INFORMATION TO USERS

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

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

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

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

AN OPTOELECTRONIC-VLSI CHIP WITH FORWARD ERROR CORRECTION TO IMPROVE THE RELIABILITY OF PARALLEL OPTICAL DATA LINKS

JULIEN FAUCHER

Department of Electrical and Computer Engineering
McGill University
Montreal, Canada
July, 2001

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements of the degree of Master of Engineering

ABSTRACT

Parallel optical interconnects (POIs) promise to deliver tremendous gains in bandwidth for applications such as massively parallel computing systems and telecommunication switches. At the core of any interconnect solution lies the fundamental problem of reliable transmission. Next-generation optical communication designers are running into hard limits when it comes to increasing data transmission rates and reducing errors. These two factors are typically in opposing balance: minimize bit error rate (BER) and data rates suffer, increase transmission rates and data integrity is compromised.

To answer these design challenges, this thesis proposes to use forward error correction (FEC) codes in an attempt to simultaneously improve data throughput, reduce BER and reduce power consumption. To verify that this is indeed possible, an optoelectronic-VLSI (OE-VLSI) chip with a FEC module was implemented in CMOS. This chip was designed in the context of a POI prototype system to demonstrate 1080 optical interconnections between two OE-VLSI chips.

SOMMAIRE

Les interconnections optiques parallèles promettent une augmentation considérable du débit de données nécessaire aux applications telles que les ordinateurs ultraparallèles et les commutateurs de télécommunication. Un des problèmes fondamentaux de toute interconnection est la fiabilité avec laquelle l'information peut être transmise. Les concepteurs d'interconnections optiques doivent donc se préoccuper à la fois d'augmenter la vitesse de transmission de données ainsi que de diminuer le nombre d'erreurs lors de la transmission. Ces deux facteurs sont diamétralement opposés. La réduction du taux d'erreurs sur les bits (TEB) nécessite habituellement une réduction de la vitesse de transfer. D'autre part, une trop forte augmentation de la vitesee de transfer peut compromettre l'intégrité des données.

Pour répondre à ces défis de conception, le présent mémoire propose d'utiliser l'autocorrection d'erreurs sans voie de retour afin d'améliorer la vitesse de transfer des données, de réduire le TEB, et de diminuer la puissance consommée. Pour vérifier que c'est en effet possible, une puce opto-électronique à intégratoin à très grande échelle avec un module d'autocorrection d'erreurs sans voie de retour a été conçue en CMOS. Cette puce faisait partie d'un projet de prototypage qui avait pour objectif de démontrer un lien optique parallèle ayant 1080 canaux.

ACKNOWLEDGEMENTS

I would like to thank Emmanuelle Laprise and Mike Venditti for their help and support ever since I joined the Photonic Systems Group. I consider myself lucky to have had a chance to work with such experienced and dedicated people. I learned a great deal from them.

I would also like to thank my supervisor, David Plant, for believing in me. He has always respected my choices and decisions and has always been very objective when I was seeking for an advice. Thanks for having been available even when we were a few states apart.

Thanks to Bodo Parady, who has given me the opportunity to do an internship at Sun Microsystems. He gave me a lot of insight on the topic of error-correction coding. Bodo was not only a good supervisor; he was also a wonderful hike partner. I am sure we will both remember the hikes, the mud, the sharks and the pancakes for a long time.

Thanks to Aaron Stark, from BAE Systems, for his help and support to make Demo 2 working. Thanks to BAE Systems for the hybridization and the packaging of the Demo 2 chip.

Thanks to Mike Ayliffe for being such a fun officemate. I am going to miss his technical knowledge, his advices and his qualities as a person.

Thanks to Mitch, Mike and Tomasz for all the squash games that we played. It truly helped me get rid of the frustrations.

I would like to thank the rest of the Photonic Systems Group: Marc, Andy, Maddy, Eric B., Eric^2 B., Feras, David R., Chris, Greg, Julianna, Alan, Fred L., Fred T.-D., Robert, Pierre-Olivier, Jean-Philippe, Rhys, Vikas, Sylviane, Kay, Lawrence, Keivan and any other person that I may have forgotten to list here. Thank you for providing a fun and interesting working environment over the past couple of years.

Emmanuelle, Mike and Dave, thank you for carefully reading my thesis and taking the time to help me improve its technical content and correct the grammatical errors.

Finally, thanks to my parents, Jacques, Martine, Marcel and Stéphanie who have always been very supportive and understanding when I was busy studying or trying to meet a deadline.

À mes parents, avec toute ma reconnaissance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Data-communication systems are required to handle ever-increasing data loads that strain their data throughput ability. Designers have made significant progress in increasing processor speeds, but new architectures like full system-on-chip (SOC) integration and the evolution of multiprocessors or dedicated machines won't be fully exploited until new high bandwidth interconnects for intra-chip, inter-chip, inter-board, or inter-computer communication are developed. Progress in the design of high-seed interconnection networks has been slow by comparison, and now represents the most significant bottleneck in today's information systems. The relatively low speed of communications between VLSI chips consumes increasing amounts of power in an effort to keep up with the faster processors. Moreover, the resistance capacitance (RC) time constants of long connections on and between chips limit the interconnect bandwidth [2, 3].

Optical interconnects satisfy a number of fundamental conditions to improve the interconnect performance regarding, for example, individual I/O channel bandwidth and the fabrication potential for massively parallel systems. The shortest interconnections are likely to remain electrical ones, due in part to the inverse relationship between electrical interconnection length and power consumption, and to a length independent minimum latency time inherent to optical interconnections caused by the time delays required for electrical-to-optical-to-electrical (E-O-E) conversion. The length at which optical interconnections become the better choice is technology dependent and lies somewhere in the mm-cm range [1]

Ideal interconnects are noise free, loss free, high bandwidth, free of latency and low cost [4]. Optical interconnections offer several advantages over metallic interconnections: higher bandwidth, higher interconnection densities, lower crosstalk, crosstalk which is independent of data rate, inherent parallelism, immunity from electromagnetic interference (EMI) and ground loops, the ability to exploit the third dimension, lower clock and signal skew, and a higher fan-in/fan-out capability. These advantages mean that optical interconnections have the potential to exhibit higher data rates at lower power consumption. Despite all these advantages, PODLs

are still an immature technology compared to electrical interconnections when packaging and reliability issues are considered. The integration of parallel optoelectronic interconnects (POIs) into high performance systems is a challenging task due to stringent technological requirements (such as power consumption, bandwidth, latency, sensitivity, homogeneity and compatibility with IC fabrication) as well as due to architectural problems encountered when designing complex systems [5].

1.1 Motivation

As bandwidth demands increase and the tolerance for errors decreases, designers of data-communication systems are looking for new ways to expand available bandwidth and improve the quality of transmission. Whether the medium is copper, aluminum, optical fiber, or free-space, the importance of getting digital data to its destination quickly is second only to getting the data to its destination without error. Whether reliable 2D interconnects with a few thousand channels can be built remains an open technological question. Yield, uniformity, and power dissipation of massively parallel interconnects are also major issues. This thesis suggests that forward error correction (FEC) coding could help with some or all of these issues. A more general term for FEC coding is "error-correction coding" (ECC). ECC can also stand for "error-correction codes". To verify the potential benefits of using ECC in a PODL, an optoelectronic-VLSI (OE-VLSI) chip with ECC was implemented in CMOS. This chip was designed in the context of a PODL demonstrator system (Demo 2) to demonstrate a link made of 1080 optical interconnections between two OE-VLSI chips (see Figure 1). Flip-chipped photodetectors (PDs) and vertical-cavity surface-emitting lasers (VCSELs) [12] provided optical inputs and outputs (I/Os) to the chips. An OE-VLSI chip will be referred to as "D2C" (for Demo 2 chip) throughout this thesis.

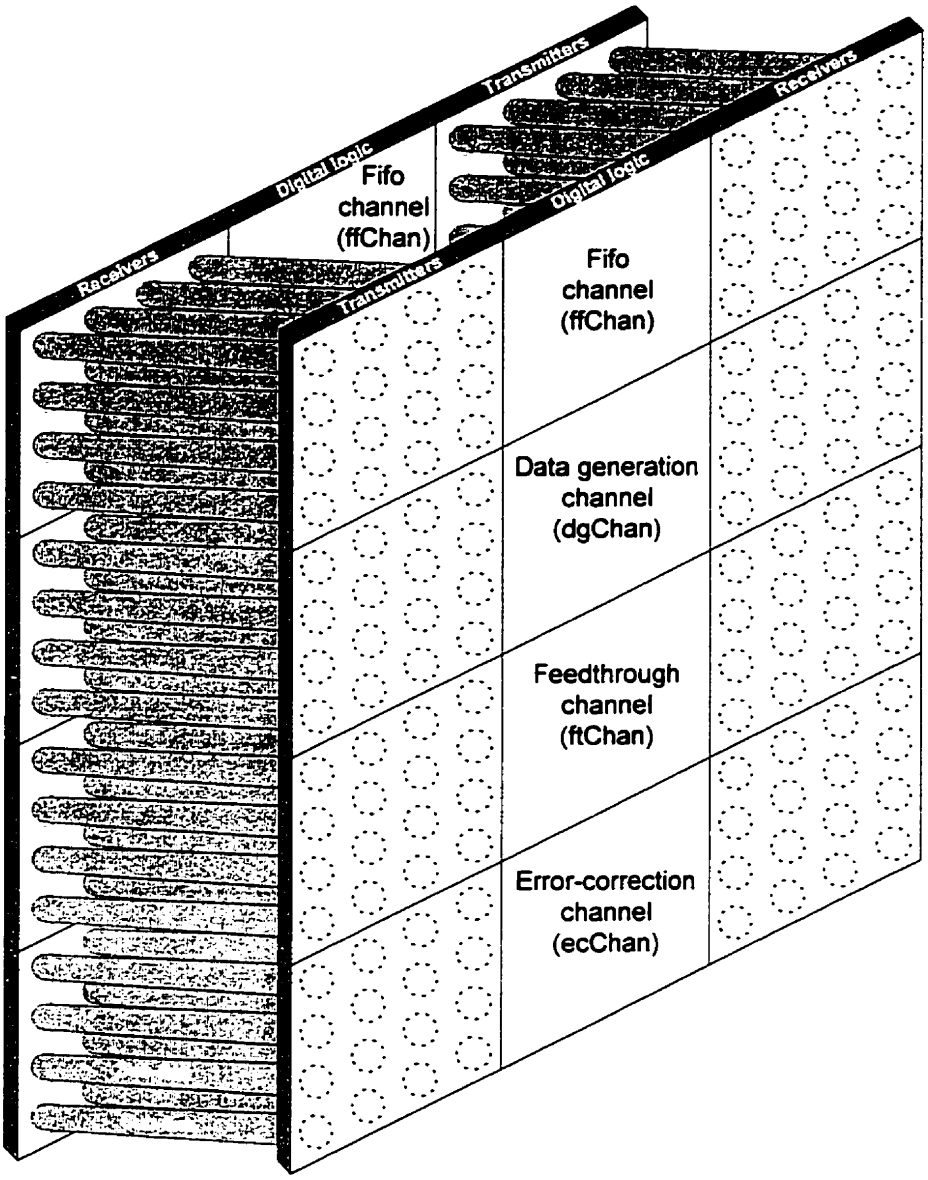


Figure 1: Two Demo 2 chips facing each other. There are 1080 optical sub-links between each of the two pairs of transmitter/receiver arrays. Note that no optics is shown in this figure. Either a fiber image guide (FIG) [9, 10, 11] or an ordered fiber array (OFA) should be used to image the transmitter arrays onto their respective receiver arrays.

An optical link between two OE-VLSI chips requires that, at a minimum, one chip has transmitters and the other has receivers. When the optical link has more than one transmitter and one receiver, it can be said to be composed of many sub-links, each sub-link creating a mapping between a transmitter and a receiver. In a massively parallel optical link, the sub-links are often grouped into channels. Figure 1 shows that the D2C is composed of four channels. This thesis will discuss the design and implementation of the error-correction channel only. The other three channels are discussed thoroughly in [8].

The error-correction channel, or *ecChan*, is divided up into two blocks: an encoder (*ecChanEnc*) and a decoder (*ecChanDec*) – see Figure 2, which puts *ecChan* in the context of the D2C. The encoder block is made of six identical sub-encoder blocks. Each sub-encoder encodes a 12-bit vector into a 24-bit vector before transmission. The redundancy is used on the decode side to correct errors. Each of the six sub-decoders decodes a 24-bit vector into a 12-bit vector, correcting up to three errors in the process.

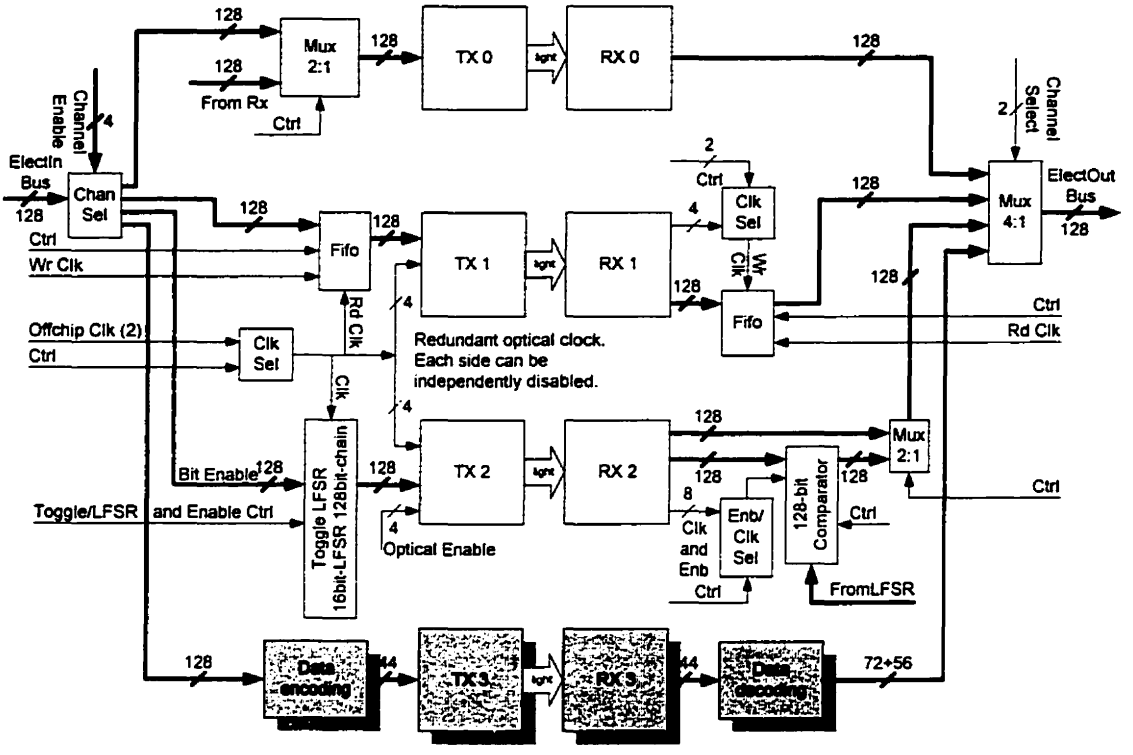


Figure 2: Block diagram illustrating the functionality of the four digital channels on the D2C.

Overall, there are 144 sub-links between the encoder and the decoder of *ecChan*. 72 of these sub-links are used to transmit information bits, whereas the other 72 are used as redundant sub-links for the purpose of ECC. In the ideal case, a total of 18 errors can be handled (three for each of the six encoder/decoder pairs). Unlike for telecommunication applications, all the encoding and decoding is performed in parallel to speed up the transfer of data.

1.2 Outline

This thesis is divided up into five chapters. Chapter 1 has motivated the need for ECC in PODLs. In Chapter 2, the types of errors that one is likely to find in a PODL will be categorized. The question of infant mortality in PODLs, which has not been addressed to date

[6], is a very important problem that will be tackled. The difference between the BER of a serial link and the BER of a parallel link will be explained qualitatively and quantitatively. This chapter will show that the distribution of the defects on an OE-VLSI chip may have a big impact on the performance of ECC. Finally, the yield requirement to obtain an acceptable BER will be evaluated.

In Chapter 3, the theory behind ECC will be covered. Two ways of performing error correction, namely "forward error correction" and "automatic repeat request", will be introduced. ECC is often perceived to cost information bandwidth owing to the transmission of redundant data for the purpose of correcting errors. The concept of coding gain and how it can be used to actually increase information bandwidth will be explained [7].

The hardware implementation of an ECC scheme will be presented in Chapter 4. The encoder and the decoder that were put on the D2C will be described in details. Because of some manufacturing delays as well as some yield problems with the hybridization process, only DC and near DC (kHz range) electrical and optical results will be presented.

In Chapter 5, performance improvement strategies that can be used in conjunction with ECC will be presented for future work. A concluding chapter will close the discussion.

1.3 References

1. D.A.B. Miller, "Rationale and Challenges for Optical Interconnects to Electronic Chips," Invited Paper, Special Issue of Proceedings of the IEEE (to be published).

2. D.A.B. Miller, and H. M. Özaktas, "Limit to the Bit-Rate Capacity of Electrical Interconnects from the Aspect Ratio of the System Architecture," Special Issue on Parallel Computing with Optical Interconnects, J. Parallel and Distributed Computing 41, 1997.

3. D.A.B. Miller, "Dense Optical Interconnections for Silicon Electronics," Trends in Optics: Research, Developments, and Applications, Vol. 3, 207-222, Ed: A. Consortini (Int'l Commission for Optics, Academic Press, 1996.

4. MEL-ARI OPTO Technology Roadmap, September 1999, http://www.cordis.lu/esprit/src/melop-rm.htm.

5. R. Lytel, H. Davidson, N. Nettleton, and T. Sze, "Optical Interconnections within Modern High-performance Computing Systems", http://www.sun.com/research/techrep/2000/abstract-84.html.

6. T.H. Szymanski, "Bandwidth optimization of optical data links by use of error-control codes", Applied Optics, Vol. 39, No. 11, 2000.

7. M.A. Neifeld, and R.K. Kostuk, "Error correction for free-space optical interconnects: space-time resource optimization", Applied Optics, Vol. 37, No. 2, 1998.

8. E. Laprise, M. Venditti, and J. Faucher, "VLSI-Photonics Demo 2 ASIC: Functional Specifications, McGill University, 2000.

9. T. Maj, A.G. Kirk, D.V. Plant, J.F. Ahadian, C.G. Fonstad, K.L. Lear, K. Tatah, M.S. Robinson, and J.A. Trezza, "Interconnection of a two-dimensional array of vertical-cavity surface-emitting lasers to a receiver array by means of a fiber image guide", Applied Optics, Vol. 39, pp. 683-689, 2000.

10. R.K. Kostuk, and J. Carriere, "Interconnect Characteristics of Fiber Image Guides", Applied Optics, Vol. 40, pp. 2428-2434, 2001.

11. D.M. Chiarulli, S.P. Levitan, P. Derr, R. Hofmann, B. Greiner, and M. Robinson, "Demonstration of a Multichannel Optical Interconnection by use of Imaging Fiber Bundles Butt Coupled to Optoelectronic Circuits", Applied Optics, Vol. 39, pp. 698-703, 2000.

12. D.V. Plant, M.B. Venditti, E. Laprise, J. Faucher, K. Razavi, M. Chateauneuf, A.G. Kirk, and J.S. Ahearn, "A 256 channel bi-directional optical interconnect using VCSELs and photodiodes on CMOS," to appear in IEEE J. Lightwave Technol., August 2001.

Chapter 2

Errors in Parallel Optical Data Links

This chapter will present qualitative and quantitative analyses of the different types of errors to expect in a PODL. Matlab simulations will be presented to demonstrate how much reliability can be gained from using ECC. A section will be devoted to the distribution of the errors in a PODL and the effect that it has on ECC. The last section will analyze the theoretical yield required to get a reliable link.

2.1 Error vs. erasure

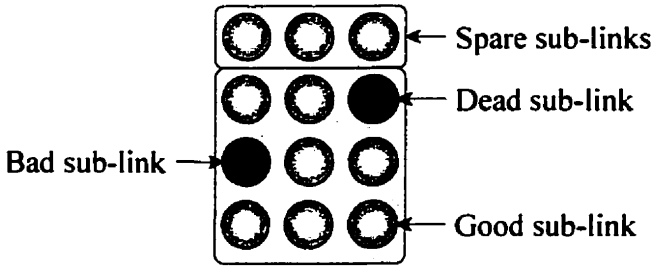


Figure 3: The four categories of sub-links in a PODL.

In looking for ways to improve the reliability of PODLs, one must learn about the types of errors that are likely to occur in the link. When transmitting data in parallel over an optical link – be it free-space [18], a FIG [15, 16, 17] or an OFA –, each sub-link in the link will belong to one of the following categories (see Figure 3): 1) good sub-link, 2) dead sub-link, 3) bad sub-link and 4) spare sub-link. A good sub-link is a sub-link that will provide a bit-error rate (BER) smaller than the target BER of the system. A dead sub-link is a sub-link on which no data can be transmitted. The cause of a dead sub-link can be any of the following: a dead VCSEL, a PD, a problem with the underlying CMOS circuitry, a dark fibre (if using a FIG or an OFA), or an alignment problem. Dead sub-links can also be referred to as erasures and are closely related to the term *infant mortality*. The bad sub-link category lies somewhere between the first two categories: data can be transmitted over a bad sub-link, but the BER is above the target BER of the system. In a PODL, bad sub-links could be attributed to skew or to some VCSELs having a higher threshold due to non-uniformities across the array. Spare sub-links, as their name imply, are used as spares in case there are too many erasures in the array.

2.2 Bit error rate (BER) vs. Packet error rate (PER)

In the telecommunication industry, the BER of a transmission link is a measure of its reliability. The BER is defined as the ratio of the number of bits that are transmitted erroneously over the total number of bits that are transmitted:

$$BER = \frac{\text{Number of bits transmitted erroneously}}{\text{Total number of bits transmitted}} \quad (1)$$

On one hand, for telecom applications such as optical communication networks, the minimum acceptable BER of the link is of the order of 10^{-12}. On the other hand, for datacom applications such as interprocessor optical interconnect networks, it is standard to require a BER of 10^{-15} or lower [3]. The Demo 2 demonstrator system falls into the category of datacom applications. Therefore, the target BER was 10^{-15} and the goal of this research was to demonstrate that ECC would help achieve this objective.

The BER is one way to assess the reliability of a serial communication system. We need to define a second measure to assess the reliability of a highly parallel system such as a PODL. This second measure is referred to as the word-error rate, the block-error rate or the packet-error rate (PER) [4]. The difference between BER and PER is that the former is a measure of the reliability of a serial link whereas the latter is a measure of the reliability of a parallel link. As we will see later, the PER of a parallel link is a function of the BER of the individual sub-links:

$$PER = f(BER_1, BER_2, ..., BER_n) \quad (2)$$

Where BER_1 is the BER of the 1st sub-link, BER_2 is the BER of the 2nd sub-link and so on. Ideally, the error-correction channel on the D2C should be tested with a parallel bit-error rate tester (parallel BERT) [9, 10]. A parallel BERT computes the PER as it computes the BER of each of the sub-links at an identical threshold level and sample times. An error in any of the sub-links at a given bit time equals an error in the received packet. When there is no ECC, the packet probability of error at a given sample time is therefore higher than the bit probability of error of the least reliable sub-link:

$$PER_{\text{NO ECC}} \geq BER_i \text{ for } i \in [1, n] \quad (3)$$

When simulation results are presented in the next section, it will become clear that Equation 3 does not necessarily hold when ECC is added in the link.

2.3 Simulation results

Let the BER of an optical sub-link before the implementation of any error control be denoted by p_{bc} (probability of a bit-error). Assuming that each individual sub-link has an identical p_{bc} (this assumption reduces computational time significantly), the probability of finding exactly t errors in a n-bit packet is [5]:

$$P(t) = \binom{n}{t} p_c^{\,t} (1 - p_c)^{n-t}, \tag{4}$$

where

$$\binom{n}{t} = \frac{n!}{t!(n-t)!}. \tag{5}$$

Equation 4 is known as the *binomial probability law* [2]. Equation 5 (read the left hand side of the equation as "n-choose-t") is the binomial coefficient. It represents the number of ways of picking t things in a pool of n things – i.e. the number of combinations.

A Matlab script was written to simulate the behaviour of a 24-bit decoder (i.e. n = 24) when confronted with erasures. Let PER_t refer to the PER obtained with an ECC scheme capable of correcting t errors. An ECC scheme with an error-correction capability of t will fail when confronted to t+1 or more erasures:

$$\begin{aligned} PER_t &= \sum_{z=t+1}^{n} P(z) \\ &\cong P(t+1) \text{ for } p_{bc} << 1 \end{aligned} \tag{6}$$

Figure 4 (a) shows a plot of $P(z)$ for z = 1, 2, 3 and 4, where $z = t + 1$, and Table 1 summarizes the main results. The decoders on the D2C can correct up to three errors. Their PER will therefore correspond to PER_3 – i.e. P(4) – in Figure 4. To confirm that the approximation in Equation 6 is valid, we will consider the case of an arbitrary p_{bc} of 10^{-8} and no erasure. There is no particular reason for choosing a p_{bc} of 10^{-8} in the analysis that follows, except that it is a relatively high value compared to the target BER of 10^{-15}. From Equation 4, $P(4) = 1.06 \times 10^{-28}$ and $P(5) = 4.25 \times 10^{-36}$. Since $P(5)$ is many orders of magnitude smaller than $P(4)$, the latter is a

very good approximation of the PER. In fact, many computing devices or software programs such as Matlab would not be able to take advantage of the accurate expression in Equation 6. This is due to the round off that occurs when adding or subtracting numbers that differ by many orders of magnitude.

Table 1: PER of a 24-bit PODL for various error-correction capabilities.

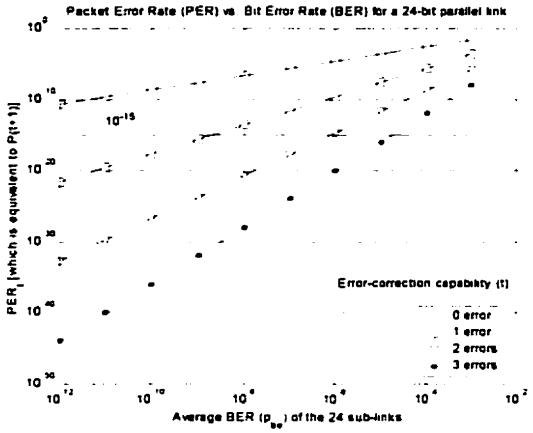
Number of erasures	Error-correction capability 0		1		2		3	
0	2.40×10^{-7}	1	2.76×10^{-14}	1	2.02×10^{-21}	1	1.06×10^{-28}	1
	"		"		"		"	
1	1	2	2.30×10^{-7}	2	2.53×10^{-14}	2	1.77×10^{-21}	2
	0.5		1.15×10^{-7}		1.26×10^{-14}		8.85×10^{-22}	
2	1	2	1	4	2.40×10^{-7}	4.4	2.76×10^{-14}	4.8
	0.5		0.25		5.50×10^{-8}		5.77×10^{-15}	
3	1	2.7	1	2.7	1	8	2.40×10^{-7}	9.1
	0.375		0.375		0.125		2.63×10^{-8}	

Ratio

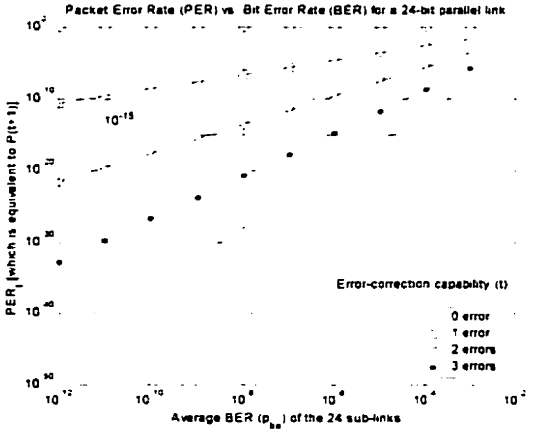
PER for balanced data

PER for unbalanced data

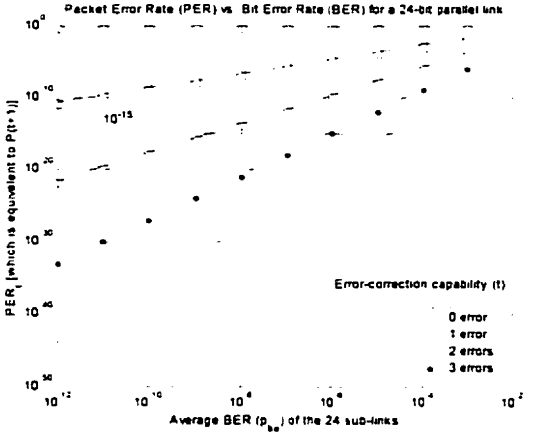
The solid lines in Figure 4 correspond to the performance of a 24-bit PODL with no erasures and appear throughout the figure to serve as a reference. Consider for example Figure 4 (b), which depicts the case of a PODL with one erasure. With an error-correction capability of three, this PODL would be as reliable as a PODL that has no erasure and an error-correction capability of two.



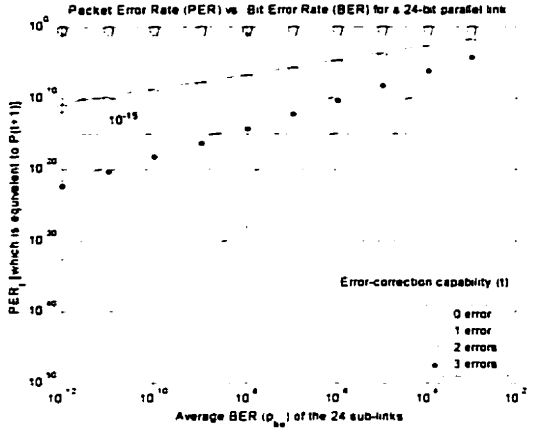
(a) $PER_3 = 1.06 \times 10^{-28}$ (no erasure)



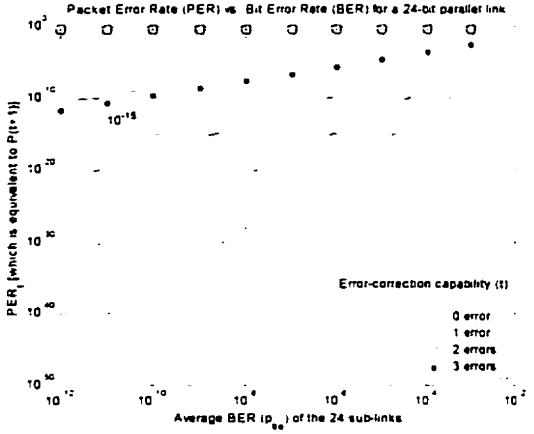
(b) $PER_3 = 1.77 \times 10^{-21}$ (one erasure)



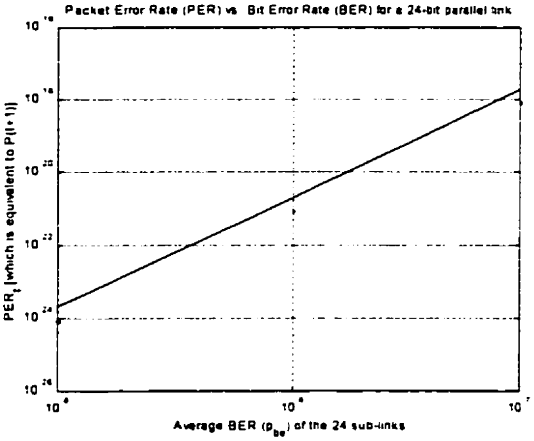
(c) $PER_3 = 8.85 \times 10^{-22}$ (one erasure)



(d) $PER_3 = 5.77 \times 10^{-15}$ (two erasures)



(e) $PER_3 = 2.63 \times 10^{-8}$ (three erasures)



(f) Blow up of (c) (one erasure)

Figure 4: PER vs. BER for a 24-bit parallel link. (a) no erasure, (b) one erasure, assuming that an erasure has a BER of 1, as opposed to 0.5 for all the other figures, (c, f) one erasure, (d) two erasures, (e) three erasures. The PERs listed below each figure are for a relatively high average BER (p_{be}) of 10^{-8}. Note that the performance improvement obtained by using balanced data is emphasized by the fact that the lines (unbalanced data) do not go through the data points (balanced data) in some of the figures.

If no ECC is used in a PODL, a received packet will differ from the transmitted packet if at least one bit was transmitted erroneously. The PER without ECC is therefore:

$$\begin{aligned} PER_0 &= P(1) + P(2) + \ldots + P(n) \\ &= 1 - P(0) \\ &\cong P(1) \end{aligned} \tag{7}$$

While *P(0)* is the probability of transmitting without any error, *P(1)* is the probability of transmitting with a single error and is a very good approximation of the PER of a PODL without ECC. From Equation 7, PER_1 is about 2.4×10^{-14} for a p_{be} of 10^{-15}. Hence, the PER may be acceptable even without ECC if the BER of each individual sub-link is sufficiently small. On the other hand, PER_1 is about 2.4×10^{-7} for a p_{be} of 10^{-8}. Clearly, if the sub-links of a PODL had such a BER, the system would not be able to meet the 10^{-15} PER requirement. ECC can help decrease this PER dramatically. The vertical in Figure 4 (a) shows that the PER is reduced by about 7 orders of magnitude for every additional bit that can be corrected: a 1-error, 2-error and 3-error ECC scheme would output a PER of 2.76×10^{-14}, 2.02×10^{-21} and 1.06×10^{-28} respectively.

Balanced data vs. unbalanced data

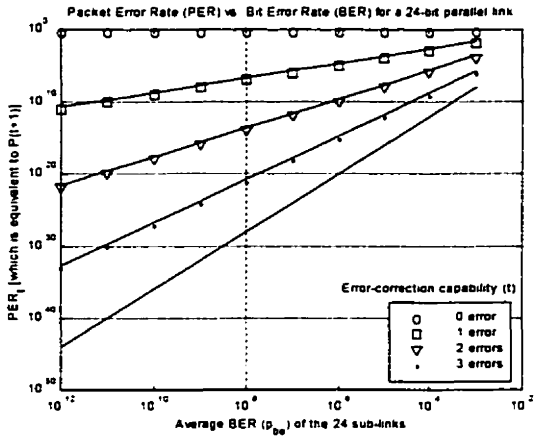
Figure 4 (a) assumed that all the sub-links were "good" (see Section 2.1 for the definition of a good sub-link). The effect that dead sub-links (erasures) have on the PER will now be presented. Suppose that the PODL has one erasure. The probability p_{be} that a bit is transmitted erroneously over that sub-link is 0.5 if the data is balanced, but could be as bad as 1 for unbalanced data. Balanced data means that the number of ones and the number of zeros transmitted over a sub-link are approximately equal. Unbalanced data occurs when long strings of 1's and 0's are transmitted. Two ways of balancing the data is to use Manchester encoding [6] or 8b/10b encoding [7].

Figures 4 (b), (c) and (f) all depict a 24-bit PODL with a single erasure. When comparing the three figures, the effect of having balanced data becomes clear. The PER_1 read from Figure 4 (c) and (f) is 8.85×10^{-22}, whereas the PER_1 read from Figure 4 (b) is 1.77×10^{-21}. This is a twofold improvement, which means that half as many errors will occur for balanced data. If there are three erasures, then using balanced data will reduce the number of errors by a factor of 9 ($2.40 \times 10^{-7} / 2.63 \times 10^{-8} \cong 9.1$) – see Figure 4 (e).

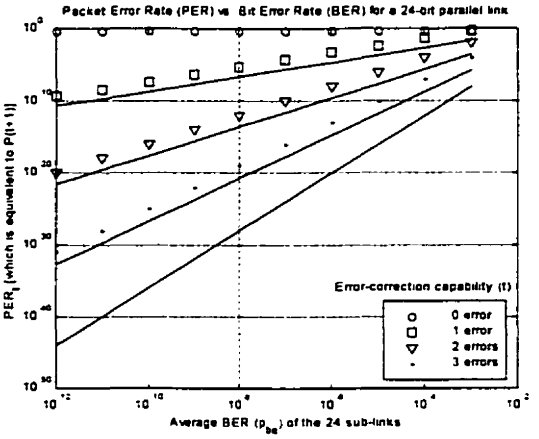
Figure 4 (c) shows a PER_3 of 8.85×10^{-22} for one erasure. Figure 4 (d) shows a PER_3 of 5.77×10^{-15} for two erasures. This means that the target PER of 10^{-15} should be met by the D2C even if there is one erasure per 24-bit packet. The target PER should almost be met if there are two erasures. This interesting result assumes a relatively large p_{bit} of 10^{-8}. If the D2C has to deal with three erasures in a 24-bit packet, then the PER_3 of 2.63×10^{-8} will be very far from the target PER of 10^{-15}. In this case, there could be two ways to decrease the PER to an acceptable value: increase the yield such that there are less erasures, or try to decrease p_{bit} by (1) checking the alignment of the PODL, (2) optimizing the bias and modulation currents of the transmitters and (3) optimizing the gain of the receivers [11, 12].

Bad sub-links

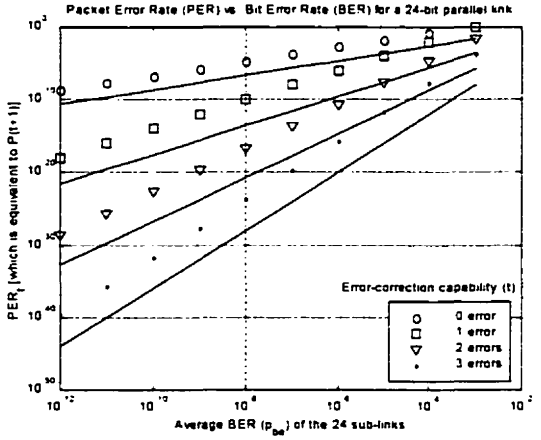
Figure 4 showed how ECC performs when dealing with erasures in a 24-bit PODL. We will conclude this section by looking at how ECC performs when it has to deal with bad sub-links in addition to erasures (see Figure 5). The quality of a bad sub-link lies somewhere between the quality of a dead sub-link and that of a good sub-link: data can be transmitted over a bad sub-link, but somewhat unreliably. Figure 5 assumes that a bad sub-link has a BER that is three orders of magnitude larger than the BER of a good sub-link. For example, if the p_{bit} of a good sub-link in a given array is 10^{-8}, then the p_{bit} of a bad sub-link in the same array would be 10^{-5}.



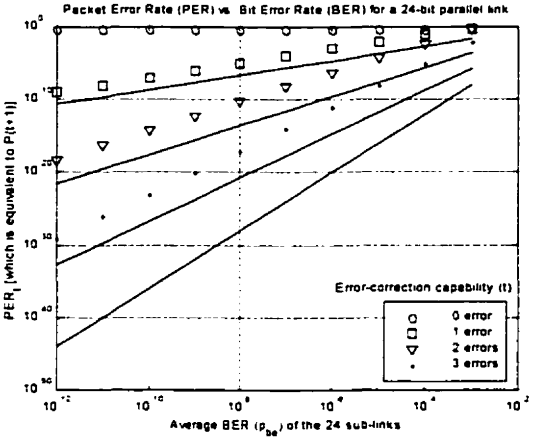
(a) $PER_3 = 8.85 \times 10^{-22}$
(one erasure)



(c) $PER_3 = 1.16 \times 10^{-19}$
(one erasure + one bad)



(b) $PER_3 = 2.34 \times 10^{-24}$
(two bad)



(d) $PER_3 = 1.07 \times 10^{-17}$
(one erasure + two bad)

Figure 5: PER vs. BER for a 24-bit parallel link that contains both, dead sub-links and bad sub-links. The bad sub-links have a BER that is three orders of magnitude larger than the BER of the good sub-links. The PER for all those cases is less than 10^{-15} for an average BER (p_{be}) of 10^{-8}.

Figure 5 (a) is no different than Figure 4 (c) and was replicated here for convenience. Figure 5 (b) depicts the case of a PODL with no erasure but two bad sub-links. The PER in this case is 2.34×10^{-24}, compared to a PER of 1.06×10^{-28} when there are no bad sub-links and no erasures in the array – see Figure 4 (a). Figure 5 (c) depicts the case of a PODL with one erasure and one bad sub-link. The PER in this case is 1.16×10^{-19}, which is less than three orders of magnitude away from the PER obtained when there is only one erasure and no bad sub-link – 8.85×10^{-22} from Figure 4 (c). Figure 5 (d) is interesting in that it shows that the PER stays below 10^{-15} (for a p_{be} of 10^{-8}) even when there is one erasure and two bad-sub-links in the array.

2.4 Effects of the distribution of the erasures on the PER

Depending on the yield, an optical link made up of two OE-VLSI chips with thousands of VCSELs and PDs is likely to have a few, indeed even many erasures. The source of each erasure can probably be tracked down to one or many of the sources of PODL erasures that were listed in Section 2.1. Erasures could either spread out randomly across the 2D array or be clustered. As far as ECC is concerned, clustered errors are to 2D parallel links what burst errors are to serial links. The term 'burst errors' applies to the telecommunication industry. For datacom applications, especially the highly parallel ones, it is more appropriate to use the term "clustered errors."

Figure 6 is representation of the 144 sub-links put at the service of *ecChan* on the D2C. Because the D2C is optically differential, every sub-link is in fact made up of one differential transmitter, two VCSELs, two PDs and one differential receiver [11].

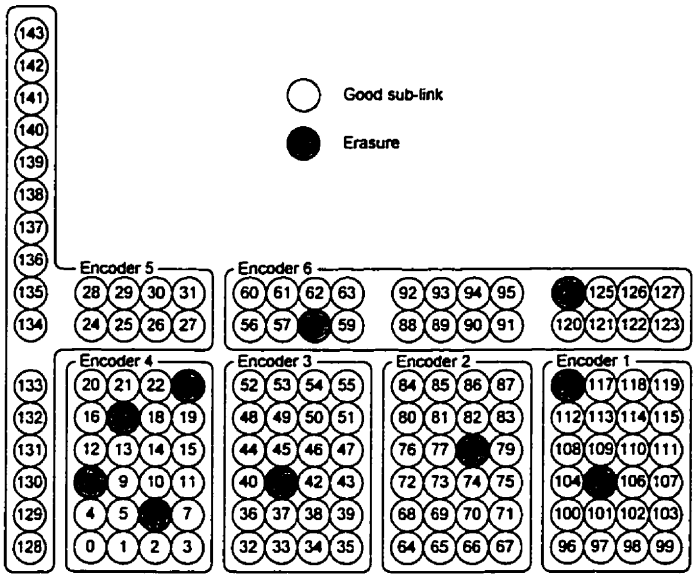


Figure 6: Example of an array containing erasures.

We will now look at how the six decoders of *ecChan* would perform if they had to deal with the erasures of Figure 6. Decoder 5, which is facing Encoder 5 in a board-to-board link, should perform very well because it does not have to deal with any erasure. Decoders 2 and 3 should also perform well as they each have to deal with only one erasure. Those two decoders, with a predicted PER of 8.85×10^{-22} for a p_{hc} of 10^{-8} – Figure 4 (c) –, should have no problem meeting the target PER of 10^{-15} because there is plenty of room to account for the errors caused by the bad sub-links, if any. According to Figure 4, Decoders 1 and 6 should produce a PER of 5.77×10^{-15} because they each have to deal with two erasures. The difference is that there is no room left to account for errors introduced by the bad sub-links and the good sub-links. The last decoder, Decoder 4, would

produce a very high PER because it has to deal with 4 errors, which is more that the error-correction capability of the decoder.

It is clear that the distribution of the erasures across the array has an effect on how well the six decoders will perform overall. Ideally, the erasures should be spread out such that the "work" is shared equally between the six decoders. If one decoder has to deal with too many errors, such as Decoder 4 in Figure 6, then this will degrade the performance of the PODL as a whole. The maximum number of erasures that the decoders in *ecChan* are able to correct is 18. This argument assumes that each decoder has to work on three erasures. In principle, it would be desirable to have one 144-bit decoder that could deal with 18 errors instead of having six decoders that could each deal with three errors. Generally, it is better to work with an ECC scheme that operates on big blocks of data to relax the assumptions that have to made on the distribution of the erasures.

To the author's knowledge, someone has yet to publish a paper on the distribution of the erasures in a PODL with thousands of VCSELs and PDs. Because there was no *a priori* information on the distribution of the erasures at the time the D2C was designed, an assumption had to be made. It was assumed that the erasures would spread out randomly across the 2D array, as opposed to being clustered. Determining how the erasures spread out across the array is very important, as it will influence the error correction scheme best suited for improving the reliability of PODLs.

2.5 How many errors should ECC be able to handle?

Ideally, one would want to use an ECC scheme that can correct as many errors as possible. ECC uses redundant sub-links to perform error correction. Sub-links that used to transfer useful information can be converted to redundant sub-links to increase the number of errors that can be corrected. More error-correction capability therefore means less information bandwidth if the baud rate and the number of sub-links are kept constant (see Section 3.4). In addition, more error-correction capability usually means more latency, power and area. The concern in adding ECC in a PODL is to not over-encode or under-encode the link. Over-encoding occurs when the link does not have enough errors to justify an error-correction capability. Under-encoding occurs when the link has too many errors for the ECC scheme selected.

The number of erasures in a PODL has a big influence on the amount of error correction required. The number of erasures can be derived from the yield of the PODL. Given that the yields of the many components around the D2C are multiplicative (see Equation 8), a low total yield was assumed.

$$\text{Yield}_{\text{total}} = \text{Yield}_{\text{digital logic}} \times \text{Yield}_{\text{transmitters}} \times \text{Yield}_{\text{VCSELs}} \times \text{Yield}_{\text{optics}} \times \text{Yield}_{\text{PDs}} \times \text{Yield}_{\text{receivers}} \quad (8)$$

The factors $Yield_{digital\ logic}$, $Yield_{transmitters}$ and $Yield_{receivers}$ could be lumped into the single factor $Yield_{CMOS}$. They were left separate because they are very different entities on the D2C, very far apart from one another and with different power, bias and control lines.

The factor $Yield_{optics}$ in Equation 8 will be relevant especially if using an OFA to establish the link between chip #1 and chip #2. A dark fiber would mean in this case the loss of a sub-link, causing an erasure. A FIG is more robust to dark fibres because every spot in the array is transmitted over more than one fiber [15]. The yield of free-space optics is expected to be very good, although it could decrease dramatically for big arrays due to aberrations. Micro-lens arrays have been proposed as a solution [19, 20], but their downside, like free-space optics in general, is their alignment complexity.

Last, even if the yields of the VCSELs and PDs turned out to be high before hybridization, it may not be the case after the devices have been flip-chip bonded to a CMOS chip. Here, the main technological problems concern the yield of flip-chip VCSEL and PD arrays involving a very large number of devices [13]. Moreover, the challenge is not only in producing arrays of VCSELs and PDs that can attach to CMOS circuits with high-yield, but it should also be possible to simultaneously operate these arrays of devices at high speeds. Cost effective standardized tests for arrays with a large number of elements have to be developed to perform the selection of known good dies and improve the yield. If the yield is hard to push up, then there is always the possibility of designing the system to accept lower yields by using ECC.

To conclude this section, Equation 8 will be used to determine what yields are necessary for a 144-bit PODL (such as *ecChan* on the D2C) to produce a PER in the order of 10^{-15}. As we saw in Section 2.2, each of the six decoders on the D2C will produce a PER of 5.77×10^{-15} if it has to deal with two erasures (assuming a p_{bc} of 10^{-8} and no bad sub-links). We will therefore

assume a total of 12 dead sub-links in the array (two for each of the six decoders). The total yield under those conditions would then be:

$$\text{Yield}_{\text{total}} = 100\% - \left(\frac{12}{288} \times 100\right) \cong 95.8\% \tag{9}$$

To simplify the discussion, assume that the six yields on the right hand side of Equation 8 are all equal to $Yield_{array}$. The average yield can be calculated using the result of Equations 8 and 9:

$$\text{Yield}_{\text{average}} = \sqrt[6]{\text{Yield}_{\text{total}}} = \sqrt[6]{95.8\%} \cong 99.3\% \tag{10}$$

This is a good average yield that may be hard to attain by some of the factors of Equation 8. For example, the reliability of the VCSELs and PDs is a key issue. Ideally, the output characteristics of lasers and other components should withstand some small temperature variations. As this constraint applies to a large number of individual array elements, a 10,000-hour lifetime requirement with a variation of less than 10% in output characteristics may be unrealistic, even for a highly advanced technology [13]. Therefore ECC schemes involving a sufficient fraction of redundant channels and sufficient error-correction capability should be used in PODLs.

2.6 Conclusion

In this chapter, the types of errors that can be expected in PODLs have been described. It was shown that ECC could play a significant role in improving the reliability of PODLs. The benefits of using Manchester encoding or 8b/10b encoding to balance the data have been demonstrated. Two assumptions have been made. First, it has been assumed that erasures would spread out randomly across the 2D array of a PODL. Second, it has been assumed that the total number of erasures in a link made of 144 sub-links was going to be less than 12. If these assumptions are correct, then a PER in the order of 10^{-15} will be obtained with the error-correction channel of the D2C.

2.7 References

1. E. Laprise, "Design and implementation of optoelectronic-VLSI chips for short reach optical interconnects", Master's thesis, McGill University, 2001.

2. Alberto Leon-Garcia, "Probability and Random Processes for Electrical Engineering", Second Edition, Addison-Wesley Publishing Company, 1994.

3. B. Webb, and A. Louri, "All-optical crossbar switch using wavelength division multiplexing and vertical-cavity surface-emitting lasers", Applied Optics, Vol. 38, pp. 6176-6183, 1999.

4. http://vada1.skku.ac.kr/ClassInfo/lower-power-DSP/viterbi/ch2.htm

5. Irving S. Reed and Xuemin Chen, "Error-control coding for data networks", Kluwer Academic Publishers, 1999.

6. Andrew S. Tanenbaum, "Computer Networks", Third Edition, Prentice Hall PTR, 1996.

7. A. X. Widmer and P. A. Franaszek, "A DC-Balanced, Partitioned Block, 8B/10B Transmission Code," IBM J. Res. Develop. 27, No. 5, pp. 440-450, 1983.

8. Peter Alfke, "Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators", Xilinx Application Note XAPP 052, 1996.

9. Parallel Bit Error Rate Tester (multiBERT MB100) from Tektronix, http:// www.tektronix.com/Measurement/Products/catalog/mb100/

10. Parallel Bit Error Ratio Tester (ParBERT 81250) from Agilent Technologies, http://literature.agilent.com/litweb/pdf/5968-9188E.pdf

11. E. Laprise, M. Venditti, and J. Faucher, "VLSI-Photonics Demo 2 ASIC: Functional Specifications, McGill University, 2000.

12. M.B. Venditti, Ph.D. thesis, McGill University, to be published.

13. MEL-ARI OPTO Technology Roadmap, September 1999, http://www.cordis.lu/esprit/src/melop-rm.htm

14. A.V. Krishnamoorthy, "Terabits/s Optical I/O directly to VLSI Chips", IEEE Southwest Symposium on Mixed-Signal Design, pp. 20-21, 2000.

15. T. Maj, A.G. Kirk, D.V. Plant, J.F. Ahadian, C.G. Fonstad, K.L. Lear, K. Tatah, M.S. Robinson, and J.A. Trezza, "Interconnection of a two-dimensional array of vertical-cavity surface-emitting lasers to a receiver array by means of a fiber image guide", Applied Optics, Vol. 39, pp. 683-689, 2000.

16. R.K. Kostuk, and J. Carriere, "Interconnect Characteristics of Fiber Image Guides", Applied Optics, Vol. 40, pp. 2428-2434, 2001.

17. D.M. Chiarulli, S.P. Levitan, P. Derr, R. Hofmann, B. Greiner, and M. Robinson, "Demonstration of a Multichannel Optical Interconnection by use of Imaging Fiber Bundles Butt Coupled to Optoelectronic Circuits", Applied Optics, Vol. 39, pp. 698-703, 2000.

18. D.V. Plant, M.B. Venditti, E. Laprise, J. Faucher, K. Razavi, M. Chateauneuf, A.G. Kirk, and J.S. Ahearn, "A 256 channel bi-directional optical interconnect using VCSELs and photodiodes on CMOS," to appear in IEEE J. Lightwave Technol., August 2001.

19. M. Châteauneuf, M. Venditti, E. Laprise, J. Faucher, K. Razavi, F. Thomas-Dupuis, A.G. Kirk, D.V. Plant, T. Yamamoto, J.A. Trezza, and W. Lui, "Design, implementation and characterization of a 2-D bi-directional free-space optical link", Conference proceedings of Optics in Computing 2000, pp. 530-538, 2000.

20. M. Châteauneuf, M. Venditti, E. Laprise, J. Faucher, K. Razavi, F. Thomas-Dupuis, A.G. Kirk, D.V. Plant, T. Yamamoto, J.A. Trezza, W. Luo, "512-channel vertical-cavity surface-emitting laser based free-space optical link: results", Conference proceedings of Optics in Computing 2001, pp. 64-66, 2001.

Chapter 3

Forward error-correction (FEC)

In addition to improving the reliability of a PODL, ECC can also help reduce power consumption and/or increase aggregate bandwidth. This contradicts with the fact that ECC is often perceived to cost information bandwidth owing to the transmission of redundant data for the purposes of detecting and correcting errors. The goal of this chapter is to present one of the most powerful ECC approaches – the Golay code [22] – and to dispel the magical atmosphere surrounding ECC.

The general theory behind ECC will be presented first. The parameters that can be used to judge the performance of an ECC will be explained. The most popular ECCs will be introduced and the emphasis will be put on the Golay code. The choice of using the Golay code on the D2C will be justified along the way.

3.1 Forward error correction (FEC) vs. automatic repeat request (ARQ)

There are two basic ECC classifications: ARQ and FEC. ARQ is a detection-only type of coding, where errors in a transmission can be detected by the receiver but not corrected. The receiver must request that bits detected as errors be retransmitted. Since these retransmissions will require valuable bandwidth, ARQ codes are generally used for "clean" transmission mediums (those with a lower probability of error). One of the most common example is simple parity checking, which is often used to detect data errors in RAM. Another example is a cyclic redundancy check (CRC), which is used to detect errors in a transmission over Ethernet. If errors are detected, the message will be retransmitted.

An OE-VLSI chip with thousands of optical I/Os is a noisy medium that stands a fair chance of introducing errors into a given transmission. Moreover, one erasure would be sufficient to put any ARQ method in a deadlock situation since retransmission would never solve the problem. Requests for resends, coupled with the retransmissions, consume bandwidth and increase latency. When the quality of a data link degenerates to a certain point, the resend

strategy will potentially be reducing system throughput to unacceptable levels. In these cases, FEC should be considered as an alternative to ARQ. FEC eliminates the need for resends by putting the burden of correcting the data on the receiver. FEC is often used for one-way communications, where the opportunity for the receiver to tell the sender to repeat the message is not available. Examples of these one-way paths include satellite transmissions, magnetic tape storage mediums and some PODLs like *ecChan* on the D2C.

3.2 Theory on FEC

Forward error correction is typically done in the digital domain, just before the transmitter block on the encode side and just after the receiver block on the decode side – see Figure 7.

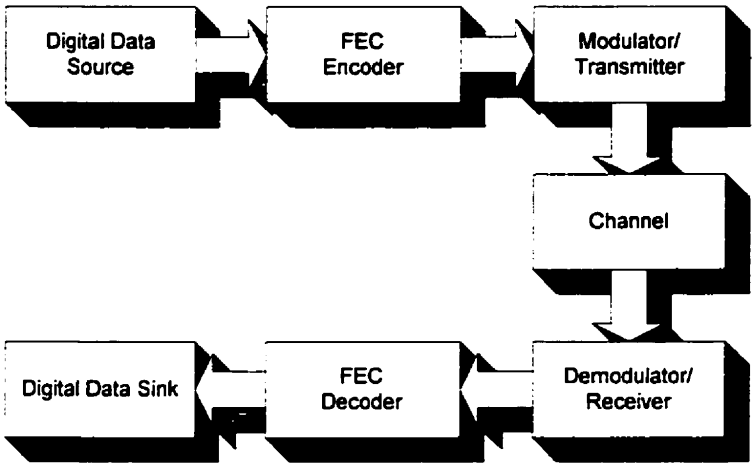


Figure 7: Block diagram of a coded channel.

The FEC encoder inserts redundant bits into a packet of data prior to transmission. The FEC decoder at the receiving end will employ the redundant bits to first determine if an error exists in the received data, and then to correct the errors if they fall within the error-correction capability of the code.

FEC codes have different characteristics in terms of error-correction capability, rate, size, hardware requirements, etc. Thus, there are trade-offs when choosing between different codes for a PODL. Below is a list of code parameters that can be used for the evaluation of error-coding methods:

Table 2: List of code parameters that can be used for the evaluation of FEC codes.

Packet-error rate (PER)	Function of the raw bit error rate, p_{bc}, of the individual sub-links (see Chapter 2)
Number of information bits	k
Coding overhead	n-k
Minimum distance	D_{min}
Maximum number of detectable errors	d
Hardware complexity	H (number of gates or VLSI area)
Parallel implementation possibility	Yes/no; if yes, degree of parallelism

Codes are first distinguished in terms of the kind of errors they can correct. Some codes like the Golay code are good at correcting randomly distributed errors whereas other codes such as the popular Reed-Solomon (RS) code [8] perform very well in burst error conditions. The second parameter in Table 2 has been treated extensively in Chapter 2.

There are two major classes of FEC: block and convolutional. A key point for block codes such as the Hamming codes [19], the Golay codes, the RS codes or the Bose- Chaudhuri-Hocquenghem (BCH) codes [20, 21] is that each codeword is formed independently from other codewords. In block coding, successive blocks of k information bits (or symbols) are formed. The coding algorithm then transforms each block into a codeword consisting of n bits (or symbols) where $n > k$. This structure is called an (n, k) code. The ratio $R = k/n$ is called the code rate (or information rate) while the difference n-k is called the coding overhead. The code rate is one of the most important parameter in Table 2 because it indicates what fraction of the codeword is actual information.

In the coding of block codes, each of the n-k redundant bits in a codeword depends only on the corresponding k information bits of the codeword. Convolutional codes such as the

Viterbi codes [8] differ from block codes in that the encoder contains memory and the n_0 outputs at any given time unit depend not only on the k_0 inputs of that time unit but also on m previous input words, in fact on $m \cdot k_0$ information bits. Hence, the convolutional encoder is said to have a memory of order m. The set of encoded sequences produced by a k-input and n-output encoder of memory of order m is called an (n, k, m) convolutional code. As for block coding, the ratio $R = k/n$ is called the code rate.

The concept of coding gain is somewhat subtle, so this parameter will be discussed in its own separate section (see Section 3.4). The next parameter to appear in Table 2 is D_{min}, or minimum distance. FEC codes map a set of desired information bits to another larger set of code bits that are then transmitted on the channel. Because not all code bit sequences (or codewords) can occur in the transmitted stream, errors introduced by the channel can be corrected by replacing the received sequence with the most likely possible sequence of code bits (given what was received), and then by inverting the transmitter mapping. This is called maximum likelihood (ML) decoding. The performance of ML decoding will increase if a FEC scheme allows only the transmission of very different codewords. The minimum number of bits by which any two codewords differ in a given FEC scheme is called the minimum distance (D_{min}) of the code. The error correcting capability, t, of a code is defined as [8, 11]:

$$t = \left\lfloor \frac{D_{\min} - 1}{2} \right\rfloor \tag{11}$$

where $\lfloor\ \rfloor$ is used to designate the integer part. The parameter t indicates that all combinations of t or fewer errors in any received sequence can be corrected. Equation 11 is called the maximum (random) error-correction capability of a code. Obviously, the bigger D_{min} is, the more errors can be detected and/or corrected.

So far, we have looked at a number of parameters that can help determine whether a FEC scheme is appropriate or not for a given application. The last few parameters listed in Table 2 (coding/decoding delay, hardware complexity, etc.) all refer to the hardware implementation of a FEC code. The number of gates (or VLSI area), the timing delays and the possibility of a parallel implementation of the algorithms determine whether a code can be efficiently implemented with electronic circuits. These parameters will be discussed for the Golay code in Chapter 4, where *etChan* on the D2C will be examined in detail.

Illustration of the principle of error-control coding

There have been many concepts, ideas and definitions exposed so far. The goal of this sub-section is to illustrate the principle of error-control coding [8]. We first begin by discussing an example of a binary block code of length 3. In this case, there are a total of 8 different possible binary 3-tuples: (000), (001), (010), (011), (100), (101), 110), (111). If all of these 3-tuples are used to transmit messages, one has the example of a (3, 3) binary block code of rate 1. In this case, if a one-bit error occurs in any codeword, the received word becomes another codeword. Since any particular codeword may be a transmitted message and there are no redundant bits in the codeword, errors can neither be detected nor corrected.

Next, consider the case where only four of the 3-tuples, namely (000), (011), (101) and (110) are chosen as codewords for transmission. These are equivalent to the four 2-bit messages (00), (01), (10) and (11) with the 3rd bit in each 3-tuple equal to the "exclusive or" (XOR) of its 1st and 2nd bits. This is an example of a (3, 2) binary block code of rate 2/3. If a received word is not a codeword, i.e. the 3rd bit does not equal the XOR of the 1st and 2nd bits, then an error is detected. However, this code is not able to correct an error. For example, suppose the received word is (010). Such an error cannot be corrected even if there is only one bit in error. The transmitted codeword could have been any of these three possibilities: (000), (010), (110).

Finally, suppose that only the 3-tuples (000) and (111) are chosen as codewords. This is a (3, 1) binary block code of rate 1/3. The codewords (000) and (111) are encoded by duplicating the source bit (0 or 1) two additional times. If the codeword is sent through the channel and the bits of the 3-tuple are not all equal on the receive side, then the received word is not a codeword and an error is detected. If the decoder uses the majority-decoding rule, then one-bit errors will always be correctable. For instance, if the received word is (010), the decoder would conclude that a 0 was sent.

3.3 The Golay code

There exist several FEC codes: Hamming, Golay, RS, BCH and Viterbi, just to name a few. Because these codes have different characteristics, it does not mean that they are all suitable for PODLs. The Golay code was favoured for the D2C and this choice will be justified in this section. The interested reader can find in the appendix a Matlab script as well as an Excel spreadsheet that outline the main steps to encode and decode the Golay codes.

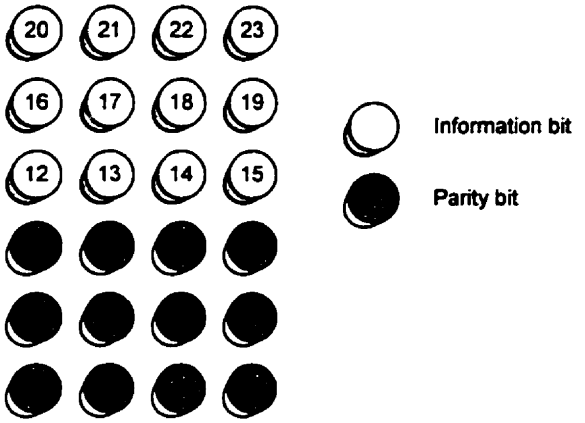


Figure 8: VCSEL array for Encoder 4 of *ecChan* (see Figure 6). Bits 23 down to 12 are information bits whereas bits 11 down to 0 are parity bits as computed by the Golay encoder.

The *perfect* Golay code is an encoding that encodes 12 bits into 23 bits, denoted by (23, 12). It allows the correction of three or fewer single-bit errors. The *extended* Golay code (the one used on the D2C – see Figure 8) contains an additional parity bit, which allows up to four errors to be detected. The resulting code is (24, 12), which is also known as half-rate Golay code.

At this point one might be led to ask the following question. Why can the Golay code correct 3-error patterns and detect 4-error patterns? For the (23, 12) Golay code, $D_{min} = 7$ [10] and therefore $t = 3$ (see Equation 11). For the (24, 12) extended Golay code, $D_{min} = 8$ [9, 10] but t is still equal to 3. It is argued in [8] that an ECC scheme that can correct t errors could still detect up to d errors provided that $t + d < D_{min}$. Since $t = 3$ and $D_{min} = 8$ for the extended Golay code, the number of detectable errors, d, is 4. The extra bit added to the extended Golay code allows the decoder to detect 4-error patterns (i.e. $d = 4$ in Table 2), which is not possible with the (23, 12) Golay code.

The Golay code is a linear block code, which means that, unlike the Viterbi code, it does not require memory. The term "linear" means that the code has a linear algebraic structure (i.e. it is based on the mathematical properties of linear algebra) that provides a significant reduction in the encoding and decoding complexity relative to that of arbitrary block codes. Very efficient decoders therefore exist for linear block codes.

The Golay encoder does not scramble the information bits and the parity bits. As it can be seen in Figure 8, the original message data is not disturbed and the redundant symbols are added separately at the end of each block. This form of linear encoding is called systematic encoding. Several implementation advantages accrue from the use of systematic codes, not the least of which is the quick-look feature: if desired, information can be extracted from the codeword without any decoding. Naturally, such a message recovery has a lesser reliability than if the full code vector were used to infer and correct the message content.

The Golay codes have a wide range of applications, but are particularly suitable for applications that require short codeword length and low latency, namely:

- Real-time audio and video communications
- Packet data communications
- Mobile and personal communications
- Radio communications

For example, the Golay code was used in the rather revolutionary design of an error-control system for the 19.9 kb/s PC modem. Also the extended binary Golay code was used to provide error control on the Voyager spacecraft [8].

Table 3: Hamming codes are usually represented as a function of the single integer $m \geq 2$ [8].

Hamming Codes	
Length	$n = 2^m - 1$
Information symbols	$k = 2^m - m - 1$
Minimum distance	$D_{min} = 3$
Error control capability	One error
Extended Hamming Codes	
Length	$n = 2^m$

The coding overhead (n-k) for the extended Golay code is 12 bits. The coding overhead should be as small as possible, but a small coding overhead usually means a poor error-correction capability. For example, the (8, 4) extended Hamming code or the (16, 11) Hamming code (see Table 3) could have been used on the D2C. Instead of one Golay block in Figure 8, there could have been three (8, 4) blocks for a total of 24 bits. The information rate and the coding overhead would have been 0.5 and 12 bits respectively, so nothing would have been gained since these figures are the same as for the extended Golay code. The drawback would have been that the (8, 4) Hamming code could only correct one error in an 8-bit block, or three errors in a 24-bit block. It seems like the (8, 4) code would have performed as well as the Golay code. The problem is that the error-correction capability of three assumes that each of the three Hamming decoders works on one error, whereas the Golay decoder does not have to assume anything. Long codewords are therefore better, which plays to the advantage of the RS codes because they can be made almost infinitely long [8] whereas the length of a Golay codeword is fixed to 23 or 24. Long Hamming codewords are possible too (see Table 3), but the error-correction capability will be one even for those long codewords.

Hamming codes are otherwise known to be easy to decode and therefore, with the BCH codes, were probably the biggest competitors of the Golay code for the implementation of FEC on the D2C. Unlike other ECC schemes such as the BCH codes and the popular RS codes, there is only one possible block size for the Golay code. The BCH codes are powerful ECC codes because there exist many decoders that are compact in size and relatively fast when implemented in CMOS. Their drawback is that their error-correction capability is not as high as that of the Golay code. The RS codes were also very attractive, but without interleaving (which adds latency), they don't do as well as the Golay code for randomly distributed errors [8]. Finally, the Viterbi codes, as mentioned earlier, necessitate the use of very low latency on-chip memory to keep up with the data throughput. The layout area taken up by the on-chip memory is another drawback of the Viterbi codes.

3.4 Coding gain

The measure of significance in the analog domain is the signal-to-noise ratio (SNR), while for digital systems it is the BER. For example, analog video to a home normally has an SNR exceeding 45 dB in a 4-MHz bandwidth, while the digitized link may specify a 10^{-7} BER at (MPEG encoded) 1.5 Mbps [15]. In a system such as a PODL where signals are converted from the digital domain to the analog domain and back to the digital domain (see Figure 7), it is useful to have a figure of merit that relates SNR to BER. For the digital components of a system, the average SNR per bit is more useful, and the bandwidth of detection nearly becomes a default value for optimum reception. Hence, the measure of BER in the digital domain is typically against E_b/N_o where E_b is the average energy per bit, and N_o is the noise spectral density. The useful figure of merit for a communication system becomes the SNR (E_b/N_o) that is required to achieve a given probability of error.

Noise in optical systems

Most communication channels add noise, distortion, and other impairments to the received signal. A PODL is no exception. As we will see in this section, the BER increases as the SNR decreases due to various sources of signal degradations. Below is a summary of the degradations that a PODL is most subject to:

- Poor alignment (lateral, angular or rotational) between the VCSELs, the optical system and the PDs. An alignment problem will cause the amount of power that reaches the detectors to decrease and therefore will have a negative impact on the SNR.
- Huge OE-VLSI chips with thousands of VCSELs and PDs are likely to be subject to non-uniformities. Some VCSELs, for example, may have a higher threshold current or a lower slope efficiency. This could have an impact on SNR, especially if the biasing is done near the threshold current to minimize power.
- Crosstalk due to electrical and/or optical coupling that occurs between adjacent sub-links may have an impact on the SNR.
- Intersymbol interference (ISI) within a sub-link. ISI may be caused by jitter or by the dispersion in an optical fiber (long transmissions only).
- Skew in highly parallel optical data links may be one of the main cause of performance degradation. If hundreds of receivers have to take their decision at the same time, some sub-links may not have reach their rail value yet due to longer propagation delays. These sub-links that are sampled before or after the optimum decision time will have a poorer SNR.

All these phenomena will either decrease the power transmitted or increase the noise level. In both cases, the SNR becomes worse. On the receive side, a low SNR at the decision instant may be insufficient for an accurate decision to be made. For instance, with high noise levels, the binary zero may occur above the threshold and hence be registered as a binary one. A low SNR is therefore synonym of a high BER. The degradations listed above apply to the system as a whole. There are two more sources of noise to be discussed. They apply to the receivers in particular.

Noise in receivers

Noise degrades the signal in optical receivers and is always present. Amplification is often used at the receiver to boost the signal strength; however, signal amplification is accompanied by an equivalent noise amplification, leaving the SNR unchanged. Two major sources of noise in receivers are thermal noise and shot noise. Thermal noise originates from the random motion of electrons in the resistive load of the receiver's amplifier circuit and is always present. Shot

noise is due to the discrete nature of electrons and arises from the random generation and recombination of free electrons and holes in a photodiode. For most optical receivers, the noise is generally thermally limited [7], which is independent of signal current.

The previous conclusion, namely that optical links are thermal-noise-limited systems, comes from [6]. It is further argued in [6] that this conclusion holds for highly parallel optical data links. According to [12], this conclusion is only correct for small arrays or for systems in which the circuits and packaging have been carefully designed to control the amount of noise in the receiver. In general, substrate noise and crosstalk on power supply lines can also be amplified by the receiver front end, thereby degrading the performance and leading to a tradeoff between signal sensitivity and crosstalk, which differs from fundamental noise considerations traditionally addressed in photoreceivers (e.g. shot and thermal noise). For instance, in [13], a 2.5-dB sensitivity penalty was measured when simultaneously operating an array of 50 such receivers. The use of guard-rings can help safeguard against noise from substrate carriers, but noise will also be present in the power and ground rails due to simultaneous switching noise, ground bounce, and other forms of crosstalk from the digital part of the circuit and the on-chip transmitter drivers. In addition, the input light source to the detector may have some noise superimposed on the data. Such noise may couple into the receivers and cause errors. Supply variations can be reduced by proper design and packaging (e.g. using separate receiver power supplier, minimizing inductive effects by using multiple power and ground leads on the chip, decoupling capacitors, etc.), but these measures are often inadequate at high levels of integration.

Whether it is the conclusion in [6] or [12] that holds for PODLs, the concept of coding gain (to be introduced next) stays the same and only the equation giving the BER as a function of SNR should differ (see Equation 12).

BER and PER as a function of SNR

Data can be transmitted over a channel according to several different modulation schemes. These include direct detection (DD), binary phase-shift keying (BPSK) and quaternary phase-shift keying (QPSK). The BER of a channel can usually be determined if the SNR at the input of the receiver is known. The exact formula for the BER depends on the modulation scheme and various other assumptions. Several different formulas are given in [2-7], the most

interesting of which is the formula for the DD modulation scheme (see derivation in [3]) because it was used on the D2C:

$$BER_{DD} = \frac{1}{2}\left[1 - erf\left(\frac{\sqrt{SNR}}{2\sqrt{2}}\right)\right], \tag{12}$$

where

$$erf(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,dt \tag{13}$$

and SNR is the ratio E_b/N_o. Equation 13 is the well-known error function.

Equations 12 and 13 imply that the higher the received SNR, the lower the BER. Equation 12 assumes a thermal-noise-limited system that exhibits AWGN. As argued previously, systems transmitting data over a serial link or over a small number of sub-links are examples of thermal-noise-limited systems. The D2C, with its thousands of optical I/Os, would not belong to this category of system. Expressions giving the BER as a function of SNR for thermally-noise-limited systems are well understood and have been studied extensively in the literature. To the author's knowledge, an equivalent of Equation 12 for highly parallel optical data links has yet to be derived. Although Equation 12 is not a perfect model for PODLs, it can still be useful to illustrate the concept of coding gain. When the various sources of noise are better understood and/or modeled in massively parallel optical data links, Equation 12 could be revised to better model the BER as a function of SNR for such systems.

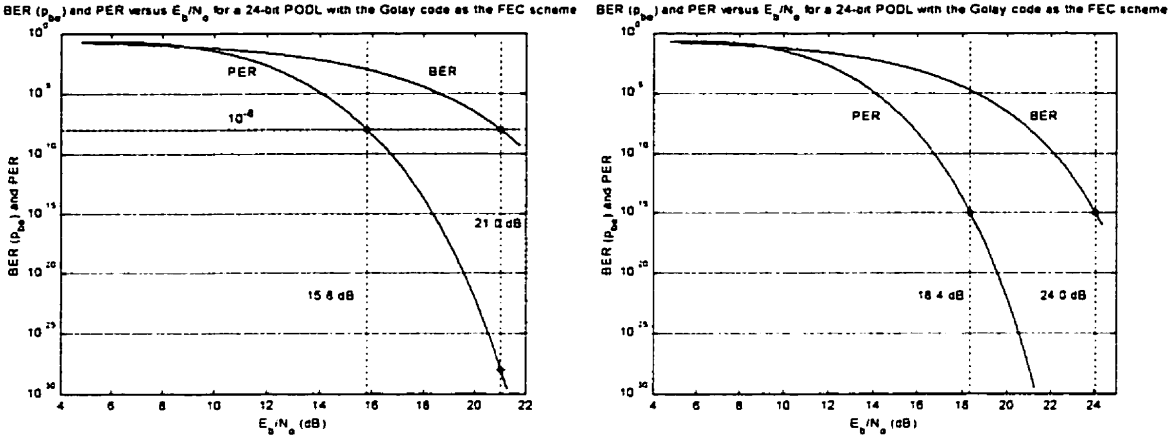
Coding gain

The coding gain of a coded channel over an uncoded channel refers to the reduction, expressed in decibels, in the required SNR to achieve a specified error performance, for example a PER of 10^{-15}:

$$\begin{aligned} Coding\ gain &= 10\log_{10}\left(\frac{SNR_{coded\ channel}}{SNR_{uncoded\ channel}}\right) \\ &= 10\log_{10}\left(\frac{P_{coded\ channel}}{P_{uncoded\ channel}}\right) \text{if } N_{coded\ channel} = N_{\text{uncoded } channel} \end{aligned} \tag{14}$$

where P and N refer to power and noise respectively.

The usual method of determining the coding gain is to plot the probability of error versus E_b/N_o (using Equation 12) for both, coded and uncoded operations, and to read the difference in E_b/N_o required at a specified error rate. The PER of the (24, 12) Golay code with hard-decision decoding is shown in Figure 9 (lower curve).



(a) Coding gain at PER = BER = 10^{-8} is 5.2 dB

(b) Coding gain at PER = BER = 10^{-15} is 5.6 dB

Figure 9: BER (p_{be}) and PER versus E_b/N_o for a 24-bit PODL with the Golay code as the FEC scheme.

The upper curve in Figure 9 shows the BER of one of the sub-links in an uncoded 24-bit link. The coding gains at $p_{be} = 10^{-8}$ and 10^{-15} are 5.2 dB and 5.6 dB respectively. A 5.2-dB coding gain means, in practice, that if the target BER of the D2C were 10^{-8}, then the required SNR would be 21 dB for an uncoded channel whereas it would be 15.8 dB for the channel with FEC.

If two channels have the same noise level, then a 5.2-dB coding gain means that the coded channels consumes 3.3 times less power than the uncoded channel for the same BER. This statement assumes that there are no erasures in the link (i.e. all the sub-links are functional). The coding gain is a useful concept only in those situations where it is meaningful to obtain performance improvements by increasing the power. For instance, if one has a communication link that is subject to random erasures, then even at high SNRs there is a floor on performance that cannot be overcome by increasing power. A coding scheme, however, might significantly reduce this floor or make it disappear entirely if the number of erasures is below the error-correction capability of the FEC code.

Although it is not clear in Figure 9, the coding gain actually becomes negative for sufficiently low SNRs – see Figure 10 (a blow up of the top left corner of Figure 9). This thresholding phenomenon is common to all coding schemes. There will always be an SNR at which the code loses its effectiveness and actually makes the problem worse. In this particular case, the minimum SNR required for the coded system to perform better than the uncoded system in terms of reliability is 9 dB.

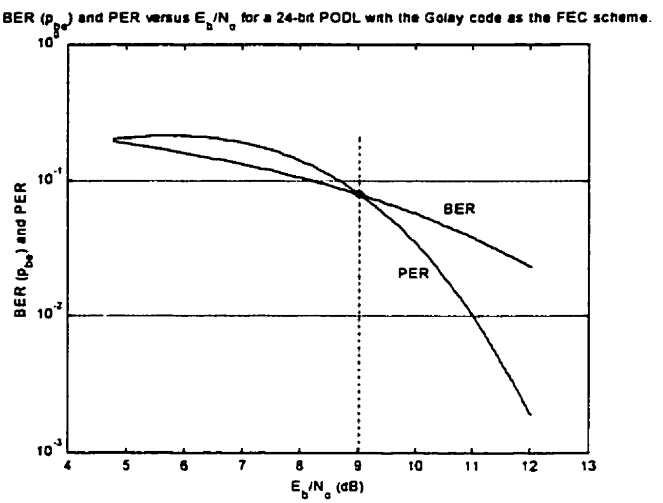


Figure 10: Blow up of the top left corner of Figure 9 (a) and (b).

In a parallel interconnect with FEC, some sub-links need to be used as redundant sub-links. Unless the baud rate or the number of sub-links is increased, FEC reduces the amount of useful information that gets transmitted. For example, if an uncoded channel on the D2C has 144 sub-links at its disposal and that the baud rate is 200 Mb/s, then the aggregate bandwidth of this PODL is 28.8 Gb/s. To make the comparison fair between the coded channel and the uncoded channel, it is fair to expect the coded channel to deliver the same aggregate bandwidth. In the coded channel of the D2C, 72 of the 144 sub-links are used as redundant sub-links for the purpose of FEC. To get an aggregate bandwidth of 28.8 Gb/s, the baud rate needs to double. The doubling of the baud rate is likely to have a negative impact on the SNR. Moreover, there is always a limit on the speed at which digital and analog circuits may operate in a given CMOS technology. Doubling the baud rate, then, may not even be an option to match the aggregate bandwidth of 28.8 Gb/s.

Alternatively, one could keep the baud rate constant but double the number of sub-links used to transfer data from point A to point B. If the Golay code is used, the number of sub-links that are necessary to keep up with the aggregate bandwidth of 28.8 Gb/s of the uncoded channel is 288 (the extended Golay code is a half-rate code, so only 144 sub-links are available for the transfer of information). In this case, the doubling of the number of sub-links means that the power consumption should roughly increase by 3 dB. Our coding gain of about 5.6 dB

at a PER of 10^{-15} is therefore reduced to 2.6 dB (or maybe a little lower to account for the CMOS circuitry needed to implement FEC). Table 4 illustrates this argument.

Table 4: Example to illustrate the concept of coding gain.

	Uncoded channel	Golay coded channel	
		Information bits	Redundant bits
Number of sub-links.	144	144	144
Useful information	yes	yes	no
Baud rate	200 Mb/s	200 Mb/s	
Aggregate information bandwidth	28.8 Gb/s	28.8 Gb/s	
Power required by each sub-link to achieve a PER of 10^{-15}	x mW	$\frac{x}{3.6}$ mW Note: A coding gain of 5.6 dB corresponds to a power ratio of 3.6 between an uncoded and a coded channel: $Coding\ gain = 10\log_{10}\left(\frac{P_{coded\ channel}}{P_{uncoded\ channel}}\right) = -5.6\ \text{dB}$ $\Rightarrow P_{coded\ channel} = \frac{P_{uncoded\ channel}}{3.6}\ \text{mW}$	
Total power	$144x$ mW	$\frac{2 \cdot 144x}{3.6}$ mW	
Power saving	$144x - \frac{2 \cdot 144x}{3.6} = 64x$ mW		
Net coding gain	$10\log_{10}\left(\frac{144x}{\frac{2 \cdot 144x}{3.6}}\right) = 10\log_{10}\left(\frac{3.6}{2}\right) = 2.6$ dB		

If x is equal to 1 mW in Table 4 (i.e. each sub-link consumes 1 mW of power), then the power saved by using FEC is 64 mW. This power saving takes into account the doubling of the number of sub-links that is necessary to get an aggregate information bandwidth of 28.8 Gb/s. The extra 2.6 dB (the net coding gain) can be used at the discretion of the designer. The following options are available:

1. Reduce the baud rate and add sub-links to the PODL to alleviate the need to design high-speed digital and analog circuits. The 2.6-dB net coding gain is used in this case to power up the additional sub-links that are necessary to keep the aggregate bandwidth constant at 28.8 Gb/s. The coded channel and the uncoded channel would, in this case, consume the same total amount of power.

2. Increase the aggregate bandwidth. The 2.6-dB net coding gain is used in this case to power up additional sub-links (apart from those used to transmit information bits and redundant bits) while keeping the baud rate constant. Like for the previous case, the coded channel and the uncoded channel would consume the same amount of power. As it was mentioned in the introduction, the use of ECC is often perceived to cost information bandwidth owing to the transmission of redundant data for the purpose of correcting errors. Because the use of a suitable code can decrease the minimum signal-to-noise ratio (SNR) required for reliable operation, the use of error correction can actually increase information bandwidth [18].

3. Reduce the total power consumption. Good performance usually prevails if the available power is limitless, but in a cost-effective design, transmitting with as little power as possible is almost always preferred. To reduce total power consumption, each individual sub-link of the coded channel needs to consume a little less power, which is likely to reduce the SNR. This would in turn increase the BER (see Equations 12 and 13). However, when the degraded data bits and redundant bits are sent to the FEC decoder, redundancy can be used to correct some fraction of the errors, thereby improving overall reliability of the PODL.

4. Improve the reliability of the system. From Figure 9 (a), a 5.2-dB coding gain can be used to transform a BER of 10^{-8} into a PER of 10^{-28}.

5. Tolerance the design by allowing bigger misalignments (free-space optics) [17] and/or reduced coupling efficiencies (guided optics). In this case, the 2.6-dB coding gain is used to burn more power in each sub-link, just as in the previous case.

3.5 Conclusion

As we just saw, FEC is not only useful in reducing the BER of a PODL; it gives the designer the flexibility to increase the bandwidth, decrease the power consumption, improve the BER, or a make mix of the three.

It was pointed out that the Golay code performs well when errors are randomly distributed. The RS code would be better suited for burst errors, but it is more complex to decode. Moreover, an RS code that has enough error-correction capability to compete with the Golay code requires huge blocks of parallel data (more data than the 144-bits that were available in *ecChan*). If it turned out that erasures are mostly clustered in a PODL, one could use a method known as "interleaving" (which will be briefly discussed in Section 5.2) to solve the problem. Because it was assumed in Chapter 2 that the erasures would occur somewhat randomly, it was decided that the D2C would not support interleaving. Since the Golay code can correct up to three errors in any 24-bit packet transmitted, some limited clustering can be supported anyways. Hamming codes and BCH codes were also considered as alternatives to the Golay code. The error-correction capability of Hamming codes and BCH codes is 1 and 2 respectively. With its error-correction capability of three, the Golay code therefore preferred.

3.6 References

1. G.C. Clark, and J.B. Cain, "Error-Correction Coding for Digital Communications", Plenum Press, New York, 1981.

2. L. Couch, "Digital and Analog Communication Systems", Macmillan, New York, 1990.

3. A. Yariv, "Optical Electronics", 4th Edition, Holt, Rinehart & Winston, New York, 1991.

4. R.E. Ziemer, and W.H. Tranter, "Principle of Communications, 2nd Edition, Houghton Mifflin Co., 1985.

5. R.E. Ziemer and R.L. Peterson, "Introduction to Digital Communication", Macmillan, New York, 1992.

6. T.H. Szymanski, "Bandwidth optimization of optical data links by use of error-control codes", Applied Optics, Vol. 39, pp. 1761-1775, 2000.

7. B. Smith, R.E. Spencer, D.C. Brown, and M. Bentley, "Optical Fibre Communications Between Radio Telescopes in the European VLBI Network", TMR-LSF RTD Sub Project 4, 2000. http://www.jb.man.ac.uk/research/fibre/fibre4vlbi.pdf

8. Irving S. Reed and Xuemin Chen, "Error-control coding for data networks", Kluwer Academic Publishers, 1999.

9. J. Wolfmann, "A permutation decoding of the (24, 12, 8) Golay code", IEEE Trans. Inform. Theory, Vol. IT-29, No. 5, 1983.

10. O. Pretzel, "Error-correcting codes and finite fields", Oxford University Press, New York, 1992.

11. G.C. Clark Jr., and J.B. Cain, "Error-correction coding for digital communications", Plenum Press, New York, 1981.

12. A.V. Krishnamoorthy, R.G. Rozier, T.K. Woodward, P. Chandramani, K.W. Goossen, B.J. Tseng, J.A. Walker, W.Y. Jan, and J.E. Cunningham, "Triggered receivers for optoelectronic VLSI", Electronic Letters, Vol. 36, No. 3, 2000.

13. T.K Woodward, A.L. Lentine, A.V. Krishnamoorthy, K.W. Goossen, J.A. Walker, J.E. Cunningham, W.Y. Jan, B.T. Tseng, S.P. Hui, and R.E. Leibenguth, "Parallel operation of 50 element two-dimensional CMOS smart-pixel receiver array", Electron. Lett., Vol. 34, 1998.

14. J.C. Palais, "Fiber Optic Communications", Prentice-Hall, Englewood Cliffs, N.J., 1984.

15. R. Howald, "Kicking Off the Communication Link", Communication Systems Design Magazine, http://www.csdmag.com/main/col9701.htm.

16. E. Laprise, "Design and implementation of optoelectronic-VLSI chips for short reach optical interconnects", Master's thesis, McGill University, 2001.

17. M.A. Neifeld, and R.K. Kostuk, "Error correction for free-space optical interconnects: space-time resource optimization", Applied Optics, Vol. 37, pp. 296-307, 1998.

18. M.A. Neifeld, and M. McDonald, "Error-correction for increasing the usable capacity of photorefractive memories", Opt. Lett. 19, 1994.

19. R.W. Hamming, Error detecting and error correcting codes", Bell Syst. Tech. J., 29, pp. 147-160, 1960.

20. R.C. Bose, and D.K. Ray-Chaudhuri, "On a class of error correcting binary group codes", Information and Control, 3, pp. 68-79, 1960.

21. A. Hocquenghem, "Codes correcteurs d'erreurs", Chiffres, 2, pp. 147-156, 1959.

22. M.J.E. Golay, "Notes on digital coding", Proc. IRE, 37, p. 657, 1949.

Chapter 4

FEC on the Demo 2 chip

FEC techniques have been invented and further developed with the idea in mind of improving the reliability of serial links. For example, the RS code is used extensively nowadays in optical fiber communication systems [15, 16]. These systems typically operate at very high speeds and may involve a limited degree of parallelism via wavelength and/or polarization multiplexing. Even with some limited parallelism, it is still possible to use multiplexers and demultiplexers to serialize the incoming bit streams and use conventional FEC techniques such as BCH, RS and Golay.

It is only in recent years that people have started wondering about adapting existing FEC techniques so that they can perform error correction in parallel. Optical storage systems [1, 2] and parallel optical interconnects (POIs) [3] are two of the main technologies that have started to explore the benefits of using parallel error correction. Parallel error correction helps get rid of some of the latency added by the multiplexing and demultiplexing stages mentioned above. Telecom applications do not care as much about latency as do datacom applications, which is why FEC is typically done serially in the former case. Moreover, a serial FEC decoder would need to handle an incredible amount of high-speed data to keep up with the high bandwidth capacity of a PODL. To ensure efficient utilization of a PODL bandwidth and avoid unwanted data bottlenecks, it is necessary to implement parallel high-speed decoders.

Szymanski argued in [3] that FEC techniques are not well suited for high-bandwidth 2D bit-parallel optical systems because they require a significant amount of hardware for implementation. Szymanski's argument was based on an implementation of the RS code [4]. Demonstrating that the Golay code could drastically improve the PER of a PODL was done in the previous chapters and is one step in the right direction; proving that it is suitable for a hardware implementation is as important. To date, the D2C is the first known attempt to perform parallel FEC in the context of a massively parallel optical data link.

Following the theory that was presented in the previous chapter, we will now look at the hardware implementation of the error-correction channel (*ecChan*) on the D2C. The setups used to test *ecChan* will then be described, followed by experimental results.

4.1 Hardware implementation

The error-correction channel is only one of the four channels on the D2C (see Figure 11 for a picture of the chip). *ffChan*, *dgChan*, *ftChan* and *ecChan* stand for fifo channel, data generation channel, feedthrough channel and error-correction channel respectively. Each channel is about 4 mm by 1 mm. The PD/receiver array (Rx) is on the left and the transmitter/VCSEL array (Tx) is on the right. Only *ecChan* will be described in this chapter. Note that *ecChan* was divided up into two blocks: *ecChanEnc* (the encoder block and its surrounding logic) and *ecChanDec* (the decoder block and its surrounding logic). The reader can find information on the other three channels in [11]. Information on the design of the transmitters and receivers can be found in [11, 14].

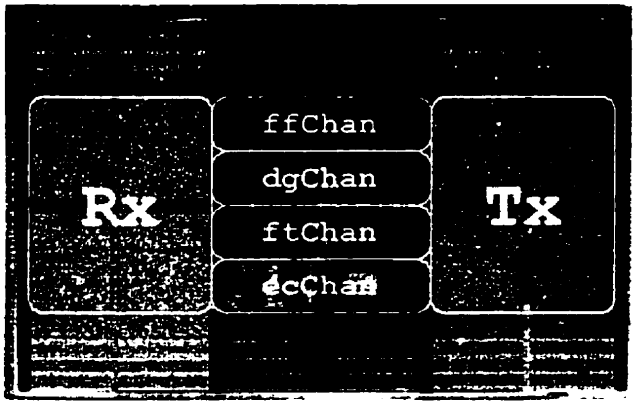


Figure 11: Picture of the Demo 2 chip. The area of the chip is 1.0977 cm^2 (14.591 mm × 7.523 mm).

4.1.1 ecChanEnc

In ECC, the encoding algorithm is usually much simpler than the decoding algorithm. The reason is that encoding is usually done in a simple pass by multiplying the input vector by a certain matrix (the matrix is specific to the ECC scheme used). For example, for the particular case of the Golay code, encoding is done as follows:

$$cw = w \times G$$

where

w is the input word to be encoded and has size 1 × 12

G is the generator matrix and has size 12 × 24

cw is the codeword (i.e. the encoded input word) and has size 1 × 24

One possible generator matrix for the Golay code was given in [7] :

$$G = \left[\begin{array}{cccccccccccc:cccccccccccc}
1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&1&1&1&1&0&0&0&1\\
0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&1&1&1&0&1&1\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&0&1&0&1&1&1\\
0&0&0&1&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&0\\
0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&1&1&1&1&1\\
0&0&0&0&0&1&0&0&0&0&0&0&1&0&1&0&0&1&1&1&1&0&1&0\\
0&0&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&1&1&0&0&1&1&1\\
0&0&0&0&0&0&0&1&0&0&0&0&1&1&1&0&1&1&0&1&0&1&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&1&1&1&0&0&1&1&0&1\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&1&1&0&1&1&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&1&1&1&1&1&1&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&1&0&1&0&1&0&0&1
\end{array}\right]$$

Note that the left half of G is the 12 × 12 identity matrix, which explains why the Golay code is systematic (i.e. there is no scrambling of the information bits). That is about all the theory there is behind the Golay encoder that was put on the D2C. It is simple, compact, and fast (as we will see shortly). The interested reader can find in the appendix the Matlab script as well as the Excel spreadsheet that were used to test the Golay encoder functionality before hardware implementation. Figure 12 is a top-level block diagram describing the functionality of *ecChanEnc*.

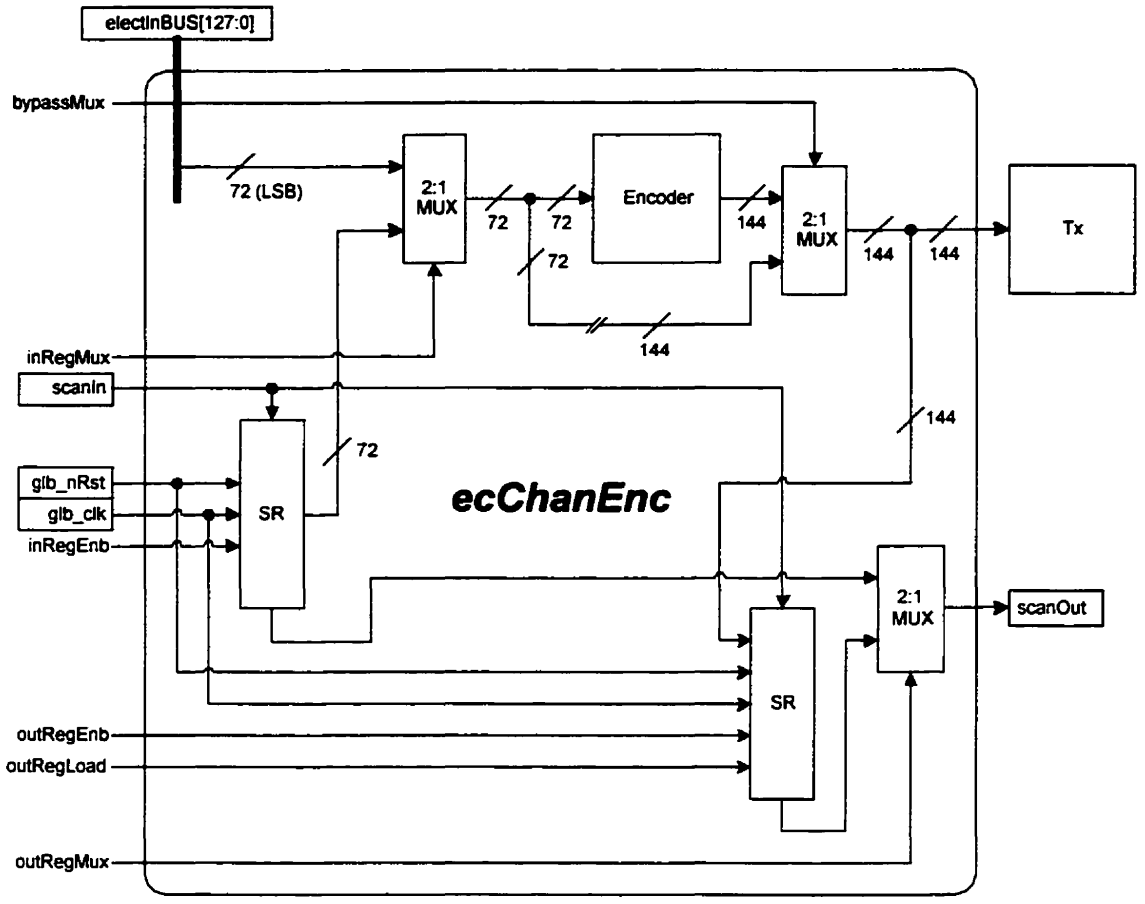


Figure 12: Top-level block diagram describing the functionality of *ecChanEnc*. Every I/O that is not in a box comes from the control register (which is described in [11]); all the other I/Os are connected to physical pins of the D2C.

The encoder block of Figure 12 is made of 6 instances of the Golay encoder. Each Golay encoder works completely independently, taking 12 bits as an input and giving 24 bits as an output. Hence, the encoder takes a 72-bit word and encodes it into a 144-bit codeword. The blocks surrounding the encoder were added for testing purposes. One can use the multiplexers to operate *ecChanEnc* in three different modes: normal mode, test mode and bypass mode.

Normal mode

In normal mode, *ecChanEnc* uses purely combinational logic to encode the data. It is therefore not necessary to provide a clock to *ecChanEnc* when it is operated in normal mode. In this mode, the data to be encoded comes from the electrical input bus (of which only 72 of the 128 bits are used). The output of the encoder goes straight to the transmitters to drive the VCSELs.

Test mode

The register on the left in Figure 12 is a serial-shift register whereas the register on the right is a parallel-load serial-shift register. These two registers were added to the design so that electrical tests could be performed on *ecChanEnc* before VCSELs get hybridized to the chip. Because registered logic is used in test mode, a clock is required. The input register can be serially loaded with data and used as an alternative to the electrical input bus to provide test vectors to the encoder. This feature is useful because a very simple board with no high-speed traces can then be designed to validate *ecChanEnc*. For example, the Validation Board (see Section 4.2) was designed without worrying about the complications of laying out a 128-bit high-speed bus.

As for the output register in Figure 12, it is useful to look at the electrical data at the output of the encoder. Without this register, flip-chipped VCSELs would be required to test *ecChanEnc* (i.e. one could only look at optical data). This requirement would complicate the validation of *ecChanEnc* unnecessarily.

Bypass mode

When operated in bypass mode, *ecChanEnc* bypasses the encoder. No data processing is performed in this mode, facilitating the testing of the registers, the multiplexers, the electrical input bus, the transmitters and the VCSELs. To operate in bypass mode, one has to steer the multiplexers on each side of the encoder accordingly.

Delay

Synopsys predicted a propagation delay of 2.23 ns for *ecChanEnc*. The propagation delay was calculated from the electrical input bus to the output of the multiplexer just after the encoder – see Figure 12. This means that *ecChanEnc* could be clocked at about 448 MHz in a pipelined design (in reality, the clock frequency would have to be a little less than 448 MHz to account for setup and hold times). In designing *ecChanEnc*, the emphasis was put on being able to characterize the encoder. Getting rid of the multiplexers could shorten the delay of *ecChanEnc*. Synopsys predicted a delay of only 1.37 ns for the encoder alone.

Area

Most of the area taken by *ecChanEnc* comes from the fact that it was designed for testability. The registers and the multiplexers add to the circuit complexity but were necessary to perform a functional verification of the design before hybridization of the VCSELs. Synopsys predicted

an area of 0.098 mm^2 for *ecChanEnc* (using the "typical case" wire model to evaluate the interconnect area). Figure 13 shows that the area taken by *ecChanEnc* is actually 0.62 mm^2 (i.e. 6.3 times bigger than predicted). The difference is mainly due to the fact that the cell placement after synthesis was done on a grid with every other row left empty to facilitate the routing. The area of the encoder (i.e. the six Golay encoders altogether) predicted by Synopsys was only 149 µm × 149 µm (0.022 mm^2), which represents just a little over 20% of the total area of *ecChanEnc*.

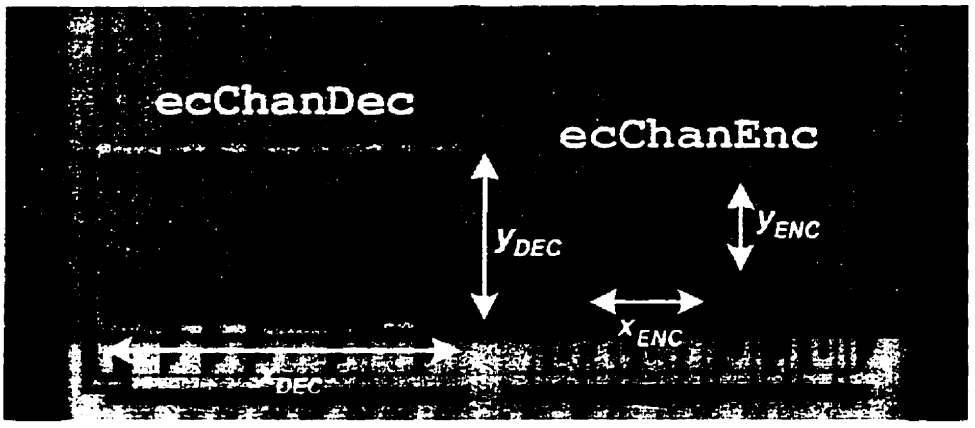


Figure 13: Zoom of *ecChanDec* and *ecChanEnc* on the D2C. x_{DEC} = 2300.56 µm, y_{DEC} = 988.30 µm, x_{ENC} = 913.44 µm and y_{ENC} = 678.48 µm.

Power

The current drawn by all the digital circuitry of the D2C was about 16 mA, for a power consumption of about 40 mW using a voltage of 2.5 V. The power consumption of *ecChanEnc* should therefore be well below 40 mW. The value of 16 mA was obtained by measuring the current drawn by the Validation Board (see Section 4.2) before the wirebonding of the digital section of a non-hybridized D2C (54 mA) and after (70 mA).

4.1.2 ecChanDec

Since the Golay code has some attractive properties such as being linear, systematic, and having a high correction capability, many efficient decoding techniques have been developed. For example, Kasami's error trapping decoding and systematic-search decoding [5], step-by-step decoding [6] and permutation decoding [8] can be used to decode the Golay code. However, none of these algorithms has been realized efficiently with parallel VLSI circuits. To solve this problem, a parallel VLSI architecture for the (24, 12) Golay decoder was presented in [10]. It is this decoder that was implemented on the D2C. The Golay decoder implements an

optimized permutation-decoding algorithm with look-ahead error correction based on Wolfmann's decoding algorithm [8]. The block diagram of the Golay decoder used on the D2C is shown in Figure 14.

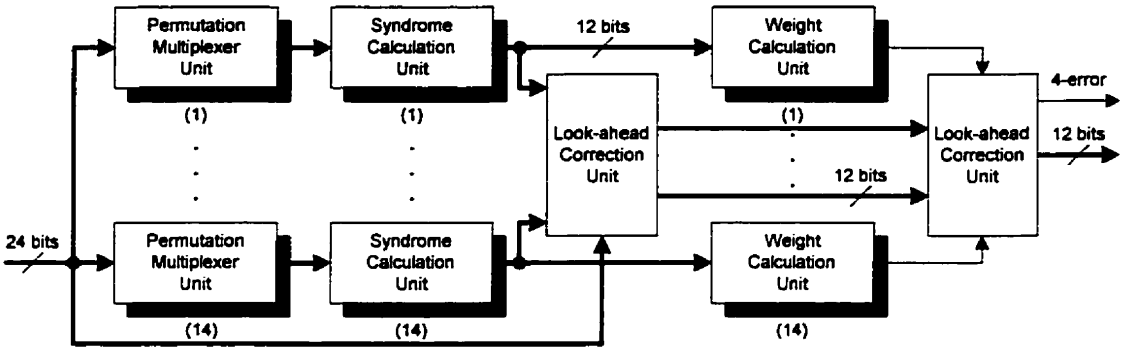


Figure 14: Block diagram of the high-speed parallel Golay decoder for optimized permutation decoding of the (24, 12) Golay code with look-ahead error correction.

The reader is referred to [8] for a complete explanation of each of the building blocks found in Figure 14. Only a top-level description of the decoder will be given here. The interested reader can find in the appendix the Matlab script as well as the Excel spreadsheet that were used to test the Golay decoder functionality before hardware implementation.

The look-ahead error correction permutation decoding technique simultaneously performs 14 error-correction operations directly in the received 24-bit word before a usable syndrome S (see step 2 in Table 5 for the mathematical definition of a syndrome) of weight[1] $w(S) \leq t$ (where $t = 3$ for the Golay code) is chosen out of the 14 syndromes. The permutation-decoding algorithm consists of the following main steps:

[1] The weight of a binary vector is simply the number of 1's that it contains. For example, the weight of 1001 is two and the weight of 1110 is three.

Table 5: Main steps of the permutation-decoding algorithm to decode the Golay code.

Step	Description	Block in Figure 14
1	Generate 14 permutation sets $\{P_1, \ldots, P_{13}\}$ from the 24-bit input word $R_m[0:23]$.	Permutation Multiplexer Unit
2	Calculate, in parallel, the syndrome S_i of each permutation P_i by using the parity-check matrix H^T [7, 8, 9]. $S_i[0:11] = P_i[0:23] \bullet H^T$ **Note:** A more complete description of syndrome decoding can be found in [12] and [13].	Syndrome Calculation Unit
3	Form 14 decoded words, R_i, by performing the parallel look-ahead error-correction operation in the information section of the received word (bits 0 to 11) with the selected syndrome S_i as listed in [10].	Look-ahead Correction Unit
4	Calculate the weight of each syndrome S_i and select the decoded word which has the weight $w(S_i) \leq 3$. The selected word becomes the output of the decoder.	Weight Calculation Unit + Select Unit
5	If all weights of the 14 syndromes $w(S_i) > 3$, a 4-error detection in the received codeword is reported.	Select Unit

Following a bottom-up approach, the optimized decoding procedure outlined in Table 5 was successfully mapped onto a parallel VLSI architecture and used as a building block in *ecChanDec*. Figure 15 is a top-level block diagram describing the functionality of *ecChanDec*.

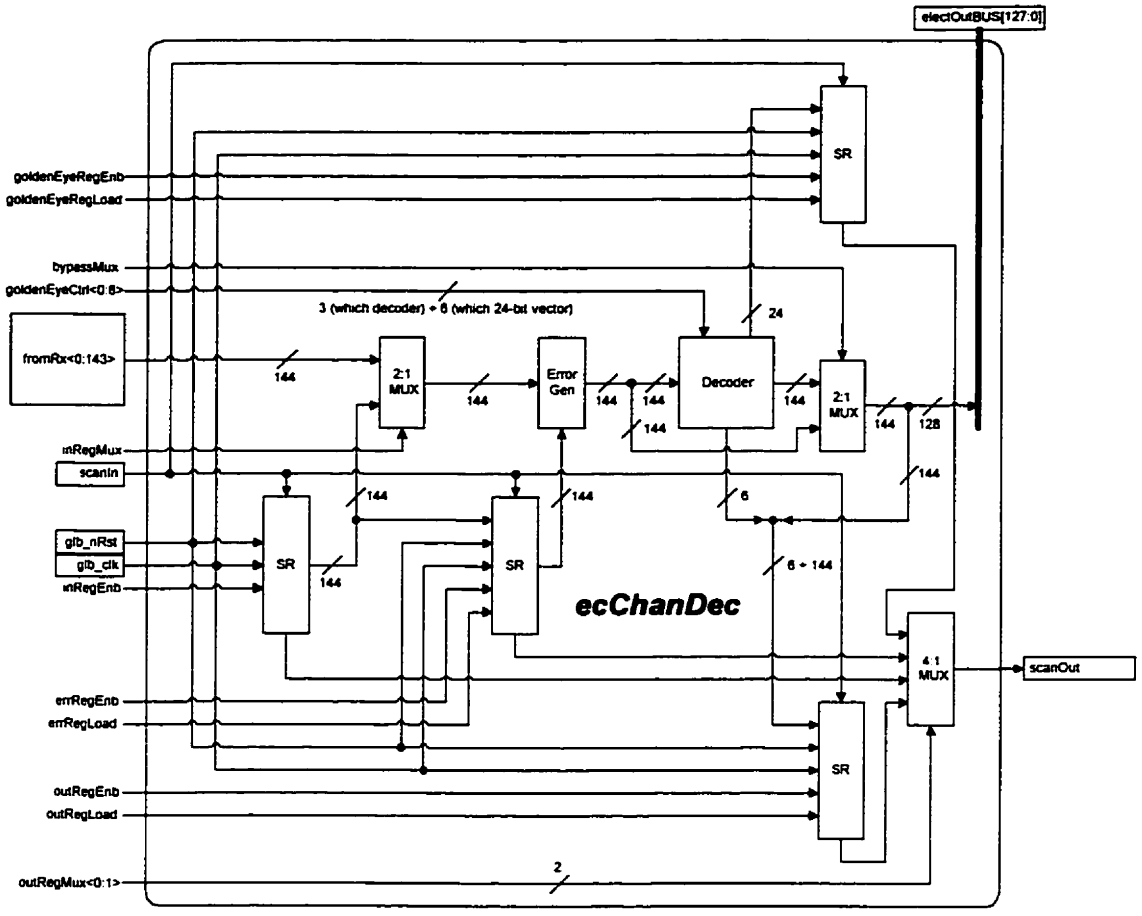


Figure 15: Top-level block diagram describing the functionality of *ecChanDec*. Every I/O that is not in a box comes from the control register (which is described in [11]); all the other I/Os are connected to physical pins of the D2C.

The decoder block of Figure 15 is made of six instances of the Golay decoder. Each Golay decoder works completely independently, taking 24 bits as an input and giving 12 bits as an output. Hence, the decoder takes a 144-bit codeword and decodes it into a 72-bit word. Figure 15 indicates otherwise, but only because the redundant bits were not dropped at the output of the decoder to facilitate the debugging and testing of *ecChanDec*. Unfortunately, not all 144 bits at the output of the decoder could be sent to the electrical output bus for verification, so some redundant bits eventually had to be dropped.

Just as for *ecChanEnc*, one can use the multiplexers to operate *ecChanDec* in three different modes: normal mode, test mode and bypass mode.

Normal mode

In normal mode, *ecChanDec* uses purely combinational logic to decode the data. It is therefore not necessary to provide a clock to *ecChanDec* when it is operated in normal mode. In this mode, the data to be decoded comes from the receivers. The output of the decoder is connected to the electrical output bus through a multiplexer.

Test mode

Because registered logic is used in test mode, a clock is required. The leftmost register in Figure 15 is a serial-shift register whereas all the other the registers are parallel-load serial-shift registers. The registers were added to the design so that electrical tests could be performed on *ecChanDec* before PDs get hybridized to the chip. For example, the leftmost register can be used as an alternative mean of providing test vectors to the decoder. Likewise, the output register offers an alternative mean to the electrical output bus of retrieving the data at the output of the decoder. This rightmost register is useful because a very simple board without the output bus can be designed to validate *ecChanDec*.

As far as testing is concerned, *ecChanDec* differs from *ecChanEnc* in that it contains two additional registers: the *error register*, and the *goldenEye register* (the topmost register in Figure 15). The *error register* works in parallel with an error generation block (a bunch of XOR gates) and is useful to generate error patterns. The generation of "artificial" errors can be useful to verify that the decoder performs as intended or to increase the BER in real-time. Because the decoder is much more complicated than the encoder, it was decided to provide for a way of looking at its internal behavior. The topmost register in Figure 15 can capture data anywhere along the Golay decoder data path shown in Figure 14.

Bypass mode

When operated in bypass mode, *ecChanDec* bypasses the decoder. No data processing is performed in this mode, facilitating the testing of the digital logic, the transmitters, the receivers, the VCSELs and the PDs. To operate in bypass mode, one has to steer the multiplexers on each side of the decoder accordingly.

Delay

Synopsys predicted a propagation delay of 8.03 ns for *ecChanDec*. The propagation delay was calculated from the output of the receivers to the output of the multiplexer just after the

decoder – see Figure 15. This means that *ecChanDec* could be clocked at about 125 MHz in a pipelined design (in reality, the clock frequency would have to be a little less than 125 MHz to account for setup and hold times). In designing *ecChanDec*, the emphasis was put on being able to characterize the decoder. Getting rid of the multiplexers could shorten the delay of *ecChanDec*. Synopsys predicted a delay of 6.77 ns for the decoder alone.

Area

As for *ecChanEnc*, most of the area taken by *ecChanDec* comes from the fact that it was designed for testability. The registers and the multiplexers add to the circuit complexity but were necessary to perform a functional verification of the design before hybridization of the PDs. Synopsys predicted an area of 1.32 mm^2 for *ecChanDec* (using the "typical case" wire model to evaluate the interconnect area). According to Synopsys, *ecChanDec* should therefore take an area 13 times bigger than that of *ecChanEnc*. Figure 13 shows that the area taken by *ecChanDec* is actually 2.27 mm^2 (i.e. 1.7 times bigger than the area predicted by Synopsys). The difference is mainly due to the fact that the cell placement after synthesis was done on a grid with some rows (as opposed to every other row for *ecChanEnc*) left empty to facilitate the routing. The area of the decoder (i.e. the six Golay decoders altogether) predicted by Synopsys was about 1.08 mm × 1.08 mm (1.16 mm^2), which represents just a little over 50% of the total area of *ecChanDec*. One Golay decoder therefore took an area of 0.193 mm^2, or about 439.3 µm × 439.3 µm. If the baud rate is 200 Mb/s, then this amount of silicon area can process 12 × 200 Mb/s = 2.4 Gb/s of data.

Power

The power consumed by the entire digital section of the D2C was 40 mW. The power consumption attributed to *ecChanDec* should therefore be well below 40 mW.

4.2 Test setups

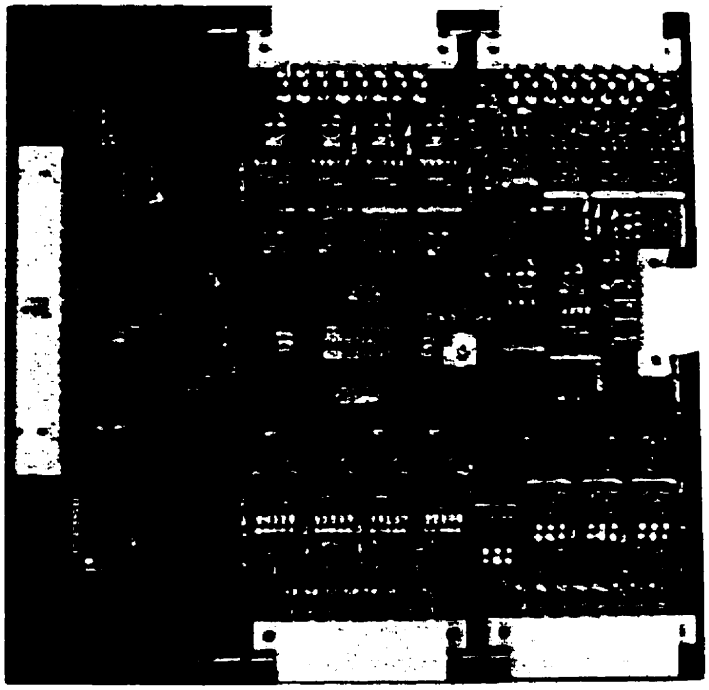
Two different test setups were designed to test the D2C: one for low-speed testing and functional verification and the other for high speed testing. In this section, we will look at the PCBs and GUIs that were part of each test setup.

PCBs

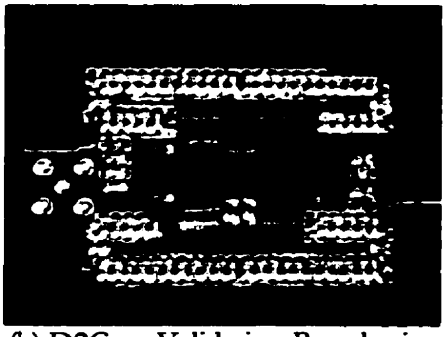
The low-speed testing solution did not provide access to all the pins of the D2C, the goal being to build a very simple validation board – see Figure 16 (a) – that would be useful to:

- Perform a functional testing of pre- and post-hybridized chips;
- Make sure that the power consumption of the chip was as expected;
- Find out the yield of the VCSEL and PD arrays;
- Test the optical link that will be used in the high-speed test setup;
- Build a PODL than can transmit data in the kHz range.

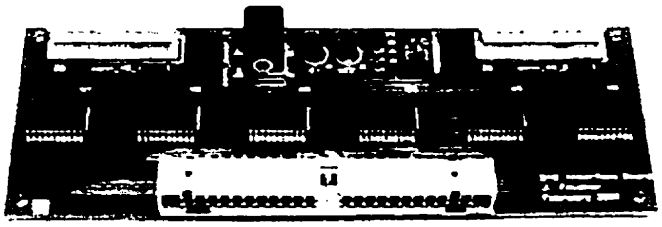
Unlike the Validation Board, the Printed Wiring Board – see Figure 16 (d) – that was built for high-speed testing did provide access to the electrical input and output buses. These high-speed buses (100 Mb/s to 200 Mb/s) were laid out very carefully to ensure data integrity.



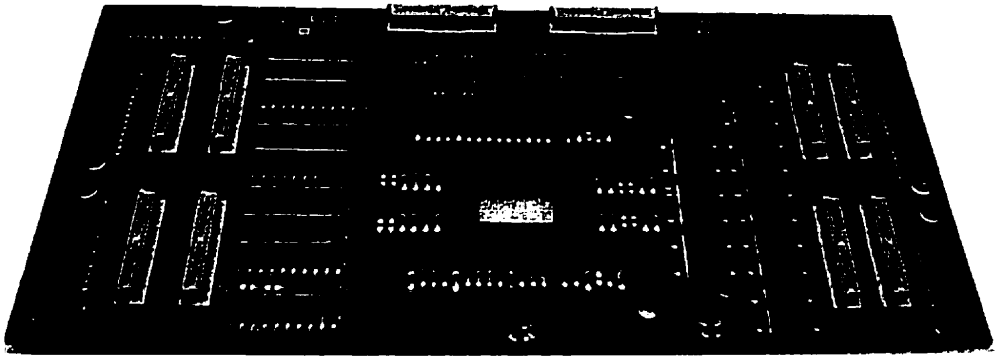
(a) Validation Board



(b) D2C on Validation Board using the chip-on-board approach



(c) DIO Interface Board



(d) Printed Wiring Board

Figure 16: Boards that were designed to test the D2C.

The DIO Interface Board of Figure 16 (c) appears in the high-speed test setup shown in Figure 17. This very simple board served as an interface between the 50-pin connector of the digital I/O card and the 40-pin AMP connectors on the Printed Wiring Board. The board on the left hand side of Figure 17 also serves as an interface, but this time between SMA connectors and the AMP connector. This setup can test one Golay encoder/decoder pair at the time because the data generator only has 12 high-speed (1.25 GHz) outputs, which is just enough to drive a (24, 12) Golay encoder in parallel. A setup similar to that of Figure 17 has to be used on the receive side. The only difference is that a signal analyzer replaces the data generator. A BER tester can also measure the BER of the link with and without FEC for comparisons.

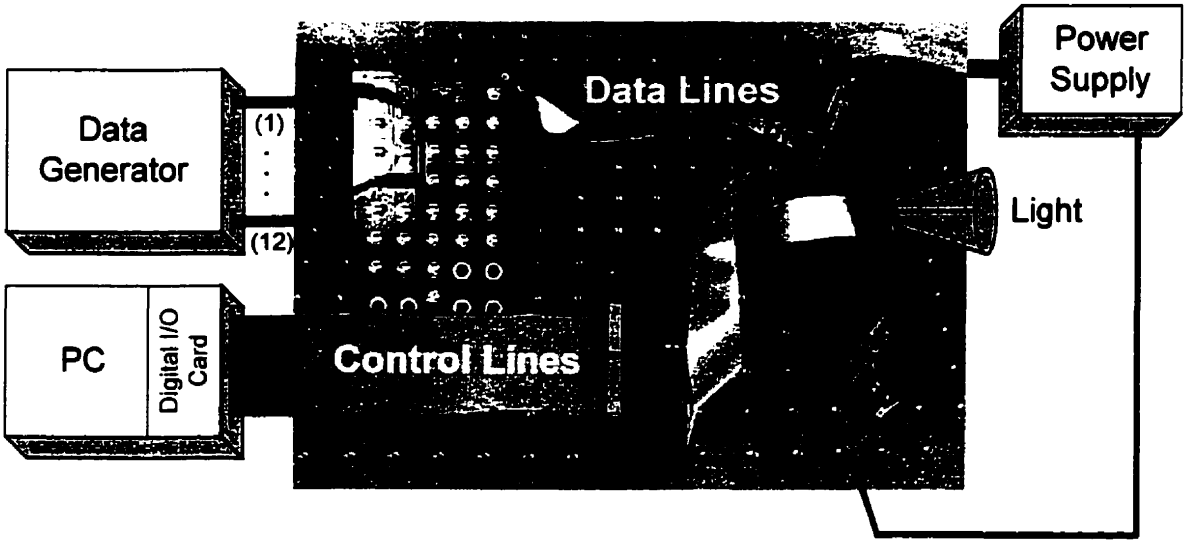


Figure 17: High-speed test setup.

Software

Testing huge arrays of VCSELs and PDs can be somewhat time consuming, especially when a lot of control bits need to be set for the chip to be operational. Figuring out these bits and setting them manually (which involves enabling the control register, scanning bits in and clocking the right number of times) can be prone to errors. This problem was solved by the three software interfaces that will be presented next.

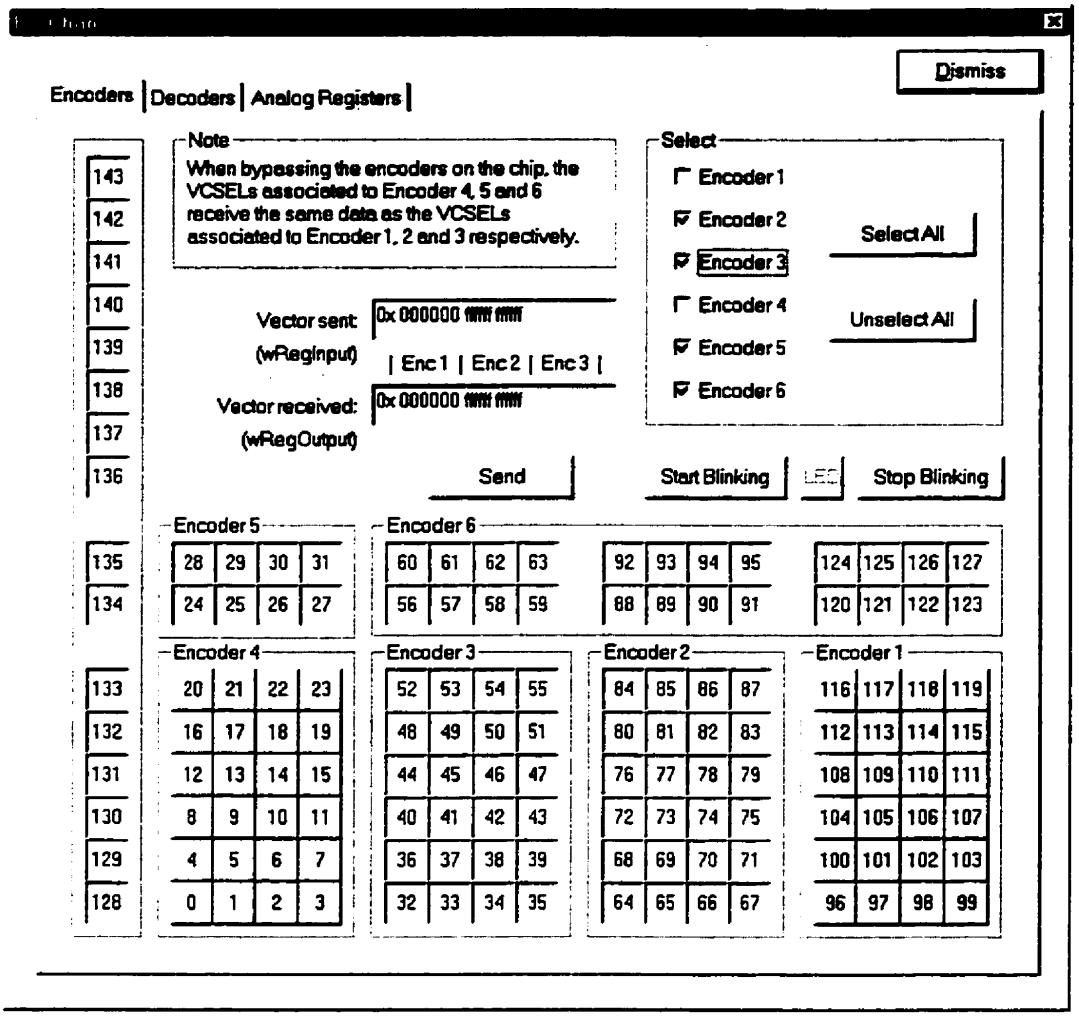


Figure 18: Software interface to test the transmitters and the VCSELs.

The software interface of Figure 18 was designed to facilitate the testing of the transmitters and VCSELs. With this interface, the user can only test *ecChanEnc* in bypass mode. Selecting the entire array and pressing the *Send* button is a good way of finding the yield of the transmitter/VCSEL array (a button that is pushed means that the associated VCSEL pair should output a '1'). One could also use this software interface to make a certain shape and try to detect the same shape with the software interface of Figure 19. One last feature of the interface is that it can make the VCSELs blink at 1 Hz to confirm that they can be modulated.

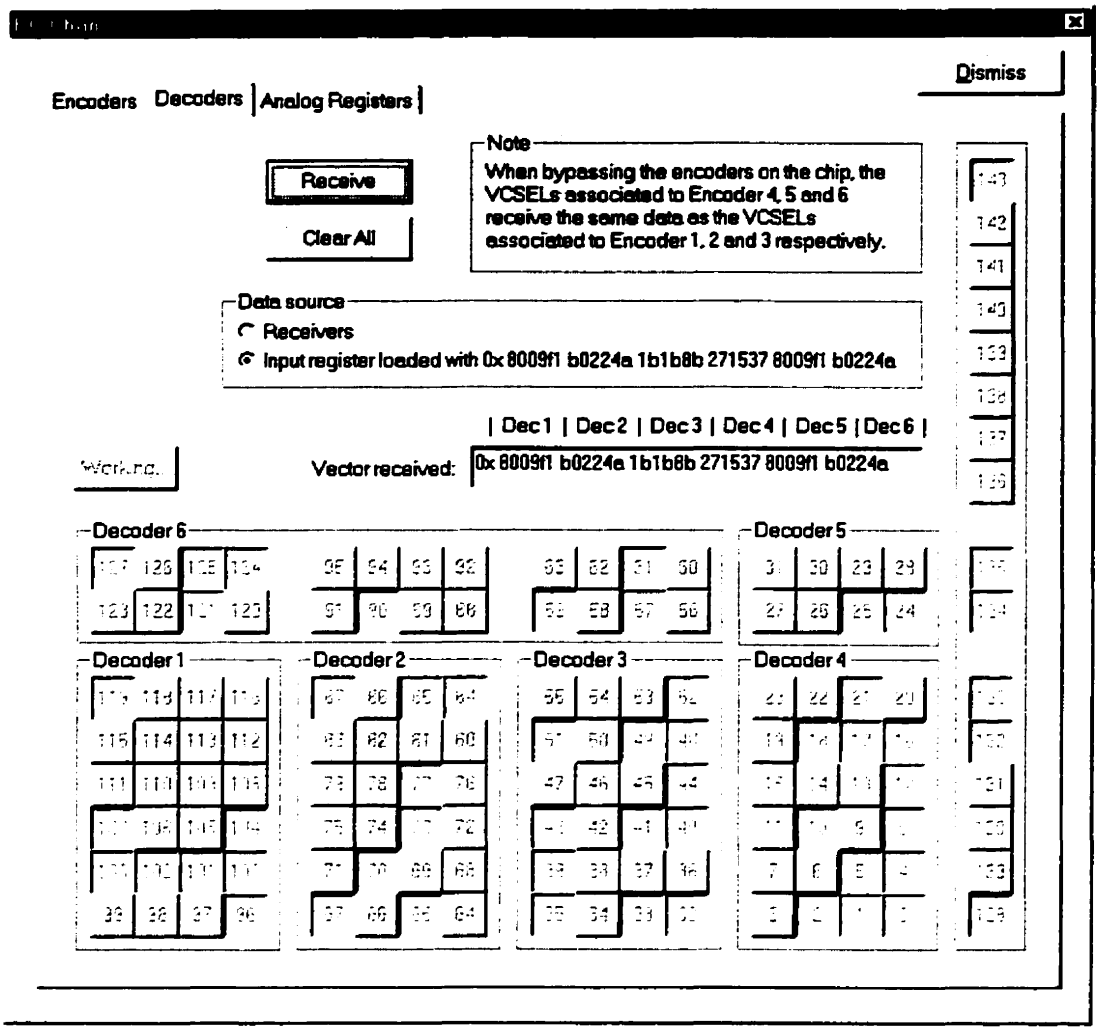


Figure 19: Software interface to test the PDs and receivers.

The software interface of Figure 19 was designed to facilitate the testing of the PDs and receivers. With this interface, the user can either operate *ecChanDec* in bypass mode (if the data source selected is *Receivers*) or in test mode (if the data source selected is *Input register*). Scanning a single beam over the array and pressing the *Receive* button at each new position is a good way of finding the yield of the PD/receiver array (a receiver reading a '1' will push its associated button). Moving an array of VCSEL beams over the array of PDs and pressing the *Receive* button repetitively could help align the two arrays. The shape obtained in the GUI could help figure out crude misalignments in x, y and θ_z.

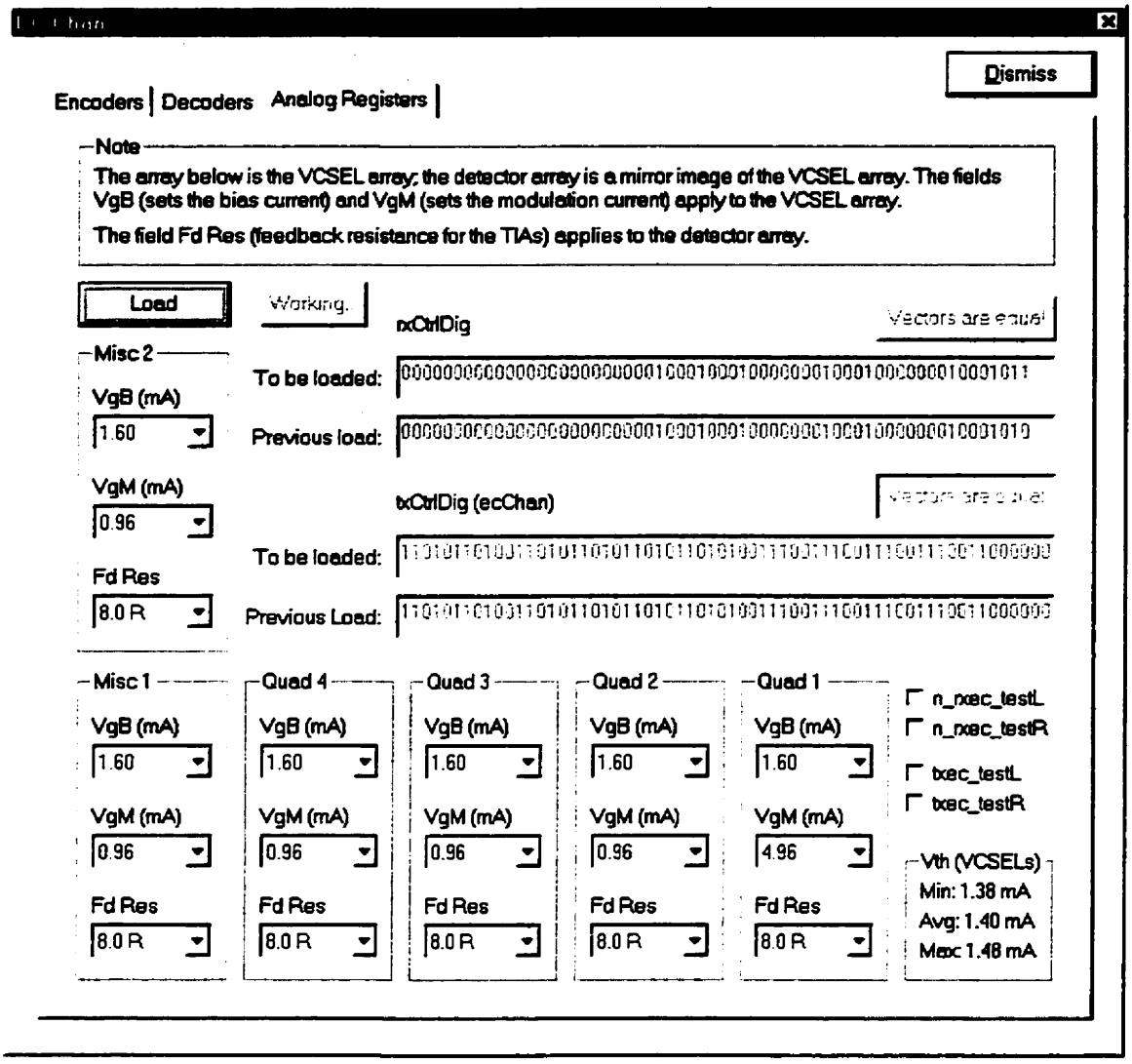


Figure 20: Software interface to set the bias and modulation currents for the VCSELs as well as the feedback resistance for the receivers.

As it can be seen in Figure 20, there are four quads and two misc sections to control on the D2C. For control purposes, Quad 1 corresponds to all the transmitters and receivers of en/decoder 1 as well as some of the transmitters and receivers of en/decoder 6. A comparison of Figures 18, 19 and 20 will make clear the mapping between the quads and the en/decoders. The parameters of each quad or misc section can be set independently. The modulation current (*VgM*) for the VCSELs can be set between 0 and 4.96 mA in 160-μA increments (this requires five control bits). The bias current (*VgB*) can be set between 0 and 2.28 mA in 80-μA increments (this requires an additional five control bits). Four more control bits were necessary to set the feedback resistance of the receivers (note that the feedback resistance sets the gain of

the receivers). The total number of control bits to set is therefore 84 (14 bits/section × 6 sections). It would be very time consuming and prone to errors to figure out which bits to set every time a parameter needed to be tweaked. The GUI of Figure 20 is very handy in that it translates the current and resistance values into control bits. Upon pressing the *Load* button, the GUI then takes the necessary actions to load the analog control registers.

4.3 Experimental Results

The experimental results that will be presented in this section were obtained with a D2C wirebonded to a Validation Board. As explained above, the Validation Board was designed to perform a function verification of the chip as well as some low-speed tests (kHz range). High-speed tests require an hybridized D2C flip-chipped onto an Interposer Board, which will itself be flip-chipped onto a Printed Wiring Board. Unfortunately, due to manufacturing problems and delays, none of this hardware was available at the time this thesis was written. The hybridization of thousands of VCSELs and PDs onto a silicon chip is not trivial and as a result, only one of the four hybridized chip delivered to us so far was healthy.

4.3.1 Electrical Tests

The setup to perform electrical tests on the D2C simply required a computer (with a digital I/O card) and a Validation Board

The testing of *ecChan* was divided up into two batteries of test. The first battery of tests verified the functionality of all the logic (registers, multiplexers, error generator) around the encoder and decoder blocks; the second battery of tests verified the functionality of the encoder and decoder blocks. The two batteries of test will now be presented.

4.3.1.1 Testing of the logic around the encoder and decoder blocks

The functionality of the shift-registers in ecChan was verified by performing serial scans of random bits in and out of the registers. The GUI of Figure 21 (a) was used to perform scan tests on all six registers of *ecChan* (see Figures 12 and 15). Figure 21 (b) shows that the six registers passed the serial scan test (i.e. the bits loaded in were identical to the bits loaded out).

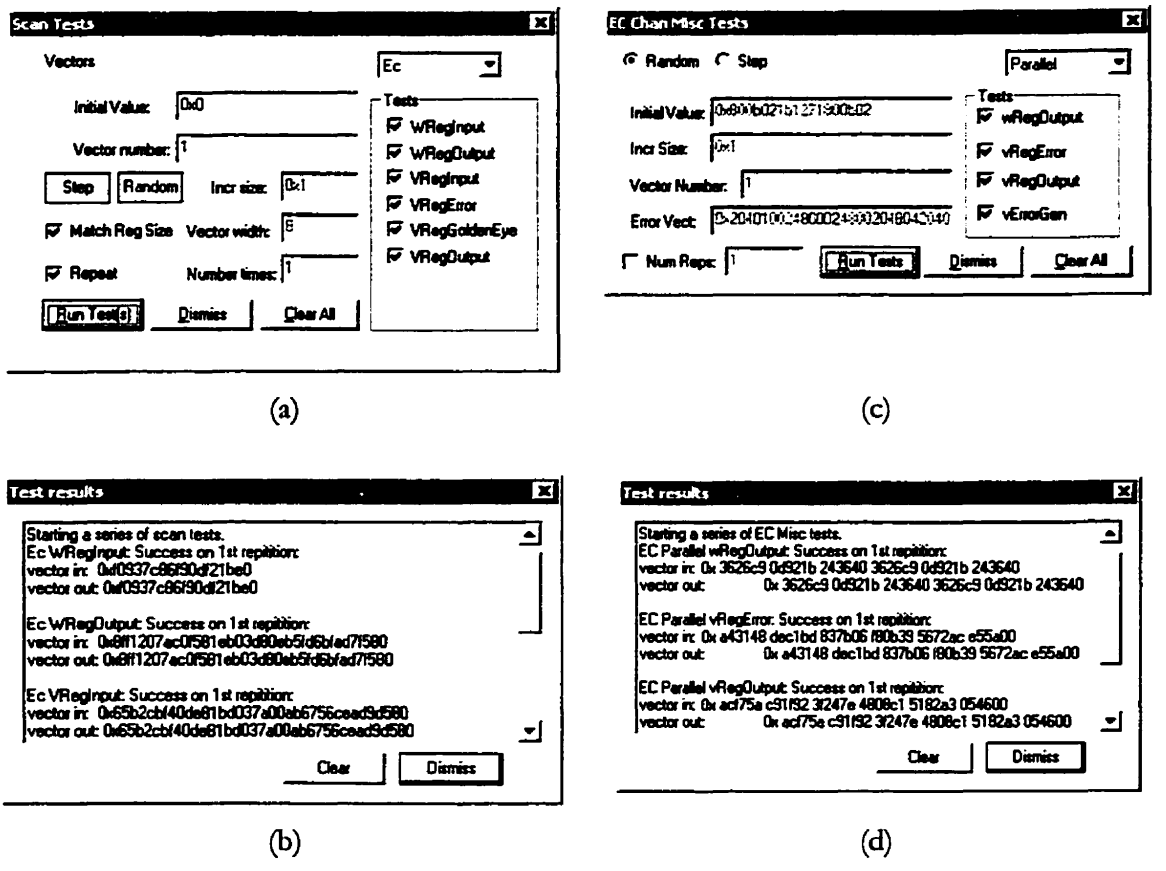
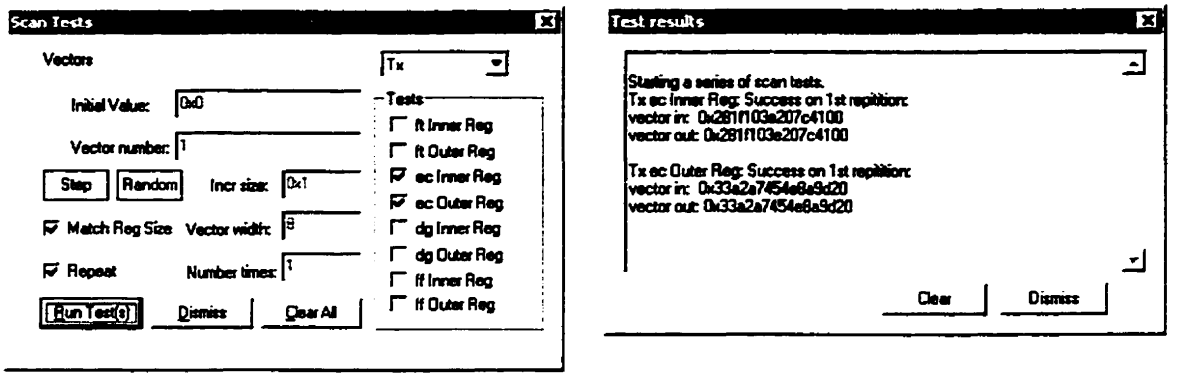


Figure 21: Testing of the registers in ecChan.

The GUI of Figure 21 (c) was used to test the parallel-load shift-registers (those registers are a subset of the shift-registers). As it can be seen in Figure 21 (d), all four registers passed the parallel-load test. The analog control resisters (one for the transmitters and one for the receivers) were the next two blocks to be tested. Unfortunately, after some testing and debugging on the hybridized D2C, it became clear that the analog control register for the receivers could not be programmed. The problem was tracked down to the fact that it was not possible to enable the programming of the register at all. Another instance of the D2C (this one had no hybridized VCSELs and PDs) was tested to find out whether the problem was a design problem or simply a fault on that particular hybridized D2C. Because it was possible to program the register on the non-hybridized D2C, the conclusion was that the problem was a fault problem. This problem simply meant that it would not be possible to tweak the gain of the receivers when optical tests are performed; the default value would have to be used (at least for this particular D2C).

The analog control register for the transmitters was tested using the GUI of Figure 22 (a). Figure 22 (b) shows that the register did pass the test, which simply meant that we were going to have total control over the bias and modulation currents for the VCSELs.



(a) (b)

Figure 22: Testing of the analog control register for the transmitters.

The last blocks that needed to be tested to confirm the functionality of the logic around the encoder and the decoder blocks were the multiplexers and the error generator. The testing of these components was integrated in the testing of the registers and therefore their functionality was verified indirectly.

4.3.1.2 Testing of the encoder and decoder blocks

The logic around the encoder and decoder blocks being fully (or almost fully) operational, the next step was to use this logic to exercise the encoder and the decoder. Figure 23 (a) shows the GUI used to test the encoder and the decoder. *ecChanEnc* and *ecChanDec* were both operated in test mode (i.e. the encoder and the decoder were not bypassed). To test the encoder, a 72-bit random vector was serially loaded into the leftmost registers of Figure 12 and then used as an input to the encoder. The encoder output (a 144-bit vector) was then collected by the rightmost register and serially scanned out for comparison with the encoded input generated in software. Figure 23 (b) shows that the encoder performed as expected.

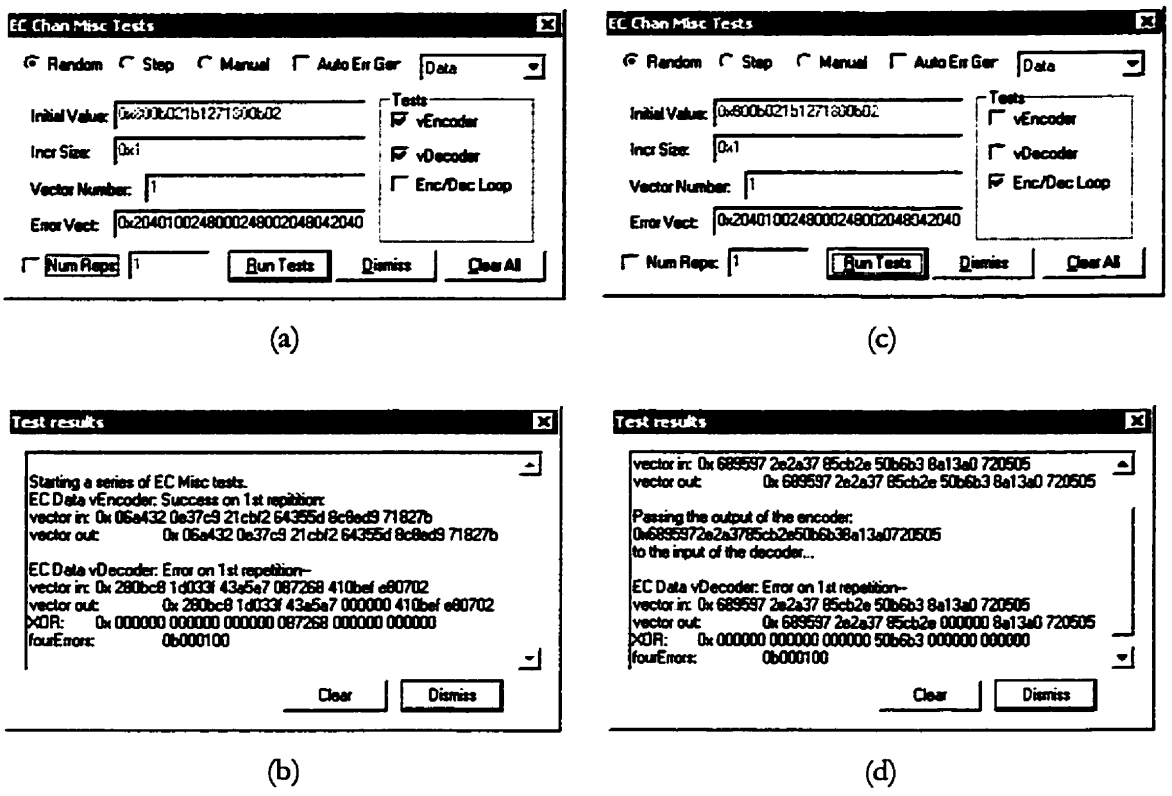


Figure 23: Testing of the encoder and the decoder.

To test the decoder, a 144-bit random vector was serially loaded into the leftmost registers of Figure 15 and then used as an input to the error generation block, which had previously been loaded with the following error pattern:

	Decoder 1 *3 errors*	*Decoder 2* *3 errors*	*Decoder 3* *3 errors*	*Decoder 4* *4 errors*	*Decoder 5* *3 errors*	*Decoder 6* *3 errors*
0x	204010	024800	024800	204804	204010	024800

This error pattern means that all the decoders but one had to correct three errors. One of the decoders (Decoder 4) was expected to flag the 4-error pattern. The output of the error generation block, which corresponds to a codeword with some errors added, was then applied at the input of the decoder. The decoder output was collected by the rightmost register and serially scanned out for comparison with the decoded input generated in software. Figure 23 (b) shows that all five decoders that had to correct three errors were successful. Decoder 4 was also successful in detecting the 4-error pattern since the fourth bit (which corresponds to Decoder 4) was set in the bottommost line of Figure 23 (d).

The last step was to run an encoder/decoder loop test. In this test, a 72-bit vector is encoded in hardware and used as an input to the decoder. The vector decoded in hardware can then be reapplied at the input of the encoder to close the loop. Errors could even be introduced in real-time via the error generation block. Figures 23 (c) and (d) show exactly the experiment just described. The encoder/decoder loop test and can be run for a very long period of time to confirm the robustness of the encoder and decoder blocks.

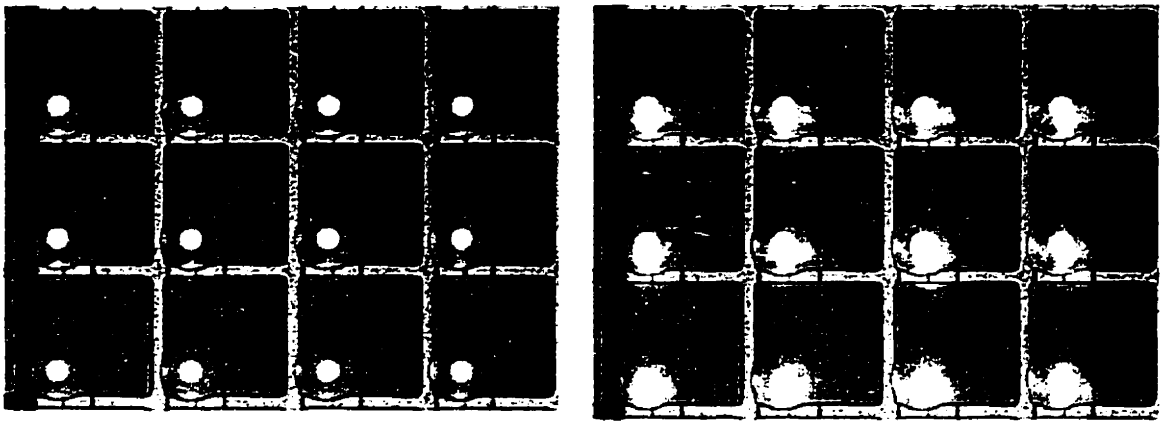
4.3.2 Optical Tests

In this section, the word transmitter will refer to the two VCSELs that implement optical outputs in addition to the underlying silicon circuitry that implements the electrical portion of the electrical-to-optical (E to O) data conversion, and to which the two VCSELs are flip-chip hybridized.

Similarly, the word receiver will refer to the two PDs that convert the optical input signals into electrical current signals (optical-to-electrical conversion) input to the underlying silicon circuitry, as well as the silicon circuitry itself.

4.3.2.1 Transmitters

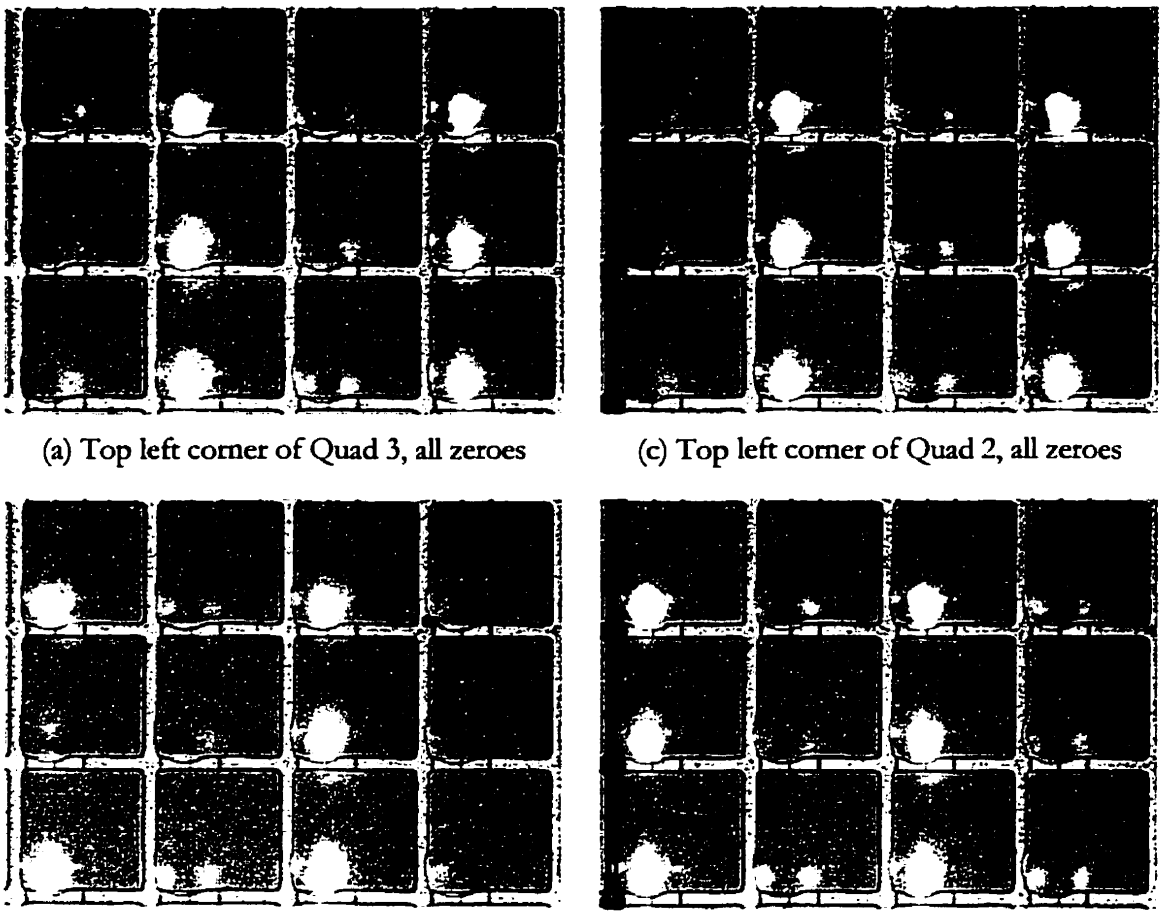
No particular setup was needed to test the transmitters. A D2C wirebonded onto a Validation Board was simply put under a microscope to observe the behaviour of the transmitters with different test inputs. Two low-speed tests were performed on the transmitter array. During the first test, the modulation current was kept to zero and only the bias current was varied. Knowing that the threshold current of the VCSELs was somewhere between 1.38 mA and 1.48 mA (with an average value of 1.40 mA), the VCSELs were first biased just below threshold – see Figure 24 (a). The bias current was then raised to 2.08 mA to make the VCSELs lase – see Figure 24 (b). Of the 288 VCSELs in ecChan, only 3 of them were dead, for a yield of 98.96%.



(a) Just below threshold (VgB = 1.36 mA) (b) Above threshold (VgB = 2.08 mA)

Figure 24: Testing of the VCSELs. Only a bias current is applied (i.e. VgM = 0 mA).

The next step was to try to modulate the VCSELs. To perform this test, the bias current was set to 0 mA and the modulation current was set to 2.08 mA. The modulation was done at 1 Hz using the software interface depicted in Figure 18. Figures 25 (a) and (c) show two sections of the transmitter array sending a '0'; Figures 25 (b) and (d) show two sections of the transmitter array sending a '1'. One of the three dead transmitters of *ecChan* can be seen in Figure 25 (b). All transmitters were successfully modulated at 1 Hz.



(a) Top left corner of Quad 3, all zeroes

(c) Top left corner of Quad 2, all zeroes

(b) Top left corner of Quad 3, all ones (one VCSEL is dead)

(d) Top left corner of Quad 2, all ones

Figure 25: Testing of the VCSELs. Only a modulation current is applied (i.e. VgB = 0 mA). Each figure shows six VCSEL-pairs being modulated by six differential transmitters with VgM = 2.08 mA.

4.3.2.2 Receivers

Figure 26 shows the setup that was used to test the receiver array on the D2C. A custom-made PCB drove the 850-nm VCSEL on the left hand side of the figure. A 4f system was used to image the VCSEL spot onto the PD array. A beam splitter was put between lens #1 and lens #2 to look at the chip with a CCD camera. This arrangement allowed us to see the VCSEL spot reflected from the chip as well as the individual PDs so that the alignment between the two can be done.

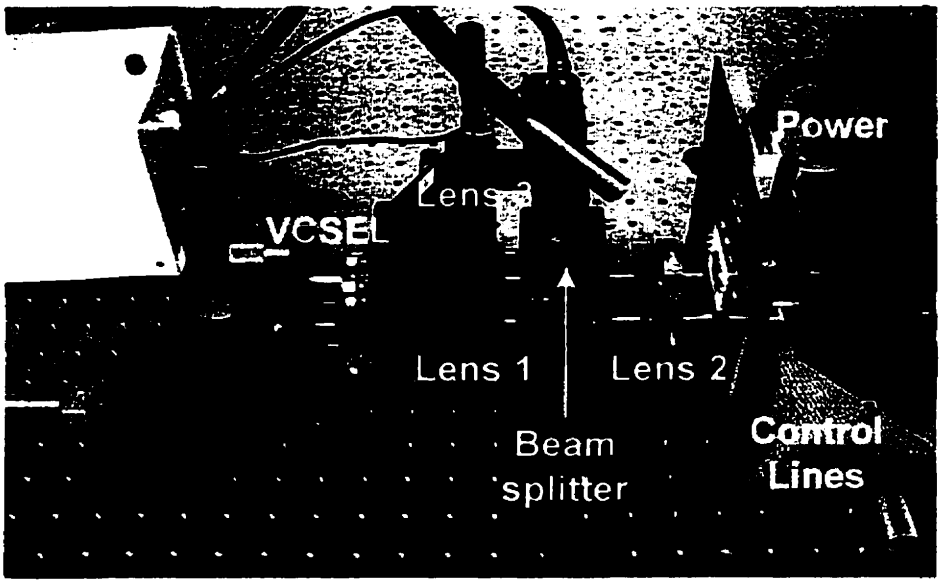


Figure 26: Setup to test the receiver array.

The power measured at the output of the VCSEL was 4 mW, whereas the power measured at the chip plane was 1 mW. The power lost through the optics was mainly due to the clipping of the VCSEL beam by the first lens. Some power was also lost through the beam splitter. Ideally, because the D2C is optically differential, two complementary VCSEL spots should have been used to test the receiver array.

Figure 27 shows the results obtained by scanning the VCSEL spot over the entire array of receivers in *ecChan*. Out of the 144 differential receivers, we found out that only 88 could successfully receive near DC modulated data. Due to manufacturing problems, no other hybridized chip was available for comparisons at the time this thesis was written. It is therefore hard to explain the low yield obtained. One possibility is that some PDs may not have survived during the flip-chipping process. The fact that the gain of the receivers could not be tweaked is another possibility that could explain the low yield obtained.

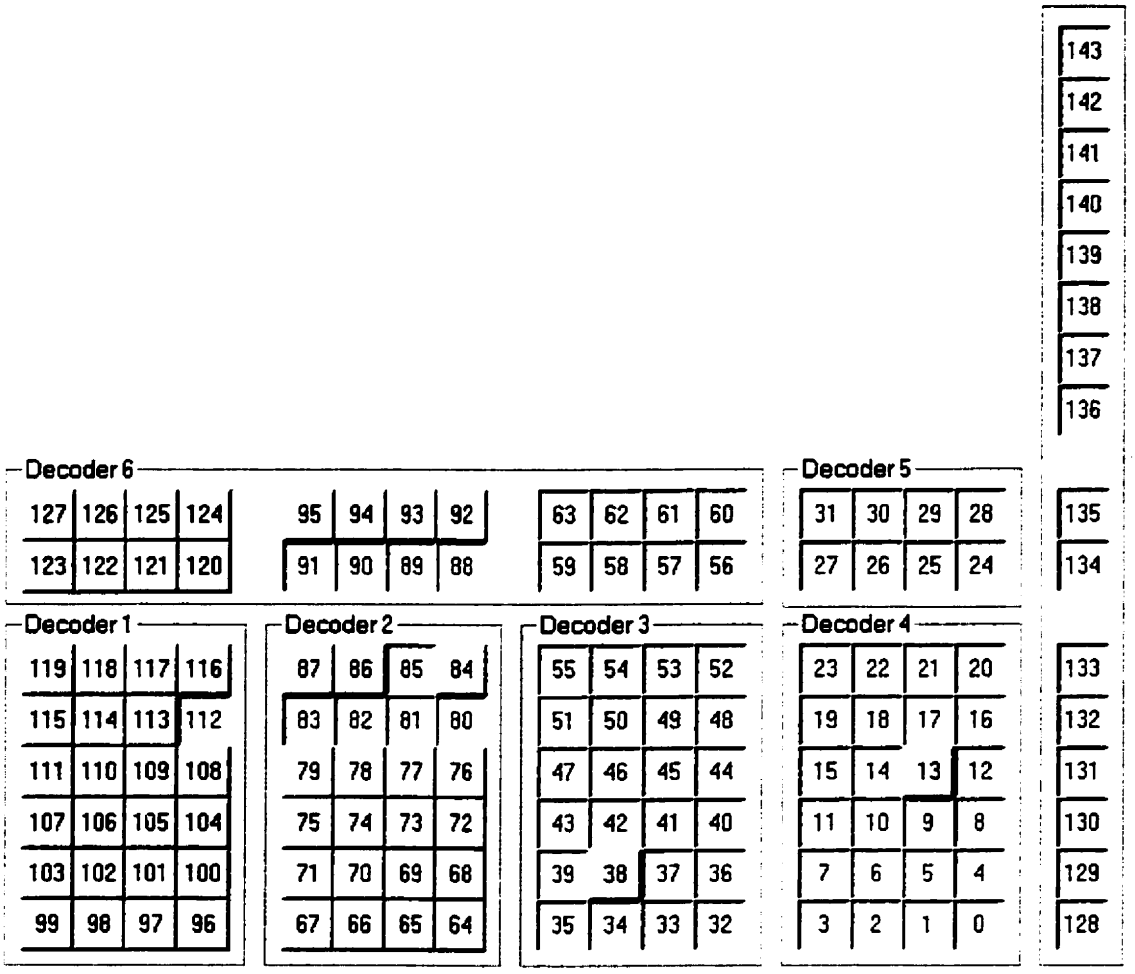


Figure 27: Yield of the receiver array. The dead receivers are those for which the button is not pushed in.

4.4 Conclusion

Unfortunately, because there was only one hybridized chip available to us, a point-to-point optical link couldn't be established in the end. When the hardware becomes available, it will be interesting to test *exChan* at high-speed using the setup of Figure 17. From Sections 4.1.1 and 4.1.2, it is expected that data will be encoded and decoded in about 2.23 ns and 8.03 ns respectively. The decoder therefore limits the maximum speed at which data can be transmitted over a sub-link. A delay of 8.03 ns does not necessarily translate into a maximum speed of 125 Mb/s. We have to account for the delays introduced by the transmitters, the receivers and/or any other component along the path from the encoder to the decoder.

Assuming a maximum speed of, say, 100 Mb/s, the aggregate bandwidth of *ecChan* is expected to be around 7.2 Gb/s (72 sub-links running at 100 Mb/s).

Other interesting experiments could be run once the manufacturing problems get solved. One of them will obviously be an experiment that will confirm that FEC can indeed, in practice, improve the reliability of PODLs. It will be interesting to find out whether FEC can help tolerance a PODL for misalignments. Another question that has yet to find an answer is whether the coding gain obtained from FEC is big enough to reduce the total power consumption of an OE-VLSI chip.

4.5 References

1. M.A. Neifeld, and R.K. Kostuk, "Error correction for free-space optical interconnects: space-time resource optimization", Applied Optics, Vol. 37, pp. 296-307, 1998.

2. M.A. Neifeld, and M. McDonald, "Error-correction for increasing the usable capacity of photorefractive memories", Opt. Lett., Vol. 19, 1994.

3. T.H. Szymanski, "Bandwidth optimization of optical data links by use of error-control codes", Applied Optics, Vol. 39, pp. 1761-1775, 2000.

4. M.A. Neifeld, and S.K. Sridharan, "Parallel error correction using spectral Reed-Solomon code", J. Opt. Commun., Vol. 17, 1997.

5. S. Lin, and D.J. Costello Jr., "Error control coding: fundamentals and applications", Prentice-Hall, N.J., 1983

6. S.-W. Wei, and C.-H. Wei, "On high-speed decoding of the (23, 12, 7) Golay code", IEEE Trans. Inform. Theory, Vol. IT-36, pp. 692-695, 1990.

7. J.L. Anderson, "On minimal decoding sets for the extended binary Golay code," IEEE Trans. Inform. Theory, Vol. IT-38, pp. 1560-1561, 1992.

8. J. Wolfmann, "A permutation decoding of the (24, 12, 8) Golay cod", IEEE Trans. Inform. Theory, Vol. IT-29, pp. 748-750, 1983.

9. D.M. Gordon, "Minimal permutation sets for decoding the binary Golay codes," IEEE Trans. Inform. Theory, Vol. IT-28, pp. 541-543, 1982.

10. W. Cao, "High-speed parallel VLSI-architecture for the (24,12) Golay decoder with optimized permutation decoding", IEEE International Symposium on Circuits and Systems, Vol. 4, pp. 61-64, 1996.

11. E. Laprise, M. Venditti, and J. Faucher, "VLSI-Photonics Demo 2 ASIC: Functional Specifications, McGill University, 2000.

12. F.J. MacWilliams, and N.J.A. Sloane, "The theory of error-correcting codes", North-Holland Publishing, New York, 1977.

13. W.W. Peterson, and E.J. Weldon Jr., "Error-correction Codes," 2nd edition, The Colonial Press, Inc., U.S., 1972.

14. M.B. Venditti, Ph.D. thesis, McGill University, to be published.

15. J.J. O'Reilly, Y. Bian, A. Popplewell, S. Fragiacomo, R. Blake, "Forward error control for future trans-oceanic optical systems", Fifth IEE Conference on Telecommunications, pp. 78-82, 1995.

16. H. Lee, M.-L. Yu; L. Song, "VLSI design of Reed-Solomon decoder architectures", Proceedings of the 2000 IEEE International Symposium on Circuits and Systems, Vol. 5, pp. 705-708, 2000.

Chapter 5

Design considerations and future work

There are situations where FEC alone won't be sufficient to keep the BER below 10^{-15}. In this chapter, we will therefore consider a few design strategies (some of which have briefly been mentioned earlier in this thesis) that could be used in conjunction with FEC to help improve the reliability of PODLs even more. These design strategies have not been implemented on the D2C (except the *optically differential design* strategy), but are simply suggestions for future work. In addition to design strategies for future work, this chapter will also contain remarks or design considerations.

5.1 Optically differential design

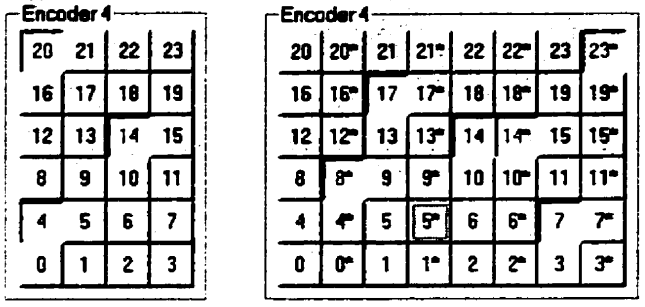
The D2C is electrically and optically differential (as opposed to being single-ended). There are advantages and disadvantages to using such an approach.

Advantages

For the VCSEL drivers, the use of a differential pair configuration could mean that a constant current is drawn from the power supply at all times [2]. This should have a positive impact on the SNR. A better SNR means a lower PER (see Equation 12), so a differential PODL should be more reliable than a single-ended PODL. A better SNR also means that the power can be reduced to obtain the same PER (see Figure 9). This would partly offset the extra power needed to convert a single-ended design to a differential one. On the receiver side, differential transmission could mean that the link does not depend on external reference levels, and such links are therefore more robust to non-uniformities [1].

Disadvantages

The main disadvantage of using a differential approach is certainly that it doubles the number of VCSELs and PDs that need to be flip-chip bonded onto the chip. This may have a negative impact on the yield and also make the optics harder to design and/or more expensive. As illustrated in Figure 28, the number of erasures in a PODL is directly related to the number of VCSELs and PDs. Consider for example the differential design of Figure 28 (b). In practice, if both VCSELs in a pair are dead (e.g. VCSELs 14 and 14*), then this sub-link is considered as an erasure; if only one VCSEL in a pair is dead (e.g. VCSEL 17), then this sub-link is still considered as an erasure. The latter argument applies to any other component along the sub-link. The number of erasures in a differential design should therefore be about twice the number of erasures in a single-ended design. This argument assumes that the erasures are randomly distributed across the array. If there is any clustering, then this doubling of the number of erasures will not hold anymore: two adjacent dead VCSELs, for example, will account for only one erasure.



(a) Single-ended (b) Differential

Figure 28: If the yield is the same for both arrays, then the number of erasures in a differential design is twice the number of erasures in a single-ended one, unless two erasures fall on a differential pair such as VCSELs 14 and 14* in (b).

While enumerating the disadvantages of a differential design, it may be tempting to say that it will consume exactly twice the power of a single-ended design. One has to be careful with this statement because a differential design doubles the number of VCSELs and PDs, but not the number of transmitters and receivers. A differential transmitter (or receiver) does not necessarily consume twice the power of a single-ended one. The extra power consumed by a differential optical design should therefore be less than 3 dB.

An interesting experiment would be to plot the PER vs. SNR for a single-ended design and a differential design (the plots obtained should be similar to Figure 9) and see how much power you can save for a given PER. One could then compare this power saving (if any) to the additional power needed for the differential design.

5.2 Interleaving

As we have seen in Section 2.4, the distribution of the erasures has an impact on the performance of the Golay decoder. Like most FEC codes, the Golay code is good at correcting randomly distributed errors. Interleaving can come at the rescue of FEC in a clustered-error environment. An interleaver spreads the clustered errors and therefore enables an improved clustered-error performance [4, 5]. Figure 29 shows where the interleaving and de-interleaving blocks would fit in the picture.

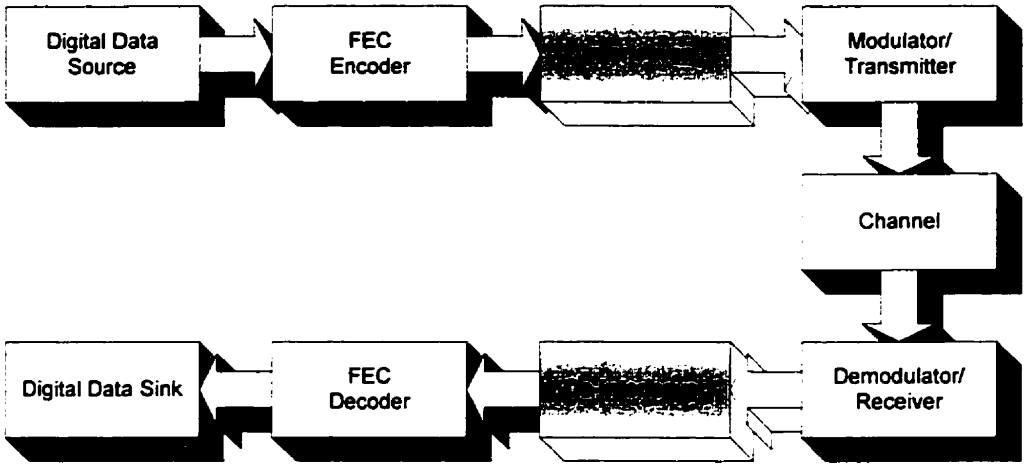


Figure 29: Block diagram of a coded channel with interleaving.

Interleaving can happen in time and/or in space. Since we are dealing with a PODL, space interleaving will be discussed; the interested reader can have a look at Ref. [3] for time interleaving over a serial link. Space interleaving in *ecChan* on the D2C would mean an organized scrambling of the VCSELs and PDs associated with each encoder/decoder. With no interleaving, the 24 bits of a Golay encoder are transmitted over adjacent sub-links (see Figure 6). Clustered errors, in this case, will kill the PER of the PODL because a decoder is likely to have to deal with more than 3 errors. With interleaving, the same 24 bits would be transmitted a few sub-links apart (see Figure 30).

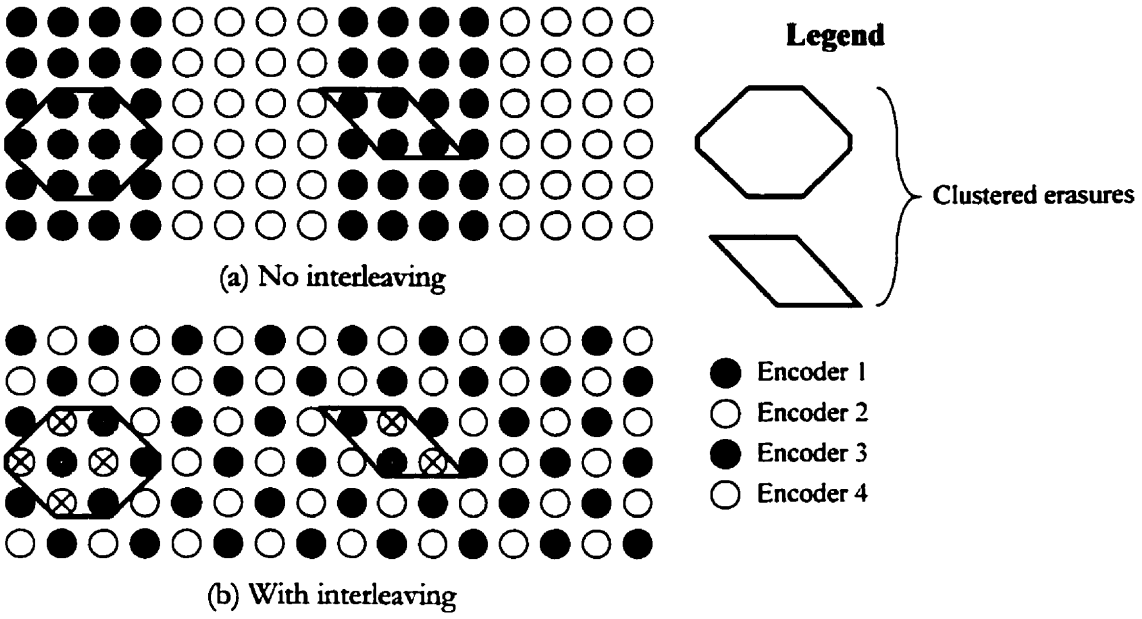


Figure 30: VCSEL array divided up into four 24-bit packets. Each packet is associated with one Golay encoder. In (a), the two clustered errors will kill the performance of the PODL because two of the four decoders will have to deal with more than three erasures. In (b), because of interleaving, all the erasures will be corrected because each decoder only has to deal with three erasures.

When de-interleaved at the receiver, clustered errors are thereby spread out randomly and are suitable for a random error-correcting decoder such as the Golay decoder. Naturally, interleaving does not provide coding gain the way the Golay code would. It simply allows the Golay decoder to achieve close to its theoretical coding gain, rather than be degraded while trying to correct clustered errors.

There are two drawbacks to using interleaving to improve the reliability of a PODL. First, as shown in Figure 29, interleaving adds two more blocks to the transmit/receive path. The latency added by the interleaving/de-interleaving blocks may not be acceptable for certain applications. The second drawback has to do with layout considerations. When laying out a chip, it is always preferable to minimize the length of metal tracks. Evidently, because interleaving causes non-locality issues, minimizing the length of metal tracks will be harder. As a result, the layout of an interleaved design will take more area, may not be optimized for speed, and may not run as fast as a non-interleaved design.

5.3 Remapping

When there are too many erasures and/or too many bad sub-links in a PODL, FEC and interleaving alone won't be sufficient. For example, only one more erasure in Figure 30 would cause a decoder to fail to correct all the errors it has to deal with. This in turn would cause the PER of the PODL to escalate very quickly. One family of codes, the Reed-Solomon codes, can take advantage of the knowledge of which sub-links are dead to give a significant performance improvement. This knowledge is otherwise useless to most of the other FEC techniques, which are only good at correcting random errors. For these FEC techniques, one may be forced to resort to remapping.

The remapping technique, like the Reed-Solomon codes, makes use of the knowledge of which sub-links are dead. In an array that has a few spare sub-links (see Figure 3), a remapping block, however it is designed, could perform some reassignments. Data that was originally supposed to be transmitted over a dead sub-link will now be transmitted over one of the spare sub-links. On-chip remapping, just as interleaving, is expected to complicate the layout significantly and may not be a suitable solution for applications sensible to latency. To the author's knowledge, a remapping strategy that is practical for a layout implementation has yet to be demonstrated. The problem is that a massively parallel optical data link would require a lot of wiring and multiplexing to give a backup to every single sub-link in the array.

5.4 Soft-decision decoding

FEC decoding can be performed using either soft [11] or hard decisions. In hard-decision decoding, when each signal value is received, a "binary" decision is made by the receiver to determine whether the signal represents a transmitted zero or a one. Then these decisions form the input sequence to the FEC decoder. In soft-decision decoding, the receiver takes advantage of the side-information generated by the bit-decision circuit. With the advent of high-speed analogue-to-digital (A/D) converters, it is now possible to digitize the incoming bit stream rather than simply "slicing" it. Instead of being represented by a 0 or 1, each input bit can be represented by a number between zero and seven (using 3-bit quantization). The usual quantization precision is three bits. More bits provide little additional improvement [9]. A high value such as seven represents a good probability that the bit is a 1, while a low value such as zero is a high probability that the bit is 0. Central values like three and four represent low

probabilities of the bit being correct. Typically a bit in error will have a low probability and this could allow the FEC decoder to make informed decisions even where the error-correction capacity of the code has been exceeded.

The D2C used hard decision decoding, but soft-decision decoding could provide better error correction. The theoretical performance gain obtained from using soft-decision decoding will be presented next.

Performance gain obtained from using soft-decision decoding

The use of soft decisions within an error-correcting code can give as much as a 2-dB SNR advantage over a simple hard-decision scheme [7, 8, 9]. As pointed out earlier, the performance gain will vary with the modulation scheme and the dominant type of noise in the system.

For the particular case of the (24, 12) Golay code, it was shown in [6] (see Figure 31) that an additional coding gain of 1.5 dB (at a BER of 10^{-4}) could be obtained by using a new decoding algorithm to perform soft-decision decoding instead of hard-decision decoding. This additional coding gain could then be used at the designer's discretion to improve the reliability of the PODL or to decrease the power consumption, just to name a few of the options available (see Section 3.4 for other options).

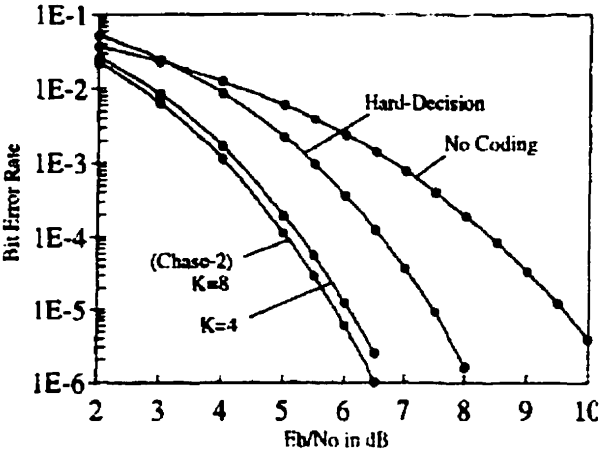


Figure 31: Performance of a new decoding algorithm to decode the (24, 12) extended Golay code in AWGN with 8-level soft-quantized BPSK. K is the number of test patterns of the algorithm. See Ref. [6] for details on the algorithm.

As for the other techniques that can be used in conjunction with FEC, soft-decision decoding has a drawback. It does provide better error correction, but at the expense of circuit complexity. The A/D converter adds to the processing time of the bits, not to mention that a FEC decoder that can handle soft-decision inputs will be more complex.

5.5 8b/10b and Manchester encoding

The D2C used intensity modulation (i.e. direct modulation) with direct detection (IM/DD). The advantage of this modulation scheme is that it is very simple. On the other hand, intensity modulated data is not inherently balanced, which may increase the PER significantly as we saw in Section 2.2. If the reliability expectations of a system are very high, this reason alone could justify the need for balanced data. But there is a second argument that works to the advantage of balanced data. The lack of a DC balance can potentially result in data-dependent heating of certain VCSELs due to a transmitter sending more 1s than 0s, resulting in higher error rates.

The addition of an 8b/10b-encoder block (and its associated decoder block) was proposed earlier as a way to balance the otherwise unbalanced data produced by intensity modulation. 8b/10b could offer another advantage. The 8b/10b-encoding scheme was designed to provide a high transition density [10], which is useful for bit synchronization and clock recovery purposes. 8b/10b has its drawbacks though. Depending on the application, it could cost too much in terms of silicon area, latency and power dissipation. Manchester encoding has also been proposed to satisfy the requirement for balanced data. The arguments against 8b/10b encoding apply equally well to Manchester encoding. Moreover, in accordance with the Manchester encoding/decoding algorithm, the Manchester encoder/decoder blocks would need to be operated at twice the clock frequency of the system. This downside of Manchester encoding would need to be taken into account in systems that need to be optimized for speed.

5.6 Conclusion

This chapter proposed that FEC be used in conjunction with other strategies to improve the PER of a PODL. Each strategy offers some great benefits, but they do not come for free. It is up to the designer to decide whether the performance improvement obtained from these strategies justifies the latency, power dissipation and silicon area that they add to a design.

5.7 References

1. MEL-ARI OPTO Achievements, 1996-2000, http://www.cordis.lu/esprit/src/melop-rm.htm

2. M.B. Venditti, Ph.D. thesis, McGill University, to be published.

3. G.C. Clark, and J.B. Cain, "Error-Correction Coding for Digital Communications", Plenum Press, New York, 1981.

4. W.-C. Chou, and M.A. Neifeld, "Interleaving and Error Correction in Volume Holographic Memory Systems", Applied Optics, Vol. 37, pp. 6951-6968, 1998.

5. C. de Almeida, and R. Palazzo Jr., "Two-dimensional interleaving using the set partitioning technique", Electronic Letters, 1996.

6. W. Cao, "High-speed parallel hard and soft-decision Golay decoder: algorithm and VLSI-architecture", IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 6, 1996.

7. Irving S. Reed and Xuemin Chen, "Error-control coding for data networks", Kluwer Academic Publishers, 1999.

8. A. Houghton, "The engineer's error coding handbook", Chapman & Hall, London, 1997.

9. C. Fleming, "A Tutorial on Convolutional Coding with Viterbi Decoding", 2001. http://home.netcom.com/~chip.f/viterbi/tutorial.html

10. A. X. Widmer and P. A. Franaszek, "A DC-Balanced, Partitioned Block, 8B/10B Transmission Code," IBM J. Res. Develop. 27, No. 5, pp. 440-450, 1983.

11. Y. Be'ery, and J. Snyders, "Soft decoding of the Golay codes", IEEE Trans. Inform. Theory, 1988.

Chapter 6

Conclusion

The high bandwidth potential of PODLs makes them an interesting interconnect technology for the next-generation massively parallel systems. One concern with PODLs is whether they can be reliable enough to provide BERs lower than 10^{-15}. Optical data links built from huge arrays of VCSELs, PDs and fibers are likely to contain non working sub-links (erasures) or poorly working sub-links. Poorly working sub-links may be caused by non-uniformities over the array whereas erasures are directly related to the yield.

FEC was proposed as a solution to lower the yield and uniformity requirements. This thesis demonstrated that an OE-VLSI chip with FEC could offer a BER many orders of magnitude lower than without FEC. An evaluation of the BER for IM/DD POIs was also performed.

The ability of a parallel FEC scheme to correct errors depends on the distribution of the errors. Most FEC are good are correcting randomly distributed errors. These FEC schemes can still perform close to their theoretical coding gain in a clustered-error environment if an interleaver block is added in the path between the encoder and the decoder. If the problem is with the number of erasures, then the use of spare cells (remapping) should be considered.

The inclusion of FEC in datacom systems could allow designers to balance reliability, latency, power consumption, data rates and other system attributes that could lead directly to reductions in system costs.

Appendix

In the appendix, we will look at one particular encoder/decoder pair for the Golay codes. As pointed out in Section 4.1, the encoder is simply a vector multiplier and the layout area required by the encoder is usually a small fraction of the layout area required by the decoder. Many decoders have been suggested in the literature. The particular decoder that was put on the D2C was picked because of its suitability for a VLSI implementation. The different steps involved in decoding the Golay codes were tested using a Matlab script. An Excel spreadsheet that shows the output of each of the blocks of the Golay decoder (see Figure 14) will follow.

Matlab script to test the algorithms for encoding and decoding the Golay codes

```
% Golay Codes

% [1] S. Roman, "Introduction to Coding and Information Theory".
% Modular arithmetic in Z_n, see [1] pp. 25-27
%     and [1] p. 182 (inner product).
clear all;

NbPermutations = 14;
wLength = 12;      % Word length.
cwLength = 24;     % Codeword length.

% [BEGIN] Encoding, see [1] p. 208
I = [1 0 0 0 0 0 0 0 0 0 0 0,
     0 1 0 0 0 0 0 0 0 0 0 0,
     0 0 1 0 0 0 0 0 0 0 0 0,
     0 0 0 1 0 0 0 0 0 0 0 0,
     0 0 0 0 1 0 0 0 0 0 0 0,
     0 0 0 0 0 1 0 0 0 0 0 0,
     0 0 0 0 0 0 1 0 0 0 0 0,
     0 0 0 0 0 0 0 1 0 0 0 0,
     0 0 0 0 0 0 0 0 1 0 0 0,
     0 0 0 0 0 0 0 0 0 1 0 0,
     0 0 0 0 0 0 0 0 0 0 1 0,
     0 0 0 0 0 0 0 0 0 0 0 1];

A = [1 0 0 1 1 1 1 1 0 0 0 1,
     0 1 0 1 0 0 1 1 1 0 1 1,
     0 0 1 1 0 1 0 1 0 1 1 1,
     1 1 1 1 0 0 0 0 1 1 1 0,
     1 0 0 0 1 0 0 1 1 1 1 1,
     1 0 1 0 0 1 1 1 1 0 1 0,
     1 1 0 0 0 1 1 0 0 1 1 1,
     1 1 1 0 1 1 0 1 0 1 0 0,
     0 1 0 1 1 1 0 0 1 1 0 1,
     0 0 1 1 1 0 1 1 1 1 0 0,
     0 1 1 1 1 1 1 0 0 0 1 0,
     1 1 1 0 1 0 1 0 1 0 0 1];
```

```
G = [I A];
H = [A' I];
Ht = H';

w = [1 0 1 1 1 0 0 0 0 0 1 0];     % Word to be encoded
cw = mod(w * G, 2);                % Codeword (encoded word).
% Errors introduced by the transmission channel.
e = [0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0];
r = xor(cw,e);                     % Received codeword.

% [END] Encoding.

% Permutation indexes
Pi = [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23;
0 1 2 12 13 14 3 4 5 15 16 17 18 19 20 6 7 8 21 22 23 9 10 11;
0 1 2 18 19 20 12 13 14 6 7 8 21 22 23 3 4 5 9 10 11 15 16 17;
0 1 2 21 22 23 18 19 20 3 4 5 9 10 11 12 13 14 15 16 17 6 7 8;
0 1 2 9 10 11 21 22 23 12 13 14 15 16 17 18 19 20 6 7 8 3 4 5;
0 1 2 15 16 17 9 10 11 18 19 20 6 7 8 21 22 23 3 4 5 12 13 14;
0 1 2 6 7 8 15 16 17 21 22 23 3 4 5 9 10 11 12 13 14 18 19 20;

12 13 14 15 16 17 18 19 20 21 22 23 0 1 2 3 4 5 6 7 8 9 10 11;
18 19 20 6 7 8 21 22 23 9 10 11 0 1 2 12 13 14 3 4 5 15 16 17;
21 22 23 3 4 5 9 10 11 15 16 17 0 1 2 18 19 20 12 13 14 6 7 8;
9 10 11 12 13 14 15 16 17 6 7 8 0 1 2 21 22 23 18 19 20 3 4 5;
15 16 17 18 19 20 6 7 8 3 4 5 0 1 2 9 10 11 21 22 23 12 13 14;
6 7 8 21 22 23 3 4 5 12 13 14 0 1 2 15 16 17 9 10 11 18 19 20;
3 4 5 9 10 11 12 13 14 18 19 20 0 1 2 6 7 8 15 16 17 21 22 23];

Pi = Pi + 1;    % Need reindexing b/c Matlab indexes start at 1.

% (1/5) Permutation Multiplexer Unit
for i = 1:NbPermutations
    for j = 1:cwLength
        P(i,j) = r(Pi(i,j));
    end
    % (2/5) Syndrome Calculation Unit
    S(i,:) = mod(P(i,:)*Ht,2);
end

% (3/5) Look-Ahead Correction Unit
zero_array = zeros(NbPermutations, wLength);
P_corrected = xor(P, [zero_array S]);

% Apply permutation inverses.
for i = 1:NbPermutations
    for j = 1:cwLength
        c(i,Pi(i,j)) = P_corrected(i,j);
    end
end

% (4/5) Weight Calculation Unit
weights = sum(S, 2);

% (5/5) Select Unit
weightsSmallerOrEqualTo3_Indexes = find(weights <= 3);
nbErrorsIs4 = isempty(weightsSmallerOrEqualTo3_Indexes);
```

```
if nbErrorsIs4 == 0
    correctedCodeword = c(weightsSmallerOrEqualTo3_Indexes(1), :);
else
    correctedCodeword = zeros(1,24);
end

% correctedWord should be identical to w if the decoder
% was successful (i.e. the number of errors was less than
% or equal to 3)
correctedWord = correctedCodeword(1:12);
```

Excel spreadsheet that shows the output of each of the Golay decoder blocks (refer to Figure 14 and the Matlab script above)

1	0	1	1	1	0	0	0	0	0	1	0	Word to be encoded
1	0	1	1	1	0	0	0	0	0	1	0	Codeword (enc. word)
0	0	0	0	0	0	1	0	0	1	0	0	Errors
1	0	1	1	1	0	1	0	0	1	1	0	Received codeword

1	0	1	0	1	1	0	1	0	1	0	1	Codeword (enc. word)
1	0	0	0	0	0	0	0	0	0	0	0	Errors
0	0	1	0	1	1	0	1	0	1	0	1	Received codeword

See below, (5/5) Select Unit - The rows with a w(S) = 1 are correctly decoded permutations. Decoded codeword

Number of errors = 4

0 0 = FALSE; 1 = TRUE

Permutation indexes

			Information Bits												Parity Bits											
			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
π_1	I		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
π_2	δ		0	1	2	12	13	14	3	4	5	15	16	17	18	19	20	6	7	8	21	22	23	9	10	11
π_3	δ^2		0	1	2	18	19	20	12	13	14	6	7	8	21	22	23	3	4	5	9	10	11	15	16	17
π_4	δ^3		0	1	2	21	22	23	18	19	20	3	4	5	9	10	11	12	13	14	15	16	17	6	7	8
π_5	δ^4		0	1	2	9	10	11	21	22	23	12	13	14	15	16	17	18	19	20	6	7	8	3	4	5
π_6	δ^5		0	1	2	15	16	17	9	10	11	18	19	20	6	7	8	21	22	23	3	4	5	12	13	14
π_7	δ^6		0	1	2	6	7	8	15	16	17	21	22	23	3	4	5	9	10	11	12	13	14	18	19	20
π_8	τ		12	13	14	15	16	17	18	19	20	21	22	23	0	1	2	3	4	5	6	7	8	9	10	11
π_9	$\tau\delta$		18	19	20	6	7	8	21	22	23	9	10	11	0	1	2	12	13	14	3	4	5	15	16	17
π_{10}	$\tau\delta^2$		21	22	23	3	4	5	9	10	11	15	16	17	0	1	2	18	19	20	12	13	14	6	7	8
π_{11}	$\tau\delta^3$		9	10	11	12	13	14	15	16	17	6	7	8	0	1	2	21	22	23	18	19	20	3	4	5
π_{12}	$\tau\delta^4$		15	16	17	18	19	20	6	7	8	3	4	5	0	1	2	9	10	11	21	22	23	12	13	14
π_{13}	$\tau\delta^5$		6	7	8	21	22	23	3	4	5	12	13	14	0	1	2	15	16	17	9	10	11	18	19	20
π_{14}	$\tau\delta^6$		3	4	5	9	10	11	12	13	14	18	19	20	0	1	2	6	7	8	15	16	17	21	22	23

(1/5) Permutation Multiplexer Unit

			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
$\pi_1 r$	Ir	P_1	1	0	1	1	1	0	1	0	0	1	1	0	0	0	1	0	1	1	0	1	0	1	0	1
$\pi_2 r$	δr	P_2	1	0	1	0	0	1	1	1	0	0	1	1	0	1	0	1	0	0	1	0	1	1	1	0
$\pi_3 r$	$\delta^2 r$	P_3	1	0	1	0	1	0	0	0	1	1	0	0	1	0	1	1	1	0	1	1	0	0	1	1
$\pi_4 r$	$\delta^3 r$	P_4	1	0	1	1	0	1	0	1	0	1	1	0	1	1	0	0	0	1	0	1	1	1	0	0
$\pi_5 r$	$\delta^4 r$	P_5	1	0	1	1	1	0	1	0	1	0	0	1	0	1	1	0	1	0	1	0	0	1	1	0
$\pi_6 r$	$\delta^5 r$	P_6	1	0	1	0	1	1	1	1	0	0	1	0	1	0	0	1	0	1	1	1	0	0	0	1
$\pi_7 r$	$\delta^6 r$	P_7	1	0	1	1	0	0	0	1	1	1	0	1	1	1	0	1	1	0	0	0	1	0	1	0
$\pi_8 r$	τr	P_8	0	0	1	0	1	1	0	1	0	1	0	1	1	0	1	1	1	0	1	0	0	1	1	0
$\pi_9 r$	$\tau\delta r$	P_9	0	1	0	1	0	0	1	0	1	1	1	0	1	0	1	0	0	1	1	1	0	0	1	1
$\pi_{10} r$	$\tau\delta^2 r$	P_{10}	1	0	1	1	1	0	1	1	0	0	1	1	1	0	1	0	1	0	0	0	1	1	0	0
$\pi_{11} r$	$\tau\delta^3 r$	P_{11}	1	1	0	0	0	1	0	1	1	1	0	0	1	0	1	1	0	1	0	1	0	1	1	0
$\pi_{12} r$	$\tau\delta^4 r$	P_{12}	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	1	1	0	1	0	1	0	0	1
$\pi_{13} r$	$\tau\delta^5 r$	P_{13}	1	0	0	1	0	1	1	1	0	0	0	1	1	0	1	0	1	1	1	1	0	0	1	0
$\pi_{14} r$	$\tau\delta^6 r$	P_{14}	1	1	0	1	1	0	0	0	1	0	1	0	1	0	1	1	0	0	0	1	1	1	0	1

(2/5) Syndrome Calculation Unit

$S[0:11] = P[0:23] \cdot H^T$

			0	1	2	3	4	5	6	7	8	9	10	11
S_1			0	1	1	1	1	1	0	1	1	0	1	1
S_2			1	1	1	0	0	0	0	0	1	0	1	0
S_3			1	1	1	1	1	1	1	1	1	0	1	1
S_4			1	0	0	1	0	0	0	0	0	1	0	0
S_5			1	1	0	0	1	0	0	1	0	0	1	0
S_6			0	1	0	0	0	1	1	0	0	0	1	1
S_7			1	1	1	0	0	0	1	0	1	1	1	0
S_8			1	0	0	1	1	1	0	1	0	1	0	1
S_9			1	1	0	1	1	0	1	1	0	0	1	0
S_{10}			1	1	0	0	0	1	0	0	0	0	1	1
S_{11}			0	1	0	1	0	1	0	0	0	0	1	1
S_{12}			1	1	0	1	0	1	1	0	0	0	1	1
S_{13}			1	0	1	0	0	1	1	0	1	1	0	1
S_{14}			0	0	1	0	0	1	1	0	1	0	0	1

(4/5) Weight Calculation Unit

w	w(S) <= e (w(S) is the weight of S and e = 3)
9	0
5	0
11	0
3	1
5	0
5	0
7	0
7	0
7	0
5	0
5	0
7	0
7	0
5	0

(3/5) Look-ahead Correction Unit

$P'_i[0:23] = P_i[0:23] \oplus 0 \cdots 0S_i$ where S_i (the syndrome) is the vector of errors that were moved to the parity section of the codeword.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
P'_1	1	0	1	1	1	0	1	0	0	1	1	0	0	1	0	1	0	0	0	0	1	1	1	0
P'_2	1	0	1	0	0	1	1	1	0	0	1	1	1	0	1	1	0	0	1	0	0	1	0	0
P'_3	1	0	1	0	1	0	0	0	1	1	0	0	0	1	0	0	0	1	0	0	1	0	0	0
P'_4	1	0	1	1	0	1	0	1	0	1	1	0	0	1	0	1	0	1	0	1	1	0	0	0
P'_5	1	0	1	1	1	0	1	0	1	0	0	1	1	0	1	0	0	0	1	1	0	1	0	0
P'_6	1	0	1	0	1	1	1	1	0	0	1	0	1	1	0	1	0	0	0	1	0	0	1	0
P'_7	1	0	1	1	0	0	0	1	1	1	0	1	0	0	1	1	1	0	1	0	0	1	0	0
P'_8	0	0	1	0	1	1	0	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	1	1
P'_9	0	1	0	1	0	0	1	0	1	1	1	0	0	1	1	1	1	1	0	0	0	0	0	1
P'_{10}	1	0	1	1	1	0	1	1	0	0	1	1	0	1	1	0	1	1	0	0	1	1	1	1
P'_{11}	1	1	0	0	0	1	0	1	1	1	0	0	1	1	1	0	0	0	0	1	0	1	0	1
P'_{12}	0	1	1	0	1	0	1	0	0	1	1	0	0	1	1	0	1	1	0	0	1	0	1	0
P'_{13}	1	0	0	1	0	1	1	1	0	0	0	1	0	0	0	0	1	0	0	1	1	1	1	1
P'_{14}	1	1	0	1	1	0	0	0	1	0	1	0	1	0	0	1	0	1	1	1	0	1	0	0

(5/5) Select Unit - The rows with a w(S) = 1 are correctly decoded permutations.

$y_i = \pi_i^{-1}(P'_i)$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	w(S) ? e (w(S)
y_1	1	0	1	1	1	0	1	0	0	1	1	0	0	1	0	1	0	0	0	0	1	1	1	0	0
y_2	1	0	1	1	1	0	1	0	0	1	0	0	0	0	1	0	1	1	1	0	1	1	0	0	0
y_3	1	0	1	0	0	1	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0
y_4	1	0	1	1	1	0	0	0	0	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	1
y_5	1	0	1	1	0	0	1	1	0	1	1	0	0	0	1	1	0	1	0	0	0	1	0	1	0
y_6	1	0	1	0	1	0	1	1	0	1	1	0	0	1	0	0	1	1	0	1	0	1	0	0	0
y_7	1	0	1	0	0	1	1	0	0	1	1	0	1	0	0	0	1	1	1	0	0	1	0	1	0
y_8	0	0	1	0	0	1	1	1	0	0	1	1	0	0	1	0	1	1	0	1	0	1	0	1	0
y_9	0	1	1	0	0	0	1	0	0	1	1	0	1	1	1	0	0	1	0	1	0	1	0	1	0
y_{10}	0	1	1	1	1	0	1	1	1	1	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0
y_{11}	1	1	1	1	0	1	1	0	0	1	1	0	0	0	1	0	1	1	0	1	0	0	0	0	0
y_{12}	0	1	1	1	1	0	1	0	0	0	1	1	0	1	0	0	1	1	0	1	0	0	0	1	0
y_{13}	0	0	0	1	1	0	1	0	0	0	1	1	0	0	1	0	1	0	1	1	1	1	0	1	0
y_{14}	1	0	0	1	1	0	1	0	1	1	1	0	0	0	1	1	1	0	0	1	0	1	0	0	0

G

	0	0	0	0	0	0	0	0	0	0	0										0	0	1
0		0	0	0	0	0	0	0	0	0	0										0	1	1
0	0		0	0	0	0	0	0	0	0	0										1	1	1
0	0	0		0	0	0	0	0	0	0	0							0	0	1			
0	0	0	0		0	0	0	0	0	0	0							0	1	1			
0	0	0	0	0		0	0	0	0	0	0							1	1	1			
0	0	0	0	0	0		0	0	0	0	0				0	0	1						
0	0	0	0	0	0	0		0	0	0	0				0	1	1						
0	0	0	0	0	0	0	0		0	0	0				1	1	1						
0	0	0	0	0	0	0	0	0		0	0	0	0	1									
0	0	0	0	0	0	0	0	0	0		0	0	1	1									
0	0	0	0	0	0	0	0	0	0	0		1	1	1									

H^T

									0	0	1
									0	1	1
									1	1	1
						0	0	1			
						0	1	1			
						1	1	1			
			0	0	1						
			0	1	1						
			1	1	1						
0	0	1									
0	1	1									
1	1	1									
	0	0	0	0	0	0	0	0	0	0	0
0		0	0	0	0	0	0	0	0	0	0
0	0		0	0	0	0	0	0	0	0	0
0	0	0		0	0	0	0	0	0	0	0
0	0	0	0		0	0	0	0	0	0	0
0	0	0	0	0		0	0	0	0	0	0
0	0	0	0	0	0		0	0	0	0	0
0	0	0	0	0	0	0		0	0	0	0
0	0	0	0	0	0	0	0		0	0	0
0	0	0	0	0	0	0	0	0		0	0
0	0	0	0	0	0	0	0	0	0		0
0	0	0	0	0	0	0	0	0	0	0	