*, "A bipolar population counter using wave pipelining to achieve 2.5x normal clock frequency," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 745–753, May 1992.

[32] P. Duthie and C. Edge, "A polarization independent guided-wave LiNbO3 electroopic switch employing polarization diversity," *IEEE Photonics Technology Letters*, vol. 3, pp. 136–137, February 1991.

[33] J. Soole, "Monolithic InP/InGaAsP/InP grating spectrometer for the 1.48-1.56 μm wavelength range," *Applied Physics Letters*, vol. 58, pp. 1949–1951, May 6 1991.

[34] C. Dragone, "An $N \times N$ optical multiplexer using a planar arrangement of two star couplers," *IEEE Photonics Technology Letters*, vol. 3, pp. 812–815, September 1991.

[35] M. Gustavson *et al.*, "Monolithically integrated 4x4 InGaAsP/InP laser amplifier gate switch arrays," *Electronics Letters*, vol. 28, pp. 2223–2225, November 1992.

[36] W. Oed, "CRAY T3D: Massively parallel processor system," Tech. report, Cray Research GmbH, November 1993. ftp.cray.com /product-info/mpp/T3D_overview.ps.

[37] R. Nordin, *Photonics in Switching*, ch. 9. Academic Press, Inc., 1993.

[38] D. K. Lewis *et al.*, "The Optoelectronic Technology Consortium (OETC)," *IEEE LEOS Newsletter*, pp. 12–13, October 1992.

[39] W. D. Hillis and L. W. Taylor, "The CM-5 Connection Machine: A scalable supercomputer," *Communications of the ACM*, vol. 36, pp. 31–40, November 1993.

[40] G. Zorpette, "The power of parallelism," *IEEE Spectrum*, pp. 28–33, September 1992.

[41] D. Gelernter, A. Nicolau, and D. Padua, *Languages and Compilers for Parallel Computing*, ch. A Future-based Parallel Language for a General-Purpose Highly-Parallel Computer. 1990.

[42] P. R. Prucnal, *Photonics In Switching, Volume II*, ch. Photonic Fast Packet Switching. Academic Press, Inc., 1993.

[43] R. Khalil, "Clock skew analysis for Si and GaAs receivers in optical clock distribution systems," Vol. 1178, Optical Interconnect in the Computer Environment, SPIE, 1989, pp. 171–176.

[44] D. Hartman, "Digital high speed interconnects: A study of the optical alternative," *Optical Engineering*, vol. 25, no. 10, pp. 1086–1102, 1986.

[45] I. Malitson, "Interspecimen comparison of the refractive index of fused silica," *Journal of the Optical Society of America*, vol. 55, pp. 1205–1209, October 1965.

[46] A. L. Fisher and H. Kung, "Synchronizing large VLSI processor arrays," *IEEE Transactions on Computers*, vol. C-34, pp. 734–740, August 1985.

[47] A. Greenberg and B. Hajek, "Deflection routing in hypercube networks," *IEEE Transactions on Communications*, vol. 40, pp. 1070–1081, 1990.

[48] J. Bannister, F. Borgonovo, L. Fratta, and M. Gerla, "A versatile model for predicting the performance of deflection-routing networks," *Performance Evaluation*, vol. 16, pp. 201–221, 1992.

[49] F. Pittelli and D. Smitley, "Analysis of a 3D toroidal network for a shared memory architecture," Supercomputing 88, pp. 42–47.

[50] K. E. Batcher, "Decomposition of perfect shuffle networks," International Conference on Parallel Processing, 1991, pp. I–255–I–262.

[51] H. F. Jordan, V. P. Heuring, and R. Feuerstein, "Optoelectronic time-of-flight design and the demonstration of an all-optical, stored program, digital computer," *Proceedings of the IEEE Special Issue on Optical Computing*, vol. 82, p. 1678, November 1994.

[52] P. Main, R. Feuerstein, V. Heuring, H. Jordan, J. Feehrer, and C. Love, "Implementation of a general purpose stored-program digital optical computer," *Applied Optics*, vol. 33, p. 1619, March 10 1994.

[53] W. Dally and C. Seitz, "Deadlock free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C-36, May 1987.

[54] C. E. Leiserson and J. B. Saxe, "Optimizing synchronous systems," *Journal of VLSI and Computer Systems*, vol. 1, no. 1, pp. 41–67, 1983.

[55] A. Gibbons, *Algorithmic Graph Theory.* Cambridge University Press, 1985.

[56] N. Balabanian and T. Bickert, *Electrical Network Theory.* John Wiley and Sons, Inc, 1969.

[57] *Cplex User's Manual.* Cplex Optimization, Inc., Houston, TX., 1990.

[58] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing; part I, graph partitioning," *Operations Research*, vol. 37, pp. 865–892, 1989.

[59] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Co., 1989.