Control and Reliability of Optical Networks in Multiprocessors

by

James Jonathan Olsen

S.B., Massachusetts Institute of Technology (1981)
S.M., Massachusetts Institute of Technology (1985)

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

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1993

Author .....
Department of Electrical Engineering and Computer Science
April 30, 1993

Certified by .....
Anant Agarwal
Associate Professor of Computer Science
Thesis Supervisor

Accepted by .....
Campbell L. Searle
Chairman, Departmental Committee on Graduate Students

Control and Reliability of Optical Networks in Multiprocessors

by

James Jonathan Olsen

Submitted to the Department of Electrical Engineering and Computer Science
on April 30, 1993, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Optical communication links have great potential to improve the performance of interconnection networks within large parallel multiprocessors, but the problems of semiconductor laser drive control and reliability inhibit their wide use. These problems have been solved in the telecommunications context, but the telecommunications solutions, based on a small number of links, are often too bulky, complex, power-hungry, and expensive to be feasible for use in a multiprocessor network with thousands of optical links.

The main problems with the telecommunications approaches are that they are, by definition, designed for long-distance communication and therefore deal with communications links in isolation, instead of in an overall systems context. By taking a system-level approach to solving the laser reliability problem in a multiprocessor, and by exploiting the short-distance nature of the links, one can achieve small, simple, low-power, and inexpensive solutions, practical for implementation in the thousands of optical links that might be used in a multiprocessor.

Through modeling and experimentation, I demonstrate that such system-level solutions exist, and are feasible for use in a multiprocessor network. I divide semiconductor laser reliability problems into two classes: transient errors and hard failures, and develop solutions to each type of problem in the context of a large multiprocessor.

I find that for transient errors, the computer system would require a very low bit-error-rate (BER), such as 10^{-23}, if no provision were made for error control. Optical links cannot achieve such rates directly, but I find that a much more reasonable link-level BER (such as 10^{-7}) would be acceptable with simple error detection coding. I then propose a feedback system that will enable lasers to achieve these error levels even when laser threshold current varies. Instead of telecommunications techniques, which require laser output power monitors, I describe a software-based feedback system using BER levels for laser drive control. I demonstrate, via a BER-based laser drive control experiment, that this method is feasible. It maintains a BER of 10^{-9}: much better than my error control coding system would need. The feedback system can also compensate for optical medium degradation, and can help control hard failures by tracking laser wearout trends.

For hard failures, a common telecommunications solution is to provide redundant spare optical links to replace failed ones. Unfortunately, this involves the inclusion of many extra, otherwise unneeded optical links, most of which will remain unused throughout the system lifetime. I present a new approach, which I call 'bandwidth fallback', which allows continued use of partially-failed channels while still accepting full-width data inputs. This provides, at a very small performance penalty, a high reliability level while needing no spare links at all.

I conclude that the drive control and reliability problems of semiconductor lasers do not bar their use in large scale multiprocessors, since inexpensive system-level solutions to them are possible.

Thesis Supervisor: Anant Agarwal
Title: Associate Professor of Computer Science

Acknowledgements

I would also like to thank Anant, my thesis supervisor, for his assistance and guidance in shaping my research program. I am grateful to my readers, Bob Kennedy and Tom Goblick, for their valuable advice. My thanks also go to the many other people at Lincoln Lab and LCS who provided me advice and assistance. I would like to thank my wife Batya for her support and encouragement in this work.

This work was sponsored by the Department of the Air Force under contract F19628-90-C-0002.

To Anna and Eva

Contents

List of Figures

List of Tables

Chapter 1

Overview

For many years, the promise of massively parallel computing (combining large numbers of inexpensive processors into powerful systems) has beckoned computer architects. This promise is now being fulfilled, and it is apparent that the interconnection network between processors is one of the primary factors determining the overall performance of a multiprocessor system. The important parameters governing interconnection network cost and performance include connection density, bandwidth, and reliability.

Recent advances in optical technology suggest that optical interconnect networks will soon be viable for use in multiprocessors. Optical networks offer the potential of vastly increased bandwidth and connection density, compared with electrical networks.

Currently, the most widely used paradigm for optical data communication relies on semiconductor lasers as optical sources. Such lasers present a number of reliability problems. In the telecommunications context, where such optical communications find increasingly wide use, these problems have already been solved.

However, telecom solutions, based on a relatively small number of long-distance links, are often ill-suited to a multiprocessor network containing thousands of short-distance links. In this thesis, I investigate reliability solutions which are more appropriate to the multiprocessor context.

1.1 The Reliability Problems

The use of optical communication networks in multiprocessors poses two main reliability questions.

1. Multiprocessor systems are exceedingly intolerant of data transmission error. How can the optical links achieve a sufficiently low data error rate?

2. Semiconductor laser failure data, when extrapolated to systems with large numbers of lasers, suggest that overall reliability of such a system might be unacceptably low; does this mean that semiconductor lasers are impractical for use in large-scale multiprocessors?

Telecommunications links intended for telephone use are often designed to achieve Bit-Error-Rates (BER) on the order of 10^{-9} or 10^{-12}, with the best systems achieving values around 10^{-15} [3]. However, data communications within multiprocessors will require much better BER levels to assure reliable operation, such as 10^{-23} for example. (A 10^{-23} rate would be needed in a 4096-channel system with 64-bit 1-Gword/sec channels, if one allowed 0.1 error/year.) Optical device research continues to seek lower BER levels, but the extremely low levels required on multiprocessor applications will almost certainly remain elusive on raw channels without error coding.

Another aspect of semiconductor laser operation is relevant to both transient-error and hard-failure performance: laser threshold current variation. As a laser ages, it wears out and eventually fails. This wearout manifests itself as an increase in laser threshold current[4], and the laser drive must provide sufficient current to compensate for the increased threshold. Threshold current also increases with laser temperature. Increased threshold, with fixed drive, will result in lower laser output and higher error rates. Also, high-speed lasers must be driven with a bias current approximately equal to the threshold, and an old (or hot) laser therefore 'fails' if the drive circuit does not supply sufficient bias current to compensate for the increased threshold.

Semiconductor laser reliability is constantly improving, but it is still potentially a limiting item for system reliability. For example, a high-reliability laser might have an MTBF (Mean or Median Time Between Failure) specification of 100,000 hours. A 1024-node multiprocessor might employ 300,000 lasers; linear extrapolation of the

failure rate would lead one to conclude that such a system would have an MTBF around 20 minutes: a completely unacceptable level of reliability. (Some researchers have gone so far as to conclude that this renders semiconductor lasers unsuitable for use in large-scale multiprocessor networks, and have instead pursued other optical sources, such as light modulators pumped by a central, high-power gas laser [5].)

Laser failures can be divided into two broad categories: wearout failures and random failures. While random failure times follow an exponential distribution, that is, they can be modeled as arrivals of a Poisson process, wearout failure times follow a log-normal distribution [4]. Therefore the simple linear extrapolation used above:

$$\text{System MTBF} = \frac{\text{Component MTBF}}{\text{Number of Components}} \tag{1.1}$$

is valid only with respect to random failures, but not with respect to wearout failures, since a set of n identical Poisson processes of rate λ is equivalent to one Poisson process of rate $n\lambda$, but such a relation does not apply to a process with log-normally distributed arrival times.

Nevertheless, the true system MTBF will still likely be inadequate without special provision for recovery from laser failure. For example, even if we have a random-failure laser MTBF of 10,000,000 hours, the system MTBF due to random failure in a 300,000 laser system will be about 30 hours, which is still unsatisfactory. As the system ages, wearout failures will also come into play, further increasing the failure rate.

1.2 Reliability Solutions

My solution to the laser reliability problem has three major parts:

- Control transient errors by Error Detection Coding, and retransmission on error,
- Control laser drive current with an intelligent software-based feedback loop based on the observed Bit Error Rate (BER), and
- Control of hard failures with 'Bandwidth Fallback', which provides the reliability benefits of large-scale redundant sparing, without the expense of providing spares.

1.2.1 Error Detection Coding

It is apparent that some form of error control coding scheme will be required to solve the bit-error-rate problem. Error control coding strategies can be placed in two broad categories: Forward Error Correction (also called error-correction coding), in which enough redundant data are included in each transmission to re-derive the original data in spite of one or more transmission errors, and Automatic Retransmission Request, in which only enough redundant information is sent to enable the detection of errors, and the sender is asked to retransmit any data received in error.

Data storage systems, such as magnetic disks, are generally constrained to use Forward Error Correction, since it is impossible to ask for a retransmission of data if it has been corrupted on the storage medium. Similarly, high-speed telecommunication systems will use Forward Error Correction because retransmission, while theoretically possible, would be impractical due to very long transmission delays (relative to bit transmission times).

By contrast, the sender and receiver in a multiprocessor system are relatively close, and Automatic Retransmission Request is quite feasible. [An exception to this would be multiprocessors which are constrained to operate in pre-set instruction sequence, such as Single-Instruction, Multiple-Data (SIMD) systems, since the times when retransmission is required can, of course, not be determined ahead of time.] A retransmission strategy, combined with a very simple error-detection coding scheme, allows the use of data links with dramatically relaxed error rate requirements.

1.2.2 Intelligent Laser Drive Control

High-speed operation of most semiconductor lasers usually requires the adjustment of the laser drive current level as the laser threshold current varies with age and temperature. Design for long-term reliable operation of an optical network requires that the problem of laser drive level control be addressed.

The standard solution to this problem in a telecommunications context is to implement a laser light output power monitor, and to use this to form a feedback loop controlling the laser drive level (I illustrate this in Figure 7-4). This is an excellent approach for telecom applications, where the size, optical complexity, and expense of

such a feedback loop is easily absorbed in an already large and expensive support system for each laser.

The multiprocessor context is different. When one considers the use of dozens of lasers on each channel, and many thousands of lasers in each system, the complexity and size of each laser link becomes much more important. Implementation of a laser light monitor system in a multiprocessor network would entail an oppressive hardware overhead, and would be much more difficult to justify than in the telecommunications context.

Fortunately, we can exploit the coding approach described in the previous section, and the error-rate flexibility it allows, to suggest an alternative approach: intelligent (i.e., software-based) laser drive feedback control using the data-link error rate as the feedback control variable. The error rate is, in fact, the data-link parameter that we ultimately wish to control. Light-monitor-based approaches control the laser light output level for feedback only as an easily-accessible and controllable (at link level) surrogate for the error rate.

Given the Automatic Retransmission Request error-control system outlined above, one may implement an intelligent laser drive control system with the addition of one or two very simple Digital-to-Analog converters per laser (or set of lasers), easily implemented *en masse* in an integrated circuit. The Digital-to-Analog converters are controlled by the sending processor, which adjusts the drive level based on how frequently its neighbors request retransmission of data.

1.2.3 Bandwidth Fallback

From even the cursory analysis I have given above in Section 1.1, it is clear that some provision must be made for continued system operation after laser failure occurs. Even if the intelligent drive control system could completely eliminate the wearout problem (not a very likely prospect), we would still have to handle random failures. In telecommunications, a common approach to failure problems is to provide redundant spares, to be used in place of failed links. I examine how the provision of redundant spare optical links could be implemented in a multiprocessor network, and show that this would be an adequate approach to the hard-failure problem if one could afford the cost of providing enough spare links.

I then examine two approaches to eliminate this cost: use of time-shared alternative paths to replace failed channels, and providing for variable-width data transmission to exploit whatever remaining data width is available (which I call 'bandwidth fallback'). Bandwidth fallback is an idea for efficient use of the remaining bandwidth in a partially-failed channel. With the addition of one multiplexer and one register per bit, the channel can be made usable at certain fractions of its full bandwidth, of the form 1/2, 3/4, 7/8, 15/16, etc. It can provide almost the same performance as large-scale redundant sparing, without needing any spares. (Bandwidth Fallback is also applicable to electrical networks, given off-board connections are generally much less reliable than on-board or on-chip connections.)

1.3 Summary of Results

As long as one avoids the type of inflexible system design that precludes the use of Automatic Retransmission Request error control[1], an optical multiprocessor network need only implement coding for error detection, as opposed to both detection and correction. With this approach, we demonstrate that a very simple linear error-detection code can to achieve the needed level of error control with minimal overhead: relaxing the optical-link BER requirement from 10^{-23} to 10^{-7}.

Given such a coding scheme, laser drive control can be accomplished without the need for light output power monitoring and the associated analog feedback loops, by using the link's bit-error-rate to control the laser drive level. I have designed and implemented a software-based laser drive control system for an experimental free-space optical data link. I have conducted experiments on the drive control system over varying temperatures, showing excellent control of error-rate. (Incidentally, it controls the laser output level quite well without the use of a light level monitor!) I show that a 5-bit Digital-to-Analog converter would be adequate for level control in this experimental setup, and that the experimental feedback system is stable, with a gain margin of at least 10 dB. Additional experiments demonstrate recovery from optical medium degradation: a capability not possessed by traditional monitor-based control schemes. I also explain how an intelligent laser drive control system can help

[1] such as Single-Instruction Multiple-Data (SIMD) design

deal with laser wearout failures by facilitating long-term monitoring of laser wearout trends.

I find that providing redundant spare lasers (if one could afford enough of them) would result in acceptable system reliability. I show that, with only a small performance penalty, the implementation of alternative routing and bandwidth fallback can provide the same reliability benefits, without the spares. Alternatively, they can provide the same performance as redundant sparing, with a great reduction in the number of spares needed.

In the final section, I offer suggestions for further research on this topic, and conclude that the laser reliability problem can, in fact, be solved by proper use of system-level reliability solutions.

Chapter 2

Why Optics?

Why would one want to use optical interconnect in a multiprocessor? Studies of multiprocessor optical interconnect have cited many advantages of optical over electronic interconnect [6, 5, 7, 8]. The major advantages are outlined below.

2.1 Connection Density

From a network theory perspective, the most interesting advantage of optical interconnect is its potential for increased connection density. This is due to the contrasting natures of the electron and the photon.

Electrons are fermions and carry a charge; they therefore interact strongly with each other, both due to electromagnetic effects and due to the Pauli exclusion principle. Photons are bosons and electrically neutral; they interact weakly, if at all. Because of this, multiple light beams can cross the same point in space without interfering, while multiple wires cannot. This property of optical interconnect may render some previous analyses of multiprocessor networks, which implicitly assumed connection density limits, inapplicable.

2.2 Bandwidth

One of the most immediate benefits of optical interconnect is increased bandwidth. While electrical network interconnects have been pushed to 1 Gbit/second [9], it becomes increasingly cumbersome as frequency components progress higher into the

microwave spectrum. Optical links now in use for telecommunications achieve 3.4 Gbit/second data rates over distances of 50 km [1]. Laser modulation rates of 15 GHz have been reported [10].

Fundamentally, an optical link is a modem (modulator-demodulator), with an optical carrier signal. Phenomenal bandwidth potential is not surprising, since the carrier being modulated typically has a frequency of 200 THz[1] or more. The bandwidth available on an optical link is not limited by the link itself, rather by the electronic circuits on either end. Progress in Opto-Electronic Integrated Circuits (OEIC's) suggests that the highest-speed electronic signals associated with the link will need to extend no further than a single chip (or chip module). This implies that serialization (multiplexing) of the electrical signals will be needed in order to exploit the full bandwidth of the optical link.

Data rates in conventional electronic signaling (that is, baseband signaling) are limited by dispersion effects, due to the very broad bandwidth relative to the center frequency. Optical telecommunications links also suffer from dispersion, but over the relatively short distances encountered within multiprocessor interconnection networks ($\ll$ 1 km), dispersion is very small.

2.3 Fanout

Electronic logic signals have traditionally supported high fanout (that is, multidrop) connections. Conventional bus architectures (for example, VMEbus or NuBus) rely on electronic fanout to broadcast signals among all the boards on the bus.

Unfortunately, the multidrop model breaks down at high switching speed, due to transmission line reflections. As Knight points out [11], while it is theoretically possible to adjust and tune the transmission line to produce a matched multidrop line, in practice it is so cumbersome as to be impractical. Practical high-speed ($\geq$ 1 Gbit/sec) electrical interconnect will generally be point-to-point.

Optical interconnect poses no such restriction. Multidrop connections are perfectly feasible, limited only by the total transmitted power available. A fundamental question, to be answered by further research, is how best to exploit this capability, or

[1] At $\lambda = 1.5\mu$m: the longest wavelength currently in wide use.

whether it is worth exploiting.

2.4 EMI Immunity

Unlike electrical signals, optical signals are virtually immune to electromagnetic interference (EMI) and similar effects, such as ground-loop noise. This is an important practical advantage, as anyone who has constructed large-scale high-speed computing systems can attest. Although quite valuable, this electromagnetic interference immunity does not appear to have a direct impact on the choice of a particular computer architecture. Instead, makes implementation of any large-scale system more practical.

2.5 Communication Energy

Power dissipation is an important factor in all high-speed computing systems, and a large part of the power is dissipated in communication, not in calculation [12]. Miller [2] argues that, fundamentally, optical communication requires less energy than does electrical communication, over all but the shortest distances.

Miller's argument is based on the characteristic impedance of free space, $Z_0 = 377$. All practical transmission lines will have a characteristic impedance much less than Z_0, since the impedance depends only logarithmically on the line's dimensions. (At one conference [13], Miller put it: "All transmission lines are 50 Ohms." A bit of hyperbole, but not far from the truth.) We can see this illustrated by the impedance formula for a vacuum-filled coaxial line: $Z = Z_0 \frac{1}{2\pi} \ln(r_2/r_1)$. Even to get the impedance up to Z_0, the outer conductor would have to be 500 times larger than the inner conductor, and any dielectric material would make it even more difficult.

Small electronic devices can pass only small currents, and are therefore high-impedance devices, poorly matched to driving transmission lines. Traditionally, large buffer amplifiers (pad drivers) are placed between the logic elements on a chip and the transmission lines off-chip. Since the transmission line must be properly terminated (somewhere), the low-impedance drivers use up considerable power in charging and discharging the line. Low-impedance drive is also the source of current-switching transient (dI/dt) noise: a major design constraint.

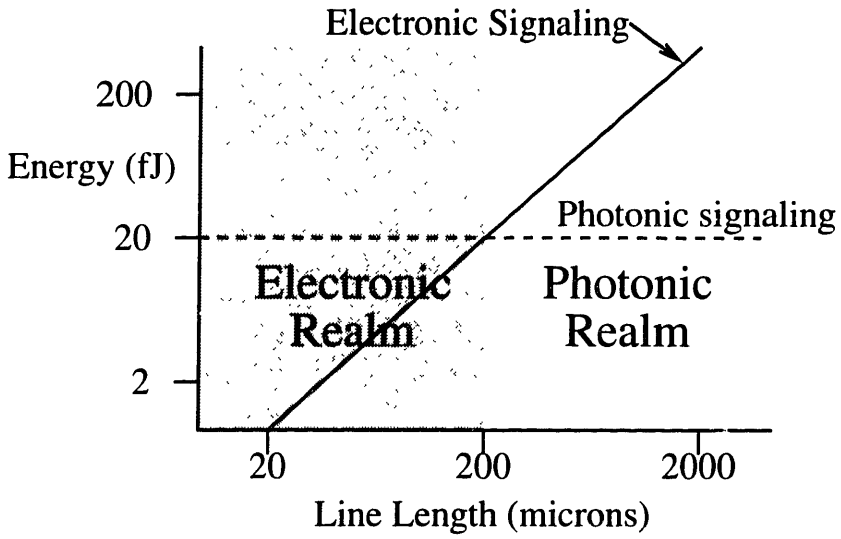

Figure 2-1: Communication energy vs. Distance (based on [1, 2])

Optical signaling avoids the free-space impedance problem, since it operates on quantum mechanical (photons↔electrons) rather than classical physical principals. Photodiodes are already well matched to logic inputs, and multiple-quantum-well techniques may produce low-threshold lasers and/or optical modulators which can be driven directly by microelectronic gates.

It now seems that optical signaling is the best approach for long-distance communications, and electronic signaling is the best approach for communication over microscopic distances. Ross [1] (probably relying on Miller's work [2]), offers a graph like the one in Figure 2-1 to argue that optical communication is preferable to electrical communication over distances longer than some critical length. The exact value of this critical length depends on the specific parameters of the links being compared, but Ross puts it at 200μm. (This is based on fundamental energy considerations. If, instead, one used present-day economic considerations, one might find the the critical length to be closer to 200cm, or even 200m.)

If energy is the ultimate limitation, it seems that optics may eventually supplant electronics for all interconnect above the intra-chip level. It does so now for communications over long enough distances, and the criterion for 'long enough' will certainly grow shorter and shorter as optical technology progresses.

Chapter 3

Multiprocessor optical networks

Optical interconnect is an active research field, and the state of the art is advancing rapidly. Here I describe the state of the art as it applies to optical networks for multiprocessors.

3.1 Data Communication vs. Telecommunication

Driven by the telecommunications market, optical communication hardware is now highly developed and continues to improve. Telecom hardware now achieves phenomenal performance, such as fibers with 0.1 dB/km attenuation, and 3.4 Gbit/sec links with 50-km distances between repeaters [1]. Current research topics, such as soliton-based signaling [14], promise even more impressive results.

Unfortunately, this superb telecommunications hardware is not directly applicable to multiprocessor networks, since it is designed to satisfy different constraints. In telecom, long-distance performance is the key, since this governs the number of repeaters required on a link, and repeaters form a major part of the cost of a link. Also, telecom size and power constraints (per link) are often less stringent than the corresponding constraints for computer hardware.

Because of this, telecom optical links tend to make the cost-performance tradeoff in favor of high performance and high cost per link: too high a cost for use in parallel processor networks, which need many, many links. (We need many links in order to increase bandwidth without increasing electrical switching speed, and to reduce the latency inherent in serializing the data for transmission. Such latency is no problem

in telecom systems, since it is overwhelmed by the speed-of-light transmission delay.)

Optical datacom (that is, short-distance communication) hardware is well behind the leading edge of performance. For example, the Fiber Distributed Data Interface (FDDI) local-area network [15] is only now coming into wide use, and it uses a signaling rate of 125 Mbit/sec: over 25 times slower than the fastest telecom links.

Given the differing cost constraints, it seems likely that datacom hardware performance will remain well behind the state of the art in telecom. However, this state of the art is advancing so rapidly that we can reasonably expect optical datacom hardware to achieve current telecom performance within a few years.

3.2 Data Communication Hardware

While there are many possible implementations of a fixed optical communication link, they share some common features. A link consists of an optical source, an optical receiver, and a transmission medium.

3.3 Optical Sources

The two major optical sources for communication links are light-emitting diodes (LED's) and semiconductor lasers. Another option is the use of optical modulators acting on an externally generated laser beam.

Light-Emitting Diodes

Light-emitting diodes are very simple devices, and have a number of advantages for use in an interconnect network: they are cheap, easy to fabricate, and reliable. In some moderate-performance applications, light-emitting diodes are the perfect choice. For example, FDDI [16] uses them as optical sources.

Unfortunately, the light-emitting diode has limited bandwidth. Light-emitting diodes are quite capable of signaling rates around 100 Mbit/sec (as in FDDI), and with some effort can be made to switch at 1 GHz, but at higher switching rates they become increasingly impractical. For >1 Gbit/sec rates, lasers or optical modulators appear to be the most reasonable sources.

Semiconductor Lasers

The most straightforward laser light source is the semiconductor laser, with its drive current modulated by the desired data stream. Modulation of semiconductor lasers at 15 GHz has been demonstrated, and it seems safe to say that the laser switching rate will likely be limited by the drive electronics before it is limited by the laser itself. Microfabrication of surface-emitting semiconductor laser arrays is showing impressive results, such as densities of 200,000,000 lasers/cm^2 [1], and threshold currents around 1 mA (potentially much lower [17]).

Semiconductor lasers have a number of problems, but probably the most difficult one for this application is laser reliability. Laser reliability figures are hard to come by, and depend on several factors, but the current state of the art is on the order of 10^5 or 10^6 hours MTBF. For a massively parallel processor with thousands of lasers, laser failure will be a fairly common event, and must be handled gracefully. A semiconductor-laser-based system will likely have to cope with the laser reliability problem by providing considerable fault-tolerance in the higher-level design.

3.3.1 Optical Modulators

To avoid the semiconductor laser reliability problem, an alternative approach using optical modulators has been proposed [5]. In this scheme, large, high-power external lasers are used to provide an optical 'power supply' which is routed to the inputs of optical modulators. The external laser can be made quite reliable and wavelength-stable. The modulator outputs produce a modulated light beam similar to that otherwise produced by the semiconductor laser. The modulators and lasers have broadly similar characteristics; in fact, multiple-quantum-well modulator arrays (see Section 3.6.1) are almost identical to multiple-quantum-well laser arrays, omitting the laser mirror on one side.

Modulator-based schemes have their own problems, chief among them the need for optical power distribution. However, a number of groups (notably Honeywell) are convinced of their superiority, especially in systems subject to wide temperature variations, such as military systems.

The modulator vs. laser question is still an open one, with no consensus yet appar-

ent. A significant factor in deciding the issue will be the extent to which it is practical to handle the laser reliability problem with inexpensive, system-level solutions.

3.4 Optical receivers

Optical receivers are relatively straightforward. The optical signal illuminates a diode, causing a photoelectric current. This is then amplified to the desired logic levels (or used directly [18]).

Either PIN (p—intrinsic—n) or avalanche diodes may be used. The avalanche diode produces a stronger signal, but is more difficult to fabricate. Either diode may be fabricated in silicon, but at the high data rates of interest here, III-V materials such as GaAs are also of interest.

This thesis assumes the use of suitable photodiodes, with high-speed amplifiers on the same chip or module, if needed.

3.5 Transmission Media

Optical transmission media are another important concern in optical network research, since the medium partly determines the topological and other constraints on the optical network.

3.5.1 Free Space

The simplest medium is free space. The pure free-space link (a light source shining directly on a receiver) is finding applications now, over short distances such as those in inter-board communication. While pure free-space links are interesting, some other free-space approaches are potentially more useful in computer networks. A number of these alternatives involve free-space links with some reflective or refractive device between source and receiver.

Free-space communication schemes (and holographic schemes, to some extent) offer the opportunity to exploit the ability, mentioned in Section 2.1, of optical signals to cross in the same point in space. This is a significant advantage, but it is not without cost. All free-space approaches require extremely accurate mechanical alignment,

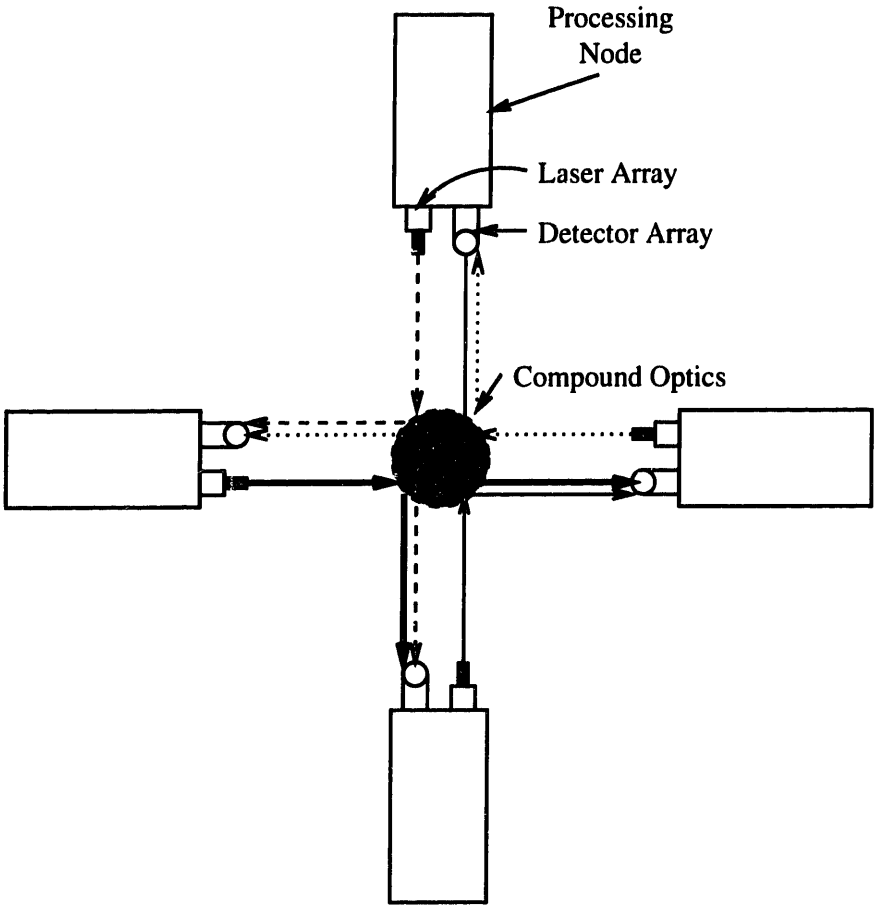

Figure 3-1: Free-Space Communication with Compound Optics

posing problems for manufacturing and, more particularly, for repair. (Re-alignment of a replacement component in the field may be quite difficult.) Some form of adaptive alignment would be a significant advantage.

Considerable research has dealt with the use of compound optics as a means of distributing light from source to receiver (see Figure 3-1). This generally imposes a constraint of 'space-invariance' on the transmission pattern: the output pattern (relative to the input beam) must be the same from all ports.

3.5.2 Holographic Networks

Holographic interconnect is another form of free-space link, with the optical distribution device being a hologram [19] (see Figure 3-2). Holograms do not impose a space-invariance constraint. The pattern reflected by the hologram is basically arbitrary, within certain angular constraints. These angular constraints can be significantly

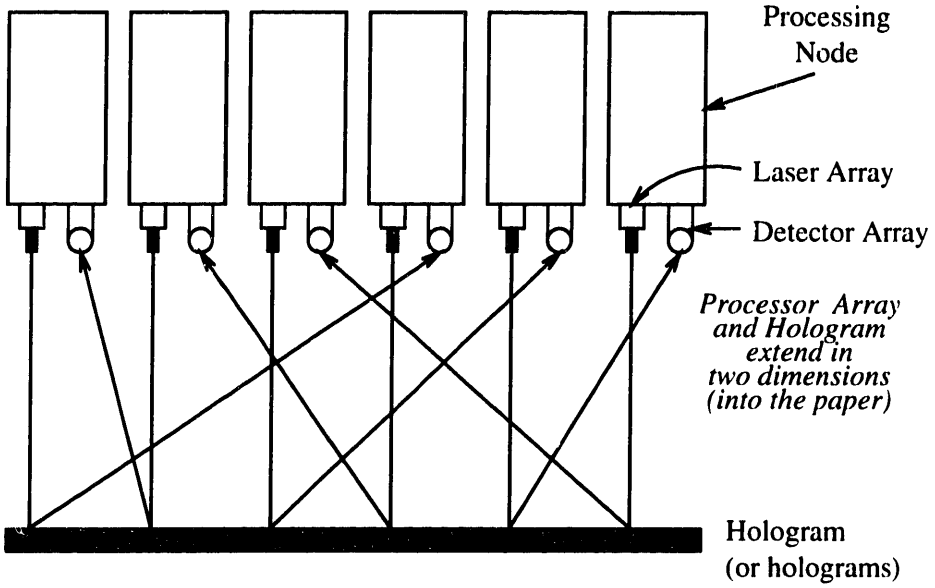

Figure 3-2: Free-Space Communication with Holograms

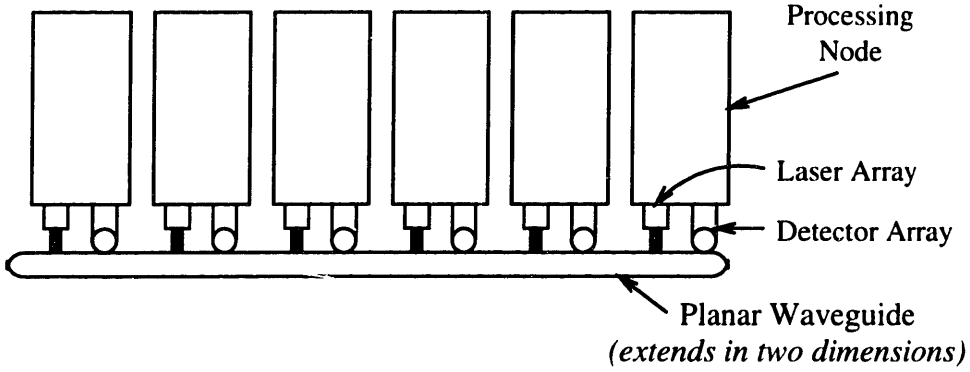

Figure 3-3: Communication via Planar Waveguide

relaxed by employing multi-layer holograms (that is, several holograms sandwiched together).

3.5.3 Planar Waveguides

Oversimplifying, one might say that a planar waveguide [20] is a 'printed-circuit board' for optical signals (see Figure 3-3). It offers connections between arbitrary points in a plane, with crossovers, combiners, and splitters available. Being self-contained, it obviates most of the alignment problems of free-space approaches such as holograms. Unfortunately, their transmission efficiency is still quite low, currently around 0.3 dB/cm. (Compare this to 0.1 dB/km for high-quality fiber!)

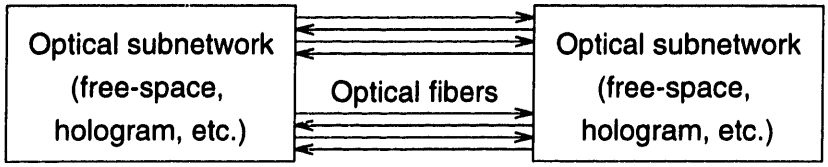

Figure 3-4: Subnetwork interconnection via optical fiber

Planar waveguides offer moderate connectivity with moderate implementation difficulty. They appear to be a possible compromise between point-to-point fiber connections (easily implemented, but low connectivity) and holograms (difficult implementation problems, but very dense connectivity). Lin [21] suggests an interesting combined hologram/waveguide approach.

3.5.4 Fiber Optics

Optical fiber is by far the most highly developed optical medium today. Optical fiber is now produced which imposes only a few dB of loss over a distance of hundreds of kilometers. Unfortunately, such long-distance performance is of little relevance to multiprocessor networks, where distances will be less than ten meters.

The basic disadvantage of fiber is that it is implemented on an individual link basis, one fiber at a time. Progress is being made on the use of fiber bundles: parallel groups of fibers terminated together [22], but it is still premature at this time to consider a large-scale multiprocessor with interconnections solely based on fiber bundles.

However, fiber does have one important advantage: flexible geometry. All of the previously-mentioned optical media require the communicating nodes to be aligned in a pre-determined geometry, and fixed in that alignment. Optical fibers, on the other hand, offer flexibility in two senses: mechanical flexibility, allowing communication between points which are not in rigid alignment, and design flexibility, allowing arbitrary placement of the two endpoints of the fiber.

Fiber bundles might therefore be of considerable usefulness in connecting optical subnetworks which were implemented by other, less flexible means. The subnetworks could be made as large as practical, and then used to create a still larger network via fiber connections. The Figure 3-4 illustrates this idea.

3.5.5 Assumptions of the analysis

Since the laser reliability questions are substantially the same for all the optical media presented here, my analysis need not assume the use of any particular medium. However, the analysis of transient errors in Section 4.1 is particularly important for the less power-efficient media, such as planar waveguides, since the importance of the power vs. error-rate tradeoff is magnified in those cases.

3.6 Optical Switching and Computing

A potentially strong motivation for the use of optical networks is optical switching and computing. There is no intrinsic distinction between optical switching and optical computing: any reasonably efficient switching system can be made to do computation. However, while 'optical switching' means what it says, the term 'optical computing' has had the special connotation of *purely* optical computing. [To confuse matters further, the meaning of 'optical computing', at least in some quarters, seems to be evolving toward 'optical communication between electronic logic devices': just the sort of thing I examine in this thesis. I shall use the older meaning.]

3.6.1 Optical Switching

Optical switching has been studied for some time, and optical switches for telecom have been well developed. However, switches for massively parallel systems must be extremely fast and compact. Presently-available switches are generally interferometer designs (such as Mach-Zehnder) which tend to be large, or liquid-crystal designs which tend to be slow. Current research into newer designs, such as Bell Lab's work on multiple-quantum-well (MQW) devices [17], is quite interesting for this application.

Multiple-quantum-well switches might serve as the modulators for the external-laser design mentioned in Section 3.3.1. The external-laser design tends to blur the dichotomy between Optical Communication and Optical Switching set forth above.

3.6.2 Optical Computing

One might consider optical computing to be the Holy Grail of this field. All of the electro-optical approaches are highly mismatched in speed, with the electrical portion as the speed-limiting factor. If true optical computing could be achieved, it might improve computer performance by orders of magnitude.

True optical gates have been demonstrated [23], but they involve high power levels, and are not fully restoring. Pure optical gates are far from practical computing implementation for the time being, and seemingly for the near future as well.

Reflecting this discouraging prospect, the term 'optical computing' is apparently evolving toward a different meaning: electronic computing with purely optical interconnection. (In a sense, this is what the purely optical gates do as well, since their optical non-linearity is fundamentally based on electronic interactions.) Bell Labs researchers vigorously assert that their self-electrooptic devices (SEED's) [24] implement 'optical computing'.

Self-electrooptic devices, and other 'optical computing' devices in the new sense of the term, while remarkable devices, are fundamentally electronic logic gates with optical input and output, and therefore limited by electronic switching speeds.

3.6.3 Assumptions of the analysis

While high-speed optical *communication* hardware is practical now and is in wide use, high-speed (that is, high reconfiguration speed) optical *switching* hardware is less advanced, and optical *computing* hardware (in the original sense) will not be practical for some time, if ever. Therefore, the analysis in this thesis treats the reliability of optical *communication* only. Analysis of optical switching and computing is an area for further research.

3.7 Multiprocessor Networks

Multiprocessor networks have been a fertile subject of research for over two decades, and continue to be so. A vast literature is available on various interconnection schemes. However, no consensus has been reached as to which approaches are optimal, even in the extensively-studied electronic regime, much less in the optical regime.

Nevertheless, some interesting mathematical results on performance limits of various networks have been achieved (for example, Dally's work on mesh networks [25]). Many of these analyses assume that network bisection bandwidth is the limiting factor. The assumption may be true for electronic interconnect (although even this is disputed), but is unlikely to be valid for optical networks. On the other hand, the analysis of Agarwal [26] would seem to be applicable to optical networks, in the case where he assumes a limit on node size instead of network bandwidth.

3.7.1 Mesh Networks

Mesh networks, also called grid or nearest-neighbor networks, possess an elegant simplicity. The basic idea is to place nodes in a plane (or a volume) in a regular pattern, and form a communication network by linking each node with its nearest neighbors. Their short communication lengths and point-to-point links make mesh networks especially well-suited to electronic implementation, negating a number of the advantages mentioned in Section 2.

An optical mesh network may still be worthwhile. As noted in Section 3.5.1, point-to-point free-space interconnect is well advanced, and is now close to practical implementation. This could be used to implement an optical mesh network. The existing research into electronic mesh networks should be directly applicable, with the optical links looking like very-high-speed wires.

If the optical mesh network is not used, a very interesting possibility is a dual network combining a non-mesh optical network with an electronic mesh. The mesh network is so well suited to electronic implementation that the addition of an electronic mesh network might provide a significant performance gain for small extra cost. This idea is illustrated in Figure 3-5.

3.7.2 Hierarchical Networks

In the current literature on multiprocessor networks, there is considerable work in hierarchical networks. These are large, complex networks consisting of a hierarchy of smaller, simple networks.

Hierarchies both of busses [27] and of crossbars [28] have been studied. They offer opportunities to exploit the ability of optics to support high-fanout connections,

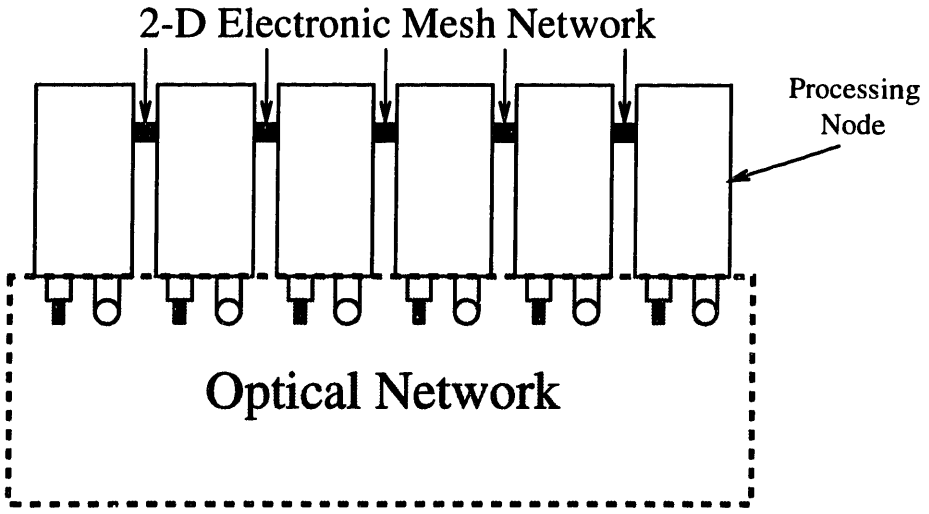

Figure 3-5: Dual Electronic/Photonic Interconnection Network

discussed in Section 2.3.

3.7.3 Butterfly-type networks

Butterfly networks (omega networks) are in wide use for multiprocessor interconnect, notably in the Bolt, Beranek, and Newman (BBN) Butterfly multiprocessor [29]. The multibutterfly, a variation on the butterfly network, is another possibility for an optical network. When the connections of a multibutterfly are randomized under certain constraints, it can achieve impressive fault-tolerant performance, as discussed by Leighton [30]. The randomized connection stage might be particularly well suited to holographic implementation.

3.7.4 Assumptions of the analysis

While the various network topologies offer fascinating opportunities for development of new multiprocessor architectures, the optical reliability questions are fundamentally similar across all of them. For my reliability analyses in Chapters 6, 8, and 9, I therefore assume the use of the simplest topology: the two-dimensional mesh.

Chapter 4

Optical data link reliability problems

Since, as was mentioned in Section 3.6.3, we are dealing with optical technology for communication only, we can define optical data link problems simply as events which cause the data received from the link to differ from the data originally transmitted. We can divide such events into two classes:

- *errors*, of short duration ('transient errors'),
- *failures*, of much longer duration ('hard failures').

The dividing line between these two classes is the amount of time required for the system to recognize, locate, and verify the problem. The exact division is not critical, since most such events are either very short (on the order of one bit transmission time) or permanent.

The goal of this chapter is to characterize the behavior of both transient errors and hard failures, and to develop mathematical models describing them. In Chapters 6-9, I use these models in developing and evaluating solutions to the reliability problems.

4.1 Transient errors

No method of data transmission is perfect; there is always a chance that a given transmission has been corrupted by transient errors.

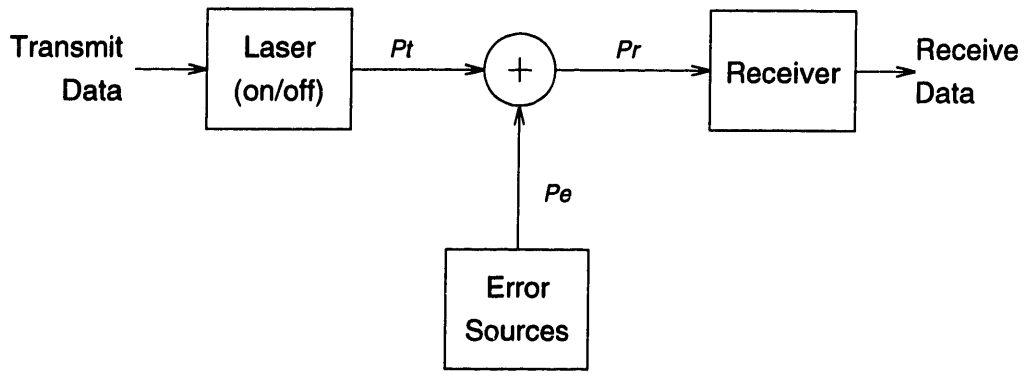

Figure 4-1: Optical communication error model

The simplest model of such errors is that of two-level modulation (such as 'on' and 'off') and a Gaussian additive noise source, which will occasionally cause the receiver to report an incorrect value [31]. Figure 4-1 shows an example of such a system.

A laser, modulated by an incoming data stream, produces a transmitted optical signal p_t at one or two intensity levels: 0 (off) or L (on). All the sources of error are lumped into one error signal p_e, a zero-mean Gaussian signal with standard deviation σ. To simplify the analysis, we set $\sigma = \sigma^2 = 1$. This implies that the Signal-to-Noise ratio (SNR)= L. The receiver compares the received signal $p_r = p_t + p_e$ with the threshold $L/2$, and reports 'on' if $p_r < L/2$ and 'off' if $p_r > L/2$.

When $p_t = 0$, the probability that a bit will be received in error is the probability that $p_e > L/2$, which is

$$\text{P(error)} = \text{BER} = \frac{1}{\sqrt{2\pi}} \int_{L/2}^{\infty} e^{-\frac{1}{2}p^2} dp = \frac{1}{2}\text{erfc}\left(\frac{L}{2\sqrt{2}}\right) \tag{4.1}$$

By symmetry, the same analysis applies when $p_t = L$. Figure 4-2 shows a plot of BER values given by Equation 4.1.

Note, however, that this idealized analysis will eventually break down at some point. When the power level is sufficiently high, the model developed above breaks down and other error sources become apparent, causing an 'error floor' in the BER vs. signal-to-noise ratio curve. The dashed line in Figure 4-2 is an illustration of such a floor.

In an optical link (especially with a power-inefficient transmission medium), the tradeoff between optical power and BER (lower power $\rightarrow$ higher BER) can be quite important. The optical sources require extremely fast electrical drivers, and the higher

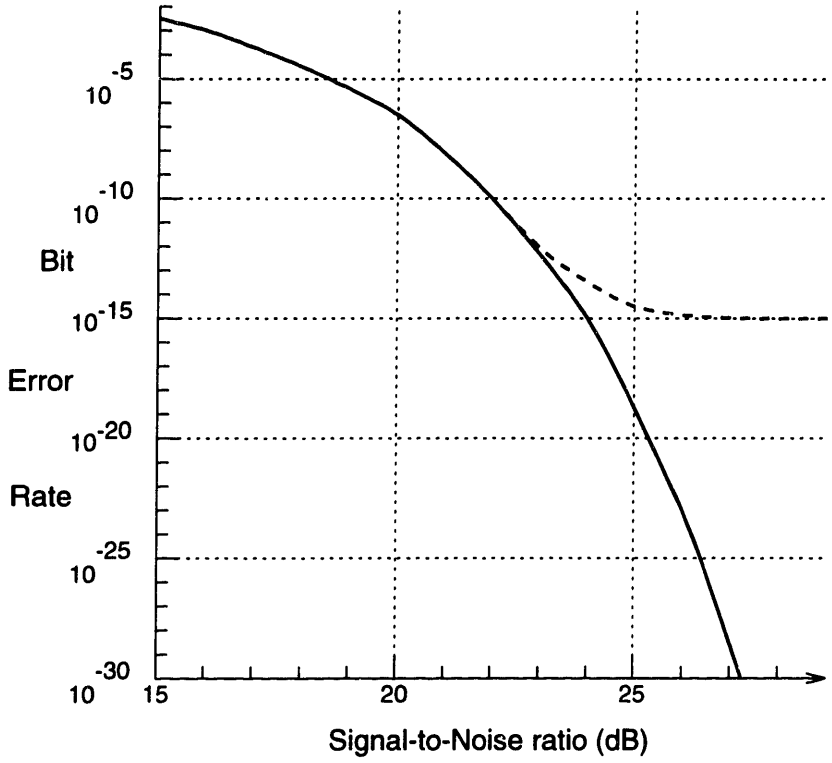

Figure 4-2: Error rate vs. Signal-to-noise ratio

the power required of the drivers, the more difficult their design becomes. Also, there is a power vs. reliability tradeoff in semiconductor lasers (lower power → higher reliability).

[The power↔BER tradeoff will also become important in Section 7, when I discuss intelligent control of laser drive power. In that case, I rely on the monotonic nature of this tradeoff to implement a power-control feedback loop with data link BER as the feedback variable.]

Generally, one needs extremely low error rates in a multiprocessor network, since an undetected error can be catastrophic. In Section 8.1, we derive a maximum tolerable BER of 10^{-23} for a prototype network. The best achievable BER values now are around 10^{-15} [3], and even these levels are only possible with painstaking care in designing every part of the optical channel.

How then can one tolerate an optical network in a multiprocessor? As we shall see in Section 6, a simple error-control coding scheme can solve this dilemma quite neatly, provided that a usable ($<10^{-7}$, for example) bit-error-rate can be maintained. (Chapter 6 gives a strategies for maintaining such error rates as the system ages.) In

addition to making optical links usable in multiprocessors, error-control coding can also make their design and implementation much simpler and more cost-effective, by relaxing the otherwise stringent error-rate requirements.

4.1.1 Assumptions of the Analysis

The most fundamental assumption I shall make in my analysis of reliability solutions is that errors in different bit positions are independent. Since I am suggesting a system with wide optical channels, having separate optical channels for each bit, the assumption is quite plausible to a first order approximation.

However, I will be pushing this independence assumption very hard in my error-control analysis. Any significant error correlation between channels (for example, electromagnetic interference between channels at the receiver module) might compromise my error-control results. (On the other hand, such correlation might simplify the drive-control tasks addressed in Chapter 6, since one might only have to control the drive level on a per-module basis, instead of a per-laser basis).

Since such error problems are likely to be implementation-specific, I shall merely note the possibility of such problems, and conduct my subsequent analyses under the independent-error assumption.

4.2 Hard Failure

All electronic components degrade with age. Figure 4-3 shows the typical relationship (called the 'bathtub curve') between age and failure rate for any electronic component. Note that the entire curve is customarily characterized by one number: 'MTBF', an ambiguous acronym which can mean either mean or median time between failures. In this thesis, I shall generally consider MTBF to refer to the median time, and I will make it clear when the other sense is meant and the distinction is important. (In this case, MTTF, mean/median time *to* fail, is more appropriate, but we will use the more common term.)

Initially, there is a high failure rate due to flawed components ('infant mortality'). There follows a long period of few, randomly-distributed failures ('random failure'). Finally there is an increasing rate again ('wearout failure'), as the component aging

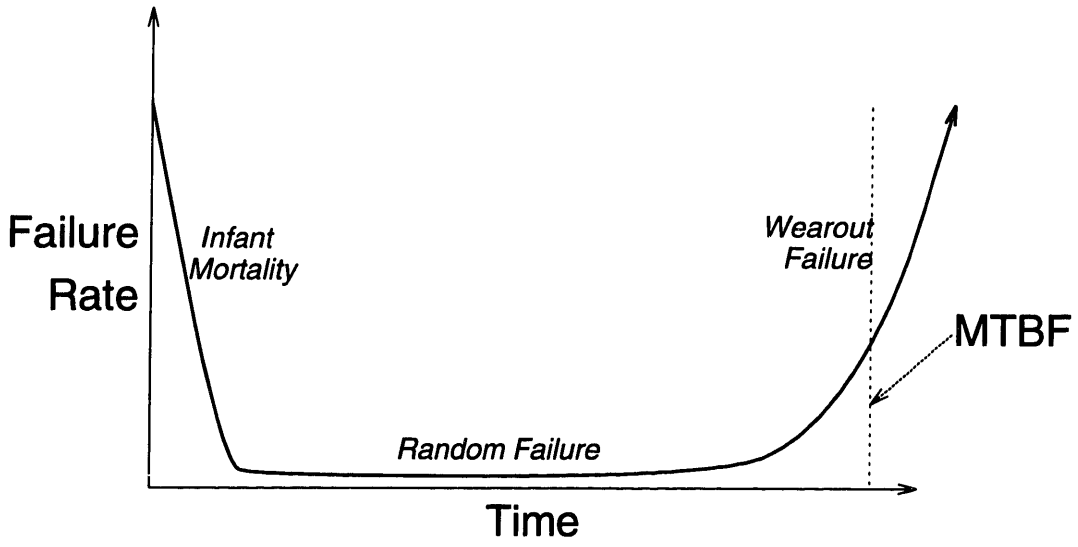

Figure 4-3: Component failure rate vs. time

processes begin to take their toll.

Infant mortality can be controlled by proper screening and burn-in procedures [32] before the system begins operation. Random and wearout failures, however, must be handled in the field, either by fault-tolerant redundancy schemes, or by repair.

Unfortunately, semiconductor lasers tend to degrade faster than most other components in an optical communication network, so their failures will tend to dominate the failure characteristics of the system. I will therefore examine the nature of semiconductor laser failure, considering wearout failures in this section, and random failures in Section 4.2.2.

4.2.1 Gradual wearout

There are numerous aging processes which lead to wearout failure in semiconductor lasers [4], but they can collectively be modeled by a log-normal lifetime distribution, where log(lifetime) is a Gaussian random variable. If we let t_ℓ denote the wearout lifetime of an individual laser, then the probability distribution of t_ℓ will be given by

$$p_{t_\ell}(t) = \frac{1}{\sqrt{2\pi}} e^{\frac{-\ln^2(t/t_M)}{2\sigma^2}} \tag{4.2}$$

where t_M is the median lifetime, and σ is the standard deviation. Figure 4-4 shows the shape of the log-normal distribution, for various values of σ.

[Note that this is one of the situations where the mean vs. median ambiguity in

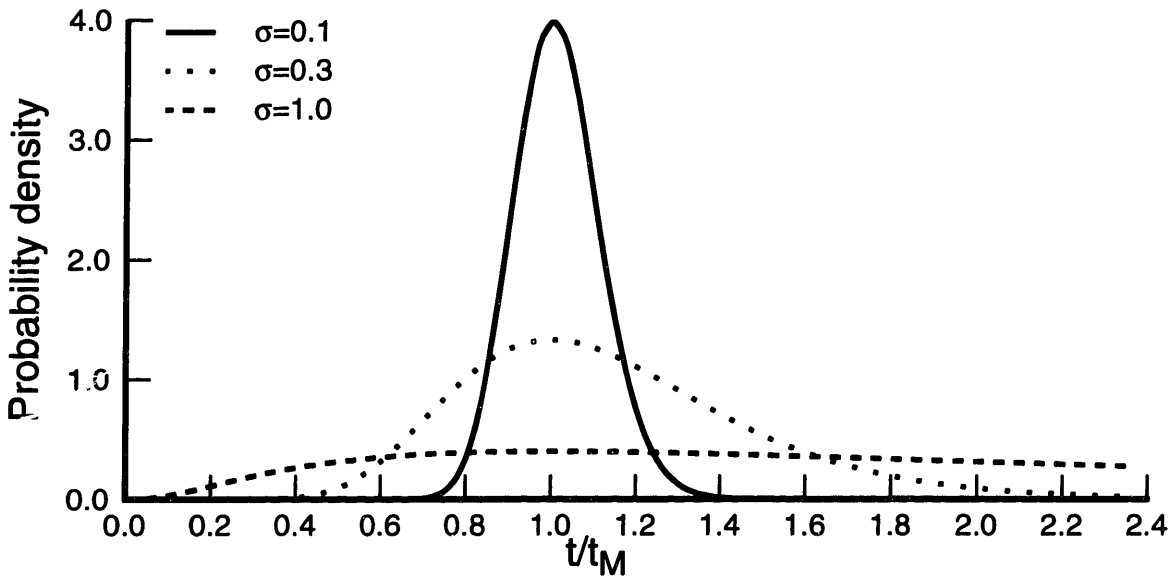

Figure 4-4: Log-normal probability distributions

acronyms such as MTBF can be misleading. We can see that for larger values of σ, the distribution is skewed toward larger values of t_ℓ, and the mean lifetime will be considerably larger than the median. For example, with $\sigma = 0.4$, the mean lifetime is 144% of the median.]

An important aspect of Figure 4-4 is that all of the curves are skewed away from time $t = 0$. This is why, for multiple lasers, the simplistic relation in Equation 1.1 does not apply. What relation *does* apply to wearout failures in multiple lasers? We shall answer this in Chapter 8, when we consider system-level solutions to the failure problem.

Laser wearout vs. time

Taken as a whole, the lifetimes of lasers in a system are random, and follow the probability density given in Equation 4.2. Taken individually, each laser has its own deterministic wearout lifetime. When this time comes, how does the wearout manifest itself?

Wearout failures are not instantaneous; they occur quite gradually. As a laser wears out, its light output gradually decreases for the same drive current. (Or, con-

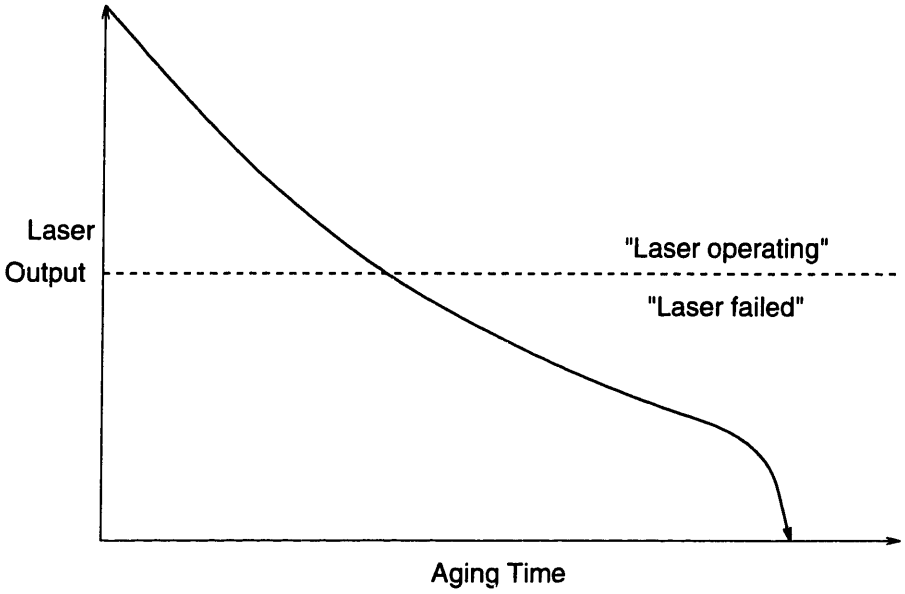

Figure 4-5: Laser wearout vs. time

versely, the drive current required to maintain the same light output slowly increases). Laser MTBF figures are obtained by setting an arbitrary wearout criterion, such as a 50% output power decrease or a 100% drive current increase, and declaring a laser 'failed' when this criterion is reached. However, the gradual degradation continues considerably beyond this point, until finally the laser stops lasing. Figure 4-5 illustrates this concept in the case of constant laser drive current.

The light output of a semiconductor laser is a linear function of the drive current (I_D) in excess of the threshold current (I_{th}) for that laser, that is

$$\text{Laser light output} \propto I_D - I_{th} \tag{4.3}$$

The wearout processes generally manifest themselves as a gradual increase in threshold current. Using $I_{th}(t)$ to denote threshold current as a function of time, we can model this gradual increase as

$$\frac{I_{th}(t) - I_{th}(0)}{I_{th}(0)} \propto t^n$$

where n is a constant characteristic of the particular laser type used ($n \approx 0.5$ [33]).

Letting α denote the drive current increase used for calculating the laser lifetime t_ℓ (for example, a 50% increase means $\alpha = 0.5$), and letting I_0 denote the initial drive current used, we have

$$I_{th}(t) = I_{th}(0) + I_0\alpha(t/t_\ell)^n \tag{4.4}$$

Therefore, with a constant drive current, the laser light output will gradually diminish. If the drive current is adjusted for constant light output, the current required will gradually increase.

If the available laser drive current limit is the same as that used for calculating the wearout lifetime statistics (for example, 50% increase from initial drive current), then the standard lifetime figures will be directly applicable. However, if the available laser drive is different, the actual wearout lifetime will vary accordingly.

This gradual degradation process presents both a problem and an opportunity. The problem is that a laser's performance will begin to decay even before its nominal wearout lifetime arrives; therefore additional power margins will have to be included in the link design, to allow for acceptable operation even with somewhat aged lasers. The opportunity to be exploited is that the increasing drive current or decreasing output power will give quite ample warning that a laser is wearing out; this could enable the repair, replacement, or re-adjustment of failing lasers to be performed at conveniently long intervals. I will consider the system implications of this in Section 7.5.2.

4.2.2 Random failure

Random failure in semiconductor lasers is rather simple compared the wearout failure discussed above. Random failure is the failure type shown in the center section of the bathtub curve, Figure 4-3. Random failures occur due to problems such as bond-wire breakage or die attachment failure. They can be modeled as arrivals of a Poisson process.

If a large number of semiconductor lasers were to be operated continuously until failure, relatively few of the lasers would fall victim to random failure, since the random failure MTBF is generally much longer than the wearout MTBF. However, unlike wearout failures, random failures occur at the same rate throughout the system lifetime, and therefore *do* follow the simple relation given in Equation 1.1.

Also unlike wearout failures, random failures are generally abrupt, clear-cut failures as opposed to gradual fadeouts. This means that random failures, as the name implies, can occur unpredictably, without warning, at any time. Therefore, system design must provide for handling of random failures whenever the system is large enough

to reduce, via Equation 1.1, the large random-failure MTBF down to a too-short system MTBF. We shall examine such designs in Chapters 8 and 9.

4.3 Hard Failure Probability

The total failure rate will be the sum of the random failure rate and the gradual wearout rate. While the random failure rate is more or less fixed, the wearout rate will grow over time. I shall analyze both types of failure in the next two sections, and then arrive at a composite failure model for use in simulations.

4.3.1 Gradual wearout probability

As I mentioned above, the gradual wearout failure rate is not constant, but instead increases with time.

The probability that a particular laser has worn out by time t (assuming no random failures) is derivable from Equation 4.2, the wearout lifetime probability distribution. If we let $P_w(t)$ denote this probability, we have

$$\begin{aligned} P_w(t) &= \int_0^t p_{t_\ell}(\tau)d\tau \\ &= \frac{1}{\sqrt{2\pi}}\int_0^t e^{\frac{-\ln^2(\tau/T_w)}{2\sigma^2}}d\tau \\ &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\ln(t/T_w)} e^{\frac{-\tau}{2\sigma^2}}d\tau \\ &= \frac{1}{2}\left(1+\mathrm{erf}\left(\frac{\ln(t/T_w)}{\sqrt{2}\sigma}\right)\right) \end{aligned} \tag{4.5}$$

where T_w denotes the median laser wearout lifetime and σ is the standard deviation of the log-lifetime $\ln(t_\ell)$.

Figure 4-6 plots $P_w(t/T_w)$ for different values of σ. Note that for small σ, wearout failure does not become significant until close to the laser lifetime (about $0.75T_w$ for $\sigma = 0.1$), and is virtually complete a short time afterward (about $1.25T_w$). On the other hand, with large σ ($\sigma = 1$, for example), wearout failure starts around $0.25T_w$, and continues at an relatively constant rate through $t = 2T_w$.

Obviously, the log-lifetime deviation σ is an important parameter in the laser wearout function. In the limit $\sigma \to 0$, every laser is pre-ordained to wear out at exactly $t = T_w$ (obviously convenient for replacement scheduling). In the limit $\sigma \to \infty$, half the

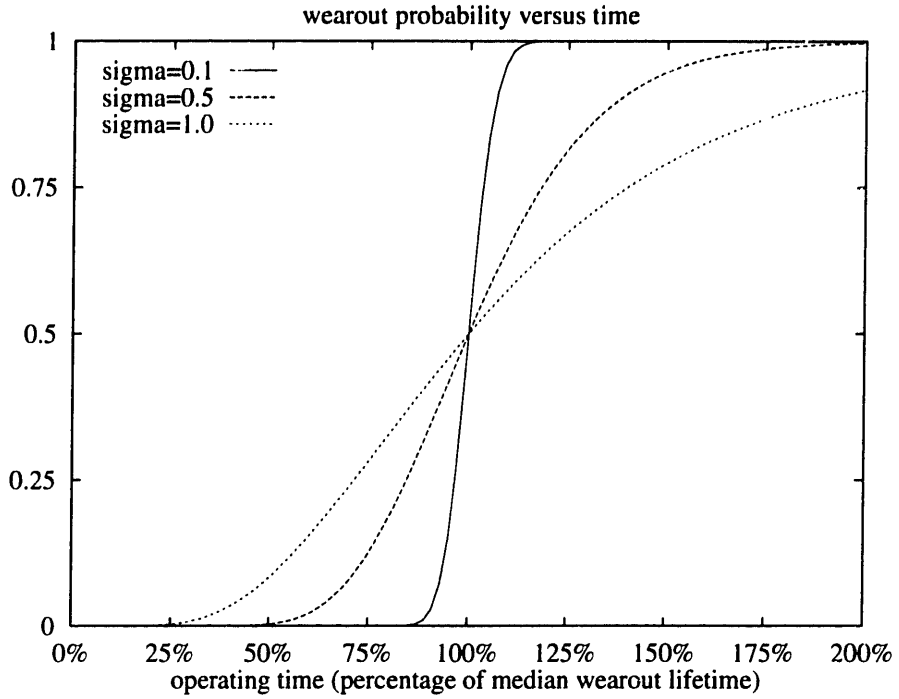

Figure 4-6: Laser wearout probability $P_w(t/T_w)$

lasers fail immediately, and the other half last forever. Between these (unreasonable) extremes, a reasonable real-world range of σ values is between 0.1 and 1.

Unfortunately, σ is often not well characterized for a particular laser design. For the moment, let us leave σ as an unspecified system parameter, and move on to consideration of random failures.

4.3.2 Random Failure Probability

As mentioned in Section 4.2.2, random failure is a rather simpler phenomenon, and can be modeled as a Poisson process. Analogous to $P_w(t)$, let us define $P_r(t)$ as the probability that a particular laser has suffered a random failure by time t (assuming no wearout failures). If we let T_r denote the mean time before random failure, we have the Poisson arrival rate $\lambda = 1/T_r$, and

$$P_r(t) = 1 - e^{\lambda t} = 1 - e^{-t/T_r} \tag{4.6}$$

Figure 4-7 shows $P_t(t/T_r)$, which is, of course, a simple exponential function. Unlike wearout failure, which is characterized by both a median lifetime and a deviation, random failure is fully characterized by its arrival rate λ or its mean interarrival time T_r.

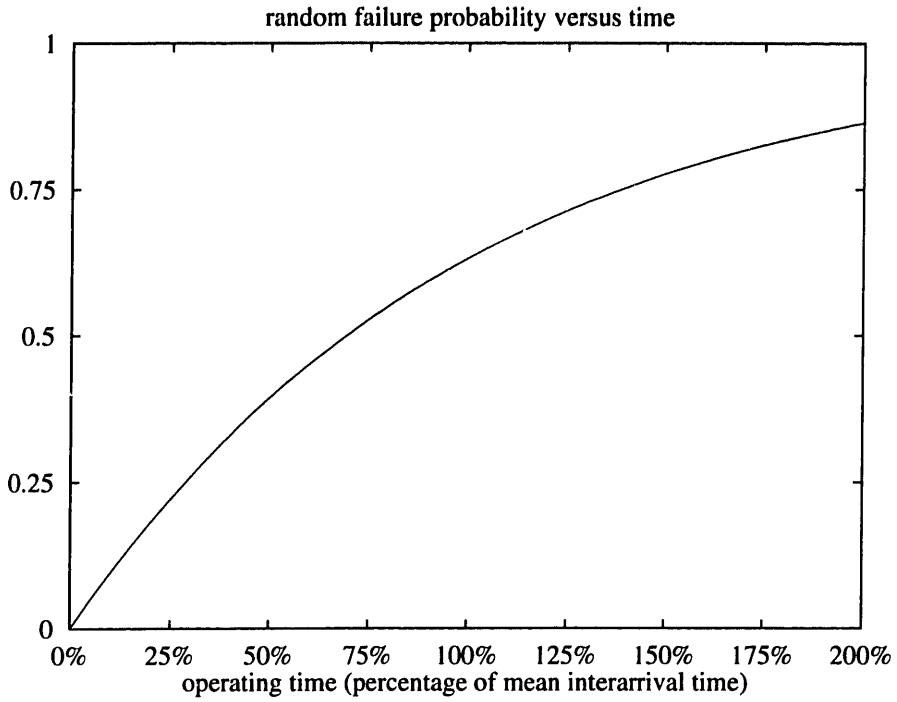

Figure 4-7: Random failure probability $P_r(t/T_r)$

Let us now consider the overall failure function, which is the combination of wearout and random failures.

4.3.3 Combined failure probability

To arrive at the combined error probability (random & wearout failures), I will assume that the two types of failure occur independently. This seems a reasonable assumption, since there are different failure mechanisms involved in the two types of failure.

If we let $P(t)$ denote the probability that a particular laser has failed by time t (either random or wearout failure), we have

$$\begin{aligned} P(t) &= P_r(t) + (1 - P_r(t))P_w(t) \\ &= 1 - e^{-t/T_r} + \frac{1}{2}e^{-t/T_r}\left(1 + \mathrm{erf}\left(\frac{\ln(t/T_w)}{\sqrt{2}\sigma}\right)\right) \\ &= 1 + \frac{1}{2}e^{-t/T_r}\left(\mathrm{erf}\left(\frac{\ln(t/T_w)}{\sqrt{2}\sigma}\right) - 1\right) \end{aligned} \tag{4.7}$$

Let us define an additional parameter $\gamma = T_w/T_r = T_w\lambda$. The value of γ is a measure of the relative importance of random failures vs. wearout failures. If $\gamma = 0$ then there are no random failures, if $\gamma = \infty$ then there are *only* random failures, and if $\gamma = 1$ then random and wearout failures are equally important.

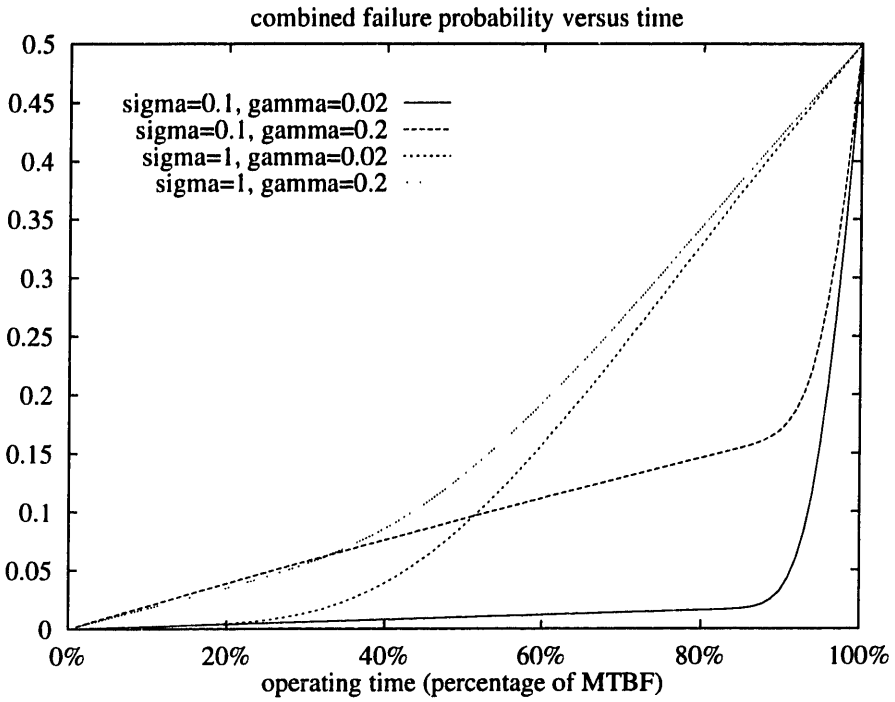

Figure 4-8: Combined failure probability $P(t/\text{MTBF})$

Laser reliability is most commonly specified with only one number: MTBF, comprehending both wearout and random failures. Let us then define T as MTBF, that is, the median time before laser failure, be it wearout or random failure. We then have three parameters to fully characterize the laser failure distribution: T, σ, γ.

The lifetimes T_w and T_r are now functions of T, σ, and γ, such that $T_r = T_w/\gamma$ and $P(T) = 0.5$. Since T is the value establishing our time scale, we must have $T \propto T_w = \gamma T_r$.

A closed-form solution for T_w and T_r (and therefore $P(t)$) as a function of T, σ, and γ is difficult to arrive at, but we can easily find numerical solutions. Figure 4-8 plots $P(t)$ for reasonable values of σ and γ. Note the importance of γ (the random failure parameter) in the early part of the lifetime axis. This demonstrates what I said earlier, in Section 4.2.2: random failure exerts its influence earlier than does wearout failure.

4.3.4 Probability model for analysis

The network failure analyses which I will describe in Chapters 8 and 9 are actually oblivious to the particular failure models and parameters we might choose to apply. The analyses and simulations work with particular values of the failure probability

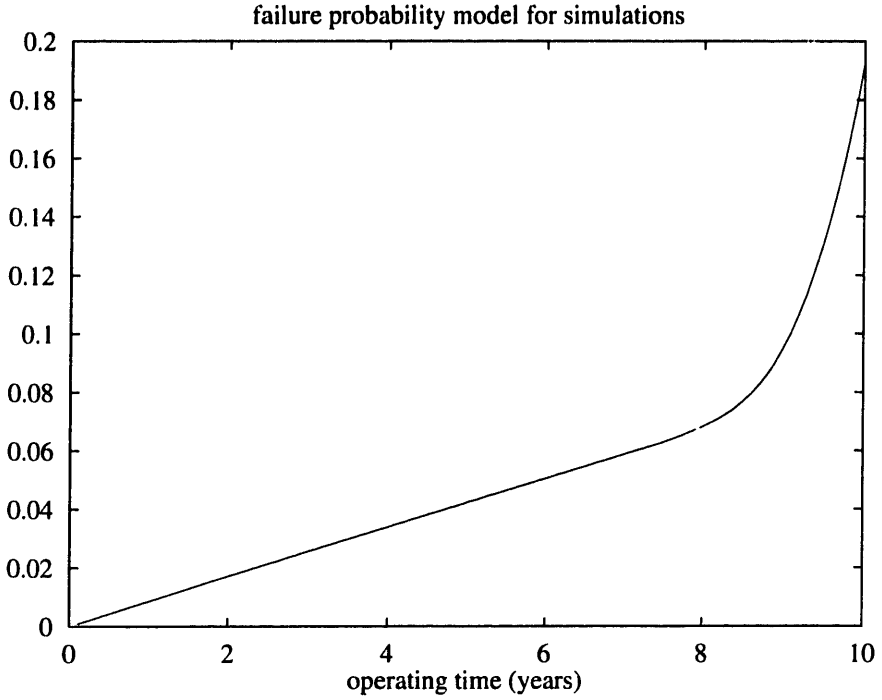

Figure 4-9: Simulation failure probability model

$P(t)$, and use the failure models only to relate these probabilities to times. Therefore, although I will set out some failure model assumptions here, the analysis and simulation results will be applicable to other failure models, with some adjustment to the system-life timeline.

For these purposes, I assume that we are using high-reliability lasers, of 100,000-hour MTBF, and I will assume a random-failure MTBF of 1,000,000 hours, and therefore $\gamma = 0.1$. As Fukuda points out [4], the random-failure rate is a strong function of the manufacturer's experience with the fabrication processes in use: as processes become established and well-known, the random-failure rate can be reduced dramatically. Of course, new processes can offer great performance advantages, at the cost of somewhat less reliability, so there is an engineering tradeoff here. The value $\gamma = 0.1$ is my best guess of the highest reasonable value for γ, which would place the most stress on the network's fault tolerance features.

I will assume a log-lifetime deviation $\sigma = 0.25$. Figure 4-9 shows the $P(t)$ function resulting from the laser lifetime assumptions. A value of 0.25 is reasonable for σ, but it could be considerably worse (perhaps $\sigma = 1$) for some lasers. In that case, the laser wearout would start around $t = 4$ years, and reach $P(t) = 0.2$ at $t = 7$ years.

One should keep this in mind when looking at the analysis and simulation results in Chapters 8 and 9: a high value of σ for the lasers in use would result in a higher rate of failure starting around $t = 4$ years.

4.3.5 Assumptions of the Analysis

As with transient errors, I shall assume that hard failures in different bits are statistically independent. Unlike my transient-error assumption, this assumption is almost certainly false.

I am assuming the use of arrays of lasers as components of very wide optical data channels. There will almost certainly be significant correlations among the lifetimes of lasers within the same array.

Modeling and simulating such correlations would complicate my simulations and analyses considerably. To make the problem more tractable, I shall assume independent failures, to arrive at a baseline result. Further research could refine this result to include correlated-failure effects.

4.4 Summary

I take two major results from here for use in the following chapters:

- for the prototype system I describe in this thesis, a reasonable undetected bit-error-rate requirement is less than 10^{-23}
- the failure probability curve given in Figure 4-9 as a reasonable basis for hard-failure simulation and analysis.

Chapter 5

Prototype network

For my analyses and computer simulations of optical network reliability solutions, I will posit a particular multiprocessor network design, whose reliability problems will hopefully be representative of those a real design would encounter.

My prototype network is a 2-dimensional mesh-network multiprocessor shown in Figure 5-1. All signaling within a node is done electrically, and all signaling between nodes is done optically.

The mesh network is interesting, and has been the subject of much study; an optically-implemented mesh network might have some significant advantages over an electrically-implemented one. However, that is not my reason for choosing the mesh for analysis. Rather, the mesh, while simple to analyze, is complex enough to involve the same type of reliability problems and solutions as do the other, more complex topologies discussed in Section 3.7.

5.1 Optical channels

Each of the node-to-node connections shown in Figure 5-1 is a bidirectional pair of optical channels, each composed of a number of one-bit optical links. As mentioned in Section 3.2, each optical link consists of an optical source, an optical receiver, and a transmission medium. The link design for my prototype network is illustrated in Figure 5-2.

Let us first consider the optical transmission medium. As I observed in Section 3.5.5, one need not make hard-and-fast assumptions about the medium to be used.

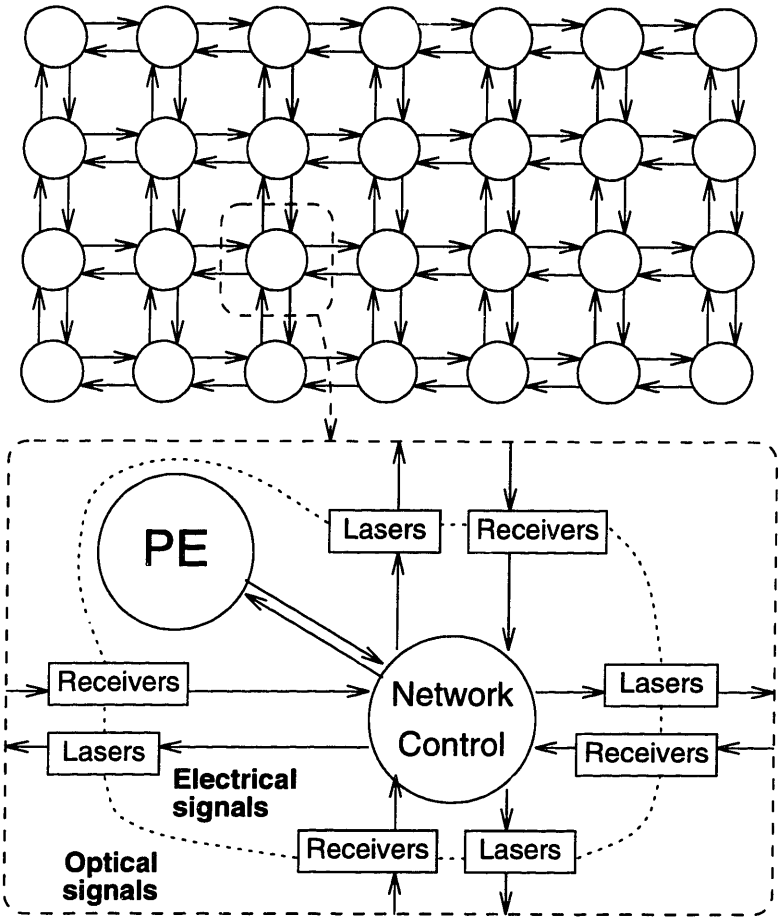

Figure 5-1: Prototype multiprocessor network

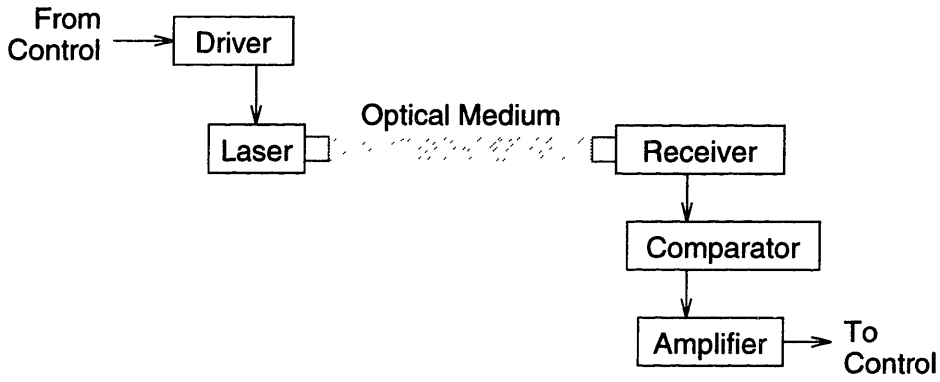

Figure 5-2: Prototype optical link

Any intra-processor links will be at most a few meters long (the distance from one cabinet to another in the same room), and probably on the order of centimeters, or even millimeters. Therefore the dispersion problems which dominate telecom optical-fiber design are completely irrelevant to my design.

However, the speed of light is a constraint which is not so readily overcome, so I will need to say something about the distances involved in order to get an estimate of the transmission delay. In free space, every 30 cm traveled introduces 1 ns of delay; in other media, there is even more delay, proportional to the square root of the dielectric constant of the medium. For my analysis, I will assume a reasonably compact system, such that transmission from one node to the next occurs within 1 ns.

Next, let us consider the data transmission rate. As I noted in Section 3.1, while optical links for telecommunication are now running at multi-Gbit/sec data rates, datacom systems are running at data rates in the low hundreds of Mbit/sec. For my prototype network analysis, I will assume that this situation has progressed enough to allow a signaling rate of 1 Gbit/sec for each laser/receiver link. (This, with the distance constraint, means that each optical transmission will be completed within one bit time.)

Next, we consider the optical channel width, that is, the number of parallel optical links in each channel. Present-day optical datacom systems are dominated by single-bit (that is, one laser and one optical receiver) approaches, since multiple arrays of lasers and optical receivers, while under intensive research and development, are not yet commercially feasible. For my prototype network, I will anticipate the availability of laser and optical-receiver arrays, and assume that each of the optical channels carries multiple bits. For the quantitative analyses and simulations I will assume that each channel carries 64 data bits (plus overhead for error control, signaling, spares, etc.), since 64-bit processors are likely to dominate the high-performance market in the near future and for some time to come.

Next, let us consider the lasers and receivers. I shall not assume any particular laser design, but rather assume different laser reliability characteristics and examine their impact on system design and reliability. For the optical receiver, I shall assume a simple photodiode design, with appropriate decision circuitry on the output.

Next, we consider the optical modulation and reception methods. Since this will

be a massively parallel system, where cost-per-link will be a vital concern, high-performance but technologically challenging approaches such as phase modulation and coherent detection will not be considered. (Soliton transmission, of course, is another challenging approach which is of vital interest to telecom, but it is completely irrelevant to the short-distance datacom links we are considering here, where dispersion is negligible.) I therefore assume simple on-off modulation of the laser, and incoherent detection at the receiver.

As Tsang demonstrates [18], the driver, amplifier, and comparator sections in Figure 5-2 needn't be complex. His data link experiments have demonstrated 1 Gbit/sec transmission using a standard emitter-coupled logic (ECL) gate for the driver, and an ECL flip-flop for both amplifier and comparator. Such circuitry could readily be integrated for use with multiple arrays of lasers and receivers, along with error-control and other circuitry to be discussed in Chapters 6 and 9. (The interesting topic of whether to have a constant laser drive level, or whether to control the drive level to maintain a constant output level, is discussed in Chapter 7.)

To summarize, my prototype network uses optical channels for inter-node communication, transmitting 64 data bits in parallel in both directions (128 data bits in all), plus overhead bits, every nanosecond. Each link is a straightforward on-off modulated laser, passing through a short optical medium to a photodiode, with each bit arriving within one nanosecond.

5.2 Network Nodes

At each node of my prototype mesh network, there is a processing element (PE), that is, a computer processor or processors, memory, and associated circuitry. The question as to how much processing power to place in each node (and therefore having a greater or lesser number of nodes) is a fascinating one, but not directly relevant to my reliability concerns.

For my purposes, the internal workings of a processing element are not considered: it is considered to be an independent, random generator of network traffic. Each processing element, at randomly-determined times, generates a data packet, of randomly-determined size, for transmission to another processing element in the net-

work. The processing element gives the packet to the data network for transmission, and tells the network to which other processing element the data is destined.

Also at each node is a network-control switching circuit which:

- accepts new packets from the processing element, codes them for error control, and sends them toward their destination node, and
- receives packets destined for this node, checks their error-control code, and gives them to the processing element, and
- receives other packets, and forwards them toward their their destination node, and
- (optionally) reconfigures the network to bypass failed optical links.

Extensive analyses of such networks are available elsewhere [25, 34]. In Section 5.3 give a brief outline of such an analysis. (The error-control coding and network reconfiguration will be discussed in Chapters 6 and 9, respectively.)

The processing element expects the network control hardware to route and deliver the data without further help from the processing element, *in most cases.* However, unlike the telecommunications system designer, we are free to blur the data-source vs. data-network boundary, whenever it is to our advantage. We may therefore consider that, conceptually, the processing element controls every aspect of the data network, and the network control logic merely helps it handle the common cases quickly.

Of course, the rate of data transfer is so fast that almost all network operations must in fact be handled by the network control logic, but the 'processor-centric' concept expressed above is valuable when considering how to handle important but comparatively rare events, such as transmission errors or laser failures. Intervention by the processor can provide a powerful, intelligent response to such occurrences, with negligible effect on total system performance because it happens so infrequently. (Note that this concept is merely a more general application of the ideas applied to the cache coherence problem in the Alewife multiprocessor [35].)

5.3 Prototype network operation

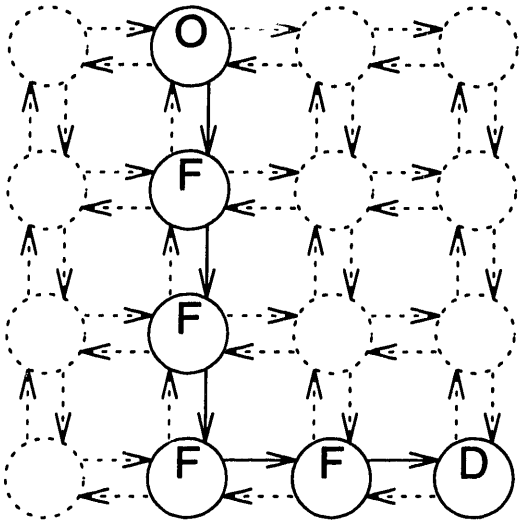

Figure 5-3: Mesh routing example

The mesh network, while providing direct connectivity only between a node and its four nearest neighbors, must be able to provide communication between any two nodes in the network. Obviously, when a message must be passed between two nodes which are not adjacent to each other, it must be forwarded through the intervening nodes. Figure 5-3 shows an example of this, with a packet originating in node 'O', forwarded by several nodes 'F', and arriving at its destination node 'D'.

The traditional approach in such situations is a 'store and forward' network, where each intervening node receives and stores the message before forwarding it on its way. Unfortunately, the store-and-forward approach has a number of problems, such as a potentially unbounded amount of storage required for messages in-transit, given certain patterns of message traffic. Of more direct importance in the present case is the delay incurred in receiving and buffering the entire message before re-transmitting it.

If the transit time from one node to the next is t_t, the time to transmit the entire packet is t_p, and the packet must be transmitted or re-transmitted (forwarded) N_t times, then the time required to store-and-forward the packet to its destination will be $N_t(t_t + t_p)$.

For my prototype simulations, I will use a different method: *wormhole routing* [36]. Rather than store the entirety of each packet in each node, it is immediately sent out as soon as received, as long as the next link in the packet's path is available. (This means that a long message will spread out, like a worm slithering across the network, hence

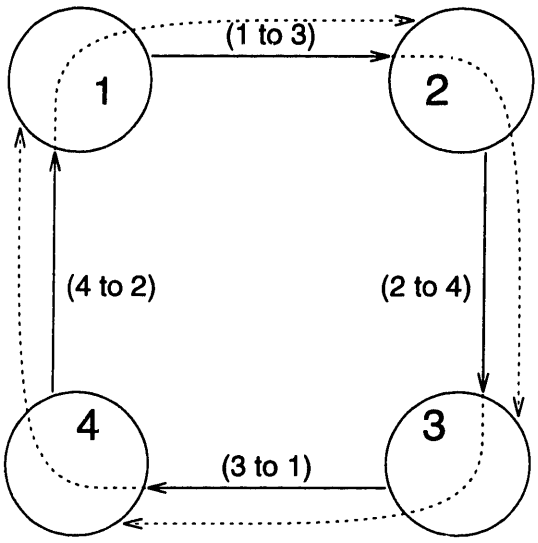

Figure 5-4: Mesh routing deadlock

the name.) If the next link is not available, the message's progress is blocked until it becomes available, and therefore each node needs to provide only a small amount of buffering. With wormhole routing, the message delivery time is reduced to $N_t t_t + t_p$, a considerable speedup for all but the smallest packets.

In such a forwarding network, it is unfortunately very easy to run into a deadlock situation, where different data transfers interact such that each is waiting for the other to complete, with the result that none of them can ever be completed. Figure 5-4 illustrates such a situation. Each of the four nodes is trying to send a two-hop message, has taken the first hop already, and cannot take the second hop until the other transfers are completed, which can never happen.

Fortunately, there is an elegant solution to the deadlock problem in mesh networks[37]. (Now, more sophisticated solutions are also available [25, 38].) If one merely ensures that all packets are routed in the same dimensional order (for example, move in the X-direction first, then in the Y-direction). This makes it impossible to form the cycle of dependencies that is required to produce a deadlock. (In my example, in Figure 5-4, node 1 is blocked by node 2, which is blocked by node 3, which is blocked by node 4, which is blocked by node 1, forming the deadlock cycle. This is possible because nodes 1 and 3 are routing their packets in the order X-then-Y, while nodes 2 and 4 are routing Y-then-X).

I will therefore assume the use of wormhole routing in my prototype mesh network.

This implies that the beginning of each packet contains an address, indicating the destination node, and that each channel has at least one reverse-direction bit for flow control, indicating whether further data can be accepted, or whether the packet must wait for a needed link to become free.

Chapter 6

Error Control Coding

Here, I consider and evaluate solutions to the transient-error reliability problems described in Section 4.1, given the systems context outlined in Section 5. My evaluation will be in the context of a prototype multiprocessor network, so we first describe this network, and my criteria for acceptable failure levels.

In this chapter, I consider the best way to provide error-control coding as a solution to the problem of transient errors. In particular, I compare Error-Correcting Codes with simple Error-Detecting Codes, and conclude that the latter are fully sufficient and much more cost-effective for this application.

6.1 Acceptable error levels

When considering the solution of a reliability problem, one must answer the question, "How reliable a system does one need?" (or the related question "How reliable a system is one willing to pay for?"). In this section, I establish the sort of reliability expectations I will assume for transient errors in conducting our analyses and simulations. A similar discussion of hard-failure expectations will be found in Section 8.1

As mentioned in Section 4, a multiprocessor network will tolerate only a very low undetected bit-error-rate, since one bit in error might crash the multiprocessor program, or worse, introduce a subtle error into the results. However, this stringent criterion only applies to *undetected* errors: a packet which suffers a transient error, when detected, can simply be discarded and retransmitted. The only impact of detected errors is the performance reduction from having to perform the retransmission.

For my analysis, I will assume that the optical network system has a lifetime of ten years (3×10^8 sec), and arbitrarily declare that an average of one undetected transient error during that time is acceptable. For a 1024-node system, with each node having four outgoing channels, each channel having 74 links (64 data + 10 overhead), and each link running at 1 Gbit/sec, we have an expected number of errors, over 10 years, of

$$\text{BER} \cdot 1024 \cdot 4 \cdot 74 \cdot 10^9 \cdot 3 \times 10^8 = 10^{23} \cdot \text{BER}$$

Undetected BER (that is, rate of bit errors which are not detected) of about 10^{-23} or less is therefore required to meet my transient-error-rate criterion.

6.2 Coding Theory

My goal in the area of transient error control is to relax the error-performance requirements on the individual optical links. If one could engineer cost-effective links with Bit-Error-Rates of 10^{-23}, the problem would disappear. Unfortunately, achieving such a BER value is impossible, and approaching it is costly, both in money and in loss of design flexibility.

Fortunately, as in many other applications, coding theory can come to the rescue. An appropriate error-control coding system can theoretically convert an abysmally poor channel into a reliable one. Of course, such a coding scheme is not guaranteed to be simple to design or practical to implement.

6.2.1 Error detection vs. Error correction

The first major decision is whether to use an error-correcting code (ECC), or merely an error-detecting one (EDC) with automatic retransmission request (ARQ) for messages received in error. In a multiprocessor network, there will be four major constraints on an error-control coding strategy:

1. Coding/decoding logic complexity

2. Transmission latency

3. Bandwidth overhead[1]

4. Undetected/miscorrected error rate

We shall see that on balance, based on these criteria, an error-correction coding strategy is less desirable than error-detection coding scheme.

Coding complexity is higher for error-correction coding than for error-detection coding, as can trivially be seen from the fact that error-detection is a subset of error-correction: an error-correction system must provide facilities to detect errors in order to know whether a corrected value is needed. In addition, it must implement a system to calculate the error-corrected value, and the error-correction section of such a system is always considerably more complex than its error-detecting counterpart.

Transmission latency is an important consideration in the choice of error-correction vs. error-detection coding. When a request for retransmission would take a 10^9 bit-transmission times to make the round-trip between receiver and transmitter (as, for example, in a satellite telecommunication application), the ARQ strategy becomes unworkable. However, in a multiprocessor network the transit times are trivially small by comparison, and an error-detection coding & Automatic Retransmission Request strategy is viable.

The coding systems work by ensuring that all possible codewords are different from each other by a specified amount, or distance. The standard metric of codeword difference is the *Hamming Distance*, which is just the number of bit positions which differ between two codewords. The important metric for our consideration here is the *minimum distance*, which we will denote by d_m, of a given code, which is the minimum Hamming distance between any two codewords that the code might generate. When the receiver sees a codeword that the code could not have generated, then it signals that an error has been detected, and either the correction circuitry (under error-correction coding) or a retransmission (under error-detection coding) should provide the correct value.

An error-detection code that will detect e_d errors must have $d_m \geq e_d + 1$. If d_m were less than or equal to e_d, then a set of e_d errors could transform one valid codeword into

[1]That is, how much of the channel is needed for transmitting extra check bits, as opposed to transmitting the actual data.

another, creating an undetectable error at the receiver.

An error-correction coding system, as we have said, contains error detection as a subset. In addition, its correction circuitry, given an invalid codeword c_r from the receiver, chooses the valid codeword c_c which is closest (in Hamming distance) to c_r. In this case, a system to correct ϵ_c errors will require $d_m \geq 2\epsilon_c + 1$, since with $d_m \leq 2\epsilon_c + 1$ a set of ϵ_c errors, while they might be detectable, can alter c_r sufficiently so that the closest valid codeword is not the one transmitted, and the correction circuitry will therefore produce an incorrect result, or miscorrection.

We can generalize the analysis by considering the possibility of coding systems to detect ϵ_d errors and correct ϵ_c errors, with $\epsilon_d > \epsilon_c$. These systems require a minimum distance $d_m \geq \epsilon_c + \epsilon_d + 1$. We see therefore that for a given level of error detection, each additional bit of error correction increases the required minimum codeword distance d_m by one bit.

Both error-correction coding and error-detection coding work by the principle of taking the data which is to be transmitted, let us say it is k bits wide, and generating based on this a set of r check bits. The check bits with the data bits form an $n = k + r$-bit-wide *codeword*, which is transmitted in place of the original data.

In normal operation, error-detection coding will consume less transmission bandwidth than error-correction coding, due to the nature of the two systems. Recall that the coding scheme transmits $n = k + r$ bits of codeword for every k bits of actual data. The coding overhead in bandwidth is therefore r/k. For a given coding scheme and data width k, number of required checkbits r is a strongly increasing function of d_m. Therefore any decision to reduce the amount of error-correction power or the code will reduce d_m, reduce r, and therefore reduce the bandwidth consumed by the coding scheme.

When an error occurs, error-detection coding will require additional bandwidth, and impose additional latency, for the automatic retransmission request and reply. The error-correction coding system will have an advantage in this case. However, if (as we shall see below) the probability of a given codeword being received in error is low, the average performance of the system is governed almost completely by the case where the codeword is received without error. Therefore the error-correction coding system, which imposes extra overhead on *every* transmission (with errors or not) and

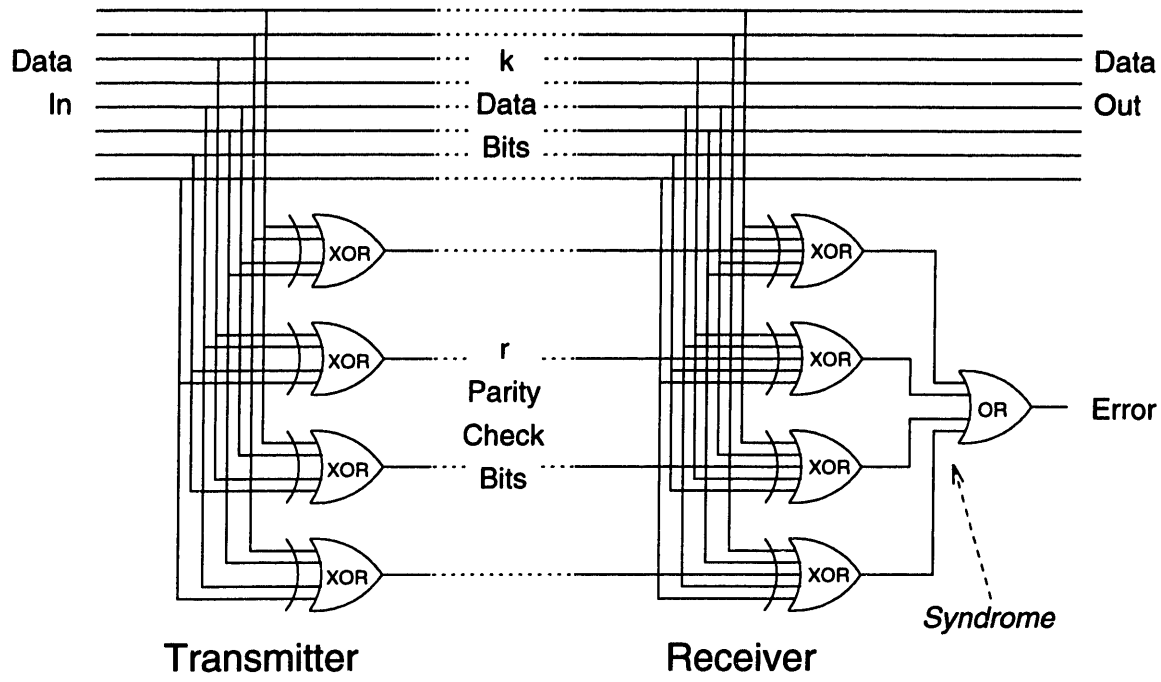

Figure 6-1: Linear block code generation and checking

involves additional logic complexity as well, is simply not appropriate. I will therefore assume the use of an error-detection code with automatic retransmission request on error.

6.2.2 Error-detection coding

There are several excellent references on coding theory and practice available [39, 40], and all we need is a basic understanding of coding techniques, since we are only interested in error detection, and not error correction.

Since the system will require an error-control code that imposes the minimum possible latency on the data transmissions, I shall adopt a *linear block code*, where each of the p check bits is generated by calculating the exclusive-OR of a given subset of the k data bits. This is illustrated in Figure 6-1.

In the receiver, each received check bit is exclusive-OR'ed with the same set of data bits used to generate it. If the word is received with no errors, the set of results from all these calculations (called the *syndrome*) should be zero. If any bit in the syndrome is non-zero, then the received word is declared to be in error, and retransmission of it must be requested.

For our purposes, there are two codes of particular interest: the parity code, and the Hamming code. The parity code is the simplest possible liner block code: merely the exclusive-or of all the data bits. The parity code adds $r = 1$ check bits to the calculation, and has a minimum distance $d_m = 2$. It can therefore detect one error.

The Hamming code is a distance-3 code, and can therefore detect two errors. A Hamming Code with r check bits can handle up to $k \leq 2^r - r - 1$ data bits. A parity check bit can be added to the Hamming Code, forming a distance-4 code which can handle $k \leq 2^{r-1} - r - 1$ data bits.

If we require a transient-error BER, after detection, of 10^{-23} or less, and our actual BER is much greater than this, then our error-detection coding system must have the ability to detect a very large percentage of the errors which do occur. Since the BER calculation in Section 8.1 assumed a codeword width of 74 bits, the equivalent word-error-rate (WER) requirement is $10^{-23} \cdot 74 \approx 7 \times 10^{-22}$, that is, of every 7×10^{-22} words received, we can afford to pass, on average, only one word with undetected errors.

Given that each data and check bit is transmitted and received separately, by different transmitters and receivers, we will assume that transient errors on different bits in the same word are independent. Therefore, the number of errors occurring in a codeword will be equivalent to the number of arrivals in a Bernoulli process, over $n = k+r$ trials. Therefore, if we denote the probability that an individual bit is received in error as p_e (note that p_e=BER), and let e_w denote the number of errors occurring in a codeword of n bits, then assuming $p_e = \text{BER} \ll 1$, the probability mass function of e_w is given by

$$p(e_w) = \binom{n}{e_w}(1-p_e)^{n-e_w}p_e^{e_w} \approx \binom{n}{e_w}p_e^{e_w} \tag{6.1}$$

Therefore, if we implement a coding system that can detect e_d errors in a codeword, the undetected WER will be $\sum_{e_w=e_d+1}^{\infty} p(e_w)$. Setting WER= 7×10^{-22}, and the data width $k = 66$ (64 data bits + 1 flow control bit + 1 error signal bit), the relations between d_m, e_d, r, and and p_e, for the codes we have discussed earlier, are shown in Table 6.1.

We therefore see that a simple code such as the distance-4 Hamming code (Hamming + parity) can relax the raw BER requirement of the optical link from 10^{-23} to more than than 10^{-7}, at a very modest bandwidth overhead of $r/k = 8/66 = 12\%$.

Let us now test the hypothesis that the retransmission overhead due to using

Type of Code	Min. Dist. (d_m)	Detected errors (e_d)	Check Bits (r)	Max. (link) WER	Max. BER (p_e)
None	1	0	0	7×10^{-22}	10^{-23}
Parity	2	1	1	10^{-11}	6×10^{-13}
Hamming	3	2	7	2×10^{-7}	2×10^{-9}
Hamming & Parity	4	3	8	10^{-5}	2×10^{-7}

Table 6.1: Error-detection code characteristics

error-detection coding (instead of error-correction) is negligible. Supposing the optical links were operating at the relaxed BER$\approx 10^{-7}$ allowed by the distance-4 Hamming error-detection code, we see that the raw Word Error Rate (before detection) would be 7×10^{-6}. Supposing that regular transmission of a codeword takes one cycle, error-correction code correction takes 1 additional cycle, and error-detection automatic retransmission takes 25 cycles (as might be the case for a heavily pipelined system), we have an error-correction calculation overhead of $1 \cdot 7 \times 10^{-6} = 0.00001\%$, and an error-detection retransmission overhead of $25 \cdot 7 \times 10^{-6} = 0.0002\%$. We see that in both cases, the performance overhead due to errors is trivial, and our hypothesis is validated.

6.2.3 Error-Detection Coding system implementation

In Figure 6-1, we have already given the basic form of implementation for our error-detection coding system. The complexity of the circuit is minimal, and the latency will depend on the number of inputs required to the exclusive-or gates at both receiver and transmitter.

Fortunately, the codes we are using are *linear*, that is, the n-element codeword C can be considered as being the product of a k-element input data vector D and a $k \times n$ code generator matrix G: $C = DG$. The receiver checks the received codeword $\hat{C}$ by multiplication with a parity-check matrix H, resulting in a syndrome which will be checked for the equality $\hat{C}H = 0$.

Since both the parity generation and checking are linear processes, they can be reconfigured by standard linear transformations. Hsiao [41] has examined such transfor-

mations on the distance-4 Hamming code, and has found its optimal minimum-weight form (that is, the form requiring the fewest exclusive-or inputs).

With a data width $k = 66$, and $r = 8$ parity check bits, the Hsiao code has a total weight of 226, or an average of 28.25 inputs to the parity-generating exclusive-or gates (6 28-input gates, and 2 29-input gates). This could be implemented in 5 levels of 2-input exclusive-or gates, in one or two 1-ns clock cycles.

6.3 Error Diagnosis

The laser drive control scheme I will describe in the following chapter assumes a knowledge of error rates for each of the optical links. The fault tolerance schemes outlined in Chapters 8 and 9 assume that failures can be easily localized and identified. Fortunately, such diagnostic tasks are straightforwardly implemented in the error-detection coding scheme I have proposed here.

6.3.1 Transient Error Diagnosis

In Figure 6-1, recall that the error-checking circuit generates a *syndrome* as a part of the error detection algorithm. In the case of the Hsiao code we considered above, the syndrome is eight bits wide. If all the syndrome bits are zero, then no (detectable) error has occurred; if the syndrome is non-zero, an error has been detected.

Each possible one-bit error has a unique syndrome associated with it. A simple table lookup could therefore identify the bit in error, if there were only one. (This is, in fact, the fundamental basis of error-correction coding schemes.) This lookup might be done in software, if error rates were low enough, or with specialized hardware, if it proved to be too great a load on the processors.

We could therefore diagnose the offending bit in each single-bit error word received, and use that information in the control schemes described in the following chapters. The rare multi-bit errors would be noted, but not otherwise diagnosed.

Would we therefore be implementing error-correction coding after all? No, because we are not actually relying on the error diagnosis to protect data integrity. Since the error diagnosis would only be used for statistical purposes, we have the luxury of occasionally omitting the diagnosis (in the rare case of 2-bit errors) or tolerating a few

misdiagnoses (in the rarer case of 3-bit errors). Also, the error diagnosis is not in the chain of data transmission, so it can be done with very little constraint on the amount of time required to perform the diagnosis.

6.3.2 Hard Failure Diagnosis

A hard failure will be detected quickly. It will first be detected by the error-detection circuitry, which will request a retransmission. The retransmission will also be in error, so another will be requested, which will be in error, and so forth. After some number of retransmissions (perhaps 3 or 4), the system would flag a possible error. Since hard failure is a rare event, its diagnosis can be handled completely in software, with negligible performance impact.

The syndrome diagnosis described in the previous section would point out the likely culprit, and transmission of a few test patterns could easily confirm the diagnosis. The failure diagnosis would then be acted on by one of the algorithms described in Chapters 8 and 9.

6.4 Summary

The multiprocessor network I described in Chapter 5 would require an undetected bit-error-rate of 10^{-23} or less. This is completely impractical to realize by direct implementation, but it can be achieved using error control coding, with moderately reliable communication links.

For this application, error detection coding (EDC) is more appropriate than error correction coding (ECC). Assuming independent errors, a Hsiao error detection code can relax the bit-error-rate requirement to around 10^{-7}, with only 12% parity-bit overhead and a pipeline delay of one or two cycles. Such a system can also diagnose transient errors and hard failures, for the use of laser control and fault tolerance systems.

Chapter 7

Intelligent Laser Drive Control

The error-control strategy proposed in the preceding chapter, in addition to directly solving the transient-error problem, can facilitate the solution of another system problem which at first glance might seem to be unrelated: the problem of laser drive level control.

My proposal for solving this problem is Intelligent (that is, software-based)[1] Laser Drive Control. Instead of presently-used methods which need monitor photodiodes, Intelligent Laser Drive Control uses the error rate itself (derived from the error detection system) to control the laser drive levels.

In addition to solving the drive control problem this can provide additional benefits, including detection of and compensation for optical medium degradation, and tracking of laser wearout trends to allow optimal repair and replacement planning.

7.1 The Laser Drive Problem

To transmit data via a laser diode, one may employ any of a number of modulation methods, such as frequency, phase, or amplitude modulation. For this thesis research, I am using one of the simplest possible (and most widely used) laser modulation methods: on-off signaling.

Figure 7-1 gives the typical light output L vs. drive current I relationship for a

[1]By using the term 'intelligent', I do not mean to imply that the software-based feedback scheme need be complex, much less that it include Artificial Intelligence. Rather, I wish to emphasize that a software-based scheme gives one the flexibility to tailor the feedback algorithm to have as much or as little sophistication as the situation requires.

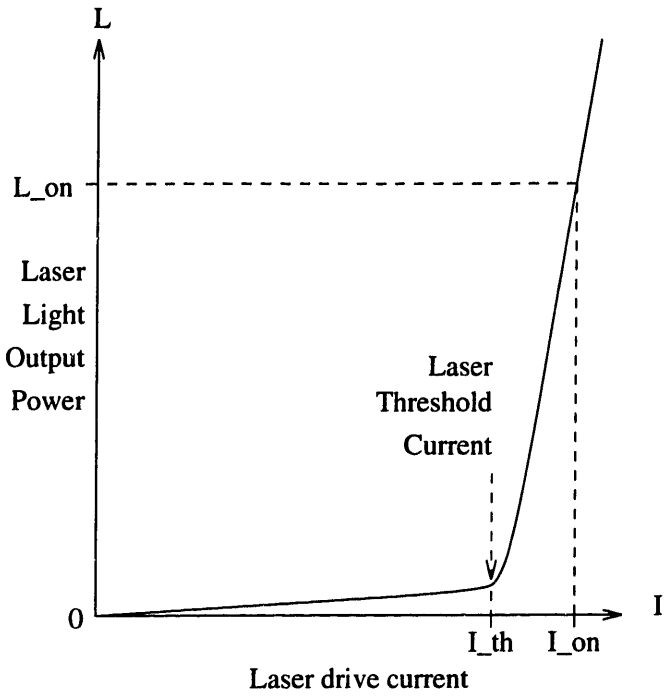

Figure 7-1: Semiconductor laser output power vs. drive current

semiconductor laser. To achieve a desired light output power level L_{on} (governed by the receiver characteristics and the Bit-Error-Rate requirements), the system must apply a corresponding drive current I_{on} for the 'on' bit signal, and remove the drive current for the 'off' bit signal.

However, the 'off' current I_{off} cannot usually be set to zero, since the laser output will take some considerable time to increase from zero to the level corresponding to I_{th}, the onset of lasing. (The laser operates as a light-emitting diode in the $0 \leq I \leq I_{\text{th}}$ range of drive currents.) High-speed operation therefore does not allow I_{off} to be set much lower than the laser threshold current I(th). Additionally, I_{on} must be set sufficiently above I_{th} to produce enough light output, since $L \propto I_{\text{on}} - I_{\text{th}}$.

Figure 7-2 shows a conceptual design for a laser drive circuit which is used to meet these constraints. The current source I_{bias} is constantly applied to the laser, and the current source I_{pulse} is switched through the laser or not, depending on the input data. We therefore have

$$I_{\text{off}} = I_{\text{bias}} \tag{7.1}$$

$$I_{\text{on}} = I_{\text{bias}} + I_{\text{pulse}} \tag{7.2}$$

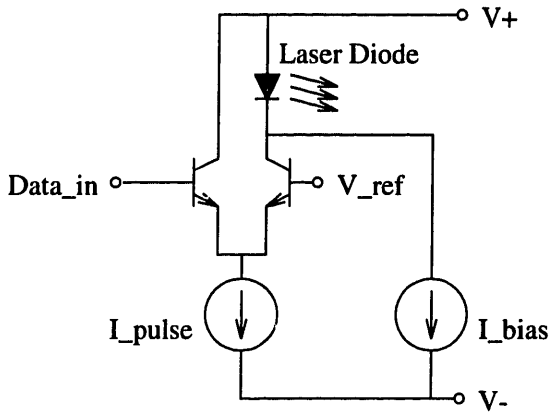

Figure 7-2: Semiconductor laser driver circuit

The problem we address here is that of properly controlling of I_{bias} and I_{pulse}.

As we learned in the analysis of laser wearout failures in Section 4.2.1, such failures can be modeled as a gradual increase in laser threshold current over time. This will result in a decrease in laser light output unless the laser drive circuit compensates for it by increasing the drive current.

There is a also more immediate laser-threshold-current problem: temperature dependence. Laser threshold current increases exponentially with temperature, so the drive current must be compensated for this as well, or else the drive level must be set high enough for the hottest laser operating temperature (and oldest laser age) that will be encountered. Of course, the high drive level may ensure that the laser actually does reach a high temperature!

7.2 Conventional Solutions

Laser-diode-based systems are in wide use now, in spite of the laser threshold current variation problem. A number of approaches to solving the problem are in use now, and are described below, but none of them seems completely satisfactory for the multiprocessor optical networks envisioned in this thesis.

7.2.1 Fixed laser drive

The simplest approach, shown in Figure 7-3 to these problems is to fix I_{bias} and I_{pulse} for acceptable performance over temperature and laser lifetime. This can

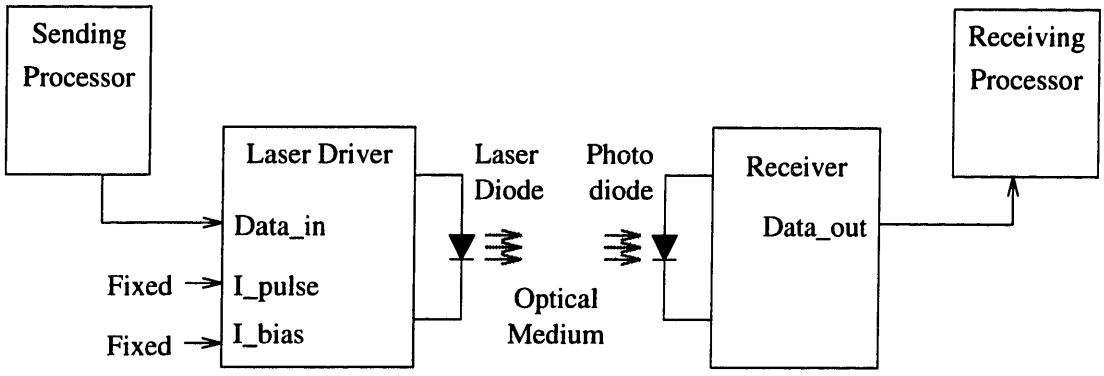

Figure 7-3: Fixed laser drive

involve significantly higher drive levels than would otherwise be required. High drive levels, of themselves, shorten laser lifetime, and they contribute to higher operating temperatures, which shorten all component lifetimes. Also, this means that at low threshold currents (that is, at low temperature and/or using new lasers) the lasers may not be completely 'off' at I(bias), degrading the laser output extinction ratio, that is, the ratio between 'on' and 'off' light levels.

7.2.2 Analog-feedback laser drive control

Figure 7-4 shows the approach conventionally used in telecommunications[42]. A small photodetector is added to the laser package, detecting some of the light leaked from the rear facet of the laser (or tapped from the laser output), and producing a photocurrent proportional to it. This current is used to control I_{bias}, by implementing an analog feedback loop to keep the laser output level constant. I_{drive} is usually kept fixed, but some of the more elaborate systems vary it as well.

[In telecommunications, the control system is often more complex: in addition to the light-monitor laser drive control loop, a temperature-control feedback loop is implemented as well. A temperature sensor is used for feedback control of a Peltier-effect thermoelectric cooler, to help keep the laser temperature constant. Even with this addition, the light-monitor control of laser drive levels is usually required.]

Unfortunately, this approach does not transfer well to the wide-path laser-array interconnects that my thesis anticipates. Implementing monitor diodes on such laser arrays adds a further complication to an already difficult optoelectronic device design. Even with the monitor photodiodes, proper implementation of the required analog

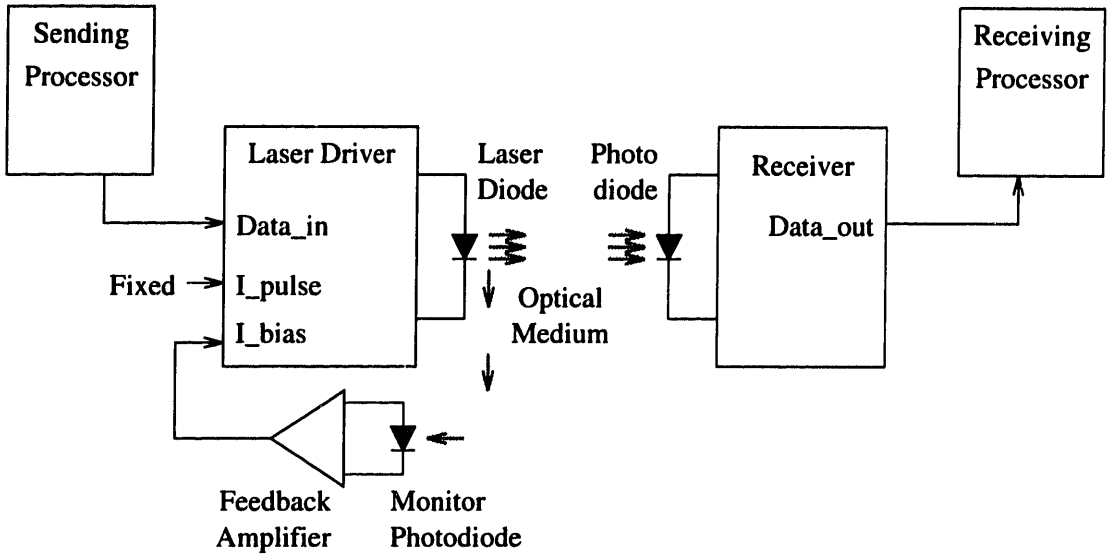

Figure 7-4: Analog-feedback laser drive control

feedback loops would be another stumbling block.

7.2.3 Very-low-threshold lasers

Because of these problems, some researchers in this field have restricted themselves to laser designs with very low threshold currents, and therefore with weaker dependencies on temperature and age, to enable them to use fixed-current drivers. This is a very sensible approach, provided that lasers with suitably low threshold currents are available.

A usable laser-drive-control system may be able to widen the field of potential laser array designs to include higher-threshold lasers. The criterion of 'suitably low threshold' would then become part of an engineering tradeoff: very-low-threshold designs using fixed drive, versus moderately-low-threshold designs using laser drive control. Even very-low-threshold designs may benefit from drive control, as it may extend their utility by restricting laser power to the minimum required, thereby reducing system power and increasing component lifetimes.

The drive control scheme I propose in the next section also gives other benefits, such as laser wearout tracking and optical medium degradation detection and compensation. These benefits might make it worthwhile to include drive control even in very-low-threshold laser designs that did not strictly require it.

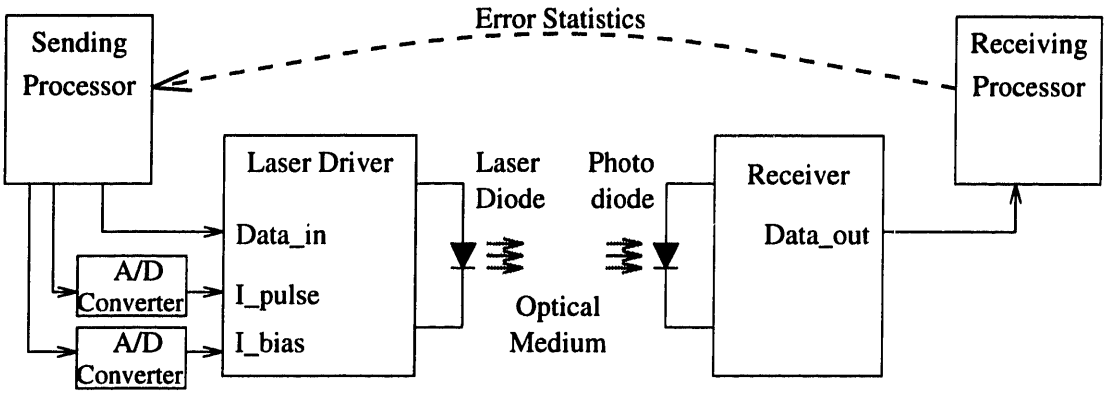

Figure 7-5: Intelligent laser drive control

7.3 Intelligent laser drive control

The new idea I propose here is to use the actual link error performance, rather than the light output, to control the laser drive levels: in the final analysis, it is the *performance* of a data link that counts, rather than the intensity of the light used to implement it.

The lowest-possible error rate, which most communications-link designers have sought, is not really required for our multiprocessor optical network. As I have already shown, a network with error-detection and automatic retransmission request can tolerate a bit-error-rate on the order of 10^{-8} or 10^{-9}, and still achieve the desired system-level data integrity.

It therefore might be possible to use the bit-error-rate, rather than a direct measurement of laser output power, to control the laser drive, since we can tolerate short periods of elevated error rates. [If one is to use the BER as a feedback control variable, one must allow it to fluctuate somewhat].

Figure 7-5 shows an intelligent (software-based) laser drive control loop. The entire control loop is digital, except for the laser driver itself and the Digital-to-Analog Converters (DAC's) controlling it. The addition of two digital-to-analog converters per laser might seem a significant increase in complexity, but this application is particularly undemanding for the digital-to-analog converters, requiring neither high speed, nor high precision, nor high accuracy. (The question of just how much precision might be required is investigated in Section 7.4.7.) They can therefore be easily implemented *en masse* in silicon very-large-scale integration (VLSI) integrated circuits for all the lasers in a particular data channel.

A fundamentally important aspect of this approach is that it is performed in software. This allows much more flexibility in the control algorithm than would be available in a simple analog control loop. For example, it can easily control two parameters (I_{bias} and I_{pulse}) instead of one. It can also record past values, enabling the system to perceive anomalies such as optical medium degradation and long-term trends such as laser threshold increase due to aging.

7.4 Laser drive control experiments

To test the feasibility of my intelligent laser drive control concept, I have implemented it on an experimental 1 Gbit/sec free-space optical data link. I conducted experiments to determine feedback system's performance with

- temperature variation, and
- optical loss variation.

The temperature variation would be a real problem in itself, and it is also a surrogate for laser aging, since a hot laser is similar to an old laser: both have elevated threshold currents. The optical loss variation could occur from medium degradation, or from age-related reduction in laser efficiency.

7.4.1 Experimental Setup

The experimental setup is shown in Figure 7-6, and a general outline of it is given here. Details of the experimental setup are given in Appendix A.

A laser transmitter board and an optical receiver board are mounted on a light table, at a distance of 35 cm. The laser board holds a laser (with a monitor photodiode), lens, laser driver, and a temperature sensor (mounted in an aluminum block with the laser). The receiver board holds a receiver photodiode, lens, and amplifier.

A data-link analysis system made by Gazelle sends a 1 Gbit/sec data stream to the laser driver, and receives data from the optical receiver. It compares the two and reports the error results via a low-speed serial interface. The Gazelle system usually communicates with a console terminal, but in this experiment it is connected to a Sun UNIX workstation instead.

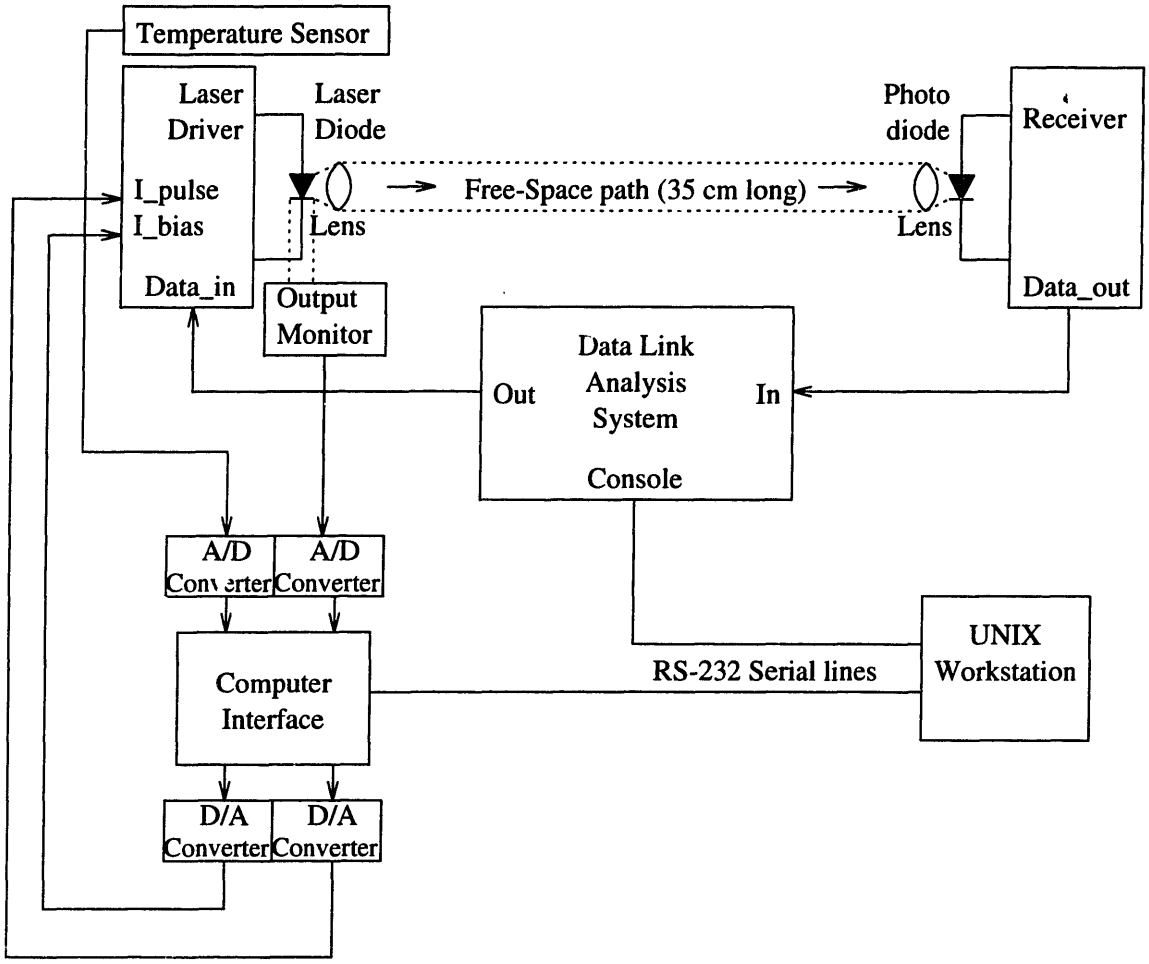

Figure 7-6: Intelligent laser drive experimental setup

A custom-made interface system communicates with the same UNIX workstation, also through a serial link. Via the interface, the workstation can control bias and pulse currents in the laser driver, and it can monitor the laser light output level and the laser case temperature.

Figure 7-7 is a photograph of the overall experimental setup. The photograph includes a later addition to the experimental step: an adjustable iris which can block part of the free-space optical path. The optical components are mounted on a benchtop vibration-isolated optical table. From left to right they are: photodiode/receiver board, optical iris, and laser/driver board. Next to the optical table, one can see the computer interface box and the data link analysis board.

7.4.2 Feedback Program

The UNIX workstation runs a program which implements the actual feedback loop. It receives the bit-error-rate (BER) results from the data-link analysis system and controls the laser bias and pulse currents based on the BER results. The program logs the experiment's progress in a disk file, recording BER, bias and pulse current, light output level, and temperature. [Note, however, that the temperature and light output level measurements are strictly for off-line analysis of the experimental results, and are not used in the feedback system.]

The program's feedback algorithm is a simple-minded one, and could almost certainly be improved with the benefit of more experience. The algorithm assumes that the open-loop response of the system (that is, drive current — BER) is monotonically decreasing, allowing the use of a simple rule:

- Low error rate —— decrease drive current
- High error rate ——→ increase drive current

The question of *which* drive current (bias or pulse) to adjust is an interesting one, which my feedback algorithm avoids by treating the two currents equally, as much as possible.

The workstation program is given a BER goal, and increases the laser drive currents whenever the observed BER is worse than the goal. When the observed BER is better than the goal, the program tests to see if it can safely reduce the drive or bias current

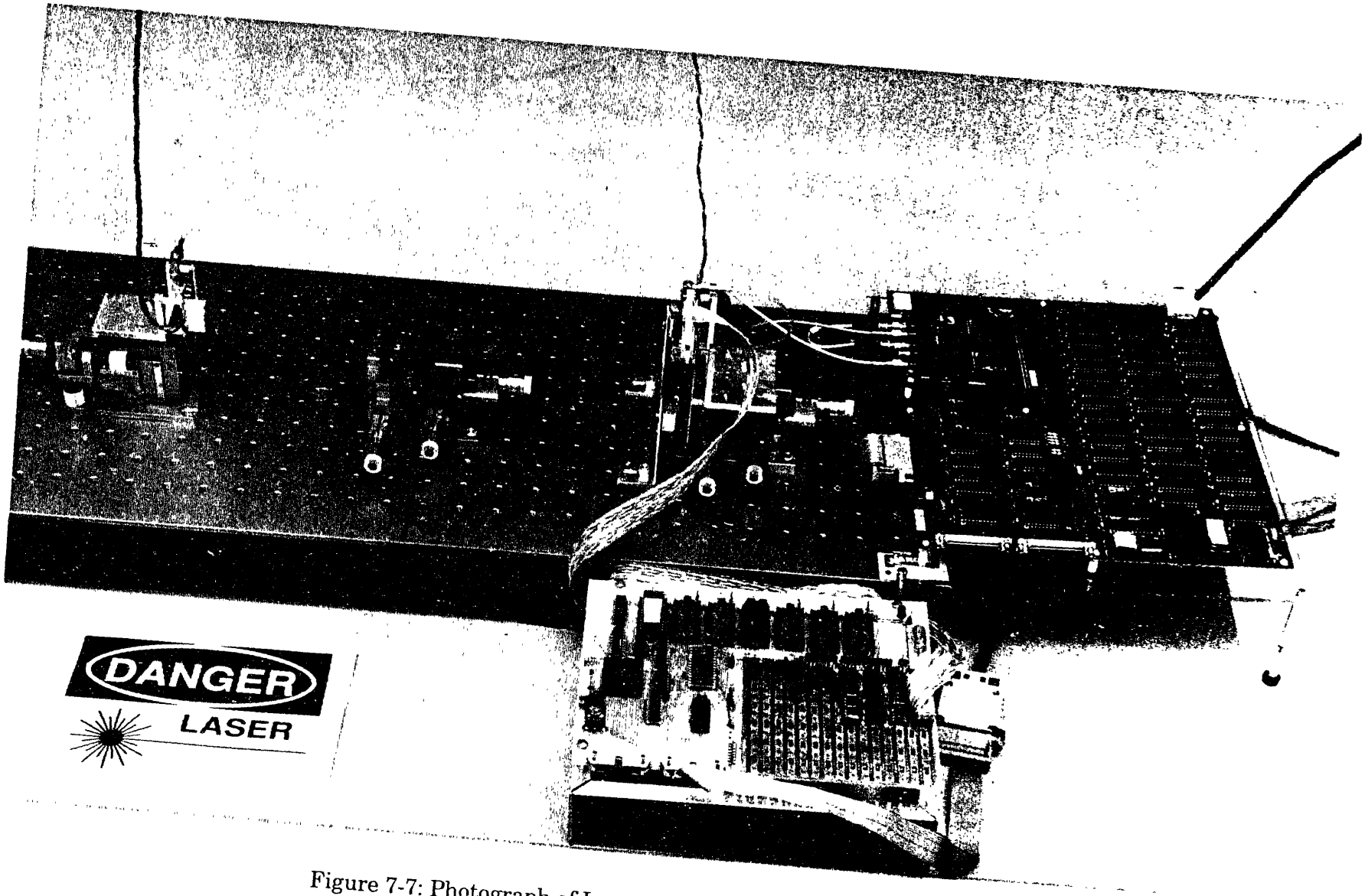

Figure 7-7: Photograph of Laser Drive Control Experimental Setup

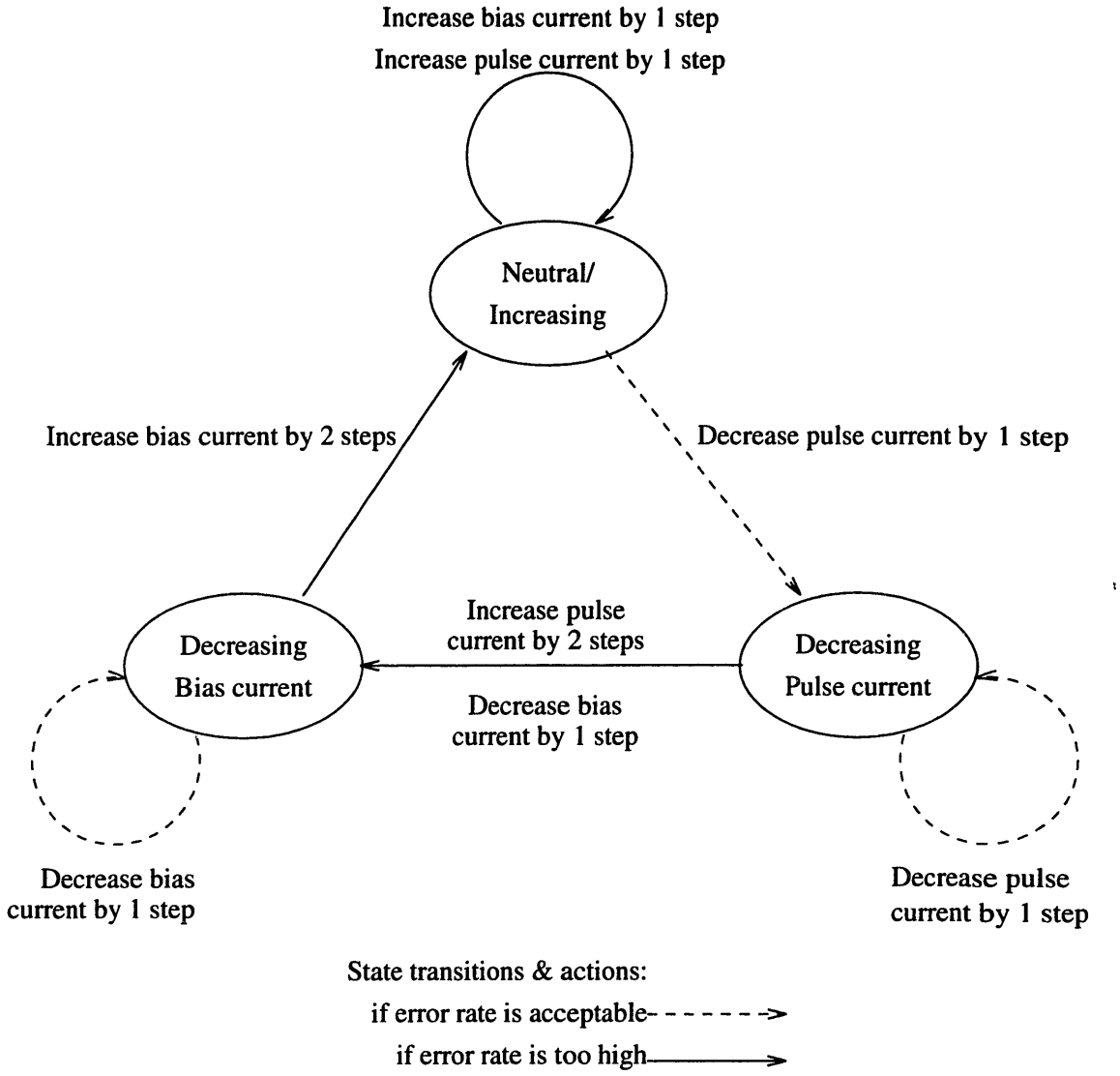

Figure 7-8: Laser Drive Control software state diagram

without increasing the BER to above the goal. The algorithm is controlled by a simple finite state machine, whose state transition diagram is given in Figure 7-8. The bias current step is 0.12mA, and the pulse current step is 0.07mA: in both cases approximately 1% of the normal operating-point value.

When faced with an excessively high error rate, a sophisticated feedback program might try to determine which drive current to increase. Alternatively, it could assume (as is most likely) that only the bias current need be increased, and only increase the pulse current if the bias current increase did not work.

As can be seen from Figure 7-8, my feedback program blindly increases both bias

and pulse drive currents until a satisfactory error rate is achieved. This will almost certainly increase one of the currents (bias or pulse) more than actually required. The program then reduces the pulse drive current until the error rate rises again, and then backtracks a small amount. The bias current is then reduced similarly. These reduction mechanisms should eventually eliminate the unneeded drive current produced by the drive-increase code.

7.4.3 Temperature experiments

It would be impractical to observe the system's control of laser wearout problems, since such wearout occurs over the course of months and years. However, a similar problem occurs much more quickly: temperature variation of laser output. I therefore observed the laser drive control system's ability to deal with the temperature variation of laser threshold current, both as a problem in its own right and because of its similarity to laser wearout.

I put an electric heating element next to the laser board, and controlled the heat output in order to vary the laser case temperature between room temperature and 40° C. First, I temperature-cycled the system with the laser bias and pulse currents fixed, then enabled the feedback program (with a BER goal of 2.5×10^{-10}) on the UNIX workstation and repeated the experiment.

The results from both experiments are shown in Figure 7-9. With fixed drive, we see a 50% reduction in light output at higher temperatures, due to increased laser threshold current; this reduction in light output is accompanied by a corresponding increase in bit-error-rate to over 10^{-3}.

With feedback enabled, even though the light output monitor values played absolutely no part in the feedback loop, and the laser temperature rose even higher than in the fixed-drive experiment, the change in laser light output over temperature is almost imperceptible. The bit-error-rate is also well-controlled, to less than 10^{-9}, aside from a 4×10^{-7} value at one data point[2].

[2]This is most likely due to the artificially rapid temperature rise, and the experimental setup's slow response time because of the low-speed interface with the data link analysis board, which was not designed for this type of application. In an actual network, error rates (when they are *high*) would be available very quickly.

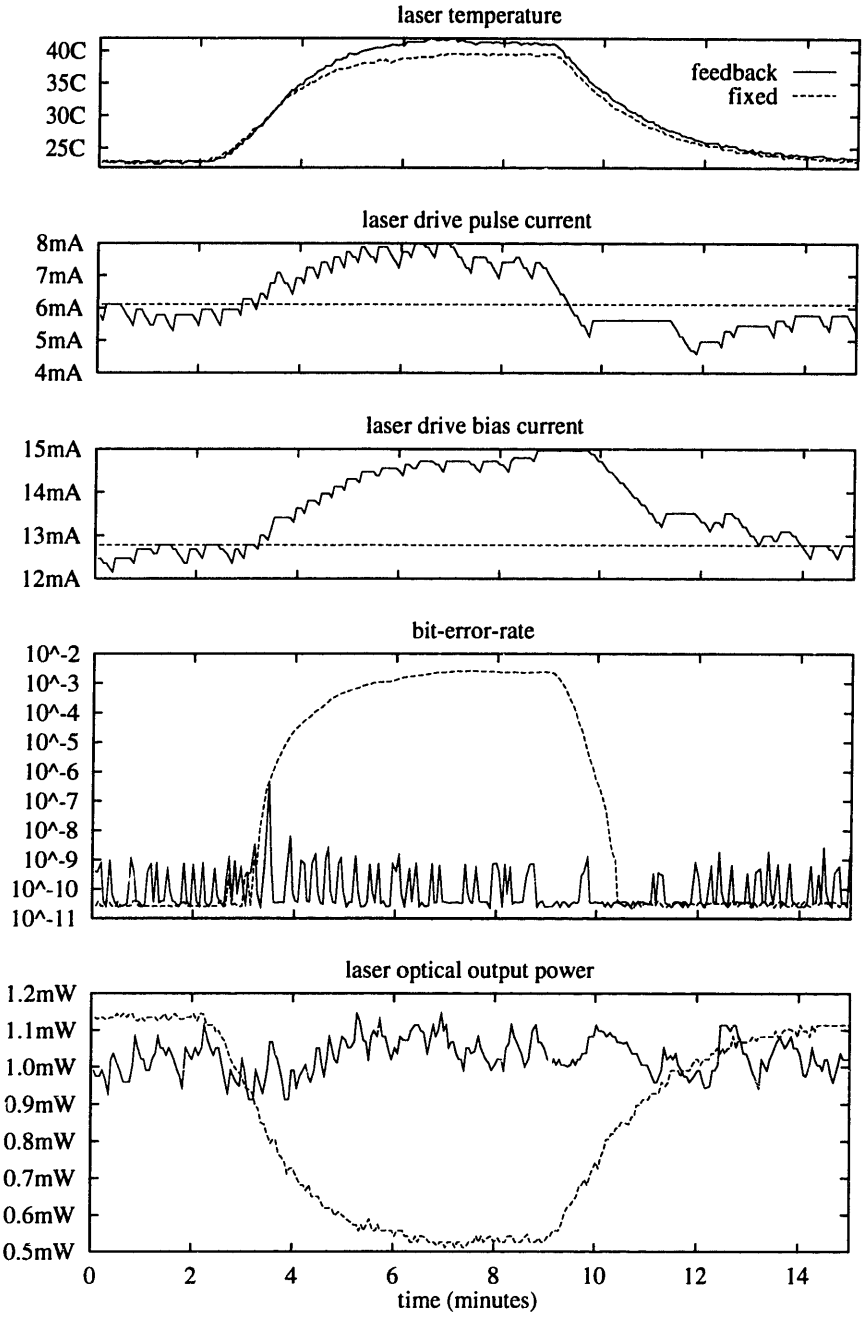

Figure 7-9: Laser Drive Control over temperature

We see from this result that even a very simple BER-based feedback algorithm can adequately control the laser drive level over changes in laser threshold current.

7.4.4 Optical loss experiments

I then examined a problem for which the conventional approach (in Figure 7-4) would be of no use: optical medium degradation. It could easily happen that the optical medium between laser and receiver could experience an increased optical loss, due to aging, misalignment, damage, or other causes. Since my approach includes the entire communication link in the feedback loop, it should be able to detect and compensate for such a problem, provided that sufficient laser output were available.

I modified the setup in Figure 7-6 to include an adjustable iris in the free-space light path. When open, the iris had no effect on the light beam. When closed, the iris blocked almost all of the beam, allowing only 2.6% of the light to pass.

I first tried the iris with the laser drive feedback control disabled, and then repeated the experiment with the feedback program running, as before. Figure 7-10 shows the results. When the iris is closed, in both cases the BER immediately increases to more than 10^{-6}. In the fixed-drive case, the BER remains at that level, while in the feedback case it almost as quickly starts falling, as the feedback loop increases the laser drive level.

The effect of my simple-minded feedback algorithm can be seen between time $t = 2$ and $t = 6$ minutes, where the bias current has been increased in lock-step with the pulse current. By time $t = 7$ minutes, the bias has been reduced back to its previous value, while the pulse current remains elevated, as one would expect. After the iris is opened, the pulse current subsides to somewhat higher than its previous level, with the bias somewhat lower. (This is an alternative operating point with approximately the same BER at the one at $t = 0$. Either point is acceptable.)

The laser optical output power readings provided a clue to a shortcoming in the experimental setup. Note that in the fixed-drive case, the power reading falls when the iris is closed, and rises when it is opened again. This cannot have any basis in reality, since (with the drive level fixed) the iris can have no effect on the laser output.

Puzzled by this incongruity, I eventually discovered that the lenses I used for the free-space optical path had been specified incorrectly, and had an anti-reflective (AR)

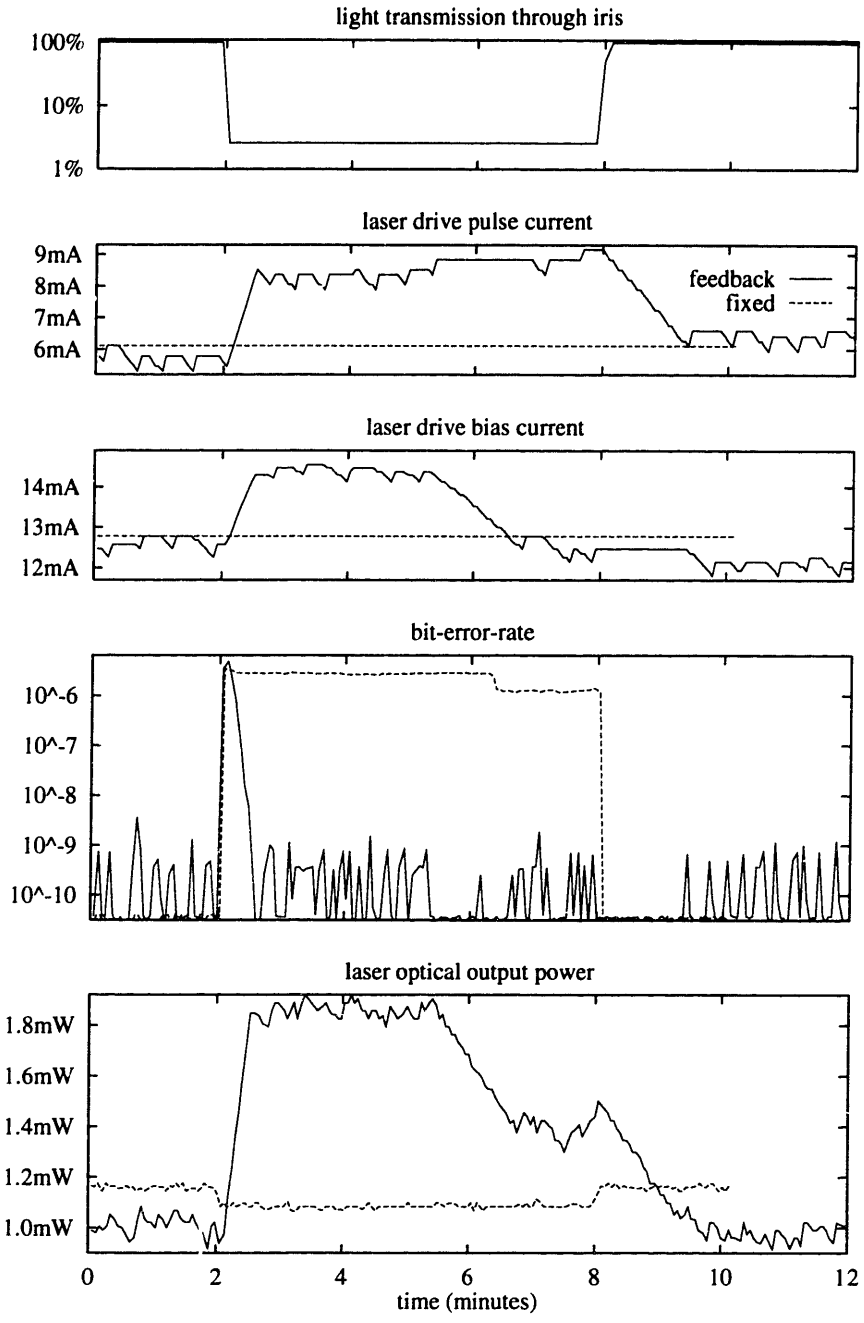

Figure 7-10: Laser Drive Control with optical loss

coating designed for different wavelength from the one I was using. The receiver board lens was reflecting some of the incident light back to the laser, and this reflected light was being detected by the monitor photodiode (behind the rear facet of the laser).

One may also note that, although the optical transmission over the free-space path was reduced by a factor of 38, the laser output power is only increased by about 50%. The most likely explanation for this is that the normal bias point found by the feedback system has non-zero optical output power even in the 'off' state, which the monitor photodiode detects in addition to the switched optical power used to transmit data. This would tend to obscure the actual increase in switched optical power.

We see that the intelligent laser drive control system can also maintain operation in spite of optical medium degradation. Since the feedback is in software, a periodic diagnostic program could easily detect the significant increase in pulse current, indicative of such a degradation, and inform the operator so that the need for repair could be evaluated, without the urgency of an actual failure.

7.4.5 Feedback Reduction Order

I next was curious about the one asymmetry between bias and pulse drive current in Figure 7-8: pulse current is reduced first, and then bias current. I reconfigured the software to reverse this situation: reducing bias current first, followed by pulse current. I repeated both the temperature and optical loss experiments in this new configuration.

Figure 7-11 shows the results of the temperature experiment, compared with those from the original configuration. Figure 7-12 is the corresponding plot of optical-loss results.

In both experiments, we see that the new configuration works as well as the original one. The major difference, as one might expect, is that excessive bias current is reduced more quickly in the new configuration, and excessive pulse current is reduced more quickly in the original configuration.

The new (bias-first) configuration is shown to best advantage in the optical-loss experiment, since that involves a significant excess bias current after iris closure. The original configuration deals better with the temperature experiment, since that involves an excessive pulse current after the initial heating.

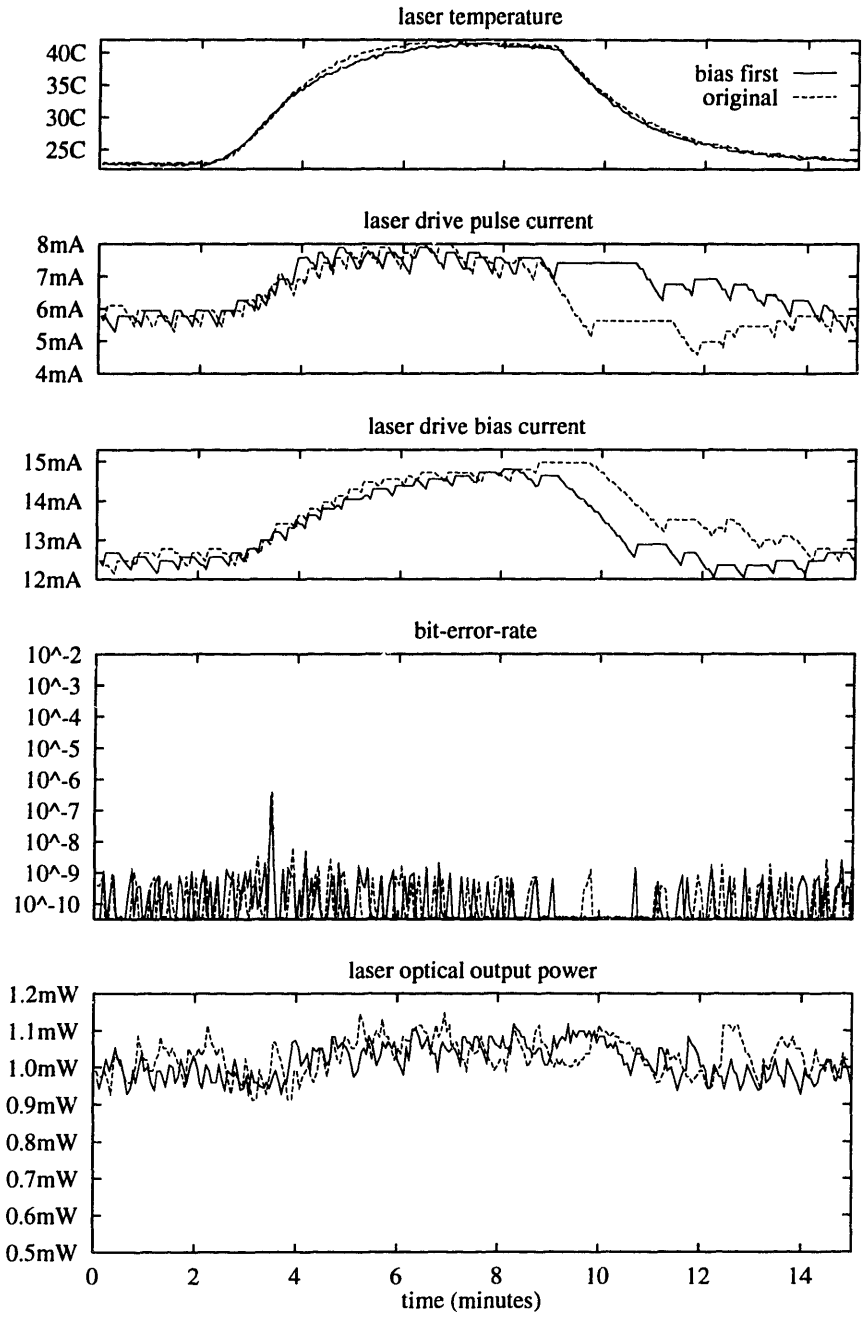

Figure 7-11: Feedback Reduction Order (temperature experiment)

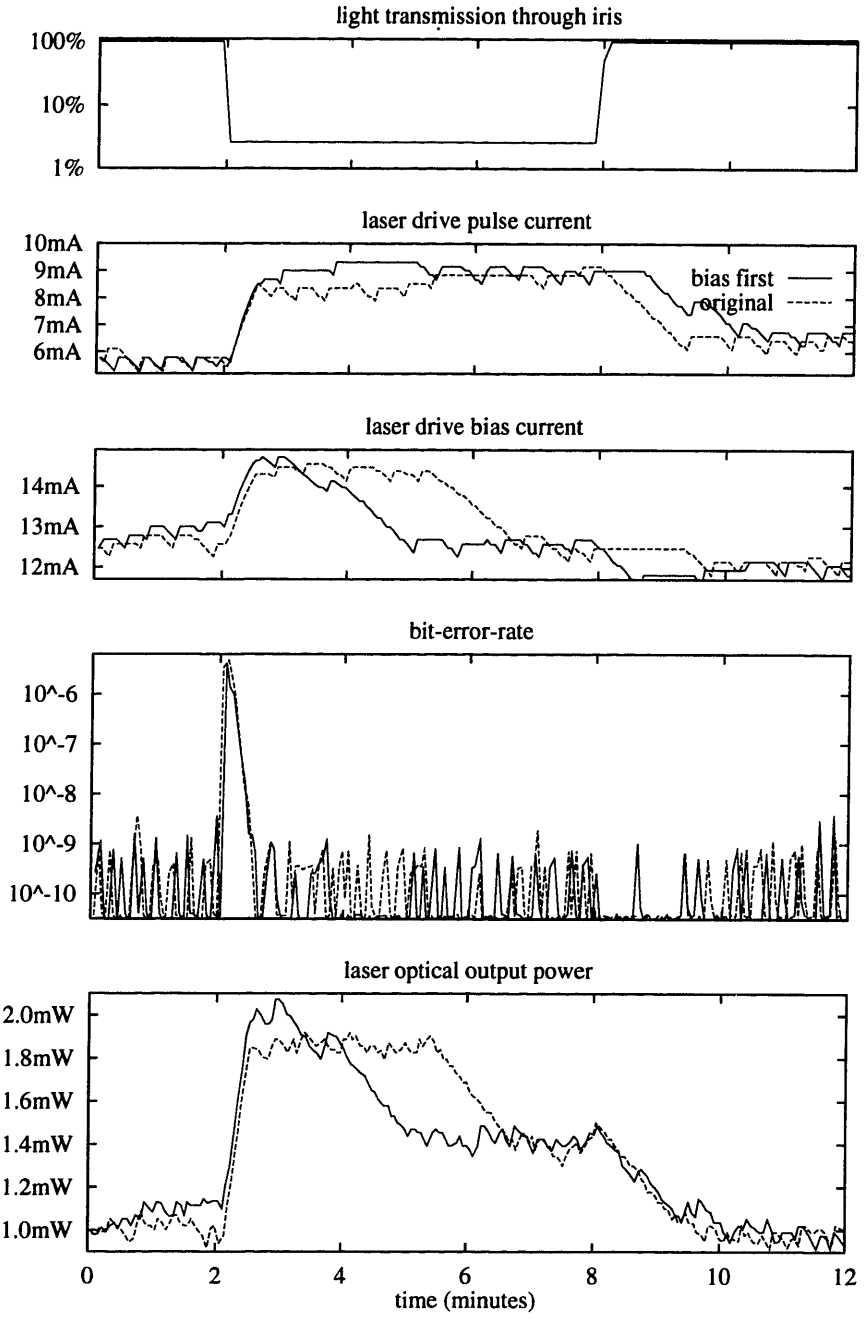

Figure 7-12: Feedback Reduction Order (optical loss experiment)

Since threshold current variation is the more likely problem, it would seem that the original state-diagram configuration (in Figure 7-8) is the better one, if one continued to use my simple-minded control algorithm. However, either configuration would seem to be adequate.

7.4.6 Bias-Only Drive Control

As I indicated in Section 7.2.2, telecommunications systems generally include control of bias current only, keeping pulse current fixed. To test its performance in this mode of operation, I reconfigured the feedback program to keep the pulse drive current fixed, and vary only the bias current.

I repeated the temperature experiment with this new feedback configuration. Figure 7-13 shows the results, superimposed on the results from the full feedback configuration. We see that the bias-only control does an equally good job. This might be expected, from the practice in telecommunications links.

I then repeated the optical loss experiment in the bias-only feedback configuration. Figure 7-14 shows the results. We see that the feedback initially has some benefit in reducing BER, since an increasing in the bias current will improve optical efficiency. Unfortunately, the system eventually leaves the region of operation in which my monotonic open-loop response assumption is valid. In a futile attempt to restore normal operation, the feedback program increases bias current so high that the BER rises again. A more sophisticated feedback control program might detect this phenomenon and reduce the bias accordingly. My feedback algorithm continues to raise the bias until it reaches a pre-set safety limit in the program.

Bias-only intelligent laser drive control, therefore, seems quite adequate to deal with threshold current variation, but does not provide the optical-loss control available with the full feedback system.

7.4.7 Digital-to-Analog Converter Implementation

I next considered the assertion I made above, in the description of Intelligent Laser Drive Control, that the digital-to-analog converters (DAC's) needed to implement it can be made particularly simple.

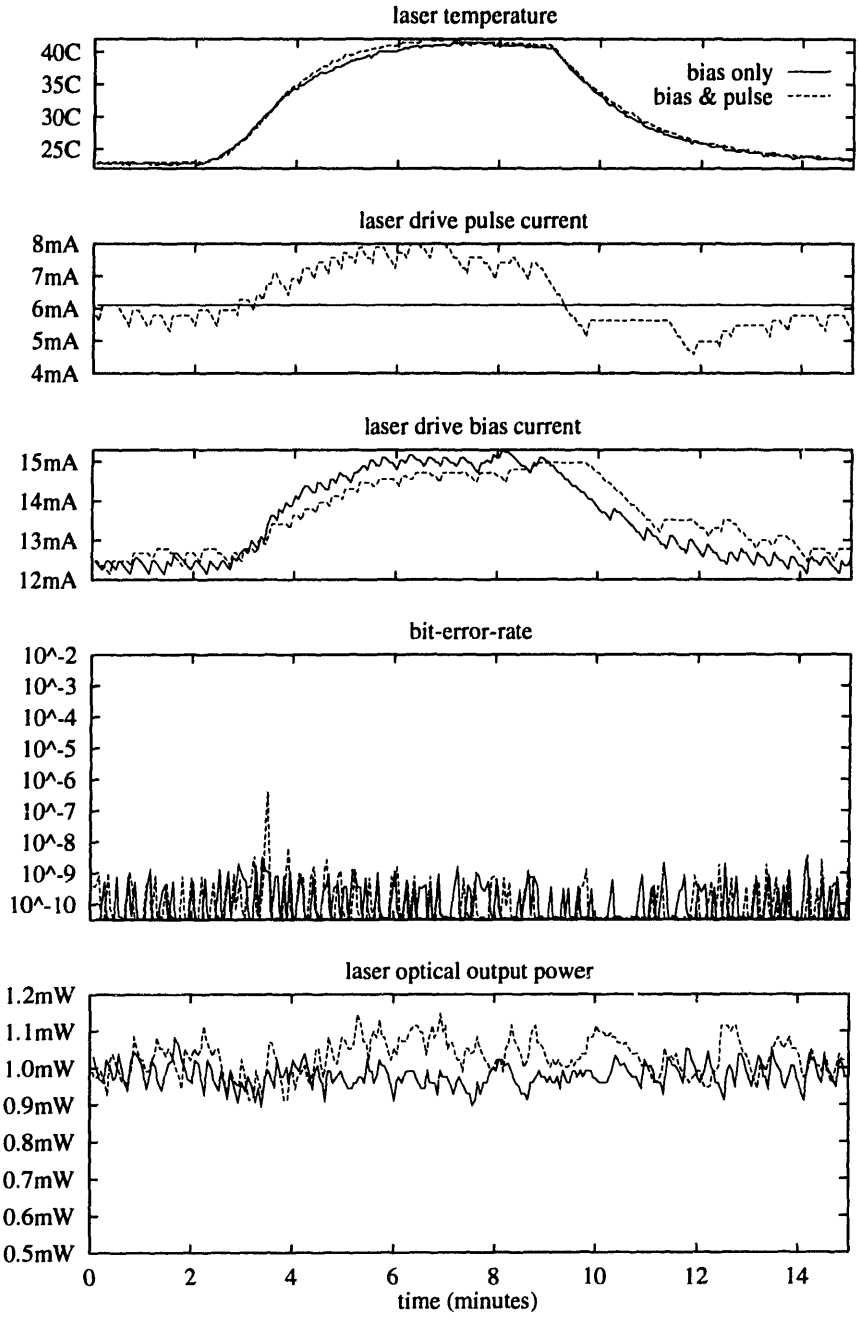

Figure 7-13: Bias-only Feedback (temperature experiment)

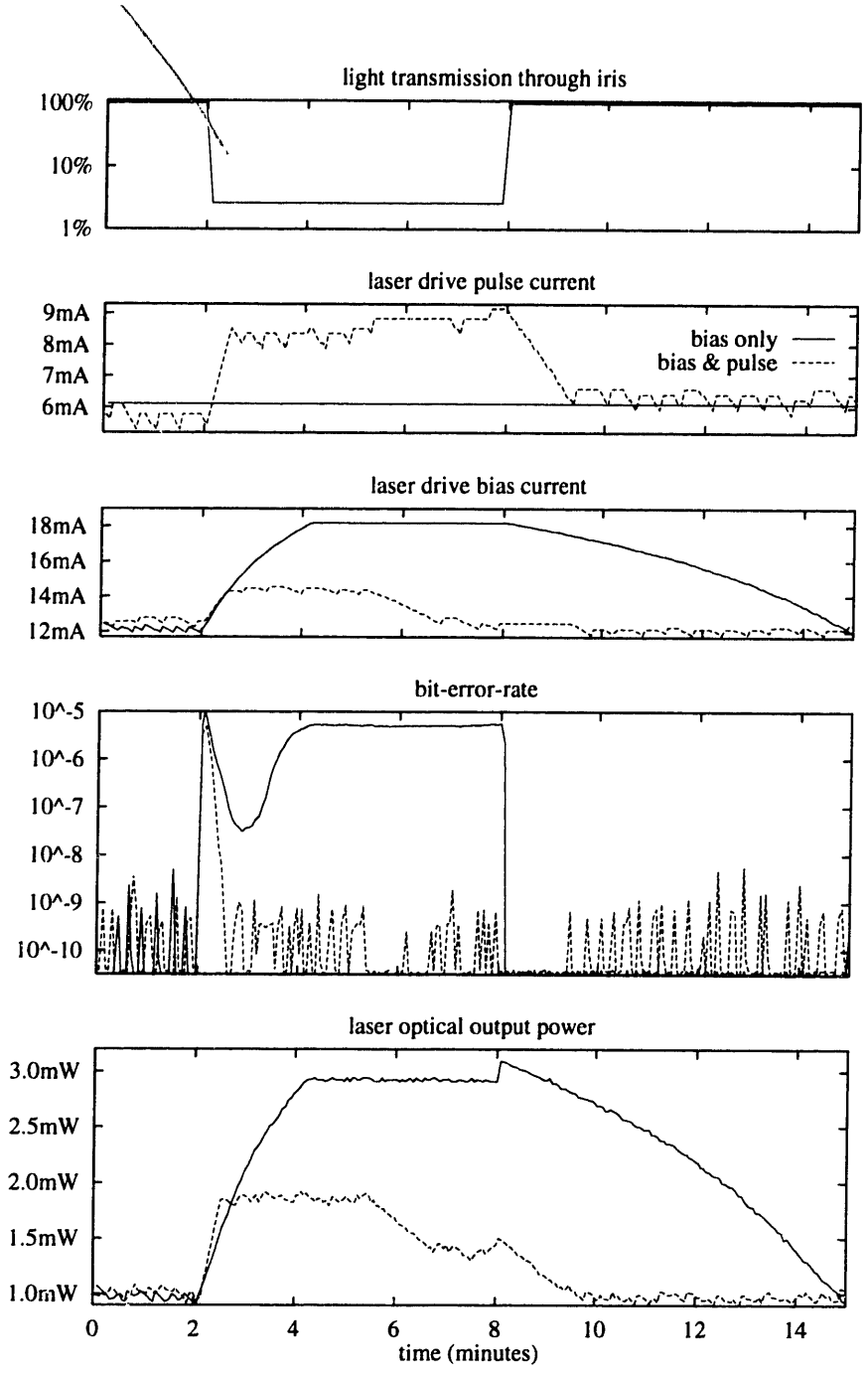

Figure 7-14: Bias-only Feedback (optical loss experiment)

Some of the digital-to-analog converter requirements can be inferred from the control system structure. Absolute accuracy is irrelevant, since the feedback system will inherently compensate for any bias or scale factor errors. Relative accuracy is important to the extent that the input/output function of the digital-to-analog converter must be monotone, else the feedback system might be caught in a sub-optimal operating point, since a non-monotonic output function might create the false appearance of a local minimum in error rate. However, even the monotonicity constraint can be relaxed if the digital-to-analog converter is calibrated before use, and a calibration table is used to eliminate any 'kinks' in the input/output function.

The required precision of the digital-to-analog converter is not immediately obvious. The experimental setup for the results given above used 8-bit converters, so we can conclude that 8 bits will probably suffice. In itself, this is enough to validate my assertion that the control system digital-to-analog converters would be easily implemented. However, I wanted to establish the required precision more accurately.

I modified the feedback control program to allow the drive current step sizes (quanta) to be increased, if desired. As I mentioned above, the original step sizes were about 1% of the normal operating values: the modified program allowed the step sizes to be increased to any multiple of the original step size. Operation with a larger step size is comparable to operation with lower-precision digital-to-analog converters. Specifically, increasing the step size by a factor of 2^n is comparable to reducing the digital-to-analog converter precision by n bits.

I tested a number of bias and pulse current step sizes. To compare the different configurations, I operated the experimental link in normal, steady-state mode (room temperature, iris open), and recorded the average laser power and average error rate for each of the step sizes.

Figure 7-15 summarizes the results with equal multiples of bias and pulse current. If we let M denote the step multiple (that is, each time the bias or pulse current is changed, the change is M times larger than in the original experiment), then the 'effective number of bits' shown in the figure is calculated as $8 - \log_2(M)$. We see that adequate control is still possible with the equivalent of 5-bit digital-to-analog converters for bias and pulse current control. Eight-bit digital-to-analog converters, as I used in the main experiments and as would be quite easily implemented in an

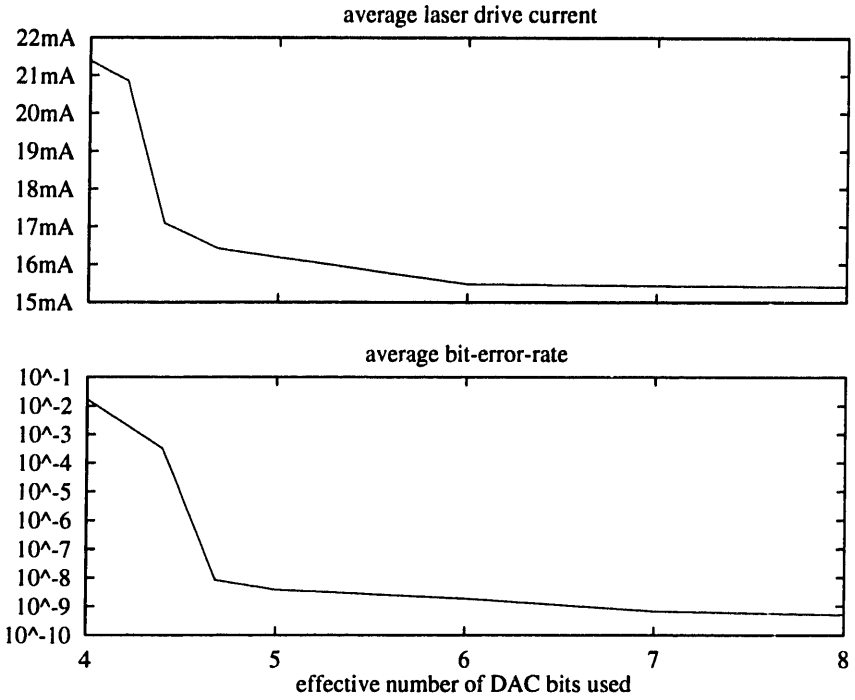

Figure 7-15: Laser control effectiveness vs. DAC precision

actual network, provide a substantial margin of additional precision above that which would be strictly required.

7.4.8 Feedback Stability

Since I have proposed a feedback control system here, one important issue is the stability of the control loop.

The open-loop response of the laser-drive/optical-link system is complex. While the BER vs. optical power relation I gave in Figure 4-2 is simple enough, the relation of bias current, to laser switching speed, to error rate, is not. An accurate mathematical model of the open-loop response function BER(I_{bias}, I_{pulse}) would be difficult to find.

Such a formal model, and stability analysis based on it, might be worthwhile subjects for further research. In the absence of such a mathematical analysis, we can still make some observations about stability, based on the experimental data.

Looking at the bias- and pulse-current plots vs. time from the experiments described in this chapter, one can see the system response to (essentially) step inputs of temperature and optical loss. Except for the coupled increase of bias and pulse currents

(explained above), the control outputs exhibit little overshoot or ringing, indicating a reasonable stability.

Figure 7-15 gives a more direct measure of stability. Increasing the step size, in addition to simulating the effect of lower precision, also increased the feedback gain. Since the step size $M = 10$ (4.68 effective bits) still produced acceptable results, we can conclude that the system has a gain margin of at least a factor of 10, that is, 10 dB. (It is unclear if the poor performance at higher multiples was due to oscillation or to other factors.)

7.4.9 Achievable error-rates

My particular experimental setup was limited to a slow response time because of its unwieldy interface to the data link analysis board. This limitation would not be present in an actual network, but the bit-error-rate, as a feedback control variable, has an intrinsic time constant (let us call it τ) associated with each error-rate value:

$$\tau = \frac{1}{\text{Bit-Error-Rate} \times \text{Bit-Transmission-Rate}}$$

this time constant τ is the expected interarrival time between errors. For example, in my experiments the transmission rate was 1 Gbit/sec, and the desired error rate was 2.5×10^{-10}, so τ was equal to 4 seconds. The time τ places a floor on the error rate that intelligent laser drive control can achieve, depending on the rate of change in the error signals (temperature, aging, optical loss, etc.) to be controlled.

The feedback system does not, in fact, receive an error rate as input. What it receives is error indications. The system must count the errors (call the count n) occurring a particular interval of time (call it T) to arrive at an estimate of the error rate: n/T.

The longer T is, the better the error-rate estimate will be. From empirical experience in designing the feedback system, acceptable operation will require T long enough so that at least two errors in time T are needed to declare the error rate too high. This implies that $T \geq \tau$. For my experiments shown above, I set $T = \tau = 4$ seconds. Any reduction in the error-rate goal would necessitate a proportional increase in T.

However, this time limitation applies only a *lower* limit to the error-rate estimate. If an excessive number of errors occurs before the period T is complete, one can justifiably

suspect that the error rate has become too high.

My experimental program, therefore, does exactly that. The error statistics are read every 1.3 seconds. If after three such readings there have been less than two errors, the error rate is judged to be acceptable. However, if after any of the three readings the error total reaches two or more, the error-rate estimation is aborted, and corrective feedback is performed.

Depending on how fast the error signals vary, intelligent laser drive control may be able to achieve error rates better than those in my experimental setup, but the time constants involved quickly become unwieldy as error-rate goals become more stringent. A strong error-control coding system, such as the one described in Chapter 6, will therefore be required.

These time constants are intrinsic to the error-based feedback system; as with many other theoretical limits they are always valid, but are not necessarily always relevant. Section 7.6 outlines one possible way that this theoretical limit might be sidestepped, to a large extent.

7.5 Benefits of Intelligent Laser Drive

In addition to the direct benefits of threshold current and optical loss variation control, intelligent laser drive control can provide two additional benefits: allowing unbalanced data transmission, and permitting laser lifetime monitoring.

7.5.1 Unbalanced data transmission

Data communications link designers are usually not at liberty to make assumptions about the nature of the data to be transmitted across the link under design. In particular, they must assume that the data may be heavily unbalanced between the bit values '1' and '0'. This can be a problem in an AC-coupled system (such as the one I used for the experiments described above), since the data imbalance will produce a DC bias which will eventually be lost through the AC coupling. (This effect is called 'baseline wander', since the baseline between '0' and '1' values at the receiver appears to wander.)

An AC-coupled system must therefore use a balanced 'line code': a logical transfor-

Data	Codeword	Data	Codeword
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

Table 7.1: 4b5b line code

mation of the incoming data stream (which may not have an equal number zeroes and ones) into another, somewhat longer, data stream which is guaranteed to have equal or nearly equal numbers of ones and zeroes. My experimental setup used an AC-coupled datapath, and a "4b5b" line code, that is, every 4 bits of data generated 5 bits on the transmission link. Table 7.1 shows this line code. Note that this code is not perfectly balanced: it could have a duty cycle imbalance of as much as 60%/40% for some input data streams. However the pseudorandom data used in my experiments, both before and after line coding, had equal numbers of zeroes and ones.

However, the use of a balanced line code can be a problem in systems employing the very wide channels I envision in this thesis, since line codes depend on a sequence of data being transmitted on each bit; word-by-word transmission might entail high overhead (up to 50%) in implementing the required line code, in addition to the coding and decoding circuitry. Unlike the error-detection codes I proposed in the previous chapter, balanced line codes can obviously not be made in systematic form (i.e., the original data accompanied by separate error check bits), so the line coding and decoding is part of the main thread of data transmission, as opposed to being an activity performed in parallel.

In some systems, therefore, it would be preferable to dispense with line coding, and design DC-coupled links. This could pose two difficulties in relation to laser drive control.

First, unbalanced data transmission makes the standard monitor-based laser feedback system (in Section 7.2.2) unusable. The monitor photodiode signal is not only a function of the laser drive and threshold currents, but also of the laser duty cycle.

The feedback system implicitly assumes a fixed and predictable duty cycle (usually 50%) in the transmitted data. (Actually, there are monitor feedback systems which compensate for varying duty cycle, but they are complex indeed: quite impractical for this application.) On the other hand, intelligent laser drive control is quite indifferent to duty cycle variations, since it concerns itself only with the actual communication performance of the link.

Second, a DC-coupled link will be significantly more sensitive to laser threshold current variation. If, because of threshold current variation, the laser is never completely 'off', there will be an abnormal DC bias on photodiode output. An AC coupling circuit would block such a bias, but a DC-coupled circuit would have difficulty with it.

Such a DC-coupled link would exhibit an error-rate minimum at a particular bias current (or range of currents), with increasing error rates at both higher and lower currents. Simple analog feedback loops (or my simple-minded program) would have great difficulty with this, but a well-designed intelligent laser drive control program should be able to track such a curve, and keep the bias current at or near the error-rate minimum.

Unbalanced data transmission is an interesting design option, which intelligent laser drive control can make more feasible.

7.5.2 Laser Lifetime Monitoring

As I indicated in Section 4.2.1, wearout manifests itself as a gradual increase in threshold current. A laser is usually declared 'worn out' when a standard monitor-based control loop has to increase the drive current by a specified amount (usually 50 or 100% of the original drive current). The laser actually fails when the threshold current has increased so much that the laser driver can no longer deliver sufficient current.

In a multiprocessor network we have the luxury of being able to rely on software to manage the system operation. Since laser wearout is a very slow process, if we periodically record the average bias current required by the intelligent laser drive control algorithm, we should be able to perceive the wearout trend in each laser, and therefore predict when it will fail (that is, when it will demand more drive current than is available).

Given this, one should then be able to give the multiprocessor system's owner a reasonably reliable prediction of the network's future reliability and repair needs. For example:

> "If you repair these lasers at time X, the system will run normally. If you wait until time Y, the system will run at reduced capacity. With no repair the system will probably fail by time Z."

Since most normal-use systems have the option of preventive maintenance open to them, such a monitoring system could significantly increase overall system reliability.

7.6 System Implementation

The intelligent laser drive control system is considerably simpler to implement than a monitor-based feedback system, since it requires only one or two simple digital-to-analog Converters in place of a monitor photodiode and analog feedback loop.

Compared to a fixed drive system, intelligent control is undeniably more complex, requiring, in addition to the digital-to-analog converters, an adjustable-level driver which the fixed drive system has no need for. However, as I have demonstrated above, the digital-to-analog converters are eminently suited to VLSI implementation. Pulse current adjustment might or might not be easily added to a fixed-level laser driver, but bias current adjustment almost certainly could, since it can be applied directly to the laser (with suitable provisions for high-frequency decoupling).

An important aspect to realize is that the system need not run all its links under the full feedback regime at all times. It could instead periodically run the feedback algorithm on each link, to characterize it and check for drift. The link would then be fixed at a known (or predicted) low-error operating point, to be changed only when the link is again re-checked, or if it develops excessive errors.

If we are therefore adjusting only one bit at a time, the consequences of a momentarily high error rate are much reduced: if only one bit is affected, the chances of an undetected (multi-bit) error are still extremely remote. The only consequence would be a drop in link performance, as more retransmissions were required.

The scheduling of such adjustment cycles would depend entirely on how quickly the system characteristics were likely to change. A system might schedule quite frequent

adjustments during initial warm-up, and gradually taper off as the system reached a steady operating state. Such scheduling would, in effect, be controlled by a higher-level feedback loop, residing entirely in software. Such dynamic adjustment scheduling is an example of the type of flexibility possible in the feedback system, because of the intelligent control available to it.

By such techniques, one may be able to sidestep the intrinsic time constants mentioned in Section 7.4.9, and have error rates (except during adjustment cycles) close to the minimum practicable.

7.7 Summary

Intelligent laser drive control seems a viable strategy for multiprocessor optical networks. It can:

- Control for temperature-based threshold current variation,
- Presumably control for age-based threshold current variation as well,
- Detect optical path degradation, and restore operation if enough capability remains,
- Allow software-based monitoring of optical-link performance,
- Make analysis long-term trends (such as laser aging) possible.

An experimental intelligent laser drive control system has been implemented and tested. It performs well under temperature and optical-loss variation. With pulse current fixed, it performs well under temperature variation, but not optical loss. Eight-bit digital-to-analog converters were used, but five bits would have sufficed. Detailed stability analysis was not performed, but the system's gain margin was found to be at least 10 dB.

An error rate around 10^{-9} was achieved, and significantly better levels should not be expected on an individual link basis, due to the inherent time scale of the error process. However, overall error performance might be made considerably better by only periodically checking each link for drift, and otherwise operating the links at a low-error operating point.

Chapter 8

Redundant Sparing

Here, I consider and evaluate a conceptually simple solution for hard failure problems: provision of redundant spares to be used in place of failed channels. I shall show how implementation of such a scheme in the type of network described in Chapter 5 can be achieved via a reconfiguration switch design which seems well suited to VLSI implementation.

Using the failure model developed in Section 4.3.4, I then analyze the efficacy of the redundant sparing approach for various network sizes. Additional approaches to the hard failure problem are discussed in Chapter 9.

I first set out my approach to acceptable failure levels, which applies to both this chapter and the next one.

8.1 Acceptable failure levels

The question of how many hard failures can be tolerated is rather different from the corresponding question for transient errors, since hard errors, while quite inconvenient, are unlikely to occur unnoticed. (See Section 6.3 for a discussion of error detection and diagnosis.)

The main criteria determining the acceptable hard-failure rate in a given situation are:

- the tolerance for occasional system unavailability, and
- the cost and feasibility of repair.

Requirements for ultra-reliable systems are generally due to one or both or these criteria. For example, an aircraft control computer system can tolerate only a very low failure rate, due to the fact that even occasional unavailability is not tolerable. An example of the other criterion is an undersea fiber-optic transmission cable, which can tolerate very few failures because of the infeasibility of repair.

Such extreme cases have been the driving force for most fault-tolerant system design. I will assume a more ordinary, less demanding environment, where a low failure rate is desired simply to reduce the inconvenience and expense of frequent downtime and repair. Therefore, my criterion for judging the efficacy of fault-tolerance schemes will be the median time before a system failure requiring repair.

8.2 Redundant Sparing

Redundant sparing is a standard approach to hard-failure problems [43]. As the name implies, Redundant Sparing is the provision of redundant spare optical links, with the ability of the network to switch in a spare link in lieu of a failed one.

As I indicated in Section 6.3, hard failures can be easily detected by the Error Control Coding circuitry. Since hard failures occur quite rarely (from a processor-cycle perspective), we need not provide special hardware to control hard-failure diagnosis and network reconfiguration: such tasks can be performed in software, with negligible impact on system performance.

8.3 Redundant channel switch

It is easy to see how one might provide spare optical links in a channel, but it is not as obvious how to use these links when they are needed. A telecommunications link where data words are transmitted serially on a single optical link is a simple case: provide one link and m spares, and connect the data stream to one of $m + 1$ possible links, depending on which links have failed.

A wide parallel data channel is more complex. It would be far too expensive to provide multiple spares for each bit in the channel, so for an n-bit-wide channel one would provide n regular links and m spares. The problem then becomes one of

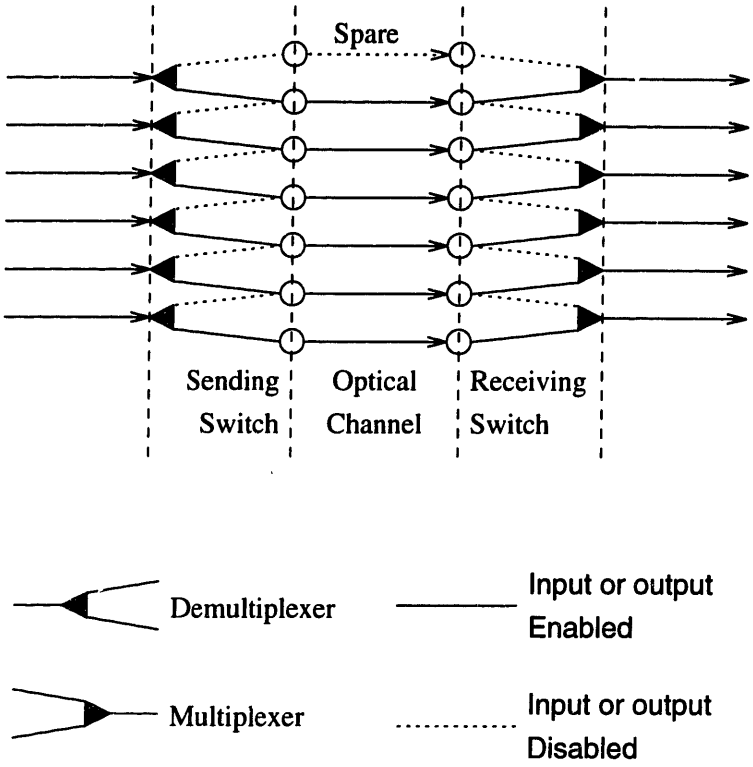

Figure 8-1: Substitution switch example

connecting the data to n of $n + m$ links, depending on which links have failed. This must also be done with the minimum possible switching latency.

I propose here the use of a substitution switch, capable of switching out failed links and substituting spare links in their place, which seems well suited to high-speed implementation in VLSI.

To illustrate the point, let us consider a six-bit channel with one additional spare link. Figure 8-1 shows a pair of substitution switches (one for sending, and a mirror-image of it for receiving) which will enable us to use the spare link to replace a failed one. Each switch contains one multiplexer (or demultiplexer) per bit. The figure shows the switch in the normal (pre-failure) state, with the spare link unused. (The choice of which link is 'spare' is actually an arbitrary one, and makes no difference to the fault-tolerance of the switch.)

Figure 8-2 shows the same channel after one of the links has failed. Note that upon reconfiguration of the substitution switches (the one switch configured as the mirror-image of the other), the channel can continue operation unimpeded, with the failed link removed from the channel.

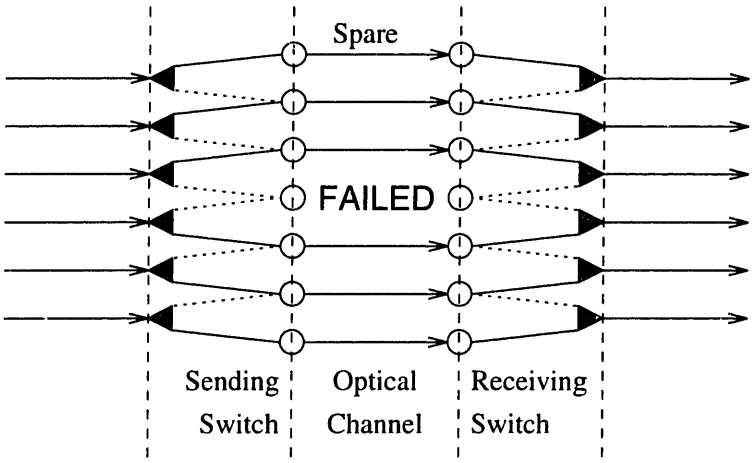

Figure 8-2: Substitution for one failed link

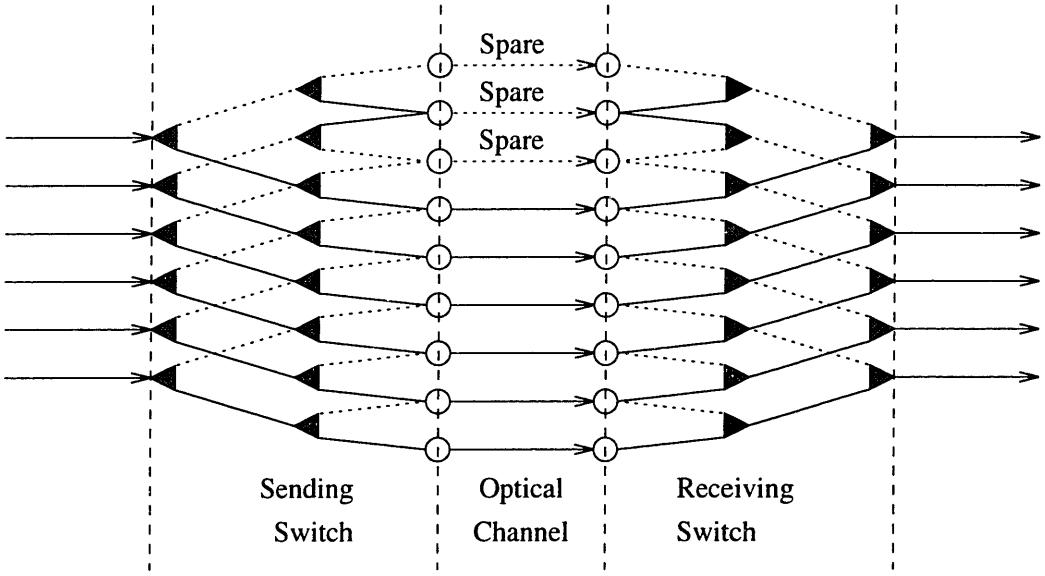

Figure 8-3: Two-stage substitution switch

The switches in Figures 8-1 and 8-2 are adequate for dealing with one error per channel. They can be extended to any number of data bits, but will accommodate only one spare link.

Fortunately, we can extend the same concept to handle additional spare links. Figure 8-3 shows the same six-bit channel as before, but now with three spare links. There are two mirror-image substitution switches, as before, but now there are two levels of multiplexing/demultiplexing.

Figure 8-4 shows the channel after three of the links have failed. As before, reconfiguration of the substitution switches (in mirror-image fashion) allows the channel to

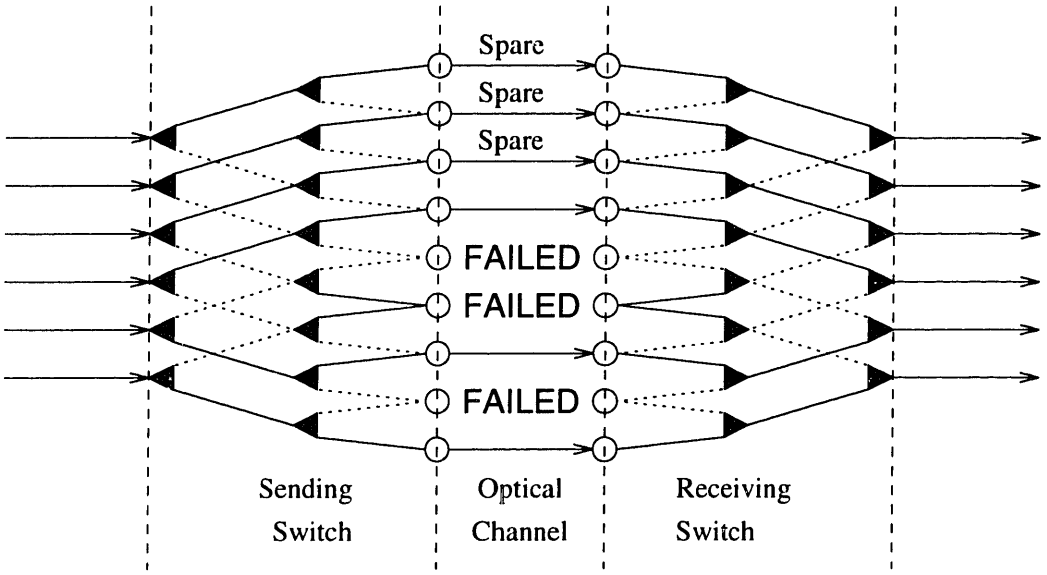

Figure 8-4: Substitution for three failed links

continue normal operation.

This concept can be extended as far as desired. Figure 8-5 shows a four-stage substitution switch, which can handle 15 spare links. (To save space, only the sending switch is shown: the receiving switch is, as always, a mirror-image of it.) Figure 8-6 shows the channel after 15 failures have occurred: again, the reconfigured switch allows full operation to resume. (Since the switch configuration gets rather complex at this level, in this figure I distinguish between the parts of the switch in active use versus those not actually connected for data transmission.)

In fact, an n-level substitution switch of this design can handle up to $2^n - 1$ spare links, and can reconfigure the channel for full operation after any set of $2^n - 1$ links have failed. Let us consider why this is so, and what is the proper algorithm for switch reconfiguration given an arbitrary set of failures.

8.3.1 Single-stage substitution switch theory

Consider the one-stage substitution switch shown in figure 8-2. It should be clear that this switch can be reconfigured to eliminate any one failed link. But suppose that there were more than one failure: what could it do? While the one-stage switch can not restore full channel operation when faced with more than one failure, it *can* map the failures to different positions, so that they are more contiguous than before. In

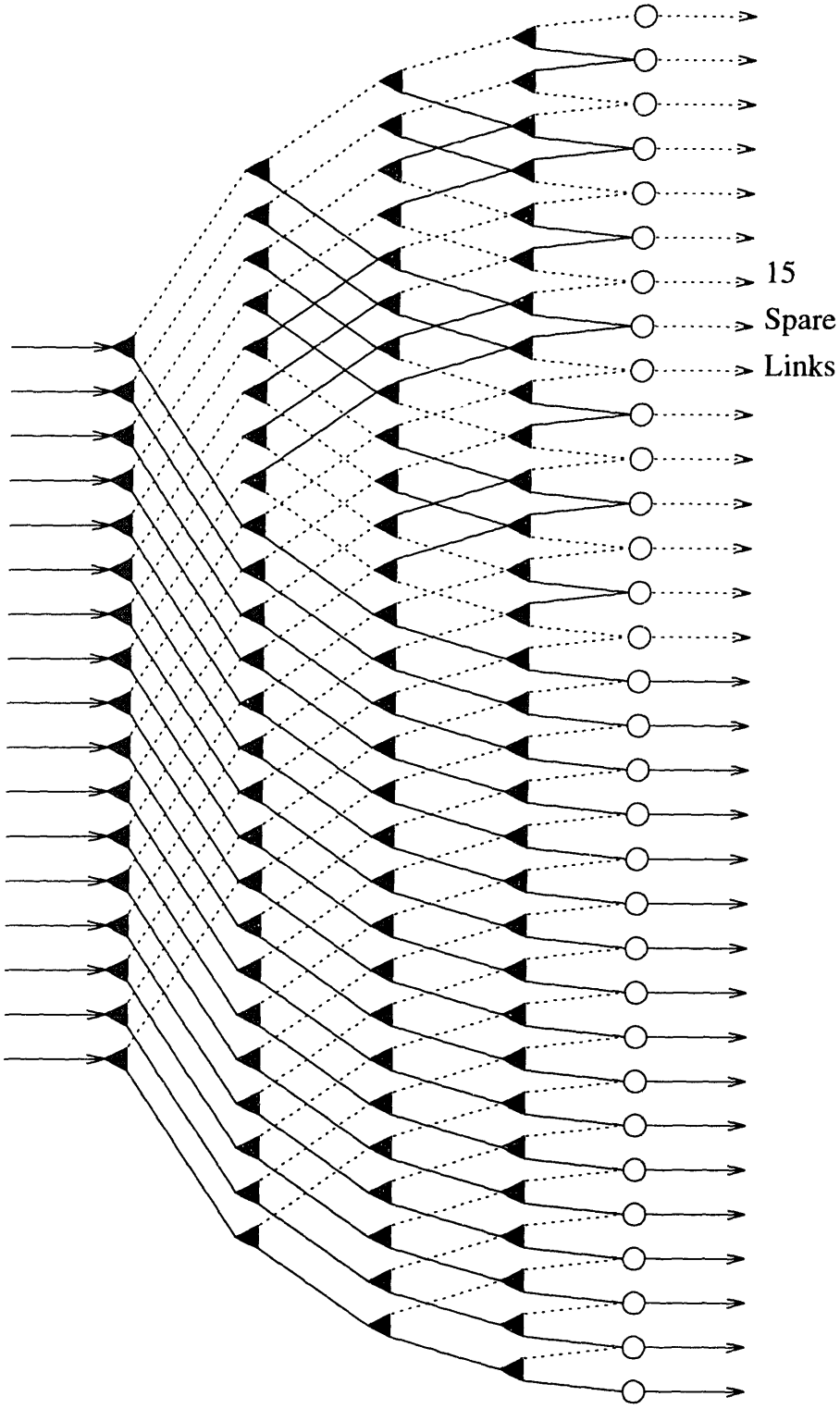

Figure 8-5: Four-stage substitution switch

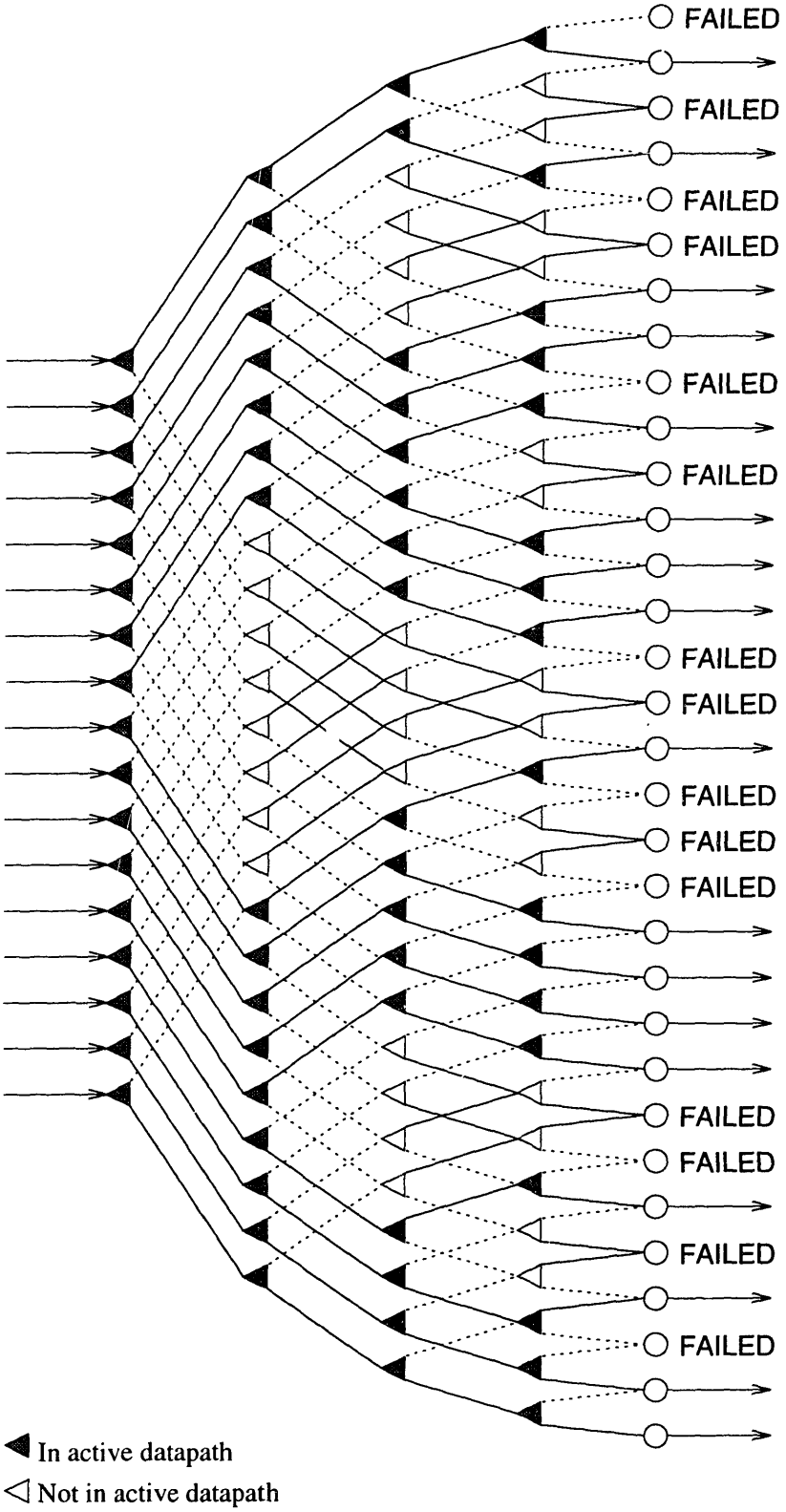

Figure 8-6: Substitution for 15 failed links

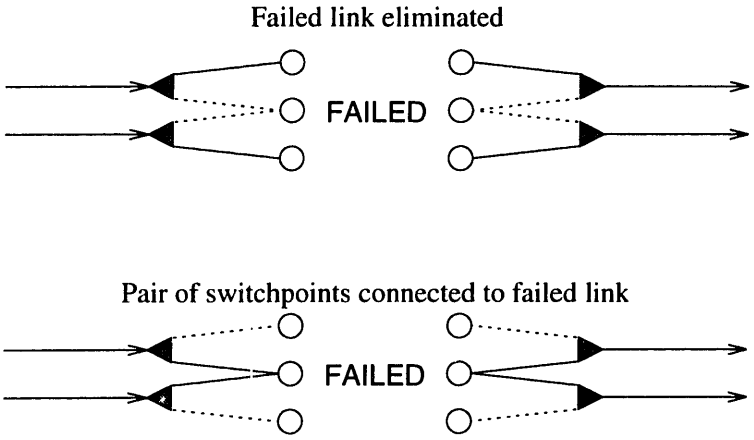

Figure 8-7: Substitution switch algorithm examples

particular, given m failed links, the one-stage switch can map every failure into $\lfloor m/2 \rfloor$ contiguous pairs (except that a 'pair' at the top of the switch might have only one failure).

Recall that the sending demultiplexers and the receiving multiplexers are configured as mirror images of one another. For convenience, let us call each pair of demultiplexer and corresponding multiplexer a 'switchpoint', and call its two switched outputs or inputs 'ports'.

Let us devise an algorithm to configure the one-stage switch. We will start with the bottom switchpoint. (Actually, we will start from one end, and define that end as the 'bottom'.) A switchpoint is in the 'up' state if its upper port is active, and is in the 'down' state if its lower port is active. Here is the algorithm:

- Start with all the switchpoints in the 'down' state.
- Scan the links one at a time, from bottom to top. At each link, if the link has failed, invert the state of all switchpoints above that link.

[Here's another way of putting it: if there is an odd number of failed links below a switchpoint, put in the 'up' state; otherwise put it in the 'down' state.]

This algorithm assures that all failures (except, perhaps, a failure in the top link) will be mapped into contiguous pairs. To see how this works, consider Figure 8-7. This shows the two possible switchpoint configurations around a failed link, since the switchpoints above and below it must be in opposite states. In the first case, the failed link has been eliminated from the network; if this is the only failure, then all the

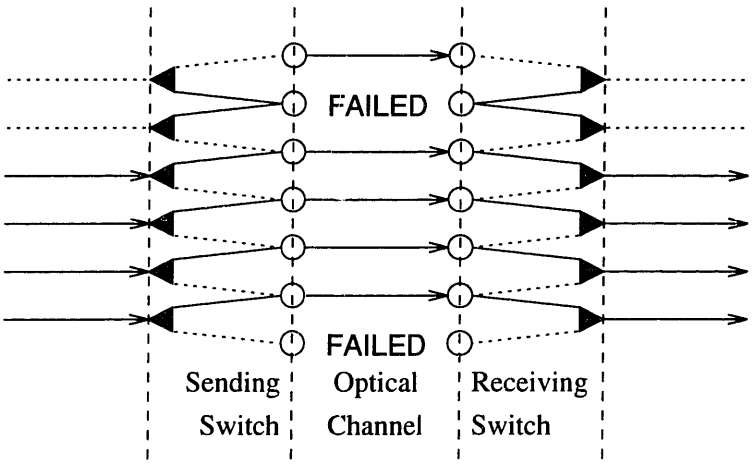

Figure 8-8: Failure pair mapping

switchpoints will be connected to working links, as we saw in Figure 8-2. Figure 8-8 shows how two separate failures are mapped to contiguous positions.

The second case in Figure 8-7 shows what happens if there is an odd number of failures below the one shown. In this case two switchpoints point at the failure, and are therefore not usable. One can think of this as effectively moving the 'odd' failure from down below to be paired with the failure here. Of course, if there were a failure in the top link with an odd number of failures below, then there would be only a single switchpoint to connect to it, so that is the one exception to the rule.

We see that with this algorithm, an isolated link failure will be shifted up until the end of the switch is reached, or until another failure is encountered. In the latter case, both failures, and any additional contiguous failures, are then propagated to the switchpoints (that is, the switchpoints are declared 'dead'), and the process then repeats. This cannot, therefore, produce isolated 'dead' switchpoints (except for the top switchpoint, if the top link has failed).

8.3.2 Multi-stage substitution switch theory

Given this result, how can we eliminate failed channels, as opposed to merely pairing them? Consider the second-level substitution switch shown in Figure 8-3. Note that the second level switchpoints are interleaved, spanning *two* positions, instead of one. We can use this to eliminate a *pair* of failed switchpoints, just as the first-level switch could eliminate a single failed link.

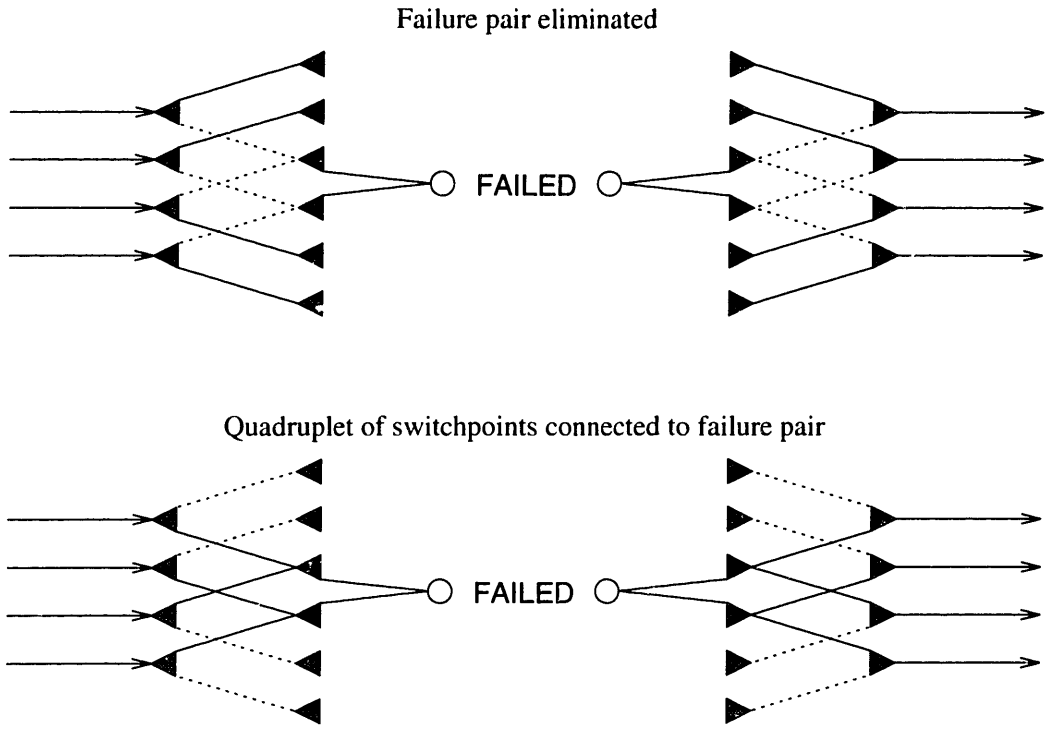

Figure 8-9: Second-level substitution algorithm examples

To configure the second-level switchpoints, we use precisely the same algorithm as we used for the first level, except that we scan for switchpoint failure-pairs instead of failed links. So, we first configure the first-level switchpoints. The second-level switchpoints are then configured as follows:

- Start with all the second-level switchpoints in the 'down' state.
- Scan the first-level failure pairs one at a time, from bottom to top. At each pair, invert the state of all second-level switchpoints above (the center of) the pair.

Figure 8-9 shows the two possible configurations around a failure pair. As with the first-level switches, there are only two possible configurations. The first case shows how a failure pair can be eliminated. The second case shows that, when the failures can not be eliminated, they will be grouped into quadruplets. If there is a first-level 'truncated pair' (that is, a singleton) at the top of the switch, it is handled like any other pair, so there may a second-level 'truncated quadruplet' at the top of the switch also.

If there are p first-level failure pairs, then there will be $\lfloor p/2 \rfloor$ second-level failure quadruplets. With m failed links, $p = \lfloor m/2 \rfloor$, so if $m \leq 3$, the two-level substitution

switch will eliminate all the failures.

We can generalize the switch to n levels, with the n-th level switch spanning 2^{n-1} positions, and grouping failures into 2^n-tuples. The configuration algorithm given applies to all levels. Since level ℓ can eliminate $2^{\ell-1}$ failures, the multi-level switch can eliminate $\sum_1^n 2^{\ell-1} = 2^n - 1$ failures. A switch to accommodate s spare channels would therefore require $\lceil \log_2(s+1) \rceil$ levels.

8.3.3 Substitution switch implementation

As we can see from Figure 8-6, the substitution switch can appear quite complex. However, the previous section shows that its configuration is actually rather simple. Also, as I mentioned in the beginning of the chapter, new failures are exceedingly rare events on a processor-cycle timescale, so we can handle the switch reconfiguration in software.

While a multi-stage reconfiguration switch would be complex, it has a number of attributes which would make it well suited for implementation in VLSI:

- it is a quite regular structure with simple control, making the design task straightforward,
- each output drives no more than two input loads, making a high-speed implementation more feasible, and
- since reconfiguration would be a rare event, we could design the multiplexer and demultiplexer[1] circuits, as much as possible, to trade off reconfiguration speed in favor of fast data propagation time.

Since the same signals are involved, one VLSI circuit could implement both the substitution switch and the error-detection coding/decoding functions. A monolithic VLSI solution might achieve quite fast switch propagation times, since it would avoid chip-to-chip communication delays within the switch. It might also be practical to include all or part of the laser drivers or photodiode amplifiers in such a VLSI circuit.

[1]Actually, one might well use just multiplexers (or demultiplexers) exclusively, in both sending and receiving switches, if that were an advantageous design; one can quite easily transform the circuits to use only one or the other. I have presented a design with both multiplexers and demultiplexers because it preserves the symmetry of the switch and thereby makes its operation easier to explain.

8.4 Redundant sparing lifetime

Let us next consider the likely operating lifetime of a network such as the one described in Chapter 5, with various levels of redundant sparing (using an appropriate substitution switch) in each optical channel. Let us consider the redundant-sparing network to have failed (i.e., to require repair) when any one of its channels has experienced more failures than it has spare links (for example, 8 failures in a channel with 7 spare links).

Looking at Figure 4-9, we see that, at least until around $t = 8$ years, we can approximate laser failure as a Poisson process, with arrival rate $\lambda \approx 0.009$/year. If each channel is n bits wide, with m spare links (and a substitution switch of $\lceil \log_2(m+1) \rceil$ levels) provided on each channel, and assuming that the spare links suffer no failures when not in use, then we can approximate the probability that a particular channel has failed by time t by the probability that more than m arrivals of a Poisson process of rate $n\lambda$ will occur in a time interval of length t (let $p_c(t)$ denote this probability). We therefore have

$$p_c(t) = \sum_{k=m+1}^{\infty} \frac{(\lambda t)^k}{k!} e^{-\lambda t} = P(m+1, n\lambda t) \tag{8.1}$$

where $P(k, x)$ denotes the incomplete gamma function[44].

If the system has a total of C such channels with independent failures, then the probability that at least one of them has failed by time t (let $p_s(t)$ denote this probability) is given by

$$p_s(t) = 1 - (1 - p_c(t))^C \tag{8.2}$$

If, as mentioned in section 8.1, we are concerned with the median time before a failure requiring repair, then we are interested in finding t_m such that $p_s(t_m) = \frac{1}{2}$, so

$$\begin{aligned} p_s(t_m) &= 1 - (1 - p_c(t_m))^C = \frac{1}{2} \\ (1 - p_c(t_m))^C &= \frac{1}{2} \\ p_c(t_m) &= 1 - 2^{-1/C} \\ P(m+1, n\lambda t_m) &= 1 - 2^{-1/C} \end{aligned} \tag{8.3}$$

Actually, we can extend the usefulness of the Poisson-process approximation beyond $t = 8$ years, by stretching out the time axis to make the laser failure probability

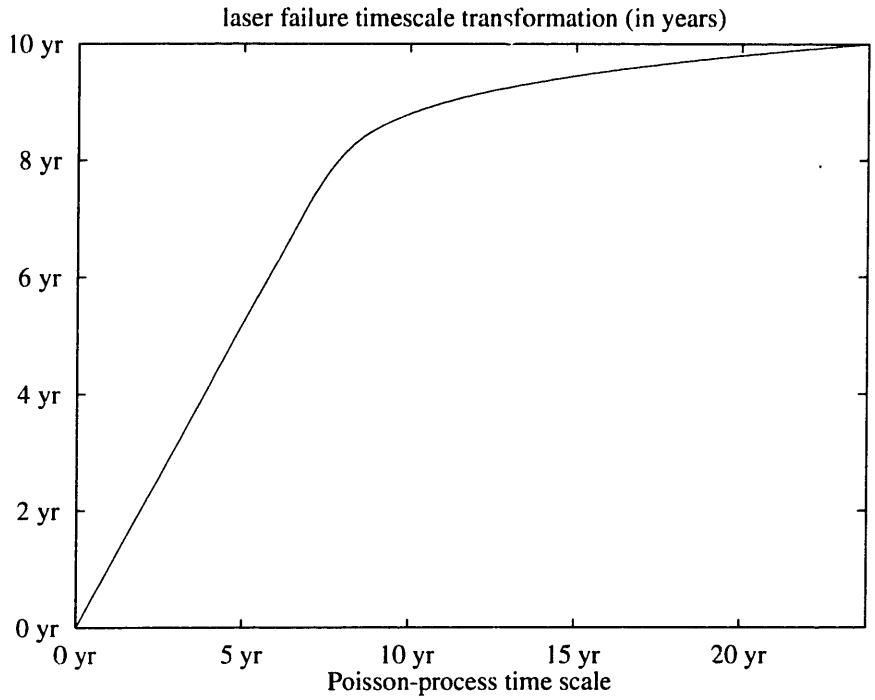

Figure 8-10: Laser failure timescale transformation

continue to match that of a Poisson process. This is a sort of Procrustean bed, where we stretch and squeeze the failure model until it fits a desired form. Figure 8-10 shows a time transformation which makes a Poisson process of $\lambda = 0.009$/year match the failure model given in Figure 4-9.

Using this method, I have obtained several numerical solutions of equation 8.3, each for networks of different sizes. Based on the the network described in Chapter 5, I set the channel width $n = 74$, and the number of channels $C = 4N$, where N denotes the number of nodes in the network.

Figure 8-11 shows the minimum sparing level m required to achieve different values of the median system lifetime t_m, for mesh networks of 16, 1024, 65,536, and 16,777,216 nodes.

We see that redundant sparing can handle the random-failure part of the laser lifetime curve, but the required level of spares rapidly becomes impractical to sustain continued operation as the lasers enter the wearout regime ($t \approx 8 - 10$ years).

[However, I have not included in these calculations any effect from a laser monitoring and replacement policy of the type suggested in Section 7.5.2. Such a policy might well allow continued operation at times when the system would otherwise by

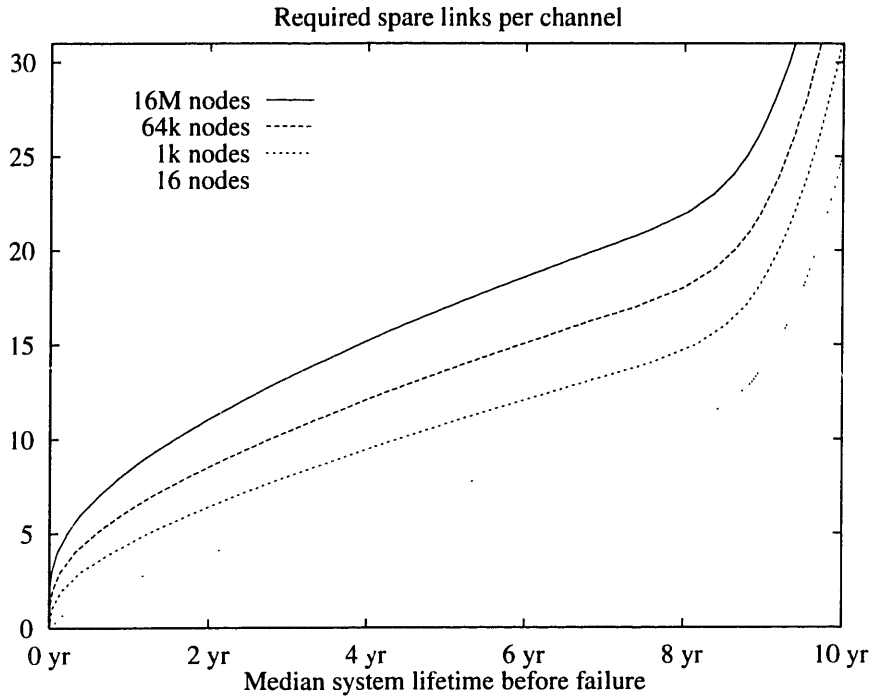

Figure 8-11: Spares/channel m vs. median system lifetime t_m

dominated by wearout failures.]

8.5 Summary

Redundant Sparing could suffice to control random laser failures and early wearout failures, if sufficient spares were provided. A multi-level substitution switch, possibly implemented in a very-large-scale integration (VLSI) chip, could make this feasible.

Bandwidth Fallback, to be discussed in the next chapter, is an interesting improvement to this approach. In fact, we shall see that it can let us omit the spare channels, and still offer performance only slightly less than Redundant Sparing.

Chapter 9

Bandwidth Fallback

In this chapter, I discuss two strategies to supplement redundancy; in particular, they specify courses of action to take when redundant spares run out. By doing this, they can reduce (or eliminate!) the level of redundant sparing that we saw was needed in the previous chapter.

The two options I discuss here are Detour Routing and Bandwidth Fallback. Both seek to continue network operation (at a reduced rate) after failures for which redundancy is inadequate: these are therefore 'graceful degradation' strategies.

9.1 Detour Routing

Detour Routing exploits the wormhole-routing concept of Virtual Channels to produce new (virtual) channels to replace failed ones. Virtual Channels is an idea which was actually conceived to enhance the performance of wormhole-routed mesh networks (see Section 5.3 for an explanation of wormhole routing). The basic idea is to take each *physical* channel in a network, and establish a number of *virtual* channels within it, that is, to handle the network routing and scheduling as if there were several independent virtual channels wherever there is an actual physical channel. Figure 9-1 depicts this idea.

Virtual channels are implemented by time-division multiplexing the underlying channel, when required. The additional overhead imposed by a virtual-channels strategy is the additional control and buffering for each additional (virtual) channel created. Wormhole routing therefore makes virtual channels more practical, because

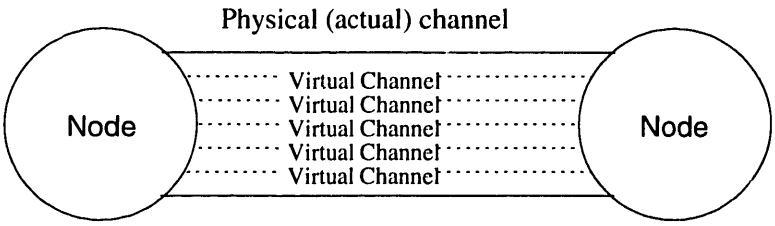

Figure 9-1: Virtual Channels

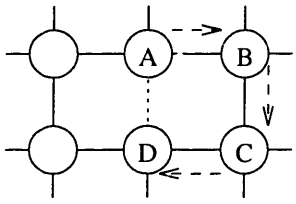

Figure 9-2: Detour Routing Example

each channel in a wormhole-routed network requires only minimal buffering.

The use of Virtual Channels can provide significant performance benefits in a wormhole-routed network as Dally has shown [45], by handling congestion more effectively. However, as Kung suggests [46], it can also provide a measure of fault tolerance.

It can do this by allowing us to construct detour routes, exploiting the existence of redundant paths in the system network topology. My prototype mesh network obviously has such redundancy, as do many other possible topologies (see Section 3.7).

Detour Routing can provide an additional backup for a channel which fails because it runs out of spare links. Traffic which would travel the failed link is re-routed on a virtual link created by time-multiplexing a redundant path in the network.

As an example, consider Figure 9-2. If the channel from A to D fails (runs out of spares), we can use virtual channels to replace it. We do this by time-multiplexing the channels A-B, B-C, and C-D, and routing the A-D traffic through these three multiplexed channels. The new channel so constructed triple the latency of the original, and must share the bandwidth of the multiplexed channels with with regular traffic (which will be impeded as well), but the network *does* function when so reconfigured, in a situation where straight Redundant Sparing would call for repair. When only small numbers of such detour channels are necessary, they will have minimal impact on system performance, while extending the system MTBF.

Note that the reconfiguration does not vitiate the argument (presented in Section 5) that our mesh network routing will not deadlock. Because we have preserved the logical topology of the network, the non-deadlock arguments still apply. All we have done is slow down some of the (logical) channels. We must, however, ensure that the regular traffic cannot monopolize the virtual channels to the extent that detour traffic would be blocked: that *would* invalidate our no-deadlock assurance.

This Detour Routing fault tolerance approach effectively views the links on the alternative channels (such as the channels shown in Figure 9-2 as another class of spare links, which in this case are not redundant. Especially if a virtual channel scheme would be implemented in any case for performance reasons, the additional overhead in implementing this fault tolerance method is minimal, and it obviously has the potential to transform a 'must-repair-now' situation into an 'almost-normal' (or at worst a 'repair-soon') situation.

In section 9.3, we shall see that Detour Routing, by softening the impact of having a channel run out of spares, will reduce the number of spares required in order to maintain a given level of reliability. We now consider another approach to reducing the amount of redundancy needed.

9.2 Bandwidth Fallback Concepts

Bandwidth Fallback is a different approach from the previous ones, in that rather than trying to preserve a fixed network channel width, it seeks to use whatever channel width the network has to offer on the desired path.

Rather than having idle spare links (waiting for an active link to fail), or having entire channels declared dead because they are one or two bits too narrow, Bandwidth Fallback takes a more flexible policy: when using a given path through the network, set the data width to the width of the narrowest channel on the path.

9.2.1 Basic Operation

The basic idea of Bandwidth Fallback is one of 'graceful degradation': when the spare links (if any) within a data channel are exhausted by previous failures, and a new failure occurs, one can reconfigure the channel to use only the remaining links, thereby

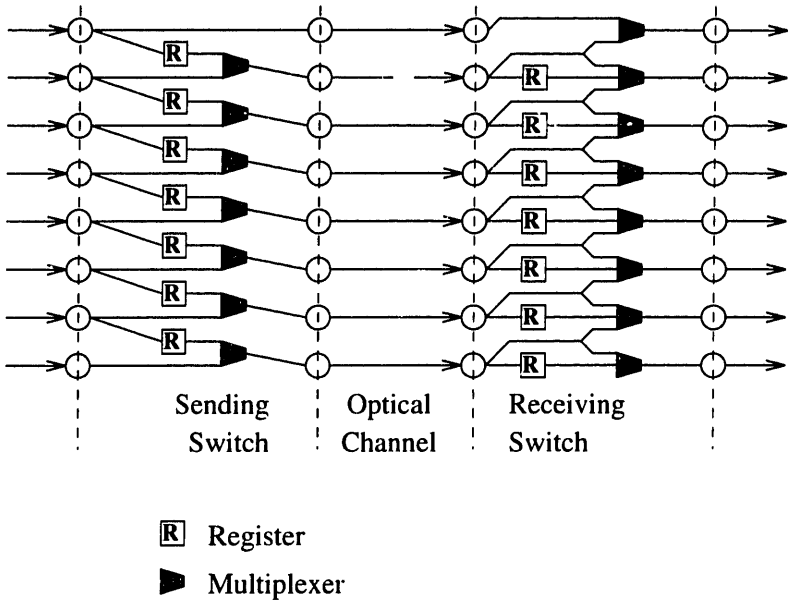

Figure 9-3: Bandwidth fallback switch

providing a channel that can still operate, albeit with reduced bandwidth. This could be preferable to the alternative: declaring the channel to have failed, and therefore providing zero bandwidth.

Unfortunately, it would rather difficult to implement Bandwidth Fallback in exactly this way, since one would have to re-package the data into words of arbitrary size, for example, converting 64-bit words into 51-bit words for transmission. This would require both sender and receiver to have at least a full-width barrel shifter and two full-width holding registers.

However, by placing a simple constraint on the set of output word widths, we can implement Bandwidth Fallback in a switch requiring only very simple hardware.

Figure 9-3 shows the basic circuit for implementing bandwidth fallback. It requires one register and one multiplexer for each bit of a channel. For full bandwidth operation, the registers are bypassed and data transfer occurs normally. For reduced bandwidth operation, we are constrained to a limited set of fractional bandwidths: $1 - 2^{-n}$, that is,

$$\frac{1}{2}, \quad \frac{3}{4}, \quad \frac{7}{8}, \quad \frac{15}{16}, \quad \frac{31}{32}, \quad \ldots, \quad 1 \tag{9.1}$$

The switch in Figure 9-3 can implement bandwidths of 1, 7/8, 3/4, and 1/2 times full bandwidth.

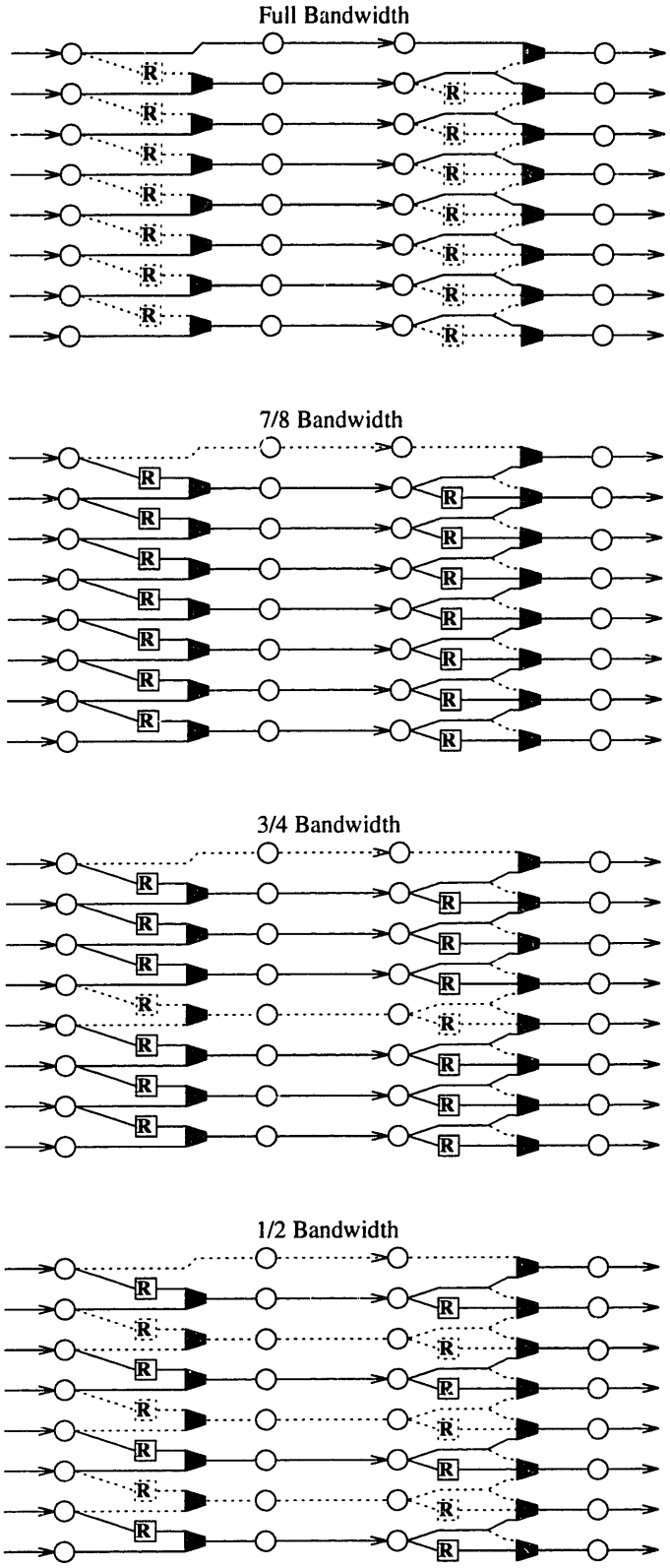

Figure 9-4: Switch configuration for various fractions of bandwidth

From sender				Data held			75% Channel			Data held			To receiver			
A3	A2	A1	A0	-	-	-	A2	A1	A0	-	-	-	-	-	-	-
B3	B2	B1	B0	A3	-	-	A3	B1	B0	A2	A1	A0	A3	A2	A1	A0
C3	C2	C1	C0	B3	B2	-	B3	B2	C0	-	B1	B0	B3	B2	B1	B0
(idle)				C3	C2	C1	C3	C2	C1	-	-	C0	C3	C2	C1	C0
D3	D2	D1	D0	-	-	-	D2	D1	D0	-	-	-	(idle)			
E3	E2	E1	E0	D3	-	-	D3	E1	E0	D2	D1	D0	D3	D2	D1	D0
⋮				⋮			⋮			⋮			⋮			

Table 9.1: Data transmission through a 75%-bandwidth channel

Figure 9-4 shows how the switch can be configured for these different fractions. The fraction 1/2 configuration may be readily understood from the figure: the bits are paired, and the paired bits are sent one after the other.

Switch operation at the higher fractions requires rather more explanation. Looking at Figure 9-4, consider either of the (now disconnected) four-bit sections in the switch configured for 3/4 bandwidth. Table 9.1 illustrates the flow of data through such a 75%-bandwidth channel.

The actual operation of the switch is somewhat complex, but its basic concept is simple: for bandwidth $\frac{k-1}{k}$ operation, the switch accepts $k-1$ input words and passes $\frac{k-1}{k}$ of the bits in each one, storing $\frac{1}{k}$ of the bits in the registers. After $k-1$ words have been received, all the registers are full. The input to the switch is then suspended for one cycle and the register contents are output. The same cycle begins again, and continues as long as input is available. The receiver reverses the process using a similar switch. The same basic operation applies to all the possible fractions $(1-2^{-n})$ mentioned above.

While the switch control may be complex, it is fixed and predictable, and can therefore be pipelined as much as necessary. The switch control program changes only when the bandwidth fallback fraction must be changed (that is, when too many additional links have failed).

9.2.2 Interaction with substitution switches

From Figure 9-4 we can see that some bits in the switch are expendable, and some are not. For example, the top channel link is quite expendable, since losing it will

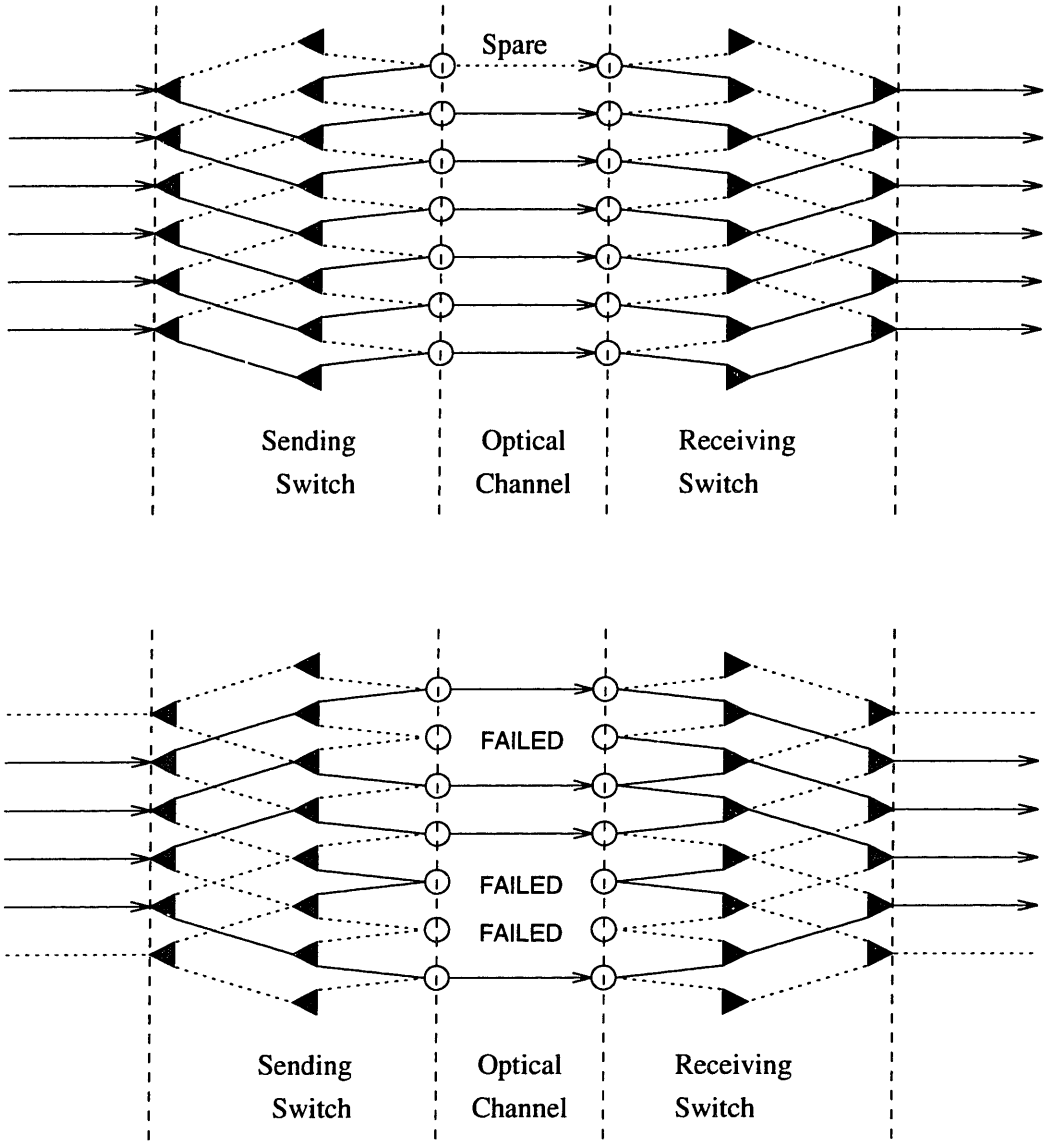

Figure 9-5: Substitution switch with bandwidth fallback

only drop the bandwidth to 7/8, while the link next to it is essential, since the fallback switch cannot work around it.

This may seem to be a problem, but there is a simple solution: ensure that the 'expendable bits' are routed to the outside channels of the redundancy substitution switch, and the essential channels are routed to the inside. If the switch is not completely full (that is, it has less than $2^{\ell} - 1$ spares, where ℓ is the number of levels), then the bandwidth fallback will essentially create new 'spare' channels from the bandwidth it stops using.

Figure 9-5 demonstrates this. In the first diagram, we see that although there is a

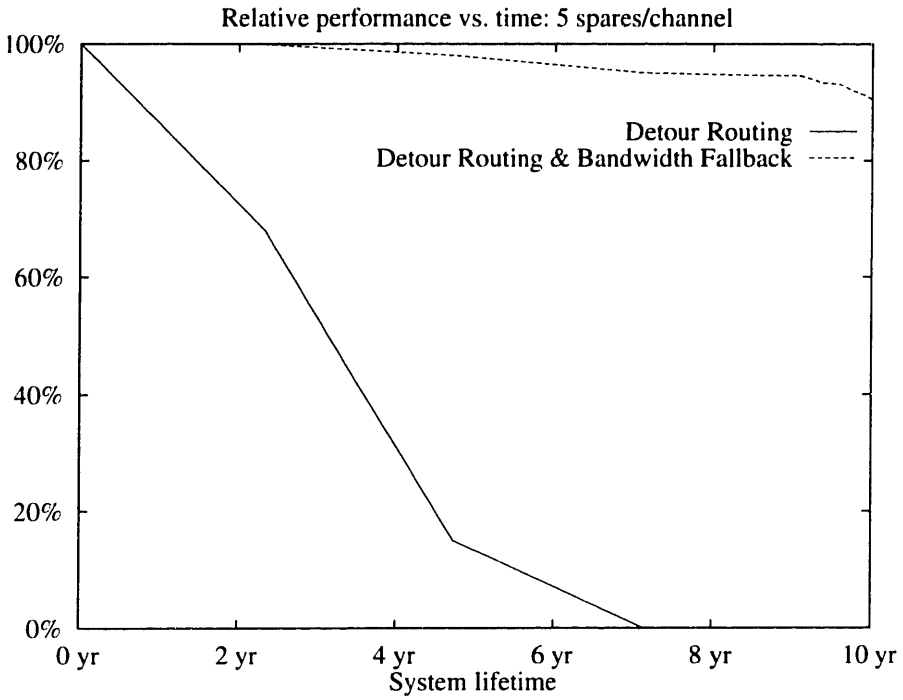

Figure 9-6: Performance vs. time with 5 spares/channel

two-level substitution switch (which can accommodate three spares), we have only one spare link. In the second diagram, we see that three links have failed, but the channel can still operate because the bandwidth fallback scheme has stopped using two of the bits in the channel, which effectively makes three spare links available, where before there was only one.

Therefore, bandwidth fallback allows us to view the regular communication links as if they were redundant spares, but unlike redundant spares, we can make full use of them both before and after failures occur.

9.3 Bandwidth Fallback Simulations

I have written simulation programs to estimate the impact of using Detour Routing and Bandwidth Fallback in a multiprocessor network of the type I described in Chapter 5. Details of the simulation are given in Appendix B, while results are given here.

I show here results of simulating the operation of a 10×10 mesh network with varying levels of redundant sparing. Figures 9-6 and 9-7 show how performance evolves over time for both Detour Routing and Bandwidth Fallback (using the failure model depicted in Figure 4-9) with 5 and 10 spare links/channel, respectively. (For

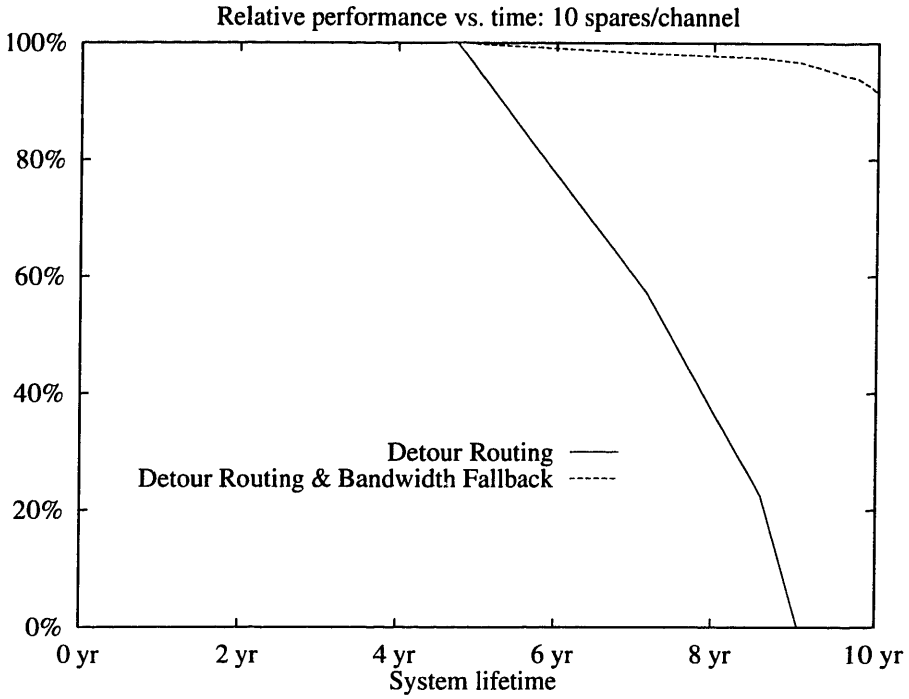

Figure 9-7: Performance vs. time with 10 spares/channel

these Bandwidth Fallback simulations, I assume the use of five-level substitution switches).

We see that the Detour Routing approach is acceptable, especially with the higher level of sparing, but the Bandwidth Fallback results are almost perfect. (Note that straight Redundant Sparing would do worse than Detour Routing, since it would just give up completely when a channel ran out of nodes.)

Let us see if we can find the limits of this phenomenally good Bandwidth Fallback performance. Figure 9-8 shows a configuration where even with Detour Routing the system fails almost immediately: *no spare links at all!* Bandwidth Fallback, on the other hand, enables the system to continue operation smoothly, with only modest performance loss.

9.4 Conclusions

I have presented two failure-tolerance schemes: Detour Routing and Bandwidth Fallback. When implemented together, they provide superb error tolerance, with performance loss of only a few percent. If adequate substitution switching is provided, Bandwidth Fallback can provide continued system operation well into the laser wearout

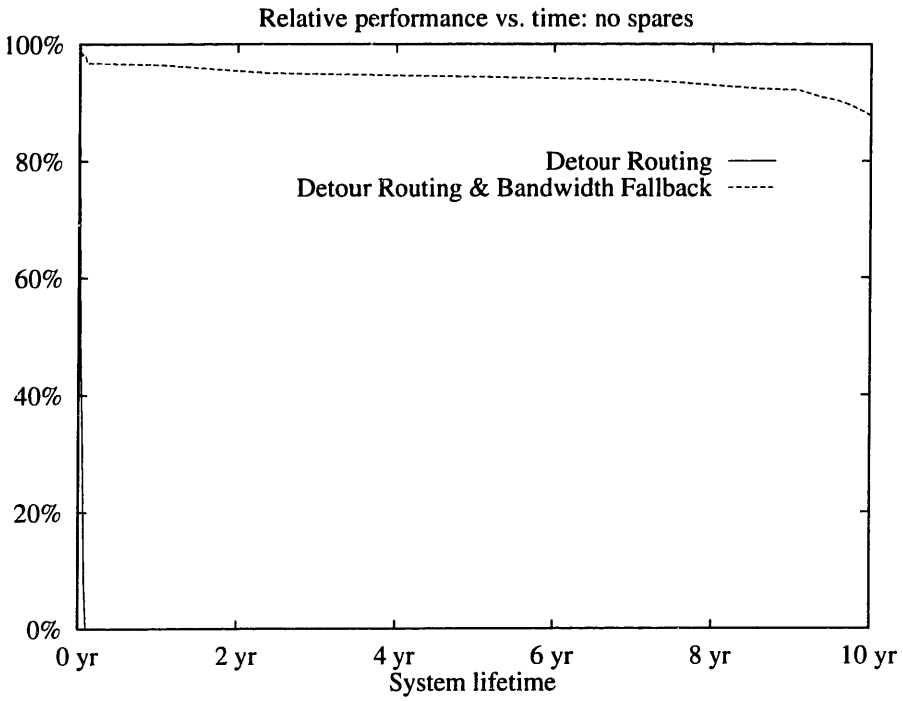

Figure 9-8: Bandwidth Fallback performance with no spares

region of system lifetime, while allowing us to completely dispense with the provision of any spare links.

Chapter 10

Conclusions

10.1 Results of the research

In this thesis, I have demonstrated a number of promising solutions to the problems of reliability and control of semiconductor lasers in large-scale multiprocessor networks.

For transient errors, an error-detecting code is preferred over an error-correcting one, because the additional circuit complexity for the error correction is not warranted by any significant system-level benefit. An error-detecting Hsiao code can be implemented at a bit overhead of around 12% will suffice to relax the Bit-Error-Rate requirement to a very tractable level, such as 10^{-7} or 10^{-8}. This is a systematic code that can be coded and decoded in parallel with the actual data transmission.

Given such leeway in the permissible error rate, my experiments have shown that it is feasible to use the link Bit-Error-Rate as a feedback variable to control the laser drive current level. The experimental feedback system is stable, with a gain margin of at least 10 dB. Such a feedback system can control both for threshold current variation (that is, temperature or age) but also for optical medium degradation, as long as the feedback system is given control of laser pulse current as well as the bias level. The intelligent drive control system also offers the possibility of controlling laser wearout failures by tracking laser wearout trends and scheduling timely replacement.

For the problem of laser failures, I have shown a switch design which enables the use of redundant spare optical links to replace failed ones. The substitution switch seems well suited to VLSI implementation. Provision of 10-15 spare links per channel seems to suffice to allow continued system operation until the effects of laser wearout

begin to assert themselves.

The use of Detour Routing to exploit redundant paths in the network topology offers some fault-tolerance benefit, but the new approach which I call Bandwidth Fallback offers a dramatically better improvement.

Bandwidth fallback allows the use of partially-failed channels via the provision of simple switching hardware which re-sizes the transmitted data to match the remaining data channel width. If an appropriate substitution switch is provided, Bandwidth Fallback can provide better fault tolerance than Redundant Sparing, even with no spare links at all. Even as it enters the laser-wearout time frame, such a system continues operation with a performance loss of only a few percent.

10.2 Topics for further study

My research presented here opens up several topics for further study. I list a few of the possibilities below. I have ordered the list roughly according to my judgement of their breadth: the first topics are potentially suitable for Masters-level research, followed by larger topics potentially suitable for doctoral theses, and ending with broad research fields, encompassing many different research opportunities.

1. It would be worthwhile to try VLSI implementations of the substitution switches and the bandwidth fallback switches from Chapter 9. Can these be implemented reasonably in VLSI?

2. Is VLSI implementation of large numbers of DAC's for intelligent laser drive control as straightforward as I suggest in Chapter 7?

3. How practical is VLSI implementation of the Hsiao-code-based EDC system I describe in Chapter 6? Combined with a substitution switch? A bandwidth fallback switch?

4. In Chapter 6, I assumed that transient errors on different channels would be independent of each other. This assumption is critical to my conclusion about the efficacy of error-detection coding. Is it valid on an actual parallel-path optical link? On a link under intelligent laser drive control?

5. As I mention in Section 4.3.5, I have assumed that laser failures will be independent. For lasers implemented together in an array, this assumption is patently false: failures of such lasers are obviously correlated. How does this alter my failure control conclusions?

6. The within-array correlation mentioned above offers an opportunity to simplify my intelligent laser drive control system: rather than control individual lasers, one could use the same control signal for all the lasers in an array. Does this yield adequate control? What impact does this have on the transient-error-control performance?

7. In Section 7.5.2, I suggest a number of possible benefits from a laser monitoring and replacement program based on laser wearout tracking. Based on a reasonable cost model, what benefits could be expected from such a monitoring program? Would it be worthwhile?

8. The VLSI devices suggested in items 1-3 above, along with the laser drivers and optical receivers, form the elements of an interface between the processors and the optical components (laser and receiver arrays). How closely can these elements be integrated? What role does electromagnetic interference in the receivers play?

9. The substitution and bandwidth-fallback switches I propose here are not specific to optical networks, but could be applied to electrical (wired) networks as well. Based on a reasonable model of electrical network failures, do my proposed solutions make sense in such a context? What are the critical differences between electrical and optical networks in this regard?

10. Increasing the laser output power makes the drive circuit more complex and shortens laser life, but it simplifies receiver design. What impact do my solutions proposed here (EDC, laser drive control, bandwidth fallback) have on the best choice of laser output power level?

11. What impact do my proposed solutions have on optimal laser array parameters: threshold current, reliability, and array size?

12. Although I have discovered some information about the stability of the feedback system, it was only by the 'brute-force' method of increasing the feedback gain. What is an appropriate mathematical model of the control system and its stability?

13. More generally, my laser drive system controls two variables (bias and pulse current) based on the arrival times of link errors. What are the limits of such a control system? What is the best control strategy? Is there a theoretically optimal control algorithm?

14. I have considered the multiprocessor to have failed when there is insufficient connectivity to any node. Much work is now being done on fault-tolerant multiprocessing via re-allocation of work from failed nodes to working ones. How do these higher-level methods interact with the methods I suggest here? Is there a synergism from combining them?

15. Data in a processor can be conveyed either electrically or optically. As optical interconnection is made cheaper and more reliable, where will electrical signalling continue to be superior?

10.3 Feasibility of Optical Multiprocessor Networks

With the use of the techniques I have described here: error-detection coding, intelligent laser drive control, redundant-spare substitution switching, and Bandwidth Fallback (especially if the circuits involved can be effectively implemented in VLSI), semiconductor laser reliability and control should not be a bar to their use in large-scale multiprocessor networks.

Appendix A

Laser Drive Control Experimental Setup

The setup for the laser drive control experiments discussed in Chapter 7 is here described in detail.

A.1 Hardware

The overall expermental setup is shown in Figure 7-6. Not shown in the figure is an adjustable iris in the free-space optical path, which offers no obstacle to the light beam when open, and blocks 97.4% of the light beam when closed. Figure 7-7 is a photograph of the overall experimental setup. The optical components are mounted on a benchtop vibration-isolated optical table. From left to right they are: photodiode/receiver board, optical iris, and laser/driver board. Next to the optical table, one can see the computer interface box and the data link analysis board.

A.1.1 Laser/Driver Board

The laser/driver board is shown in Figures A-1 and A-2. On the front of the board, there is an aluminum block, holding a Corning 350110 aspheric lens, a Mitsubishi ML7761 laser diode (1300-nm wavelength), and an Analog Devices AD590KH temperature transducer. (Note that this particular lens was actually ill-suited to this application, having an anti-reflective (AR) coating designed for a wavelength of 775 nm, instead

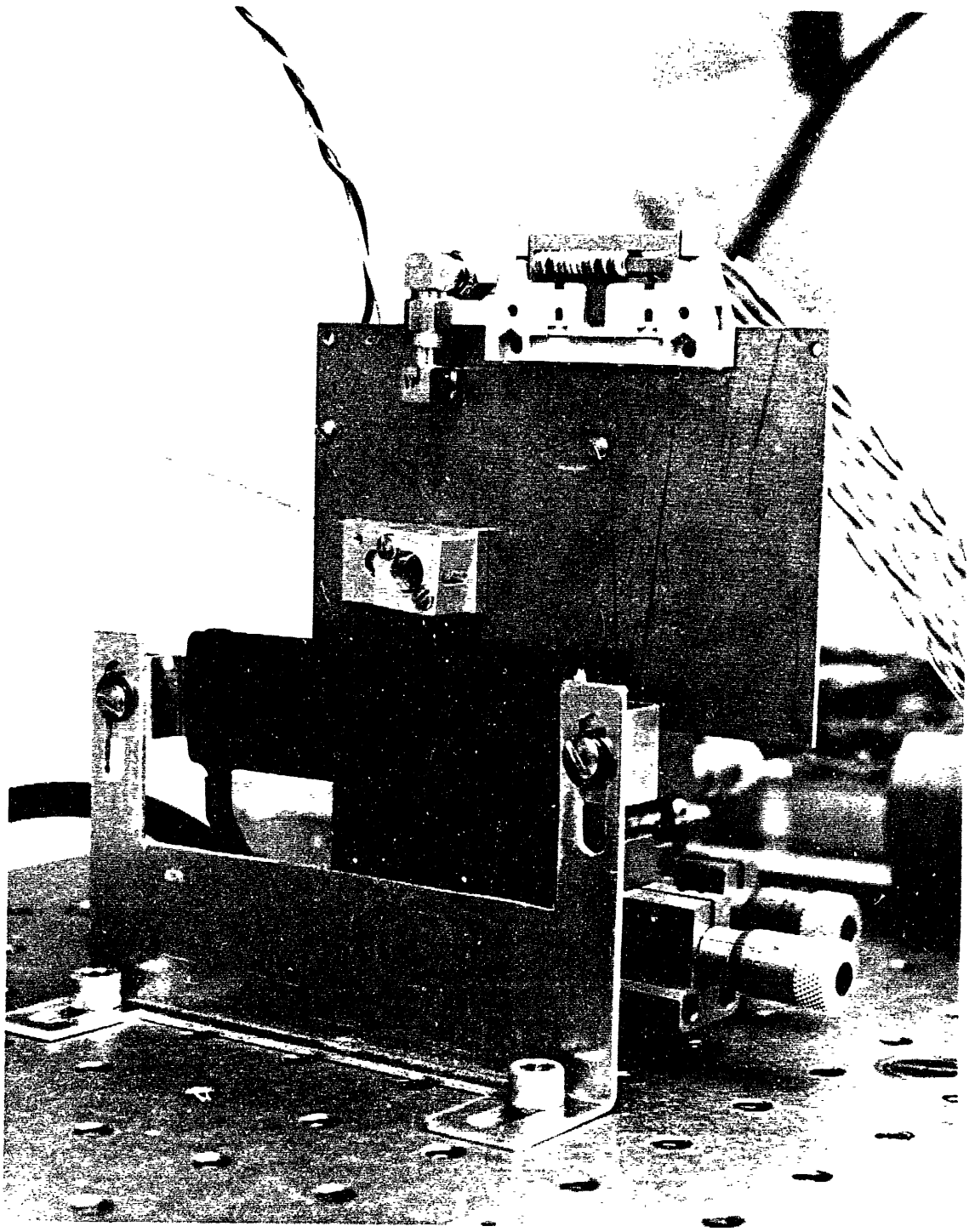

Figure A-1: Photograph of Laser/Driver Board, front view

of 1300 nm. This helps explain some of the anomalous light output readings seen in Section 7.4.4.)

The aluminum block and the laser diode were fixed in position when the board was constructed. The lens was then inserted, and its position adjusted for optimal collimation of the output beam. The lens was then fixed in position with a nylon setscrew.

In Figure A-1, there can also be seen a large power resistor mounted just in front of the laser/driver board, This was connected to line power through a variac, and was used to control the laser temperature in the temperature-based tests.

On the rear of the board, there is an NEL NL4512-2 laser driver circuit, a Motorola MC33074 operational amplifier, and several discrete components. A simplified schematic of the board is given in Figure A-3, omitting power bypass capacitors and similar details. All control inputs (bias current and pulse current) and monitoring outputs (temperature and light output) are current based, to avoid ground-loop problems and to ease the interconnection between the -4.5V-powered driver board and the $+5$V-powered computer interface.

Control of the NEL driver circuit was rather difficult to understand, and it required some experimentation to arrive at a working design. The data sheet (apparently translated from the Japanese) stated that the V_{CSDC} pin controlled the bias current and the V_{CSAC} pin controlled the pulse current. Connected this way, the circuit did not work correctly. Eventually, I arrived at the circuit shown in Figure A-3. The pulse and bias level current commands from the computer interface are applied to 270 resistors to V_{SS}, to develop the required control voltages (V_{SS} to $V_{SS} + 0.7$). The V_{CS} input presented an unexpected non-linear load, so that the addition of a buffer amplifier was needed to control it adequately.

The laser driver input is AC-coupled, to simplify level-shifting between the Gazelle data link analysis board output and the NEL laser driver circuit. Since the Gazelle board uses a balanced (zero-DC-bias) line code on its data output, we can use AC coupling without fear of baseline wander.

A.1.2 Photodiode/receiver board

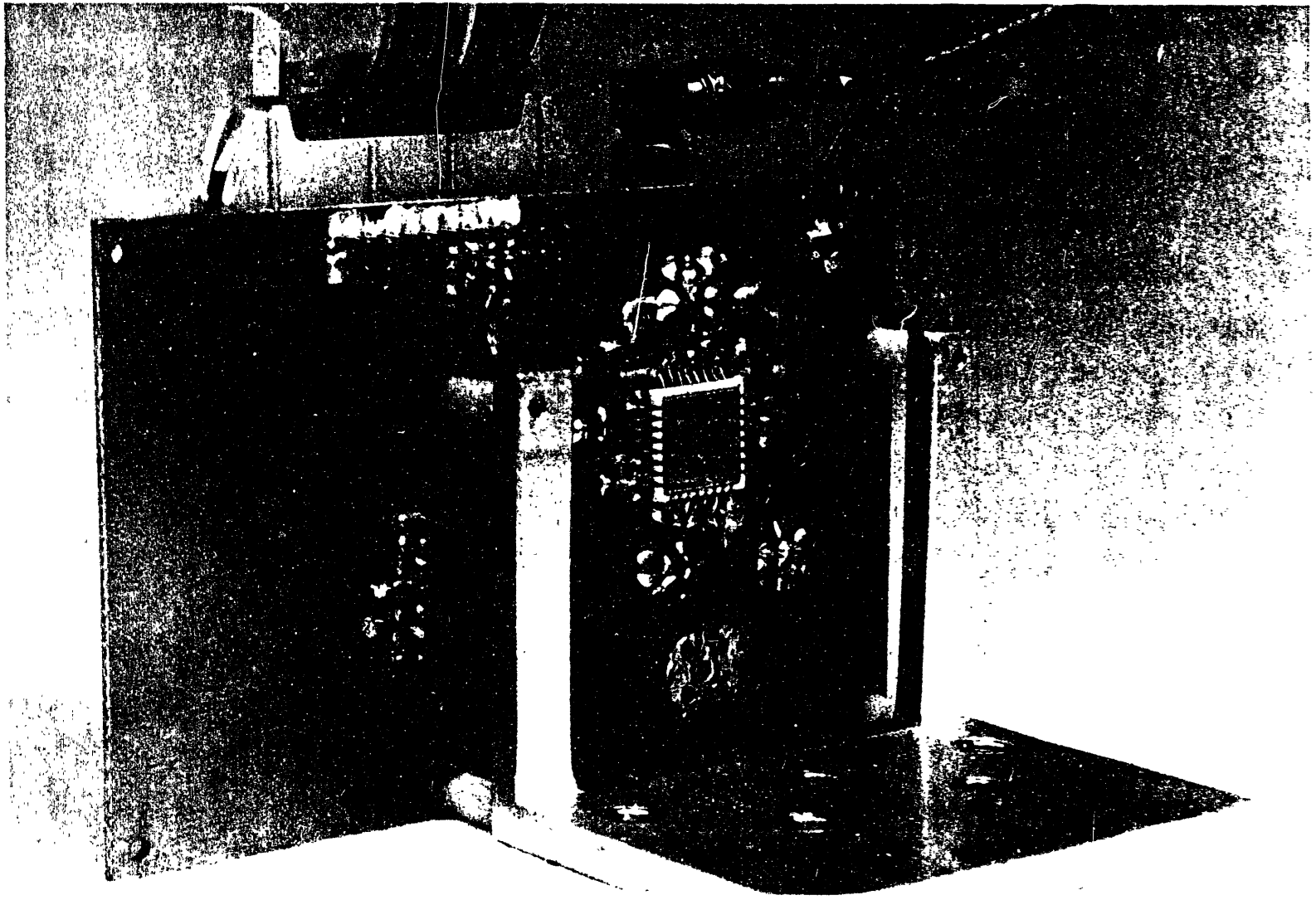

Figure A-2: Photograph of Laser/Driver Board, rear view

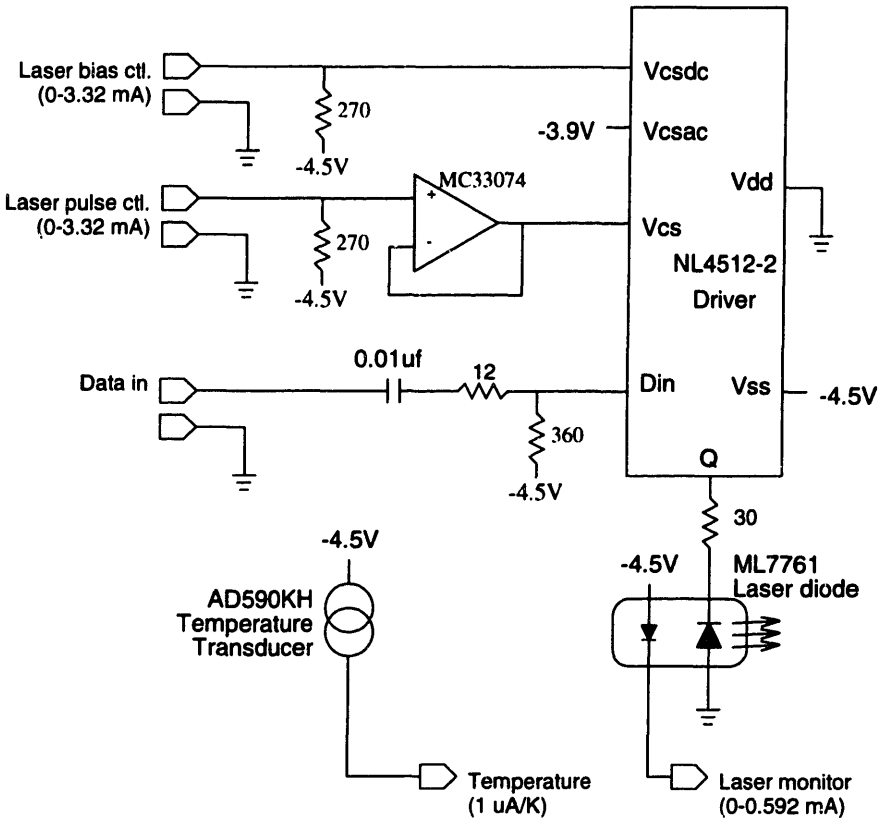

Figure A-3: Simplified Laser/Driver board Schematic

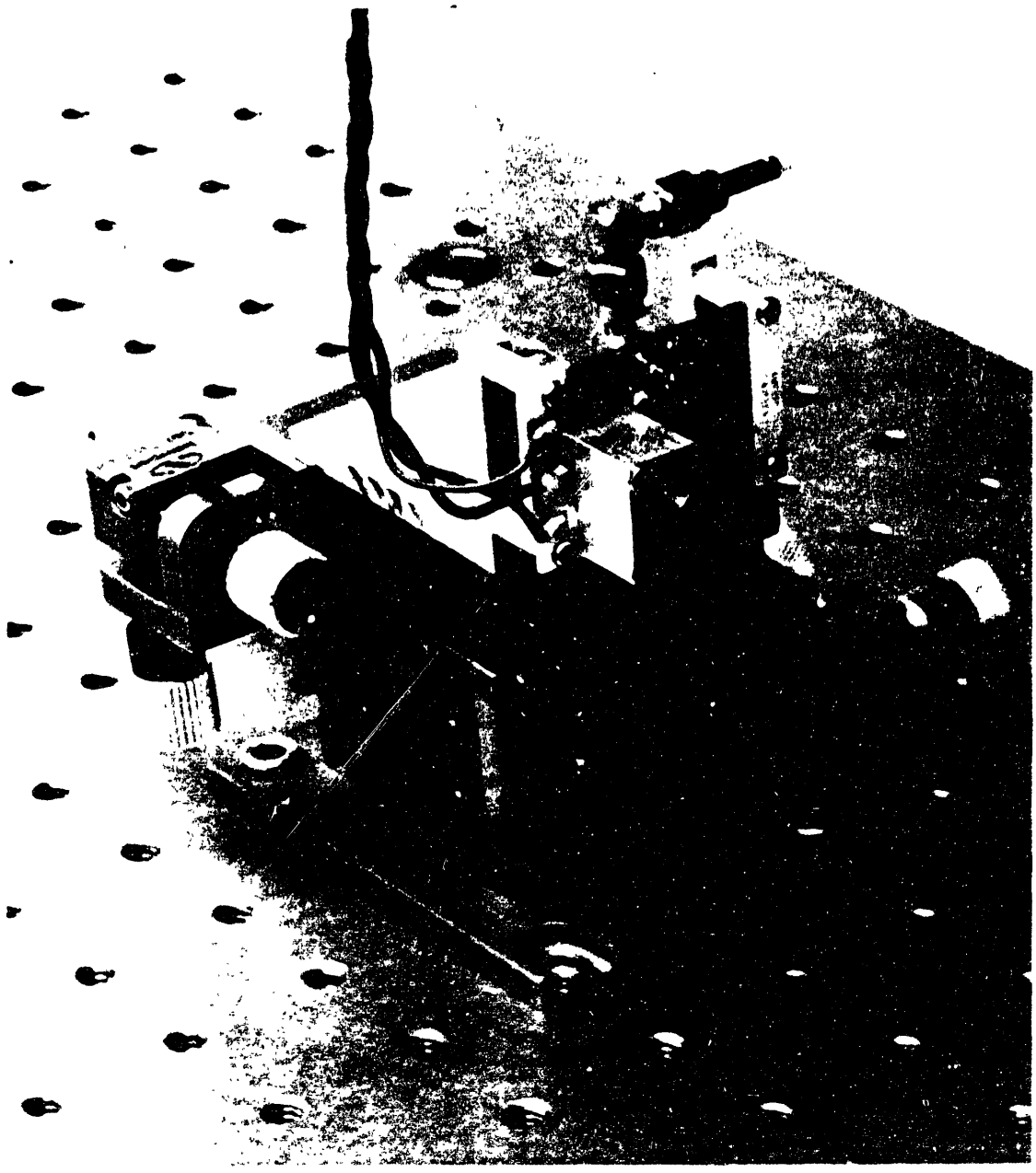

Figure A-4: Photograph of Photodiode/receiver Board

Figure A-4 shows the photodiode/receiver board, and the Newport optical iris used in the experiments. The photodiode/receiver board is much simpler than the laser/driver board, since it has no control inputs or monitoring outputs. Another Corning 350110 lens (also incorrectly AR coated) is mounted with a GE C30617 PIN photodiode. The lens/photodiode assembly is mounted and collimated in the same manner as the laser/driver board.

The photodiode current is amplified by an Avantek MSA-0370 amplifier before output to the Gazelle data link analysis board. As with the laser driver input, the photodiode output is AC-coupled.

A.1.3 Computer Interface

The computer interface box, shown in Figure A-5, provides a means for our UNIX workstation to control the laser drive level, and to read the temperature and laser monitor outputs for experiment logging. The interface is based on a Motorola MC68HC11E2 microcontroller, which includes an eight-channel Analog-to-Digital converter on-chip.

The schematic of the analog section of the interface is shown in Figure A-6. Two Analog Devices AD558 eight-bit D/A converters develop output voltages between 0 and 2.56V on command from the microcontroller. Each of these voltages is then converted to current by an op-amp and FET. As noted above, all signals between the computer interface and the laser/driver board are current-mode signals. The laser monitor and temperature current signals are converted to voltage, and sent to the on-chip A/D converters.

The microcontroller communicates with a UNIX workstation via a 9600-baud RS-232 serial interface. It was originally connected directly to a workstation, but the connection was later transferred to an Annex terminal server, which communicates with the workstation via an Ethernet connection.

A.1.4 Data Link Analysis Board

The data link analysis board is a Gazelle HRDS 'HOT ROD' Development System, shown in Figure A-7. In these experiments, it generates a test data pattern and transmits it at a 1 Gbit/sec line rate, using a DC-balanced line code, to the laser/driver

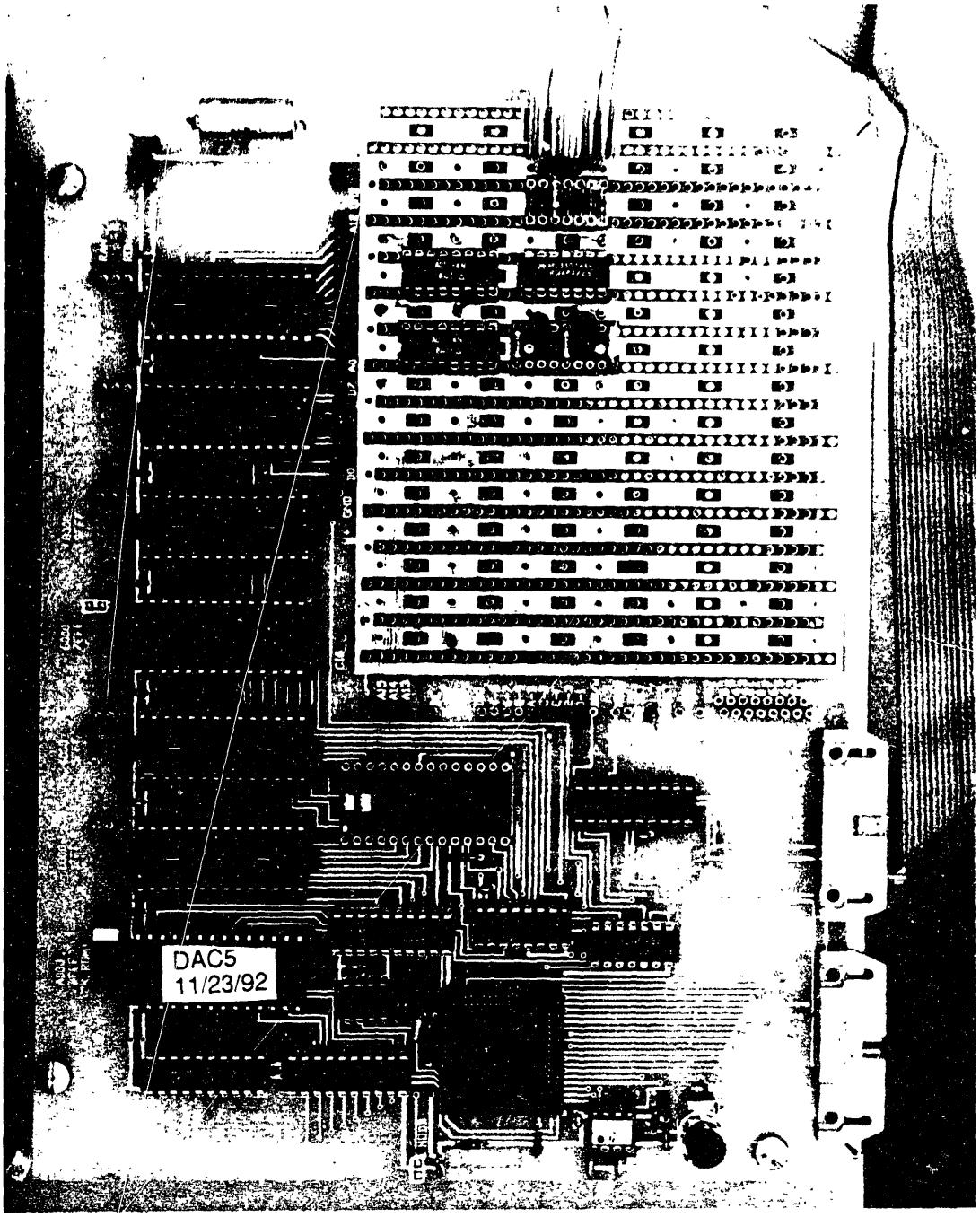

Figure A-5: Photograph of Computer Interface

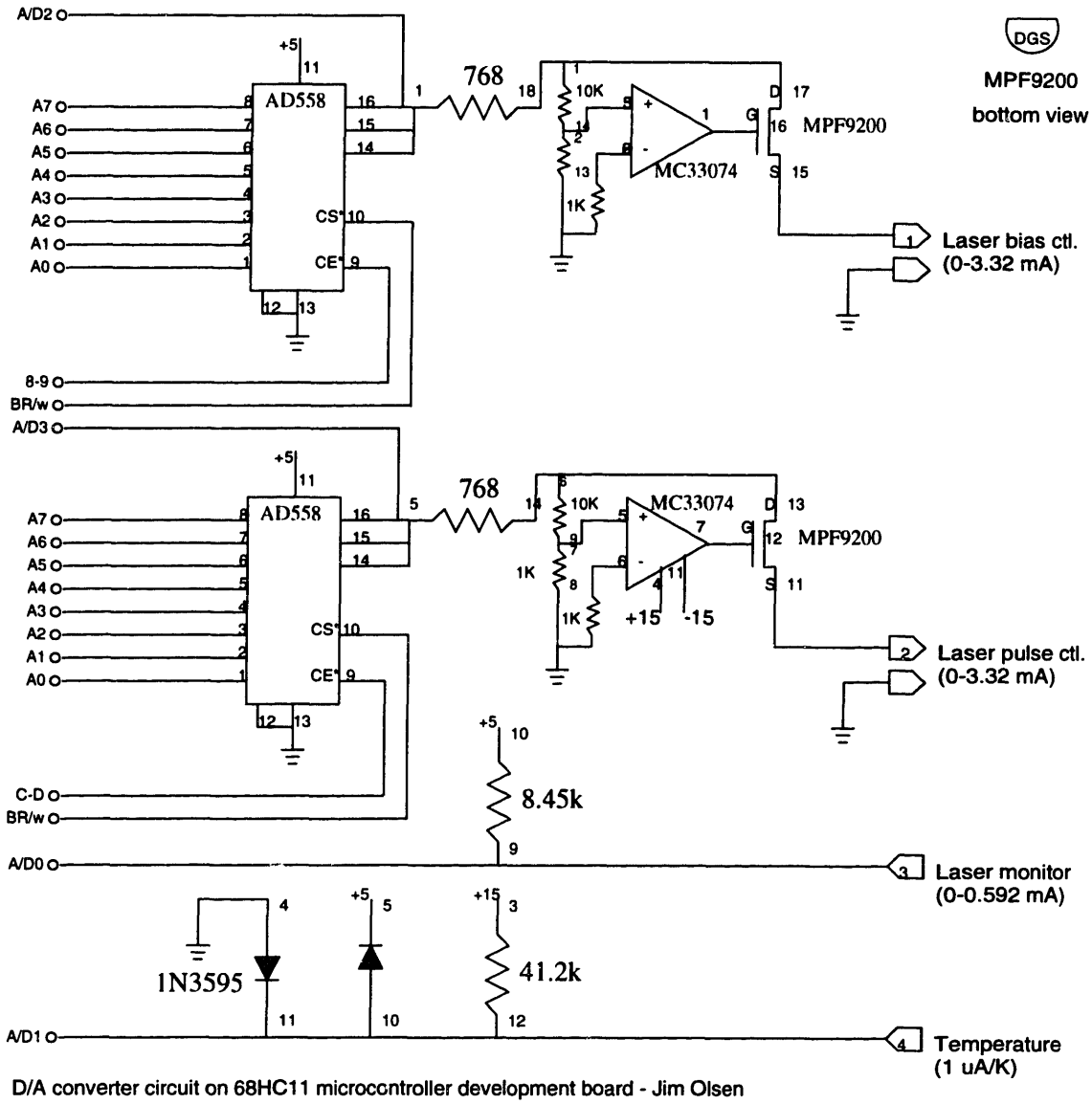

Figure A-6: Computer Interface Schematic (analog section)

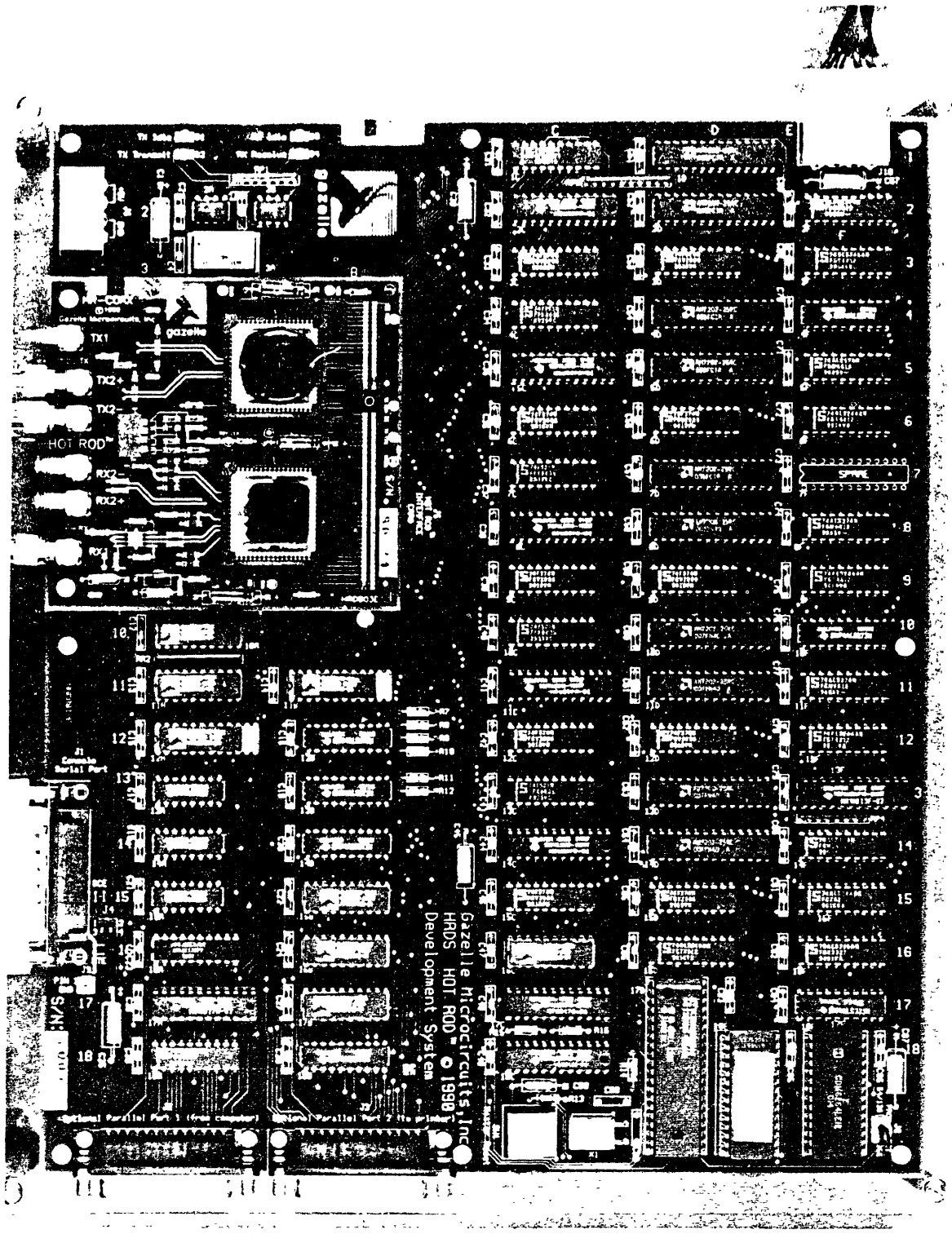

Figure A-7: Photograph of Data Link Analysis Board

board. It receives the resulting data stream from the photodiode/receiver board, and compares the received and transmitted data, keeping a running count of errors.

The analysis board is actually intended to facilitate design of systems using the Gazelle 'HOT ROD' communication boards, one of which can be seen in Figure A-7, mounted as a daughterboard on one corner of the main Gazelle HRDS board. The HRDS board actually generates and receives data in 40-bit-wide words, at 1/40th of the line rate, via the HOT ROD. The HOT ROD performs parallel-to-serial and serial-to-parallel conversions with some of Gazelle's GaAs integrated circuits.

The Gazelle board has a 4800-baud RS-232 serial interface for connection tc a terminal, through which the operator can start and stop the data gathering, and get error reports. Instead of a terminal, this serial interface was connected to the UNIX workstation, through the Annex terminal server mentioned above. The software described in the next section had therefore to emulate a person sitting at a terminal in order to control the Gazelle board.

A.2 Software

The laser drive control experimental software is written in C, and runs on a UNIX workstation. A Sun Sparcstation was used, but any comparable UNIX computer would have sufficed.

The first problem to be solved was interfacing with the experimental equipment. My custom-made computer interface was not a problem, since its communication protocol could be changed at will, by modifying the microcontroller firmware. The Gazelle board was another matter. Its protocol (assuming a human operator) is fixed into proprietary firmware whose source code is unavailable to me. I had therefore to make the UNIX workstation emulate a person.

Fortunately, tools for doing this are readily available. I used the 'expect' library of Libes [47] (itself based on the 'Tk' library of Ousterhout [48]) which was expressly written for this sort of operation. The feedback control program handles all the hardware interface chores via 'expect' library routines.

The feedback program's strategy is implemented via a finite-state-machine (FSM), described in Section 7.4.2 and diagrammed in Figure 7-8. The state machine's basic

timing was limited by the slow interface to the Gazelle board. Since the board's control firmware insists on sending two screenfuls of text, at 4800 baud, in response to every error-status query, the error status could only be obtained every 1.3 seconds.

As I mention in Section 7.4.9, the basic cycle of the FSM entails gathering three error measurements, which will take 3.9 seconds. If the number of errors during that time does not exceed a user-specified threshold (in the experiments shown in Chapter 7, the threshold was one error), then the error rate is judged to be acceptable. If, after any of the three error readings the error threshold is exceeded, the FSM cycle ends immediately, and the error rate is judged to be too high.

Every three seconds, the feedback program writes an entry into an experiment log file. The entry gives the following parameters:

- laser bias current
- laser pulse current
- bit-error-rate
- temperature reading
- laser output monitor reading

The experimental data plots in Chapter 7 are derived from these log files.

Appendix B

Network Simulation Software

The bandwidth-fallback simulation results presented in Chapter 9 were produced by a suite of simulation programs I wrote in C, running on a UNIX workstation.

I separated the simulation task into stochastic and deterministic components, and wrote different programs for each.

The stochastic tasks were performed by two relatively simple programs: a laser failure simulator and a packet generator. The laser failure simulator took as parameters the laser failure probability, channel width, and network mesh size, and generated a file indicating how many lasers had failed in each channel by a simple Bernoulli process.

The packet generator took as parameters the network mesh size, number of packets for each node to send, and the maximum packet size. It then created descriptions of the desired number of packets for each node, specifying the destination node and the packet length (in flits). The length was chosen as a uniformly distributed random variable between 1 and the maximum packet length. The choice of destination node was a little more constrained. Rather than assign destinations completely at random, the packet generator ensured that each node received and sent exactly the same number of packets. It therefore constructed a pool of packets, with each node being the destination of an equal number of packets. The pool was then shuffled and dealt out to the nodes; this process determined which node would be the sender of the packet. All the packet information was then written out onto a file.

The rest of the simulator was deterministic. It received as parameters an ensemble

of laser-failure and packet-assignment files, performed simulations on each combination of them, and reported the mean and deviation of the performance results from the various simulation runs. For each run, the simulation result was the estimated number of cycle required for all the nodes to transfer all their data packets.

The structure of the simulation program might be more easily understood if we examine some of the major data structures.

```
struct node_struct {
    int sleep_time,xfers_left;
    unsigned char x,y;
    struct node_struct *next; /* linked list for events */
    struct channel_struct ch[4];
    /* the following data describe the packet
       originating at this node */
    struct xfer_struct *my_xfer;
    enum {PK_READY,PK_OPENING,PK_XFERING,PK_CLOSING} state;
    int min_fract;
    struct channel_struct *next_chan,**detour_chan;
    struct vc_struct *head_vc;
};
```

The node structure is the most important one in the program. The structures describing the four channels leaving this node reside here. Because each node can have at most one active packet at a time, the packet information is kept here too. We see that the packets go through three phases: pushing through the network (establishing a path), transferring data, and tearing down the path. The packet also keeps track of the smallest fallback fraction on its path, as this affects the packet's throughput. This is an event-driven simulator, so the nodes are kept on a linked list which functions as an event queue.

```
/* virtual channel structure */
struct vc_struct {
  struct node_struct *user;
  struct vc_struct *prev;
  struct channel_struct *c;
};

struct channel_struct {
  int fraction,n_vc,n_users;
  struct vc_struct vcs[MAX_VC];
```

```
  struct node_struct *src,*dest;
  struct channel_struct *prev,**detour;
};
```

The channel structure defines the basic topology of the network via its `src` and `dest` pointers, although this might be overkill for a simple mesh network. Each channel is associated with four virtual channels, so a packet is refused entry to an actual channel only when all four of its virtual channels are used up. However, we must make special provision for the Virtual Channel detours around failed channels. If a channel forms a part of such a detour path, then that detour path has one of the virtual channels dedicated to it exclusively. This is necessary to avoid deadlock: if normal transactions could freeze out a detour path, then the detour would no longer be logically equivalent to the channel it replaces.

At the start of the simulation, one `node` structure is created for each node in the network to be simulated. The packet data is read from the packet-generator output file, and the channel capacity is read from the laser failure file. Each channel is checked for adequate remaining capacity; it capacity is inadequate (and the network can't fix it with the given network parameters), then the simulation run is declared 'failed' and aborted. In the results given in Chapter 9, I counted such runs as zero performance, and averaged this with the performance of the non-failed runs, if any. The performance of normal runs was calculated as the normal (no-failure) execution time divided by the actual execution time with failures.

The simulation results were based on runs of a 10×10 mesh network, with 64-bit-wide paths, 16-flit maximum packet length, and 32 packet transfers per node. The simulations were performed for 11 values of the laser failure probability $P(t)$, from 0 to 0.2 in steps of 0.02. These results were then transformed into system lifetimes using the model given in Section 4.3.4.

Bibliography

[1] Ian M. Ross. Electronics/photonics technology: Vision and reality. *Proceedings of the SPIE*, volume 1389:2–26, 1990.

[2] David A. B. Miller. Optics for low-energy communication inside digital processors: Quantum detectors, sources, and modulators as efficient impedance converters. *Optics Letters*, 14(2), January 15 1989.

[3] S. Ryu et al. Field demonstration of 195 km-long coherent unrepeatered submarine cable system using optical booster amplifier. *Electronics Letters*, 28(21):1965–1967, October 1992.

[4] Mitsuo Fukuda. *Reliability and Degradation of Semiconductor Lasers and LEDs*. Artech House, Norwood, MA, 1991.

[5] Aloke Guha et al. Optical interconnections for massively parallel architectures. *Applied Optics*, 29(8):1077–1093, 10 March 1990.

[6] Davis H. Hartman. Digital high speed interconnects: A study of the optical alternative. *Optical Engineering*, 25(10):1086–1102, October 1986.

[7] P. R. Haugen, S. Rychovsky, and A. Husain. Optical interconnects for high speed computing. *Optical Engineering*, 25(10):1076–1984, October 1986.

[8] Michael R. Feldman et al. Comparison between optical and electrical interconnects based on power and speed considerations. *Applied Optics*, 29(9):1742–1751, 1 May 1990.

[9] Thomas F. Knight and Alexander Krymm, Jr. A self-terminating low-voltage swing CMOS output driver. *IEEE Journal of Solid-State Circuits*, 23(2):457–464, April 1988.

[10] M. Nakamura and M. Obara. Advanced semiconductor devices for optical communication systems. *Toshiba Review*, 44(7), 1989. In Japanese.

[11] Thomas F. Knight, Jr. Technologies for low latency interconnection switches. In *Proceedings of 1989 ACM Symposium on Parallel Algorithms and Architectures*, pages 351–358. ACM, 1989.

[12] Robert W. Keyes. Physical limits in digital electronics. *Proceedings of the IEEE*, 63(5):740–767, May 1975.

[13] David A. B. Miller. Quantum-well devices for optics in digital systems. *Proceedings of the SPIE*, volume 1389:496–502, 1990.

[14] Jack L. Bufton. Experimental study of soliton transmission over many thousands of kilometers in fiber with loss periodically compensated by raman gain. In *Summaries of Papers Presented at the Conference on Lasers and Electro-Optics*, pages 150–151. Optical Society of America, 1989.

[15] American National Standards Institute. Ansi standard X3.166-1990, fibre data distributed interface, (FDDI) - token ring layer medium dependent (PMD), September 1990.

[16] Floyd E. Ross. Overview of FDDI: The fiber distributed data interface. *IEEE Journal on Selected Areas in Communications*, 7(7):1043–1051, 1989.

[17] A. G. Dickinson et al. Free-space optical interconnect using microlasers and modulator arrays. *Proceedings of the SPIE*, volume 1389:503–514, 1990.

[18] Dean Z. Tsang. Free-space optical interconnects. *Proceedings of the SPIE*, 994:73–76, 1988.

[19] L. A. Bregman et al. Holographic optical interconnects for VLSI. *Optical Engineering*, 24(10):1109–1118, October 1986.

[20] Charles T. Sullivan and Anis Husain. Guided-wave optical interconnects for VLSI systems. *Proceedings of the SPIE*, volume 881:172–176, 1988.

[21] Freddie Lin et al. Optical multiplanar VLSI interconnects based on multiplexed waveguide holograms. *Applied Optics*, 29(8):1126–1133, 10 March 1990.

[22] E. E. E. Frietman et al. Parallel optical interconnects: Implementation of optoelectronics in multiprocessor architectures. *Applied Optics*, 29(8):1161–1177, 10 March 1990.

[23] Hideki Kobayashi, Hirohisa Kanbara, and Ken'ichi Kubodera. Optical gating performance using a semiconductor-doped glass etalon. *IEEE Photonics Technology Letters*, 2(4):268–270, 1990.

[24] D. A. B. Miller et al. Field-effect transistor self-electrooptic device: Integrated photodiode, quantum well modulator and transistor. *IEEE Photonics Technology Letters*, 1(3):62–64, 1989.

[25] William J. Dally. *A VLSI Architecture for Concurrent Data Structures*, chapter 5. Kluwer Academic Publishers, Norwell, MA, 1987.

[26] Anant Agarwal. Limits of interconnection network performance. *IEEE Transactions on parallel and distributed systems*, 2(4):398–412, October 1991.

[27] Andrew W. Wilson, Jr. Hierarchical cache/bus structure for shared memory multiprocessors. In *Conference Proceedings - Annual Symposium on Computer Architecture*, pages 244–252. IEEE, 1987.

[28] Michael R. Feldman and Clark C. Guest. Nested crossbar connection networks for optically interconnected processor arrays for vector-matrix multiplication. *Applied Optics*, 29(8):1068–1076, 10 March 1990.

[29] W. Crowther et al. Performance measurements on a 128-node butterfly parallel processor. In *Proceedings of the 1985 International Conference on Parallel Processing*, pages 531–540. ACM, 1985.

[30] Tom Leighton. The role of randomness in the design of parallel architectures. In William J. Dally, editor, *Advanced Research in VLSI: Proceedings of the Sixth MIT Conference*, pages 177–178. MIT Press, 1990.

[31] Amnon Yariv. *Optical Electronics*. Holt, Rinehart, and Winston, 1985.

[32] O. Fujita, Y. Nakano, and G. Iwane. Screening by aging test for highly reliable laser diodes. *Electronics Letters*, 21(24):1172–1173, November 1985.

[33] S. P. Sim et al. High reliability InGaAsP/InP buried heterostructure lasers grown entirely by atmospheric MOVPE. In *Fourteenth European Conference on Optical Communication (ECOC)*, pages 396–399. IEE, 1988. IEE Conference Publication 292, part 1.

[34] William C. Athas and Charles L. Seitz. Multicomputers: Message-passing concurrent computers. *Computer*, 21(8):9–25, August 1988.

[35] David Chaiken et al. Directory-based cache coherence in large-scale multiprocessors. *Computer*, 23(6):49–59, June 1990.

[36] William J. Dally and Charles L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 36(5):547–553, May 1987.

[37] Herbert Sullivan and T. R. Bashkow. A large scale, homogeneous, fully distributed parallel machine, i. In *Conference Proceedings - Annual Symposium on Computer Architecture*, pages 105–117. IEEE/ACM, 1977.

[38] Christopher J. Glass and Lionel M. Ni. The turn model for adaptive routing. In *Conference Proceedings - Annual Symposium on Computer Architecture*, pages 278–287. ACM, 1992.

[39] Shu Lin and Daniel J. Costello. *Error control coding: fundamentals and applications*. Prentice-Hall, 1983.

[40] T. R. N. Rao and E. Fujiwara. *Error-Control Coding for Computer Systems*. Prentice-Hall, 1989.

[41] M. Y. Hsiao. A class of optimal minimum odd-weight-column SEC-DED codes. *IBM Journal of Research and Development*, 14:395–401, 1970.

[42] P. W. Shumate et al. GaAlAs laser transmitter for lightwave transmission systems. *Bell System Technical Journal*, 57(6):1823–1836, August 1979.

[43] Daniel P. Siewiorek and Robert S. Swarz. *The Theory and Practice of Reliable System Design*, chapter 3. Digital Press, Bedford, MA, 1982.

[44] William H. Press et al. *Numerical Recipes in C*, page 176. Cambridge University Press, 1988.

[45] William J. Dally. Virtual-channel flow control. In *Conference Proceedings - 17th Annual International Symposium on Computer Architecture*, pages 60–68. IEEE Computer Society, 1990.

[46] H. T. Kung and O. Menzilcioglu. Virtual channels for fault-tolerant programmable two-dimensional processor arrays. Technical Report CMU-CS-87-171, Carnegie-Mellon University, Computer Science Dept., 1986.

[47] Don Libes. expect: Curing those uncontrollable fits of interaction. In *Proceedings of the Summer 1990 USENIX Conference*, pages 183–192. USENIX Association, Berkeley, CA, June 1990.

[48] John K. Ousterhout. Tcl: An embeddable command language. In *Proceedings of the Winter 1990 USENIX Conference*, pages 133–146. USENIX Association, Berkeley, CA, January 1990.