Low-Complexity Erasure Decoding of Staircase Codes

by

William Stewart Clelland

A thesis
submitted to the University of Ottawa
in partial fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University's regulations concerning plagiarism, including those concerning consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

This thesis presents a new low complexity erasure decoder for staircase codes in optical interconnects between data centers. We developed a parallel software simulation environment to measure the performance of the erasure decoding techniques at output error rates relevant to an optical link. Low complexity erasure decoding demonstrated a 0.06dB increase in coding gain when compared to bounded distance decoding at an output error rate of 3×10^{-12}. Further, a log-linear extrapolation predicts a gain of 0.09dB at 10^{-15}. This performance improvement is achieved without an increase in the maximum number of decoding iteration and keeping power constant. In addition, we found the optimal position within the decoding window to apply erasure decoding to minimize iteration count and output error rates, as well as the erasure threshold that minimizes the iteration count subject to the constrained erasure decoding structure.

Acknowledgments

I would like to thank my research supervisor, Dr. Claude D'Amours, for his guidance, encouragement, and technical expertise. I also thank my wife, Tara, for her ongoing patience and support throughout this process.

Table of Contents

List of Tables

List of Figures

Glossary

Symbol	Definition
α	Primitive element of field
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BCJR	Bahl, Cocke, Jelinek and Raviv
CFEC	Concatenated Forward Error Correction
DCI	Data Center Interconnect
DMC	Discrete Memoryless Channel
FEC	Forward Error Correction
FPGA	Field Programmable Gate Array
GF	Galois Field
K	Message length
ML	Maximum Likelihood
MAP	Maximum A Posteriori
N	Codeword length
OIF	Optical Internetworking Forum
SNR	Signal to Noise Ratio
VLSI	Very Large Scale Integration

Chapter 1: Introduction

The progress of optical communications systems has long been driven by the pursuit of higher data rates while also reducing costs. This dynamic has driven an immense amount of innovation, pushing the boundaries of material science, VLSI and communication theory. Historically, the improvements in transceiver performance occurred primarily in the long haul optical space. In these applications, the cost of the fiber plant dominates that of the transceivers, even while the cost per bit has declined dramatically in the last 20 years[36]. This pushed the industry to develop transceivers with high spectral efficiency, allowing increased capacity in a pre-existing fiber deployment.

While long haul systems continue their approach to the Shannon limit, the demand for connectivity between data centers has grown dramatically in the past decade. The market for optical Data Center Interconnects (DCI) has increased from $1.69B in 2016 to $5.08B in 2020[33], with an expected market of $17B by 2026[34].

The nature of the DCI market forces transceiver manufacturers to re-evaluate architectural trade-offs when compared to long haul. The relative proximity of data centers (120km) makes the fiber plant no longer a dominant cost. Instead, there is an emphasis on the transceiver unit cost and the associated operating expenses. The expenses are primarily driven by the cost of electricity to power and cool the transceiver. Even if we ignore the costs of operation, the density of transceiver deployments can pose fundamental challenges to thermal management. In current generation deployments, a 2U switch may house up to 32 transceivers operation at 400Gbps—each drawing 15W for an aggregate throughput of 12.8Tbps at 480W(transceivers only)[38]. While next generation systems designed for 800Gbps may increase the transceiver power budgets to 25W[20], it is clear that power per

bit must continue to improve.

Since progressing beyond 100Gbps per wavelength, data center interconnects are almost exclusively the domain of coherent optics, as dispersion penalties at 120km make intensity modulated schemes impractical[21]. In recent years, the industry has moved to adopt the Zero-Reach (ZR) series of OIF standards for coherent optical interfaces, which is targeted at DCI applications. In 2019, 400ZR was approved and, subsequently, saw broad market adoption from major cloud providers such as Microsoft[44]. The standard is designed to allow compliant solutions to use low cost optics and low power DSP, driving down both capital and operating expenses.

During the development of the 400ZR standard, the need to reduce the DSP power consumption drove the OIF committee to adopt a novel concatenated FEC (CFEC) consisting of an outer staircase code and an inner Hamming code[3]. The concept of staircase codes was first introduced in 2012 by Smith *et al.*[48] and was adopted as the line side FEC for OTU4[2]. The architecture of the staircase code admits an efficient decoder implementation with competitive performance, an important consideration when the forward error correction circuit consumes approximately 10% of total power in a transceiver[41].

In less power constrained applications, a common method of improving the performance of a FEC decoder is to use more sophisticated decoding algorithms, such as soft decision algorithms. A soft decoder uses reliability estimates of the received symbols in conjunction with extrinsic information to increase decoding performance by 2dB-2.5dB for realistic values of SNR[42]. However, a soft decoder requires far more power than a hard decision decoder due to an increase in the computational complexity, making it impractical for low power applications such as DCI.

In the pursuit of hardware efficiency, this thesis presents a low complexity erasure de-

coder for staircase codes that improves decoder performance with little to no increase in complexity. This approach is modeled and validated in an optimized parallel software simulation, enabling reliable measurement at bit error rates relevant to optical communication systems analysis.

First, this thesis begins with an overview of BCH code theory in Chapter 2, along with a set of efficient decoding routines. BCH codes are widely used as component codes for staircase and other similar braided structures (such as OFEC[49]) and a majority of the energy used by a staircase decoder is used in their decoding.

Second, in Chapter 3, we present the theory of erasure decoding and common approaches used to implement the technique. We also discuss theoretical optimal erasure thresholds and how they effect decoder performance.

In Chapter 4, we motivate the development of staircase codes by reviewing the ideas and FEC techniques that led to their development. The staircase code structure uses techniques borrowed from block codes, convolutional codes and product codes. We discuss the limitations of those prior techniques and how they lead to staircase codes.

Chapter 5 reviews the architecture of the specific decoder used for this thesis. We discuss common techniques to increase the efficiency of the software implementation along with improving performance.

Chapter 6 discusses the results of our experiment and the key conclusions.

Chapter 2: BCH Codes

Bose–Chaudhuri–Hocquenghem (BCH) codes are a popular class of cyclic code which were independently discovered by both Alexis Hocquenghem[28] in 1959 and Bose and Ray-Chaudhuri[9] in 1960. In the subsequent 60 years, these codes have become ubiquitous in communication systems such as optical[3], deep-space[52], wireline[29], and storage[1].

A BCH code is commonly specified using a generator polynomial with coefficients from a base field GF(q). Consider a BCH codeword v of length N containing a message sequence of length K. By definition, this sequence is a valid codeword if its polynomial representation $v(x)$ of order $N-1$ has the generator polynomial $g(x)$ of order $N-K$ as a factor.

There are several families of BCH codes that can be used depending on system requirements. If a communication system is prone to small bursts of errors, we may select a non-binary base field q such that a burst is likely to be smaller than q. A popular non-binary BCH code is Reed-Solomon. We also have the ability to select a non-primitive BCH code such that $N \neq q^m - 1$, which allows BCH codes to fit various higher order protocols without the use of techniques such as shortening. The BCH codes we discuss past this point are binary (i.e. $q = 2$) and primitive (i.e. $N = q^m - 1$).

The generator polynomial is the lowest order polynomial which has $2t$ consecutive powers of the primitive element α^j as root where $j \in [1, 2t-1]$ and t is the number of correctable errors. This definition is equivalent to the generator polynomial being the least common multiple of the minimal polynomials $\phi_j(x)$ of elements α^j. As the code $v(x)$ is divisible by $g(x)$, it must also have elements α^j as roots. The process of decoding a received codeword $r(x)$ involves finding the "closest" polynomial $\hat{v}(x)$ that also has the set of elements $\{\alpha^j\}$ as roots. Please note that a full exposition of BCH codes is beyond

the scope of this thesis and the reader may wish to consult a reference text on the topic [32][51][40].

BCH codes provide many advantages to engineers when designing an error control system. The primary advantages are the ease of codeword design and options for hardware efficient decoding strategies. While the design criteria for a given BCH code may be motivated by a number of factors, we are most commonly concerned with the rate of information transmission R, the amount of resources required for decoding, and the error correction capability t per codeword. These parameters are not orthogonal; rather, they are coupled by the following relationships in a standard BCH code:

$$\begin{aligned} t &\leq \frac{N-K}{m} \\ R &= K/N \end{aligned} \tag{2.1}$$

With an increase in the order of the extension field $GF(2^m)$, the rate R increases for a given error correcting capability t. However, the probability of a received codeword containing more than t errors also increases according to the Binomial distribution, if we assume channel errors are independent and identically distributed (IID). In Figure 2.1, we show an example of the upper bound on the probability of a decoding failure for double error correcting BCH codes subject to a binary symmetric channel (BSC) with a 1% transition probability.

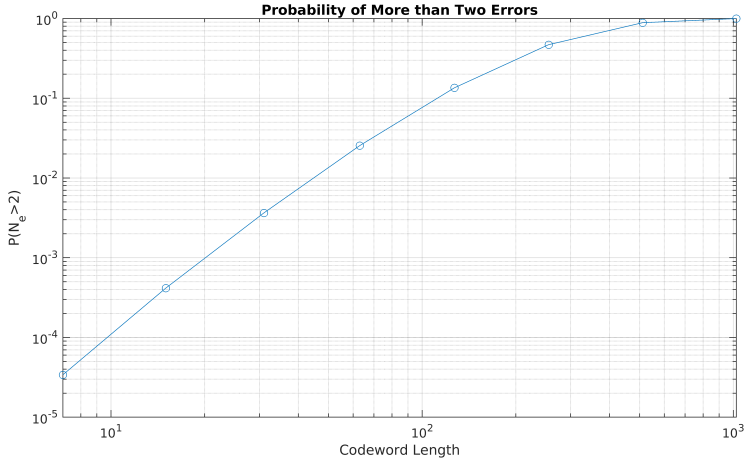

Figure 2.1: Probability of Two or More Errors

For a bounded distance decoder and a binary symmetric channel, we can examine the trade-offs involved when choosing the length of a code using the coding gain. This metric measures the reduction in the required signal-to-noise ratio (SNR) of a coded channel when compared to an uncoded channel for a given bit error rate.

$$\gamma_c = \frac{(E_b/N_0)_{uncoded}}{(E_b/N_0)_{coded}} \tag{2.2}$$

Once the codeword designer has chosen a basic BCH structure, they can further modify the code to better suit the application space. The most popular modifications are shortening and extending the block code.

Shortening reduces the number of message symbols while maintaining the same number of parity symbols. This modification retains the same minimum distance as the original code, while potentially reducing the number of valid codewords at the minimum distance. Shortening is commonly used to modify the length of a codeword to fit the framing requirements of a higher layer protocol such as in OTU4[2].

Extending a code involves adding additional parity equations to the codeword, reducing the number of valid codewords at the minimum distance. For example, we might extend a code by modifying the generator polynomial such that $g'(x) = (x+1)g(x)$. This modification adds an embedded parity bit and roughly halves the number of valid codewords at d_{min}, reducing the probability of a false correction. A codeword extension is particularly useful when selecting candidate codewords during a soft decoding procedure. The technique may also be used in combination with shortening to fine tune the rate, block length and performance when designing a full communication system.

2.1 BCH Encoding

Encoding a binary BCH codeword is the process of uniquely mapping a message m with K bits onto a set of N bits. This mapping may result in either a systematic or non-systematic code. In a systematic code, the original message is found directly in the first (or last) K bits of the code with the remaining $N-K$ bits used as parity bits. In a non-systematic code, the BCH encoded sequence does not contain the initial message. To reduce the number of errors in an incorrectly decoded message sequence, it is common practice for block codes to use a systematic mapping. With the message embedded in the codeword, a decoding failure would still leave a large majority of the message intact as the receiver output.

We will work through an example in which we encode a BCH(15,7) code capable of correcting all combinations of two or fewer errors (i.e. $t = 2$). For the primitive polynomial

$p(x) = 1 + x + x^4$, we find the following minimal polynomials for α and α^3:

$$\begin{aligned} \phi_1(x) &= 1 + x + x^4 \\ \phi_3(x) &= 1 + x + x^2 + x^3 + x^4 \end{aligned} \tag{2.3}$$

As previously discussed, the generator polynomial will be the least common multiple of the two minimal polynomials:

$$\begin{aligned} g(x) &= \text{LCM}\{\phi_1(x), \phi_3(x)\} \\ &= \phi_1(x)\phi_3(x) \\ &= (1 + x + x^4)(1 + x + x^2 + x^3 + x^4) \\ &= 1 + x^4 + x^6 + x^7 + x^8 \end{aligned} \tag{2.4}$$

To generate a valid non-systematic codeword $v(x)$, we simply multiply the message polynomial $m(x)$ by the generator polynomial $g(x)$. It is trivial to show that the resulting codeword is divisible by $g(x)$. For a systematic encoding, we must find a polynomial $b(x)$ such that $v(x) = m(x)x^{N-K} + b(x)$ is divisible by $g(x)$. The parity polynomial $b(x) = m(x)x^{N-K} \mod g(x)$ can be found using Euclidean division implemented using a shift-register.

Suppose we encode the following message sequence:

$$\begin{aligned} m &= [1100101] \\ m(x) &= 1 + X + X^4 + X^6 \end{aligned} \tag{2.5}$$

Using a non-systematic encoding:

$$\begin{aligned} v_{ns}(x) &= m(x)g(x) \\ &= (1 + X + X^4 + X^6)(1 + x^4 + x^6 + x^7 + x^8) \\ &= 1 + x + x^5 + x^9 + x^{11} + x^{13} + x^{14} \\ v &= [1100010001010110] \end{aligned} \tag{2.6}$$

Using a systematic encoding:

$$\begin{aligned} b(x) &= m(x)x^{N-K} \mod g(x) \\ &= x^4 + x^5 + x^6 \\ v_s(x) &= m(x)x^{N-K} + b(x) \\ &= (x^8 + x^9 + x^{12} + x^{14}) + (x^4 + x^5 + x^6) \\ &= x^4 + x^5 + x^6 + x^8 + x^9 + x^{12} + x^{14} \\ v &= [00001110, 1100101] \end{aligned} \tag{2.7}$$

There is an equivalent approach expressed using matrix algebra, which is more often employed in practice due to the explicit parallel structure which more closely resembles the resulting *xor* tree. Instead of using $g(x)$ directly, we create a Toeplitz convolution matrix $\mathbf{G}$, the systematic equivalent of which is a reduced row echelon form $\mathbf{G}'$.

2.2 BCH Decoding

The determination of error locations in a given codeword is a major factor in the decoding complexity of hard decision algebraic codes. There are several methods that can be used, the suitability of which depends on the order of the field N and the error correcting capability t. We will limit our discussion to binary BCH codes.

Syndrome Computation

In all popular decoding algorithms, we begin by computing the syndromes of the received codeword r. The calculation involves evaluating $r(x)$ at t values. The result is up to N additions in GF(2^m) for each of the t syndromes.

$$S_i = r(\alpha^i) \tag{2.8}$$

Algebraic Decoding

The Abel-Ruffini Theorem[4] states that no closed form solution exists for general polynomials of order greater than four. As a result, algebraic solutions for BCH codes are only possible for t smaller than 5. For these codes, we have efficient algorithms to compute the roots of an error locator polynomial[17][39][18][25].

The syndromes of the received codeword are used to determine the appropriate order for the error locator polynomial $\Lambda(x)$ of order ν. This is generally done using Peterson's direct-solution but can be simplified in specific cases.

Suppose a received codeword $\boldsymbol{r}$ has ν errors located at indices $j_1, j_2, \ldots, j_\nu$, the polynomial representing the received codeword will be of the form $r(x) = c(x) + e(x)$. The error polynomial $e(x)$ is expressed as follows:

$$\begin{aligned} e(x) &= e_{j_1}x^{j_1} + e_{j_2}x^{j_2} + \cdots + e_{j_\nu}x^{j_\nu} \\ &= \sum_{k=0}^{N-1} e_k x^k \end{aligned} \tag{2.9}$$

We compute a set of syndrome values for the received codeword r where the i-th syndrome is defined as:

$$
\begin{aligned}
S_i &= r(\alpha^i) \\
&= e(\alpha^i) + c(\alpha^i) \\
&= e(\alpha^i) \\
&= (\alpha^i)^{j_1} + (\alpha^i)^{j_2} + \cdots + (\alpha^i)^{j_\nu} \\
&= \sum_{l=1}^{\nu} (\alpha^{j_l})^i
\end{aligned}
\tag{2.10}
$$

We commonly refer to the set of elements $\{\alpha^{j_l}\}$ as error locators $\{X_l\}$. The set of syndromes for a received codeword form a set of power-sum symmetric equations which are a function of the error locators:

$$
\begin{aligned}
S_1 &= \alpha^{j_1} &&+ \alpha^{j_2} &&+ \cdots &&+ \alpha^{j_\nu} \\
S_2 &= \alpha^{2j_1} &&+ \alpha^{2j_2} &&+ \cdots &&+ \alpha^{2j_\nu} \\
&\vdots \\
S_{2t} &= \alpha^{2tj_1} &&+ \alpha^{2tj_2} &&+ \cdots &&+ \alpha^{2tj_\nu} \\
\\
S_1 &= X_1 &&+ X_2 &&+ \cdots &&+ X_\nu \\
S_2 &= X_1^2 &&+ X_2^2 &&+ \cdots &&+ X_\nu^2 \\
&\vdots \\
S_{2t} &= X_1^{2t} &&+ X_2^{2t} &&+ \cdots &&+ X_\nu^{2t}
\end{aligned}
\tag{2.11}
$$

Any method of finding j_i such that (2.11) holds is a decoding algorithm for BCH codes[32]. However, since the set of nonlinear equations would be difficult to solve directly, we can simplify the problem using Newton Identities. First, let us define the error locator polynomial $\Lambda(x)$ following Peterson's derivation [39]:

$$\begin{aligned}\Lambda(x) &= \Lambda_0 + \Lambda_1 x + \cdots + \Lambda_\nu x^\nu \\ &= \prod_{l=1}^{\nu}(1 - X_l x)\end{aligned} \tag{2.12}$$

The coefficients Λ_i of the error locator polynomial form the elementary symmetric equations of $\{X_l\}$[51]:

$$\begin{aligned}\Lambda_0 &= 1 \\ \Lambda_1 &= \sum_{i=1}^{\nu} X_i \\ \Lambda_2 &= \sum_{i<j}^{\nu} X_i X_j \\ &\vdots \\ \Lambda_\nu &= \prod_{l=1}^{\nu} X_l\end{aligned} \tag{2.13}$$

The power-sum symmetric(2.11) and elementary symmetric(2.13) equations are related by Newton's Identities:

$$\begin{aligned}0 &= S_1 + \Lambda_1 \\ 0 &= S_3 + S_2\Lambda_1 + S_1\Lambda_2 + \Lambda_3 \\ 0 &= S_5 + S_4\Lambda_1 + S_3\Lambda_2 + S_2\Lambda_3 \\ &\vdots\end{aligned} \tag{2.14}$$

We simplify by noting that $S_i^2 = S_{2i}$ in any extension field formed using a binary base:

$$\begin{aligned} 0 &= S_1 + \Lambda_1 \\ 0 &= S_3 + S_1^2\Lambda_1 + S_1\Lambda_2 + \Lambda_3 \\ 0 &= S_5 + S_1^4\Lambda_1 + S_3\Lambda_2 + S_1^2\Lambda_3 \\ &\vdots \end{aligned} \tag{2.15}$$

We may select from a number of algorithms to determine the coefficients of the error locator polynomial Λ_i. The most popular methods are from either E. Berlekamp and J. Massey or W. Peterson Once the error locator polynomial is found, the roots of $\Lambda(x)$ may then be calculated by a Chien search [16].

We can compute error locations more efficiently by using ideas from Peterson's direct method[39][51] given a specific codeword. First, let us review the direct method formulated by Peterson. We may express equation (2.15) using matrix notation.

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ S_1^2 & S_1 & 1 & \cdots & 0 \\ S_1^4 & S_3 & S_1^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ S_{2t-2} & S_{2t-3} & S_{2t-4} & \cdots & S_{t-1} \end{bmatrix} \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \\ \vdots \\ \Lambda_t \end{bmatrix} = \begin{bmatrix} -S_1 \\ -S_3 \\ -S_5 \\ \vdots \\ -S_{2t-1} \end{bmatrix} \tag{2.16}$$

Peterson showed that given a binary code, the above syndrome matrix $\boldsymbol{S}$ is non-singular if a codeword contained t or $t-1$ errors[39]. To decode, we reduce the size of the above matrix until it becomes non-singular and then solve the linear system. Solving (2.16) is equivalent to solving (2.15).

If our interest lies in a particular binary code, we can simplify the decoding procedure. Let our code of interest be a triple error correcting code (as in the G709.2 Staircase code[2]),

where the coefficients of the error locator polynomial Λ_i have a closed form solution after reducing 2.15 to three terms:

$$\begin{aligned}\Lambda_1 &= S_1 \\ \Lambda_2 &= \frac{S_1^2 S_3 + S_5}{S_1^3 + S_5} \\ \Lambda_3 &= S_1^3 + S_3 + S_1 \Lambda_2\end{aligned} \tag{2.17}$$

We define two more quantities which arise naturally when working with equation 2.15:

$$\begin{aligned}D_3 &:= S_1^3 + S_3 \\ D_5 &:= S_1^5 + S_5\end{aligned} \tag{2.18}$$

No Error

A codeword is valid if all the syndromes are zero.

Single Error

In the event of single codeword error at location j_1, we observe the following:

$$\begin{aligned}\Lambda_1 &= S_1 \\ &= \alpha^{j_1} \\ &\neq 0\end{aligned} \tag{2.19}$$

$$\begin{aligned}\Lambda_2 &= \frac{S_1^2 S_3 + S_5}{S_1^3 + S_5} \\ &= \frac{\alpha^{5j_1} + \alpha^{5j_1}}{\alpha^{3j_1} + \alpha^{5j_1}} \\ &= 0\end{aligned} \tag{2.20}$$

$$\begin{aligned}\Lambda_3 &= S_1^3 + S_3 + S_1\Lambda_2 \\ &= \alpha^{3j_1} + \alpha^{3j_1} + S_1 \cdot 0 \\ &= 0\end{aligned} \tag{2.21}$$

The received codeword has a single correctable error if $\Lambda_1 \neq 0$ while $\Lambda_2 = \Lambda_3 = 0$. These conditions reduce 2.15 to the following system of equations:

$$\begin{aligned}0 = S_3 + S_1^3 \quad &= D_3 \\ 0 = S_5 + S_1^5 \quad &= D_5\end{aligned} \tag{2.22}$$

$$\begin{aligned}\Lambda(x) &= 1 + \Lambda_1 x \\ &= 1 + S_1 x \\ 0 &= 1 + \alpha^{j_1} x \\ \Rightarrow j_1 &= 1/S_1 \\ &= -\log S_1\end{aligned} \tag{2.23}$$

In summary, the error locator polynomial $\Lambda(x)$ has root $1/S_1$ which implies the error location is $j_1 = -\log S_1$.

Double Error

If the received codeword contains two correctable errors, then $\Lambda(x)$ is a second order polynomial with $\Lambda_3 = 0$ along with the following constraints derived from the requirement that $|\boldsymbol{S}_2|$ and $|\boldsymbol{S}_3|$ are nonzero while $|\boldsymbol{S}_4| = 0$:

$$\begin{vmatrix} 1 & 0 \\ S_1^2 & S_1 \end{vmatrix} \neq 0 \tag{2.24}$$
$$\Rightarrow S_1 \neq 0$$

$$\begin{vmatrix} 1 & 0 & 0 \\ S_1^2 & S_1 & 1 \\ S_1^4 & S_3 & S_1^2 \end{vmatrix} \neq 0 \tag{2.25}$$
$$S_1^3 + S_3 \neq 0$$
$$\Rightarrow D_3 \neq 0$$

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ S_1^2 & S_1 & 1 & 0 \\ S_1^4 & S_3 & S_1^2 & S_1 \\ S_1^6 & S_5 & S_1^4 & S_3 \end{vmatrix} = 0 \tag{2.26}$$
$$S_1^6 + S_1^3 S_3 + S_5 S_1 + S_3^2 = 0$$
$$S_1(S_1^5 + S_5) + S_3(S_1^3 + S_3) = 0$$
$$\Rightarrow S_1 D_5 = S_3 D_3$$

To summarize Eq. 2.24 - 2.26, the following must be true for a received codeword to be at a Hamming distance of 2 from a valid codeword:

$$S_1 \neq 0 \qquad D_3 \neq 0 \qquad S_1 D_5 = S_3 D_3 \tag{2.27}$$

The error locator polynomial is then formulated using the coefficients found in 2.17 and substituted into 2.12:

$$\Lambda(x) = 1 + S_1 x + \frac{D_3}{S_1} x^2 \tag{2.28}$$

With the substitution $y := x/S_1$, we obtain a suppressed quadratic $\Lambda(y)$ with roots related to those of $\Lambda(x)$ by a factor of S_1:

$$\Lambda(y) = S_1^2 (\frac{D_3}{S_1^3} + y + y^2) \tag{2.29}$$

Given we found roots $\{r_1, r_2\}$ of $\Lambda(y)$, we note that the two roots are related such that $r_2 = r_1 + 1$:

$$\begin{aligned} 0 &= \frac{D_3}{S_1^3} + r_1 + r_1^2 \\ &= \frac{D_3}{S_1^3} + r_1(1 + r_1) \end{aligned} \tag{2.30}$$

The two roots $\{r_1, r_2\}$ of quadratic $\Lambda(y)$ may be computed ahead of time using a method such as Chien search, or any number of computer algebra systems. We store r_1 in a lookup table with 2^m entries of m bits, the value of r_2 is computed at runtime by XOR-ing the lowest bit of r_1[17][8][48].

$$\begin{aligned} \Lambda(y) &:= (1 - \alpha^{j_1} y)(1 - \alpha^{j_2} y) \\ \Rightarrow \Lambda(x) &= (1 - \alpha^{j_1} S_1 x)(1 - \alpha^{j_2} S_1 x) \end{aligned} \tag{2.31}$$

Therefore, the error locations $\{j_1, j_2\}$ are:

$$\begin{aligned} j_1 &= \log S_1 r_1 \\ j_2 &= \log S_1 (r_1 + 1) \end{aligned} \tag{2.32}$$

Triple Error

A received code is treated as though it has three detected errors if all coefficients of the third order error locator polynomial are non-zero. To efficiently determine if a received code meets the criteria, we first check that its syndromes do not satisfy the requirements of single or double error correction. Next, we check that the syndrome matrix is non-singular, while ensuring that Λ_2 is finite (i.e. $D_3 \neq 0$), before attempting to decode.

$$\begin{aligned}
\begin{vmatrix} 1 & 0 & 0 & 0 \\ S_1^2 & S_1 & 1 & 0 \\ S_1^4 & S_3 & S_1^2 & S_1 \\ S_1^6 & S_5 & S_1^4 & S_3 \end{vmatrix} &\neq 0 \\
S_1^6 + S_1^3 S_3 + S_5 S_1 + S_3^2 &\neq 0 \\
S_1(S_1^5 + S_5) + S_3(S_1^3 + S_3) &\neq 0 \\
\Rightarrow S_1 D_5 &\neq S_3 D_3
\end{aligned} \tag{2.33}$$

$$\begin{aligned}
\Lambda_1 &= S_1 \\
\Lambda_2 &= \frac{S_1^2 S_3 + S_5}{S_1^3 + S_5} \\
\Lambda_3 &= S_1^3 + S_3 + S_1 \Lambda_2
\end{aligned} \tag{2.34}$$

We will reference the result found in the Appendix of Smith et al. [48], where they derive simplified forms for the error locator polynomial depending on the value of D_5.

$$\begin{aligned}
\Lambda(y) &= D_3 + y^3 && \Leftrightarrow D_5 = 0 \\
\Lambda(y) &= \left(\frac{D_3^5}{D_5^3}\right)^{1/2} + y + y^3 && \Leftrightarrow D_5 \neq 0
\end{aligned} \tag{2.35}$$

The resulting roots are always of the form $\{r_1, r_2, r_1 + r_2\}$. In the first case where $D_5 = 0$, we find the roots of $f_Y(y)$ where $y = x + S_1$.

$$\begin{aligned} \Lambda(y) &= D_3 + y^3 \\ 0 &= D_3 + y^3 \\ \Rightarrow y &= \sqrt[3]{D_3} \end{aligned} \tag{2.36}$$

In the second case where $D_5 \neq 0$, we perform a further substitution of $y = \sqrt{D_5/D_3}z$.

$$\Lambda(z) = \left(\frac{D_5}{D_3}\right)^{3/2}\left(z^3 + z + \sqrt{\frac{D_3^5}{D_5^3}}\right) \tag{2.37}$$

To solve for the roots of the original error locator polynomial, we first find the roots r_z of $\Lambda(z)$. The roots $r_y = r_z\sqrt{D_5/D_3}$ while $r = r_y + S_1$.

Note that the roots (if they exist) of this class of polynomial are of the form $\{r_1, r_2, r_1 + r_2\}$. A Computer Algebra System (CAS) is used to generate two lookup tables of length 2^m with $2m$ bits per entry.

2.3 Decoding Complexity Comparison

To evaluate the merits of the algebraic decoding technique described above, it is compared with a traditional approach using the binary Berlekamp-Massey algorithm to find $\Lambda(x)$ followed by a Chien search to compute roots $\{\alpha^{j_i}\}$. The section will evaluate the complexity in terms of hardware units and ignore latency. The latency of either implementation will depend heavily on parallelism in the hardware design. In an iterative algorithm such as Chien search, a pure sequential implementation would cause a worst case latency penalty

proportional to the size of the field while a parallel implementation has a fixed latency penalty.

The practical complexity of correcting a codeword depends heavily on nature of the channel error statistics. In a channel with high error rates, we expect the decoding heat to be dominated by triple error correcting circuit of a BCH3 code — the complexity of which will vary significantly across implementations. For the staircase decoding structure presented later in this manuscript, the decoder usage near threshold is noted in Table 2.1. The data is the result of measurements performed in simulation at an input error rate of 4.65×10^{-3} and BCH(1022,990) code.

$\hat{t}$	**Usage**
1	18.6%
2	14.1%
3	67.2%

Table 2.1: BCH Decoder Usage for Representative Workload

Algebraic Decoder

Using the equations derived in Section 2.2, we compute the average number of operations required to decode a codeword given a known set of syndromes.

Decoder	**Add**	**Multiply**	**Division**	$\log \mathbf{x}$	$\sqrt{\mathbf{x}}$	$\alpha^{\mathbf{n}}$	**LUT**
BCH1	2	0	0	1	0	2	0
BCH2	2	2	1	2	0	3	1
BCH3	6	3	2	3	2	4	1
Average	4.7	2.3	1.5	2.5	1.3	3.5	0.8

Table 2.2: Algebraic Complexity for Representative Workload

Berlekamp-Massey Complexity

The most common method of finding the error locating polynomial $\Lambda(x)$ is an iterative algorithm developed by Berlekamp and Massey[7][35]. Unlike the algebraic approach previously described, The procedure can be applied to an arbitrary BCH or RS code. The algorithm proceeds by refining our estimate of $\Lambda(x)$ by checking if the μ-th Newton identity holds. If the identity evaluates to non-zero, the result is termed a discrepancy d_μ and is used to adjust $\Lambda(x)$. For a binary BCH code, we require $\lceil (t+\hat{t})/2 \rceil$ iterations to determine the final value of $\Lambda(x)$[51].

This complexity of this algorithm is often eclipsed by the root finding routine. Analyzing the cost of the Berlekamp-Massey algorithm is rather involved due in part to the length $\Lambda(x)$ varying throughout the procedure. We find that the routine requires at most $2\lceil n^2/4 \rceil - n + 1$ multiplications and additions to compute $\Lambda(x)$ with n coefficients[27][37]. We make the approximation that all i-th order error locators contain $i+1$ terms.

Decoder	Add	Multiply
BCH1	1	1
BCH2	4	4
BCH3	5	5
Average	2.8	2.8

Table 2.3: Berlekamp-Massey Complexity for Representative Workload

Chien Search Complexity

A Chien search is an iterative root finding algorithm for finite field polynomials using the primitive α[16][40]. The algorithm performs an exhaustive search of all elements of the field while exploiting the relative efficiency of a constant coefficient multiplier and the cyclic nature of the BCH code. It is important to note that a constant coefficient multiplier can be up to 40% less expensive than the usual implementation[10]. A sequential circuit depicting a Chien search engine is shown in Fig. 2.2 where each iteration requires we perform $\hat{t}$ additions and $\hat{t}$ constant multiplications.

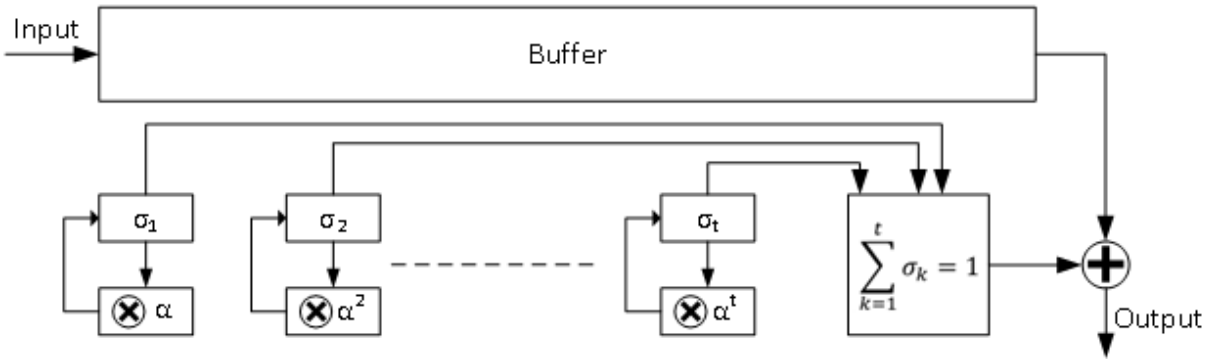

Figure 2.2: Sequential Chien Search Hardware Implementation[16]

The amount of work performed in a Chien search is not only governed by the hardware complexity of an iteration, but also by the number of iterations expected given the order of $\Lambda(x)$ is $\hat{t}$. The probability distribution which describes the number of iterations used to locate $\hat{t}$ roots is known as the $\hat{t}$-th order statistics. Let the order of roots be uniformly distributed according to $Y_1, Y_2, \ldots, Y_{\hat{t}} \sim U\{1, N\}$ such that $y_1 \neq y_2 \neq \ldots \neq y_{\hat{t}}$ with corresponding roots $\{\alpha^y\}$. We forego the analysis of order statistics for mutually exclusive discrete random variables and instead approximate the distribution of roots using random variables X from an IID uniform continuous distribution. The approximation holds if we restrict the domain of analysis to codes where $\hat{t} \ll N$ rendering the probability of $Y_i = Y_j$ negligible.

Given $X_1, X_2, \ldots, X_{\hat{t}} \sim U(0, 1)$.

$$\begin{aligned} X_{(\hat{t})} &\sim \beta(\hat{t}, 1) \\ E[X_{(k)}] &= \frac{\hat{t}}{\hat{t}+1} \\ E[Y_{(k)}] &\approx N\frac{\hat{t}}{\hat{t}+1} \qquad \text{(for large N)} \end{aligned} \tag{2.38}$$

For the code used in our experiment, we find the following distribution of Chien iteration.

Decoder	**Add**	**Multiply**	**Iterations**
BCH1	1	0.6	511
BCH2	2	1.2	681
BCH3	3	1.8	767
Total	1391.3	834.8	0

Table 2.4: Chien Search Complexity for Representative Workload

Conclusion

As expected, the complexity of a traditional decoder is dominated by root finding. The Chien search requires 220 times the number of multipliers in comparison to the entire algebraic decoding algorithm when applied to the specific BCH3 component code used in this thesis.

Chapter 3: Erasure Decoding

3.1 Introduction

In a modern communication system, a receiver often produces an estimate of the reliability of demodulated symbols during symbol slicing. This estimate, often referred to as soft information, can be used by the error correction circuit to improve decoding capability using a soft decoder. In addition to soft information, the error correcting decoder also obtains the receivers estimate of the most likely transmitted bit, termed a hard-decision. When a FEC decoder ignores soft information, using only the hard decision, it is called a hard decision decoder. The choice of either soft or hard decoding is a trade-off between complexity and performance. While a soft decoder improves performance, it comes at the cost of extra hardware requirements and power dissipation. In this Chapter, we discuss a third alternative called erasure decoding, a technique that lies between hard and soft decoding in complexity and performance.

3.2 Discussion

The soft information generated by the receiver may be used in the soft-decision decoding of error correcting codes. This information is commonly expressed as a Log-Likelihood Ratio (LLR)[19]

$$LLR(r_i) = \ln\left(\frac{\sum P(t_i = 1|r_i)}{\sum P(t_i = 0|r_i)}\right) \tag{3.1}$$

where t_i is the transmitted binary symbol and r_i is the associated decision variable. The exact expression for conditional probability depends on the channel statistics. In the simplest channel, the reliability of each binary symbol is independent and affected only by the channel noise $n(i) \sim \mathcal{N}(0, \sigma^2)$. The LLR expression is derived (3.2) for a sequence of equiprobable Binary Phase-Shift Keying (BPSK) symbols with levels $\{-\sqrt{\varepsilon}, \sqrt{\varepsilon}\}$ across an AWGN channel.

$$
\begin{aligned}
LLR(r_i) &= \ln \frac{\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(r_i-\sqrt{\varepsilon})^2}{2\sigma^2}\right)}{\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(r_i+\sqrt{\varepsilon})^2}{2\sigma^2}\right)} \\
&= \ln \frac{\exp\left(-\frac{(r_i-\sqrt{\varepsilon})^2}{2\sigma^2}\right)}{\exp\left(-\frac{(r_i+\sqrt{\varepsilon})^2}{2\sigma^2}\right)} \\
&= -\frac{(r_i-\sqrt{\varepsilon})^2}{2\sigma^2} + \frac{(r_i+\sqrt{\varepsilon})^2}{2\sigma^2} \\
&= \frac{4r_i\sqrt{\varepsilon}}{2\sigma^2}
\end{aligned}
\tag{3.2}
$$

With more advanced modulation formats, the calculation of an exact expression for the LLR also depends on other factors such as the symbol mapping function (in the case of higher order modulation), or the apriori visitation probability (in the case of probabilistic shaping). The reliability information contained in the demodulated symbol value enables a dramatic improvement of receiver performance. For example, Proakis[42] shows that for a given codeword, we see a soft-decoder achieve an asymptotic coding gain of 3dB relative to a hard-decision decoder—even for realistic values of SNR the coding gain is be greater than 2dB.

However, improved decoding performance of a soft decoder comes at the cost of complexity. Each bit of a codeword requires the storage of a multi-bit LLR while the decoder structure requires more algebraic decoders and consumes more power. Take Chase-2

decoding[15] as soft-in soft-out (SISO) decoder commonly used by iterative decoders for turbo product codes[43]. For each codeword, we create a set of candidate codewords where we flip the N_{lrp} least reliable bit positions, creating $2^{N_{lrp}}$ candidate codewords. Each candidate is decoded using bounded distance decoding with the resulting extrinsic information being used to update the codeword and its reliability metric. Ignoring the additional storage required by soft information, the complexity of the Chase algorithm scales exponentially with N_{lrp}[5].

In applications with high sensitivity to power consumption or receiver latency, we may not have the resources available to employ a soft decoding algorithm and, thus, make full use of the additional information of an LLR. In these situations, instead of resorting to hard-decision decoding, we may use the reliability information to find symbol locations in a received codeword where reliability is below some threshold. For those locations, we may ignore the received symbol altogether while decoding using an *erasure decoding* routine.

The textbook algorithm for erasure decoding doubles the number of bounded distance decoding operations per received codeword. The extra computational complexity has the potential to increase the correcting capabilities. While a standard decoder is limited to an error correcting capability $\lfloor (d_{min} - 1)/2 \rfloor$, an erasure decoder will successfully decode a received codeword if:

$$(2e + f) < d_{min} \tag{3.3}$$

where we correctly identified f erased locations and have another e locations with errors.

If we employ an erasure decoder, one of the design decisions will be a selection of an optimal erasure threshold. In the basic implementation of Alg. 1, any symbol location in the received codeword will be marked an erasure if its reliability metric is below a static threshold T. While simple, this approach has the drawback of creating a potentially

Algorithm 1 Erasure Decoding

procedure ErasureDecoding($\mathbf{r}$, T)
    $\mathbf{r}_0 \leftarrow \mathbf{r}$
    $\mathbf{r}_1 \leftarrow \mathbf{r}$
    for $i = 1, \ldots, N$ **do**
        if $|\mathbf{r}[i]| \leq T$ **then**
            $\mathbf{r}_0[i] \leftarrow 0$
            $\mathbf{r}_1[i] \leftarrow 1$
        end if
    end for
    $\hat{\mathbf{c}}_0 \leftarrow BDD(\mathbf{r}_0)$ ▷ Bounded Distance Decoder
    $\hat{\mathbf{c}}_1 \leftarrow BDD(\mathbf{r}_1)$
    if $d_H(\hat{\mathbf{c}}_0, \mathbf{r}) \leq d_H(\hat{\mathbf{c}}_1, \mathbf{r})$ **then**
        $\hat{\mathbf{c}} \leftarrow \hat{\mathbf{c}}_0$
    else
        $\hat{\mathbf{c}} \leftarrow \hat{\mathbf{c}}_1$
    end if
    return $\hat{\mathbf{c}}$ ▷ Closest valid codeword to $\mathbf{r}$
end procedure

unbounded number of erasures in a received codeword. Per Equation 3.3, a greater number of erasure locations reduces our ability to correct regular errors and has the potential to degrade decoder performance.

To further illustrate the trade-off in selecting an erasure threshold, suppose we receive a codeword encoded using a binary BCH(127,113) scheme. This code has a $d_{min} = 5$ and corrects two errors when using bounded distance decoding. If we set our erasure threshold T to a low value relative to the channel reliability, we might infrequently mark a single received symbol as an erasure — potentially improving the decoding capabilities to three errors. If we increase the relative threshold, we would observe erasures with a higher frequency. This increased probability of erasure will improve decoding if the error locations are correctly marked, as every erasure "costs" half the minimum distance as an error. However, increasing the threshold can also have a negative performance impact and degrade decoding by introducing extraneous erasures. These erasures create candidate received codewords (r_0 and r_1 in Alg. 1) that may have a larger minimum distance than the originally received codeword r.

If we choose to employ BPSK modulation across our AWGN communication channel, the probability density function of the received symbols resembles Fig. 3.1.

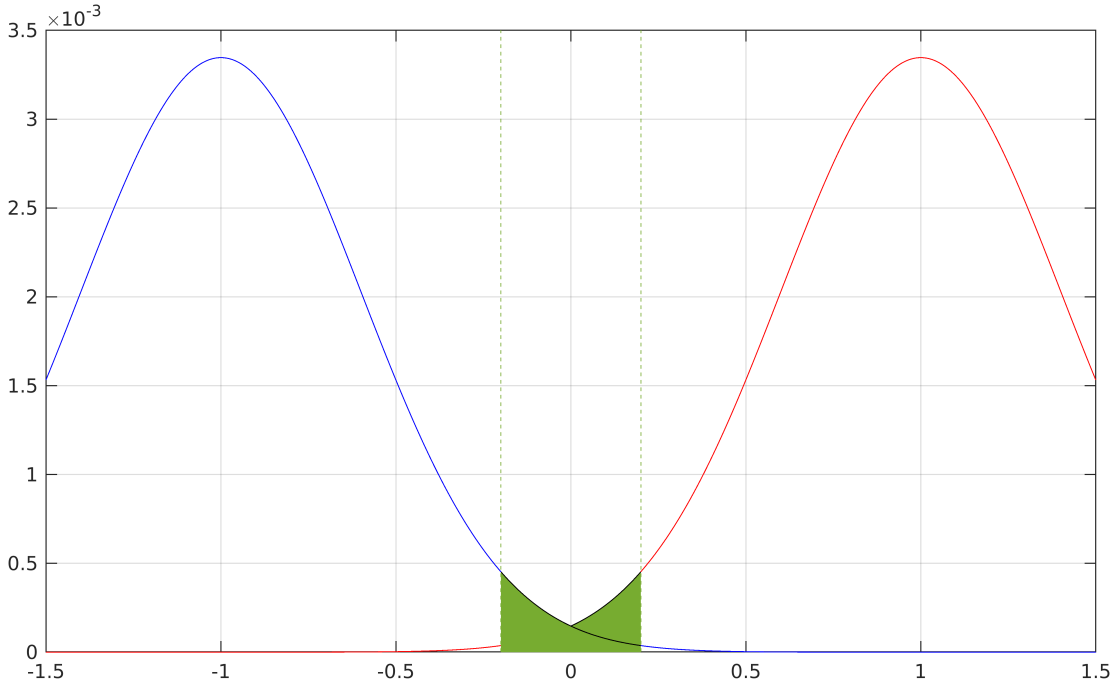

Figure 3.1: Probability Density of BPSK Symbols with an SNR of 8dB

Depending on the communication system, our opinion on an optimal decoding goal will differ. For an optical link, we minimize the number of decoding failures—the number of bit flips introduced in a decoding failure is irrelevant. In contrast, some wireless links may have error tolerance built into a protocol layer and prefer a single bit flip rather than an entire frame filled with errors. For our application, we choose to reduce the probability of a decoding error:

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}\in C} P(\mathbf{r}|\mathbf{c})P(\mathbf{c}) \tag{3.4}$$

For a given transmitted codeword $\mathbf{c}$, let the received codeword be $\mathbf{r}$ with the most likely transmitted codeword $\hat{\mathbf{c}}$. This decoding is referred to as *maximum a priori* (MAP) decoding. In the event that each codeword is equiprobable, and the channel is symmetric, the expression below is equivalent and called *maximum likelihood* (MAP) decoding.

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}\in C} P(\mathbf{r}|\mathbf{c}) \tag{3.5}$$

We begin with a channel where noise is independently distributed according to $n \sim N(0, \sigma^2)$. For the purpose of this analysis, the modulation format may be any symbol mapping such that the error statistics are identical for each bit (i.e. there exists a single bit class such as in BPSK).

We first derive the probability density function that a receiver symbol r will be flagged as an erasure given a threshold T.

$$\begin{aligned}
P_f &= P(s = +\sqrt{\varepsilon})P\big(-T < r \le T | s = +\sqrt{\varepsilon}\big) \\
&\quad + P(s = -\sqrt{\varepsilon})P\big(-T < r \le T | s = -\sqrt{\varepsilon}\big) \\
P_f &= \frac{1}{2}\int_{-T}^{T} \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{\frac{-(x-\sqrt{\varepsilon})^2}{2\sigma^2}\right\} \\
&\quad + \frac{1}{2}\int_{-T}^{T} \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{\frac{-(x+\sqrt{\varepsilon})^2}{2\sigma^2}\right\} dx \\
&= \int_{-T}^{T} \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{\frac{-(x-\sqrt{\varepsilon})^2}{2\sigma^2}\right\} dx \\
&= \int_{\sqrt{\varepsilon}-T}^{\sqrt{\varepsilon}+T} \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{\frac{-x^2}{2\sigma^2}\right\} dx \\
&= \int_{-\infty}^{\sqrt{\varepsilon}+T} \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{\frac{-x^2}{2\sigma^2}\right\} dx - \int_{-\infty}^{\sqrt{\varepsilon}-T} \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{\frac{-x^2}{2\sigma^2}\right\} dx \\
&= \frac{1}{2}\left[\operatorname{erf}\left(\frac{\sqrt{\varepsilon}+T}{\sigma\sqrt{2}}\right) - \operatorname{erf}\left(\frac{\sqrt{\varepsilon}-T}{\sigma\sqrt{2}}\right)\right]
\end{aligned} \tag{3.6}$$

Similarly, the probability of receiver symbol having an error is:

$$
\begin{aligned}
P_e &= P(s=+\sqrt{\varepsilon})P\big(r<-T|s=+\sqrt{\varepsilon}\big) \\
&+ P(s=-\sqrt{\varepsilon})P\big(r>+T|s=-\sqrt{\varepsilon}\big) \\
&= \frac{1}{2}\int_{-\infty}^{-T}\frac{1}{\sqrt{2\pi}\sigma}\exp\left\{\frac{-(x-\sqrt{\varepsilon})^2}{2\sigma^2}\right\}dx \\
&+ \frac{1}{2}\int_{T}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}\exp\left\{\frac{-(x+\sqrt{\varepsilon})^2}{2\sigma^2}\right\} \\
&= \int_{-\infty}^{-T}\frac{1}{\sqrt{2\pi}\sigma}\exp\left\{\frac{-(x-\sqrt{\varepsilon})^2}{2\sigma^2}\right\}dx \\
&= \frac{1}{2}\left[1+\operatorname{erf}\left(\frac{-T-\sqrt{\varepsilon}}{\sigma\sqrt{2}}\right)\right] \\
&= \frac{1}{2}\left[1-\operatorname{erf}\left(\frac{T+\sqrt{\varepsilon}}{\sigma\sqrt{2}}\right)\right]
\end{aligned}
\tag{3.7}
$$

As mentioned in Eq. 3.5, our goal is to maximize the probability of a correct decoding. We select an erasure threshold $\hat{T}$ that maximizes the probability of a received codeword containing a combination of erasures and errors that is correctable (Eq. 3.3)[31].

$$
\hat{T} = \arg\max_{T} P(2N_e + N_f < d_{min}) \tag{3.8}
$$

Since we have assumed a channel with independent identically distributed (IID) noise, the probability of N_e errors or N_f erasures both follow a binomial distribution.

$$
\begin{aligned}
P(N_e = x) &= \binom{N}{x} P_e^x (1-P_e)^{N-x} \\
&= P_{N_e}
\end{aligned}
\tag{3.9}
$$

$$
\begin{aligned}
P(N_f = x) &= \binom{N}{x} P_f^x (1-P_f)^{N-x} \\
&= P_{N_f}
\end{aligned}
\tag{3.10}
$$

Using Eq.16 of Kwon *et al.*[31], we have an exact expression for the probability of a codeword decision error when using an erasure decoder. To mirror the nomenclature in the reference, note $d \triangleq d_{min}$ in the equation below.

$$\begin{aligned} P(N_z > d) &= P(2N_e + N_f > d) \\ &= \sum_{x=0}^{d} \binom{d}{x} P_f^x \sum_{k=\lceil \frac{d-x}{2} \rceil}^{d-x} P_e^k (1 - P_e - P_f)^{d-x-k} \end{aligned} \tag{3.11}$$

The quantity above can be exactly minimized with a given T for some channel SNR η. However, the optimization routine may not be appropriate for real-time applications with time varying noise. The authors also derive a closed form approximation which holds for the high SNR regime relevant in short haul optics.

$$\hat{T} \approx \frac{\ln 4\pi\eta}{6\eta} + 1/6 \tag{3.12}$$

Further research into optimal thresholds was performed for generalized LDPC codes by Rapp *et al.* [45]. They analyzed the optimal erasure threshold for product and staircase codes using density evolution and simulation. Their analysis was limited to a shorter subset of BCH component codes and reduced block dimensions. For the relevant coding rate of 0.93, a simulation measured up to 0.06dB improvement when compared to hard decision decoding when using an erasure threshold of 0.04. We note that their results are for an output BER of 1×10^{-4} which is higher than expected in optical communications.

3.3 Conclusion

In conclusion, this Chapter describes erasure decoding and the selection of optimal erasure thresholds. We present the canonical erasure decoding algorithm and the selection of erasures using soft information. We also derive an optimal erasure threshold to minimize codeword errors while citing existing publications discussing erasure thresholds blocks and generalized LDPC codes.

Chapter 4: Staircase Codes

4.1 Introduction

Staircase codes are a class of hard-decision codes first presented in 2012 by Smith et al.[48]. In the subsequent years, the design was adopted within the ITU-T G709.2 standard[2] and OIF 400ZR standard[3]. While the coding scheme was originally designed for use in the 100G Optical Transport Network standard (OTU4) to transport 100GE Ethernet, it was also selected by the OIF as the outer code in a concatenated FEC for 400GE zero-reach application. The staircase structure was designed to enable hardware efficient decoding algorithms while also coupling component codes across neighbouring product blocks. To properly motivate the design of the staircase structure, we will review the ideas that the code borrows and challenges that it is designed to mitigated.

4.2 Block Codes

We first consider a simple forward error correction scheme consisting of a set of independent block codes of length N. In a bounded distance decoder, any individual codeword which encounters in excess of t errors will result in a decoding failure.

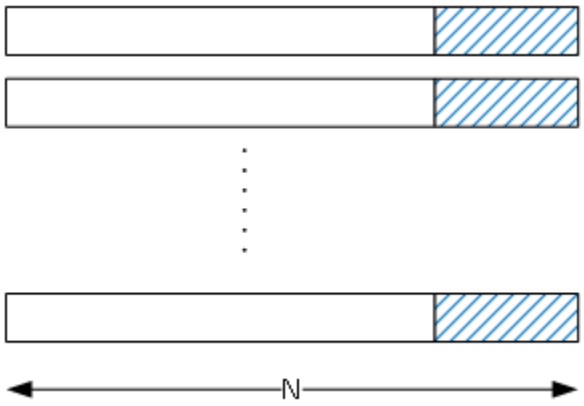

Figure 4.1: Sequential Block Codes of Length N

The probability of any given codeword being uncorrectable by a bounded distance decoder is upper bounded by the channel error rate p_e (assuming a DMC):

$$P(N_e > t) = 1 - \sum_{k=0}^{t} \binom{N}{k} p_e^k (1 - p_e)^{N-k} \tag{4.1}$$

For each bit transmitted using a block code, the information content is reduced due to the encoding overhead. Rate R is a measure of the efficiency of that transmission overhead, equal to the information content per transmitted bit.

$$R_b = K/N \tag{4.2}$$

4.3 Product Codes

If we rely on a set of disjoint codewords to perform error correction, the probability of a decoding failure in any given codeword would be excessive, even when using low rate codes. In order to reduce the probability of any given codeword being uncorrectable or

decoded in error, we can couple codewords together such that the successful decoding of one codeword improves the probability of decoding others. One strategy is to create a longer code from short component codes in a product code. This approach was described early in the development of the field of error correction by Peter Elias in 1954[23]. The concept was later extended to cyclic codes in 1965[11]. The product code remains relevant with the use of capacity approaching Turbo codes[43] and other structures.

Given two systematic codes: $C_1 = (N_1, K_1)$ and $C_2 = (N_2, K_2)$, we form a product code (N_1N_2, K_1K_2) with code C_1 computed across the rows and code C_2 computed across the columns.

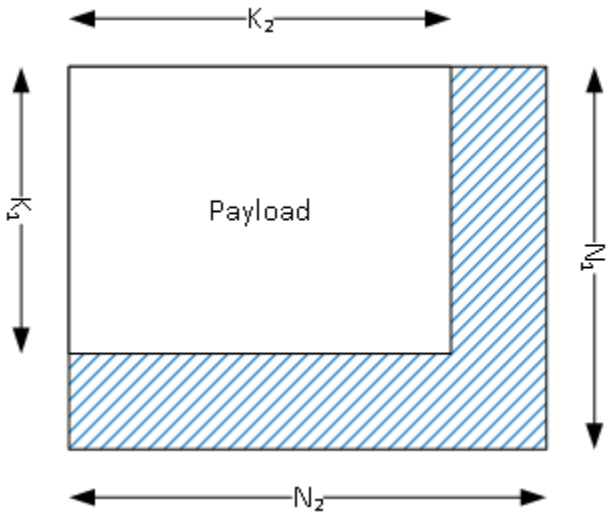

Figure 4.2: Product Codes of $C_1 \times C_2$

The minimum distance of the resulting product code is increased to $d_p = d_1d_2$[23] where d_1 and d_2 are the minimum distances of C_1 and C_2, respectively. The rate R_p of a product code is the product of the constituent component code R_1 and R_2. Assuming the component codes of the row and column code are the same, we express rate as:

$$R_b = \frac{K^2}{N^2} \tag{4.3}$$

As an example, consider a BCH(255,239) block code with $t = 2$ and rate $R = 0.937$. The product code with BCH(255,239) component has an error correcting capability of $t = 12$ at the expense of a reduced rate of $R_p = 0.878$. The decoding process used may vary. Among the simplest would be an "iterative" hard-decision approach where component codewords along the rows are decoded followed by the columns (or vice-versa). Further, the product structure demonstrates improved burst tolerance, since a grouping of errors within a horizontal codeword are likely to present a correctable pattern to the orthogonal component codewords [11].

4.4 Convolutional Codes

While a product structure enables joint decoding of component codewords within a block, we are still limited by the finite dimensions of the block size. In the event of a burst of errors within block B_i, a fully decoded adjacent block B_{i-1} will not increase the probability of successfully decoding B_i. In 1955, Elias initially proposed a solution to this problem for block codes: convolutional coding[13]. These codes are famously in use by several NASA deep space missions [50]. A convolutional structure allows the decoder to make a correction by relying on the previous m received symbols without any discontinuities as would be encountered by block codes.

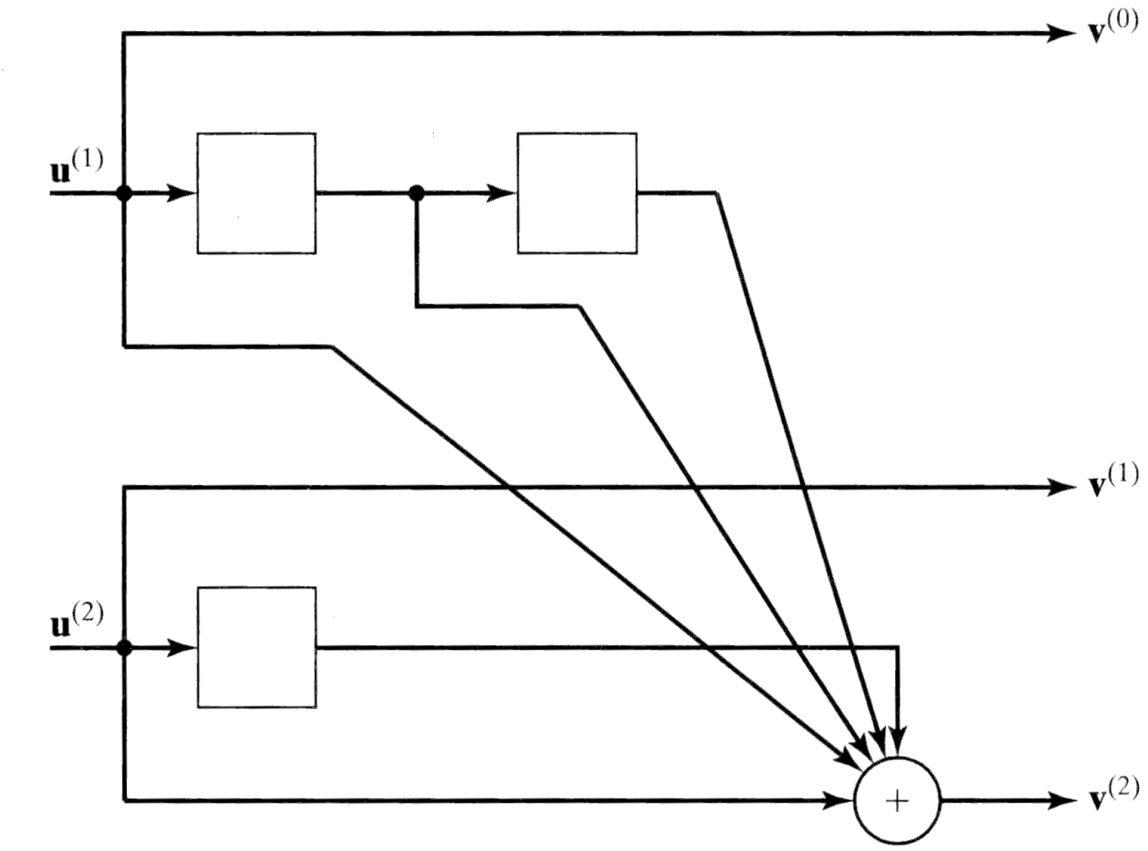

Figure 4.3: Rate 2/3 Convolutional Encoder [32]

A convolutional encoder can be implemented efficiently using a method similar to block codes, multiplying the message sequence by a convolutional generator matrix. While on the receiver side, the decoder has both a well known ML and MAP optimal decoding structure named Viterbi[24] and BCJR[6] respectively.

However, there are a few drawbacks that prevent a standard convolutional code from being appropriate in short haul optical systems. First, the state size and computational complexity for Viterbi and BCJR decoders increase exponentially with an increase in code-

word constraint length. If we choose to instead use a non-optimum sequential decoding algorithm such as ZJ(Stack)[53], the high variance in decoding latency makes hardware efficient implementation very challenging with a high probability of buffer over/under flows (up to 10^{-3}). Second, while convolutional codes are capable of delivering suitable error correction performance for wireless networks with post correction error rates of $P_e \approx 10^{-6}$, they are generally unsuited for optical links which require $P_e \approx 10^{-15}$.

4.5 Staircase Codes

Finally, we arrive at an examination of the structure of Staircase codes. The scheme borrows from both product and convolutional codes. We retain the two dimensional structure of the product code with component codes C computed on rows and columns. However, in staircase codes, the row codewords of block B_i contain data from B_{i-1} and the columns of block B_i are contained in row codewords of B_{i+1} ad infinitum. In figure 4.4 we show the basic structure of a staircase code. The computed parity is shown in hatched blue along with the location of two component codes shown in dotted red.

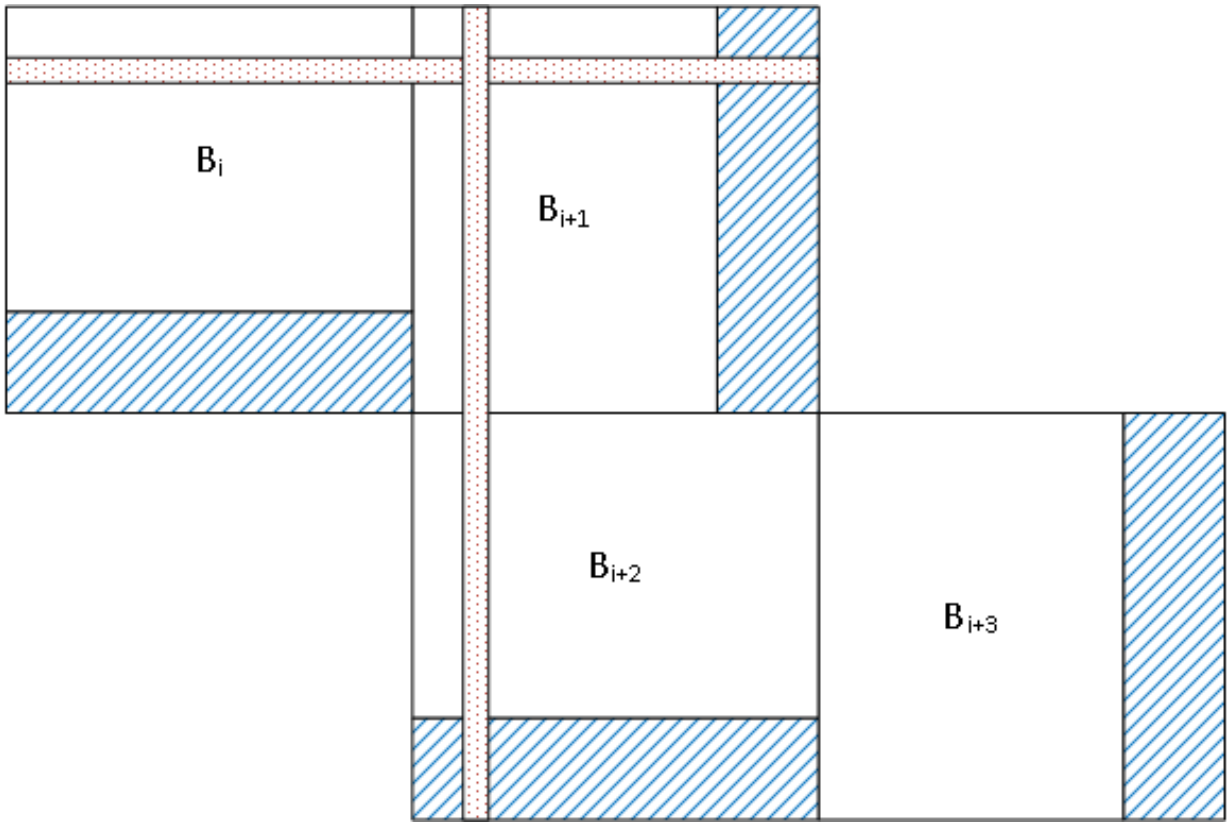

Figure 4.4: Staircase Block Code Structure

As with a product code, we have the option to choose a component code from among the wide selection. Given a staircase code with blocks of size $L \times L$ and component code (N, K), we transmit an additional $(N/2)(K - N/2)$ bits of payload data per staircase block. The resulting code rate is as follows:

$$\begin{aligned} R_s &= \frac{(N/2)(K - N/2)}{(N/2)^2} \\ &= \frac{K - N/2}{N/2} \\ &= \frac{2K}{N} - 1 \end{aligned} \tag{4.4}$$

We can compute the rate loss penalty of a staircase structure when compared to a standard product code. Note that we assume the use block code has been extended such that $N/2$ is an integer value. The penalty in efficiency is derived in Eq. 4.5 and plotted in Fig. 4.5 for a BCH code of varying field order m and error correcting capability t.

$$
\begin{aligned}
\eta &= \frac{R_s}{R_p} \\
&= \frac{K - N/2}{N/2}\frac{N^2}{K^2} \\
&= \frac{2N(K - N/2)}{K^2} \\
&= \frac{2NK - N^2}{K^2} \\
&= \frac{2N(N - mt) - N^2}{K^2} \qquad \text{Given that } K \approx N - mt \\
&= \frac{N^2 - 2Nmt}{K^2} \\
&= \frac{K^2 - m^2t^2}{K^2} \\
&= 1 - \frac{m^2t^2}{K^2} \\
&= 1 - \frac{m^2t^2}{(N - mt)^2} \\
&= 1 - \frac{m^2t^2}{(2^m - mt)^2} \\
\frac{\partial \eta}{\partial m} &= \frac{t^2 2^{m+1} m(m \log 2 - 1)}{(2^m - mt)^3}
\end{aligned}
\tag{4.5}
$$

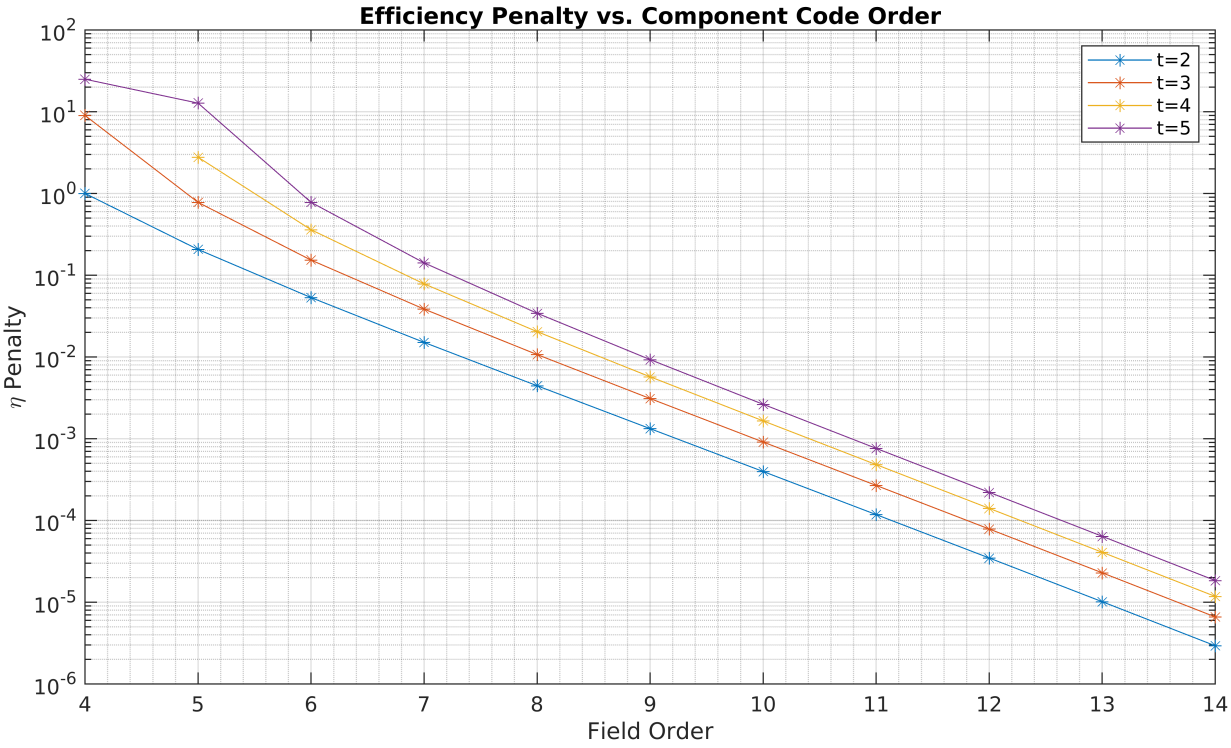

Figure 4.5: Efficiency Penalty of Staircase Codes

A penalty in the rate for a given component code may be worthwhile if the overall performance of the resulting system is improved. A more sophisticated comparison relies on the net coding gain (NCG, γ_{nc}) at a given output error rate of interest[46]. This figure includes the effects of reduced spectral efficiency of coding overhead as a loss along with improvement in BPSK error rates as a coding gain.

$$\gamma_{nc} := Q_{out} - Q_{in} + 10\log(\eta) \tag{4.6}$$

In the above equation, we use Q to represent the equivalent BPSK SNR as is common in optical application. Note that γ_{nc} is a more meaningful figure of merit but is still subject

to misinterpretation. Care must be taken to ensure that the structure of decoders under evaluation are of comparable complexity. If we allocate more latency or hardware to an implementation, we almost invariably observe an increase in the input BER required for a given output BER and thus an improved NCG.

In Fig. 4.6, we see a plot of the NCG of a reference staircase implementation. With an input BER of 4.50E-3 to obtain a post FEC BER of 1E-15, it achieves a NCG of 9.25dB. This figure was then revised based on the results of an FPGA simulation to 9.41dB[48]. In comparison, a competing proposal put forth a product code using a nearly identical triple error correcting component code of BCH(1023,992)[30]. The paper predicted an identical 4.5E-3 input BER to achieve the desired optical standard 1×10^{-15} output BER. However, they found that performance was heavily degraded by misconvergence and, thus, required an input of 4E-3. This degraded the performance of the code from a predicted NCG of 9.39dB to a measured NCG of 9.26dB.

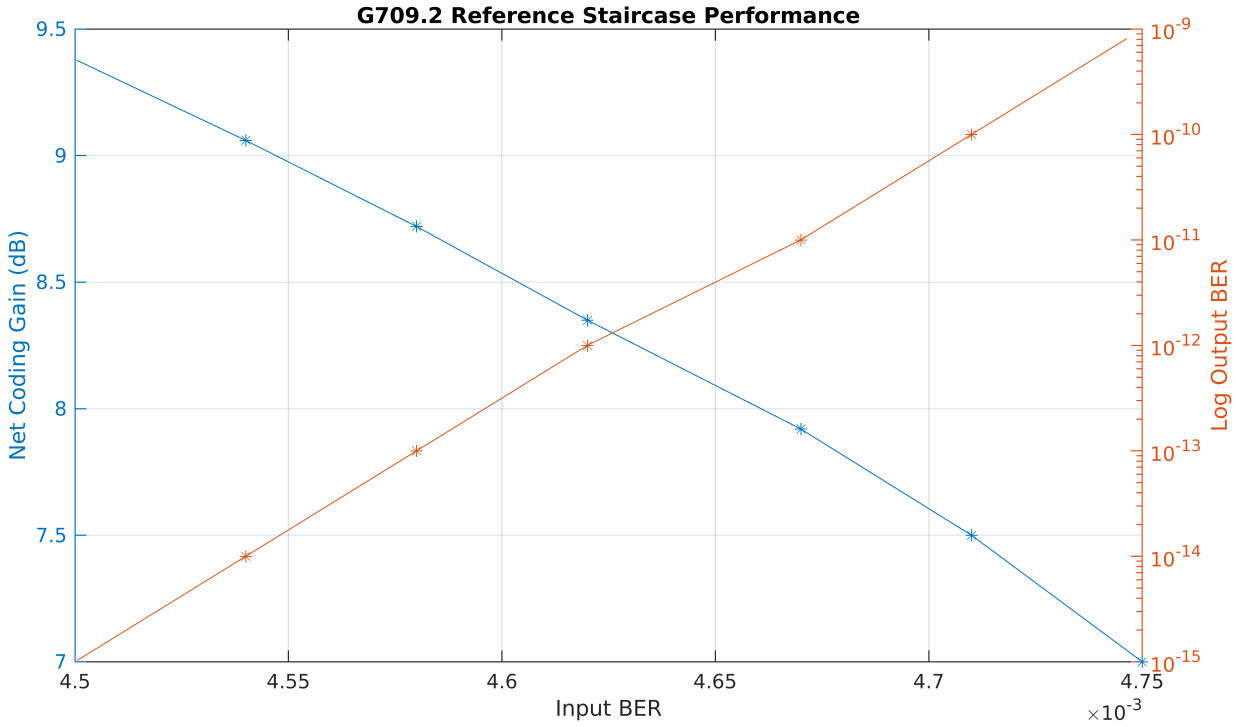

Figure 4.6: Reference Performance of Staircase Codes with BCH(1022,990)[2]

4.5.1 Decoding Algorithm

The decoding approach used in a hard-decision staircase code bears a striking resemblance to iterative decoding procedure used in hard decision product codes as early as 1954[23] and described in the previous subsection. In contrast with product codes, we store a finite number of previous blocks in a *decoding window* (Fig. 4.7) due to the convolutional nature of staircase codes.

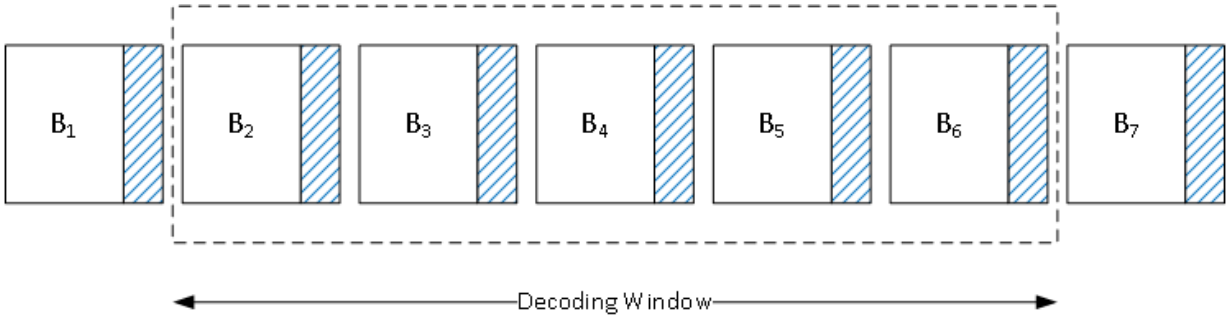

Figure 4.7: Staircase Code Window

The number of blocks we store is a trade-off between decoding latency, hardware complexity and error correction performance. The longer the decoding window, the better the error correcting performance at the cost of increased latency and hardware. Refer to section 5.4 for a detailed description of the decoder architecture used for staircase codes.

Chapter 5: Experimental Methodology

5.1 Introduction

Experimental validation is typically required to accurately evaluate the performance of a complex error correcting system. While the convergence properties of iterative systems can be theoretically analyzed through tools such as Extrinsic Information Transfer Chart (EXIT) charts[22], predictions can differ significantly from experimental results[45]. Experimental validation of a code can be implemented with hardware in a lab or as a software simulation. A full hardware experiment is often more accurate and capable of measuring performance in the high SNR regime; however, it is also significantly more expensive and less flexible. The findings in this thesis are the result of a set of heavily parallelized computer simulations using codes written in the MATLAB language.

Forward error correction simulations are often prototyped in MATLAB or Python then re-written using C/C++. This approach allows for early prototyping to be done using a flexible interpreted language, while evaluation of final *production* performance is conducted using the much faster compiled binaries. However, porting the software descriptions of a complex decoder between languages is error-prone with a divergence between code bases being difficult to identify and rectify due to the statistical nature of error correcting codes. The simulations described in this thesis were performed entirely in MATLAB using a custom finite field routines to achieve an average 50Mbps of decoder throughput on a

single desktop.

5.2 Coding Parameters

In section 2, we briefly touch on different attributes we consider when selecting a codeword. The trade-offs between the impact of rate, minimum distance and decoding complexity can make performance comparisons of two different codes difficult and potentially misleading. To facilitate performance comparison of our implementation against existing codes, we select a standardized component code employed in OTU4 and 400ZR. The code is a modified triple error correcting BCH(1022,990) based on a standard BCH(1023,993) code with a primitive polynomial $x^{10} + x^3 + 1$. The modified code is the result of extending with $(x^2 + 1)$ and shortening the code by a single message coordinate.

$$g(x) = (x^{10} + x^3 + 1)(x^{10} + x^3 + x^2 + x + 1)(x^{10} + x^8 + x^3 + x^2 + 1)(x^2 + 1) \tag{5.1}$$

The resulting BCH block code has a rate $R = 0.969$. When used in a staircase structure, the nominal code rate is further reduced to $R_s = 0.937$[1] (Eq. 4.4). The modifications to the structure of the staircase code are necessary to ensure that two consecutive 122368 bit OTU4-SC frames fit into the payload portion of single staircase block of 512x478 bits.

The OTU4 staircase block described above is not square, we therefore deviate from the ideal structure shown in Fig. 4.4 by modifying the parity on the first two rows of block B_i. These rows have no corresponding columns in block B_{i-1}, so we replace the missing message coordinates in the black hatched area with zeros resulting in a further shortening of the component BCH code. The structure of the encoding of B_i is shown in Fig. 5.1.

[1]Due to the rectangular frame structure of 512x510, the actual rate is slightly lower at $R = 0.934$

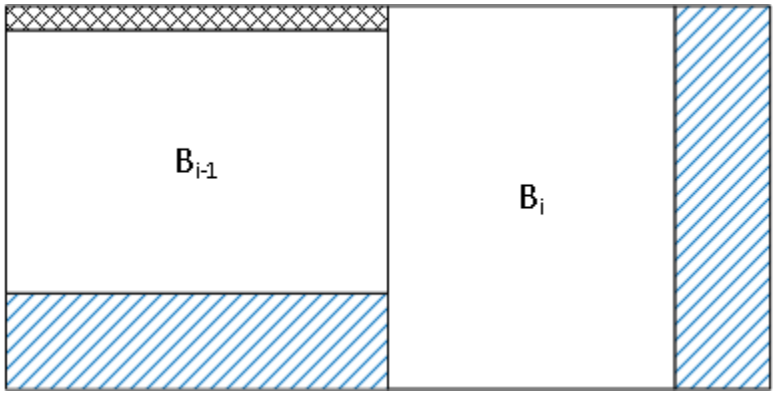

Figure 5.1: G709.2 Staircase Block Structure

As described in Ch. 2, a valid primitive binary BCH codeword has consecutive powers of α^{2i} as roots. In the language of matrix algebra, this equates to the codeword being in the null space $\mathcal{N}(\mathbf{H})$, which in turn has dimensionality K. We reduce the computational complexity of the decoder by permuting the columns of $\mathbf{H}$ such that a linear mapping exists between the codeword roots and the location of errors. This modification to the column mapping avoids a GF(2^{10}) logarithm to translate the roots of $\Lambda(x)$ into error locations $\{j_l\}$. The permutation also makes it possible to row reduce $\mathbf{H}$ after the addition of the extended parity check bits.

5.3 Encoder Structure

The encoder structure of a staircase code shares attributes found in both block and convolutional encoders. The parity of block $\mathbf{B_i}$ is a function of its own payload along with the content prior block $\mathbf{B_{i-1}}$. An example of the constituent code is shown in Fig. 5.2 with the message portion in green cross-hatching. Each constituent code contains 512 message coordinates from the prior block and 478 message coordinates from the current block. Note,

the message coordinates within the prior block can be either payload or parity.

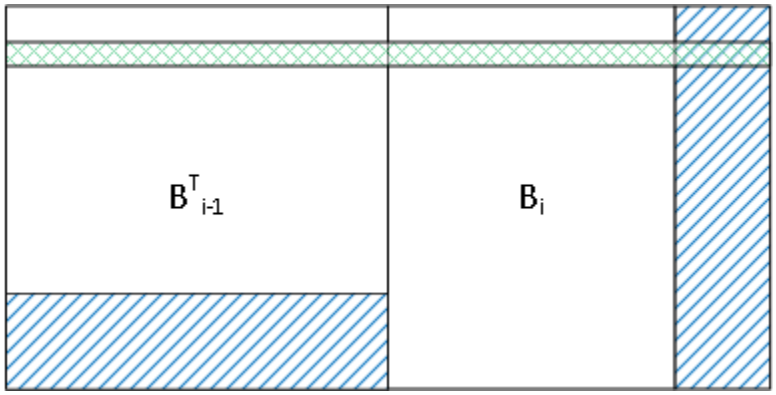

Figure 5.2: G709.2 Staircase Block Structure

The parity of each block row is (blue hatching) is generated by multiplying each message vector by the parity matrix **P**. The parity matrix is a non-identity block matrix from a row reduction of **G**.

$$\mathbf{p_j} = \mathbf{P}\mathbf{m_j} \tag{5.2}$$

5.4 Decoder Structure

The canonical structure of a staircase decoder uses a sliding window spanning multiple received blocks. The latency of the decoder pipeline is a function of the depth of the window and the channel data rate. For example, if we ignore overhead for line framing and inner codes, a signal with a data rate of 400Gbps will experience an additional latency of 611µs per block in the decoding window, given that each block contains 244736b of payload. The number of blocks within a decoding window, as described in most standards[2], is five active with a total width of seven blocks. The latency incurred within a standard

window staircase decoder is therefore 4.2ms. If we further increase the decoding window to improve performance, there will be an increase in both latency and hardware complexity. In addition, the die area required to implement a staircase decoder grows super-linearly with the length of the decoder window due to the increase in wiring complexity between the syndrome storage and algebraic decoders.

The nominal data rate also limits the number of decoding iterations we can perform during a given state of the decoding window. If we again consider the 400Gb line rate, a new block arrives at the decoder approximately every 611ns. Suppose we operate with a digital clock rate of 1GHz, we have at most 611 clock cycles to perform corrections before a new staircase block enters the decoding window. The number of corrections iterations performed on each block will be limited by instances of decoder engines operating in parallel.

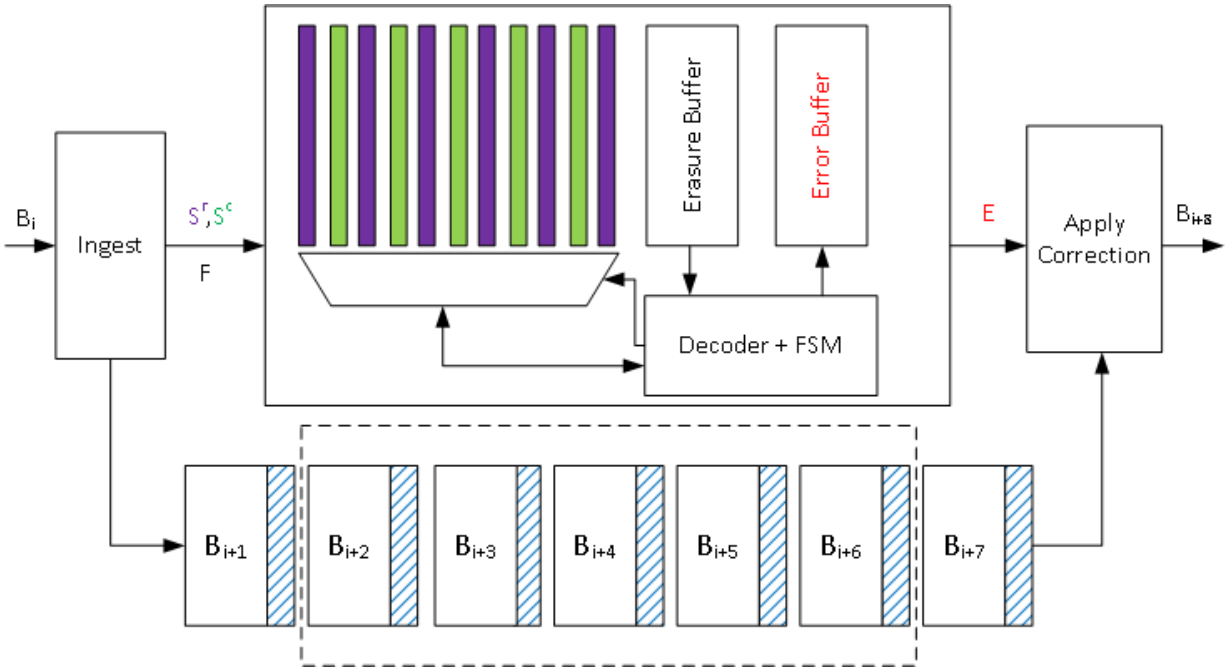

Figure 5.3: Staircase Decoder Structure

A diagram of the decoder implementation used for simulation is shown in Fig. 5.3. The execution of the decoding steps would be completed in parallel within a real hardware decoder. For the purposes of simulation, the decoder operates serially.

First, we compute the row and column syndromes $\mathbf{S^r}$ and $\mathbf{S^c}$ of the latest received block $\mathbf{B_i}$ and store the result as a series of 32 bit words. Error decoding is performed by computing the error location of constituent codewords where the syndrome is:

$$\mathbf{S_i} = \mathbf{S^c_{i-1}} + \mathbf{S^r_i} \tag{5.3}$$

Computing the error locations $\{j_i\}$ is performed as described in section 2.2 using algebraic decoders with specific optimizations for the $GF(2^{10})$ extension field. For codewords

that were successfully corrected (i.e. a valid codeword was found), we store the error locations in a memory and update the row and column syndromes of $\mathbf{B_i}$ and $\mathbf{B_{i-1}}$ to reflect corrective bit flips at the error locations.

When using erasure coding, we buffer a vector of erasure locations $\mathbf{F_i}$ for each staircase block $\mathbf{B_i}$. To produce the error locations, we use the erasure procedure described in algorithm 1 on page 28 along with the bounded distance decoder defined above.

In the case of either bounded distance decoding or erasure decoding, we correct a pair of blocks at a time. The sequence of blocks to be corrected is determined by a finite state machine within the decoder. The scheduling of block decoding order is a balance between incorporating new information from the most recent block while also ensuring the oldest block is error free before it exits the decoding window.

Several other optimizations have been made to reduce the total time bounded distance decoding is performed on a given block. The most impactful optimization is to avoid performing decoding on a constituent code that has remained unmodified since the prior decoding iteration. We use a *dirty* flag to mark rows and columns where syndromes have been modified by corrections to an orthogonal codeword. In the absence of a *dirty* flag on a constituent row and column syndrome, we skip erasure or bounded distance decoding. The results of decoding in this state would give us no new information to correct the codeword.

In addition to reducing decoding complexity based on constituent codeword states, we also avoid decoding a staircase block in any way if it meets the criteria to be *sealed*. We seal a block when we have a high confidence that it has no uncorrected errors remaining and that any further corrective actions are more likely to induce errors than eliminated them. In such cases, we discard any correction to an adjacent block that flips a location in the sealed block. This has the added benefit of reducing the chances of a misconvergence

on the other blocks.

When a block exits the window, we apply the net sum of all the corrective bit flips computed during the iterative decoding procedure. To determine if the block is in fact error free, we check whether the payload CRC is valid while all the component code have zero syndromes. The probability of an undetected block based solely on a CRC32 is a function of the checks weight distribution function for a payload of 244664 bits. It is known that the 802.3 IEEE standard check codeword has a minimum distance of 3 for codewords between $[91608, 2^{32} - 1]$[14][26]. While the full weight enumeration is unknown for codewords of this length, we can compute an upper bound on the probability of undetected frame errors using d_{min}. Even given a highly degraded output BER of 1×10^{-10}, the probability of a block exiting with undetected errors is at most 1×10^{-20}. However, at the nominal output error rate of 1×10^{-15} the probability drops to 1×10^{-40}, a negligible quantity for any feasible data rates.

Chapter 6: Discussion

6.1 Complexity

In Chapter 2, we discussed the complexity of BCH decoding using either an algebraic implementation or a generic approach. The analysis captured the cost of decoding a single BCH3 codeword. In the context of a hard-decision iterative decoding, an iteration is the process of using a codeword syndrome $\mathbf{S}$ to compute the set of error locations $\{\hat{e}_i\}$. We then use the error locations to update the syndrome to reflect the best estimate of the transmitted codeword $\hat{\mathbf{c}}$. This iterative procedure continues while the block is in the decoding window. A decoder failure occurs when a block exits the decoding window with one or more uncorrected errors.

The amount of work performed while decoding is not just a function of the *cost per iteration*, but also the *number of iterations* required per block. When comparing staircase decoding strategies, it is often sufficient to use the number of algebraic iterations as a proxy for total work. A suite of simulations was performed to measure the number of BCH decoding iterations as a function of input bit error rate.

We see in Fig. 6.1 that an increase in channel noise results in a greater number of iterations required. At a certain noise threshold, the window length and available decoder iterations are insufficient to correct all the errors in a block and the number of iterations climbs in a stepwise fashion.

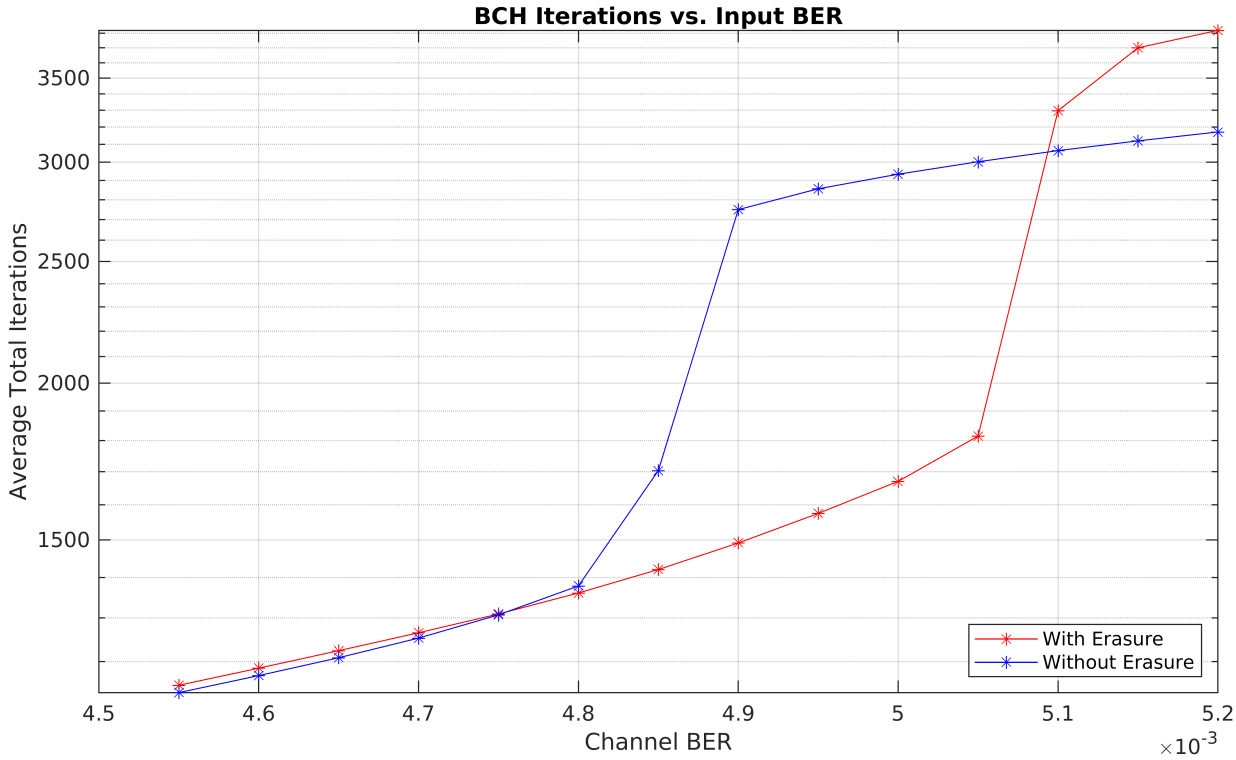

Figure 6.1: BCH Iterations as a function of Input BER

We observe that the average number of decoder iterations is nearly identical with and without erasures when operating at an input BER below 4.8×10^{-3}. In addition, we see in table 6.1 that the maximum number of decoder iterations required with erasures is lower below threshold. This reduction in maximum iterations required can be used to reduce the required clock speed or the instances of algebraic decoders operating in parallel.

We also measure the probability of a block requiring a given number of iterations at an input BER of 4.8×10^{-3}. The distribution has a long tail of 3431 and 2196 iterations with and without erasures respectively. Note that the red distribution without erasures is

[1]Average complexity of erasure decoding relative to baseline without erasures

	Without Erasures				With Erasures				
Input BER	**Min**	**Mean**	**Max**	σ	**Min**	**Mean**	**Max**	σ	**Complexity**[1]
4.55×10^{-3}	765	1133	1845	95	767	1148	1665	99	101.3%
4.6×10^{-3}	781	1169	1780	98	795	1185	1737	101	101.4%
4.65×10^{-3}	818	1208	1859	102	835	1224	1790	105	101.3%
4.7×10^{-3}	841	1252	2108	109	861	1264	1905	109	101.0%
4.75×10^{-3}	865	1307	2514	120	832	1309	2152	114	100.2%
4.8×10^{-3}	929	1378	3134	146	898	1360	2145	121	98.7%
4.85×10^{-3}	965	1703	3431	505	901	1420	2196	131	83.4%
4.9×10^{-3}	1009	2749	3594	239	976	1491	2721	143	54.2%
4.95×10^{-3}	1078	2856	3669	187	1019	1574	3173	159	55.1%
5.0×10^{-3}	1085	2934	3739	181	1080	1669	3630	177	56.9%
5.05×10^{-3}	1072	3001	3813	178	1080	1814	4278	301	60.4%
5.1×10^{-3}	1294	3063	3943	176	1040	3296	4481	640	107.6%
5.15×10^{-3}	1195	3119	3929	175	1209	3699	4576	228	118.6%
5.2×10^{-3}	1104	3170	4035	175	1245	3820	4686	208	120.5%

Table 6.1: Algebraic Iteration Distribution for Various Error Rates

operating slightly above threshold, we see the iteration distribution is skewed with a higher number of iterations being more probable.

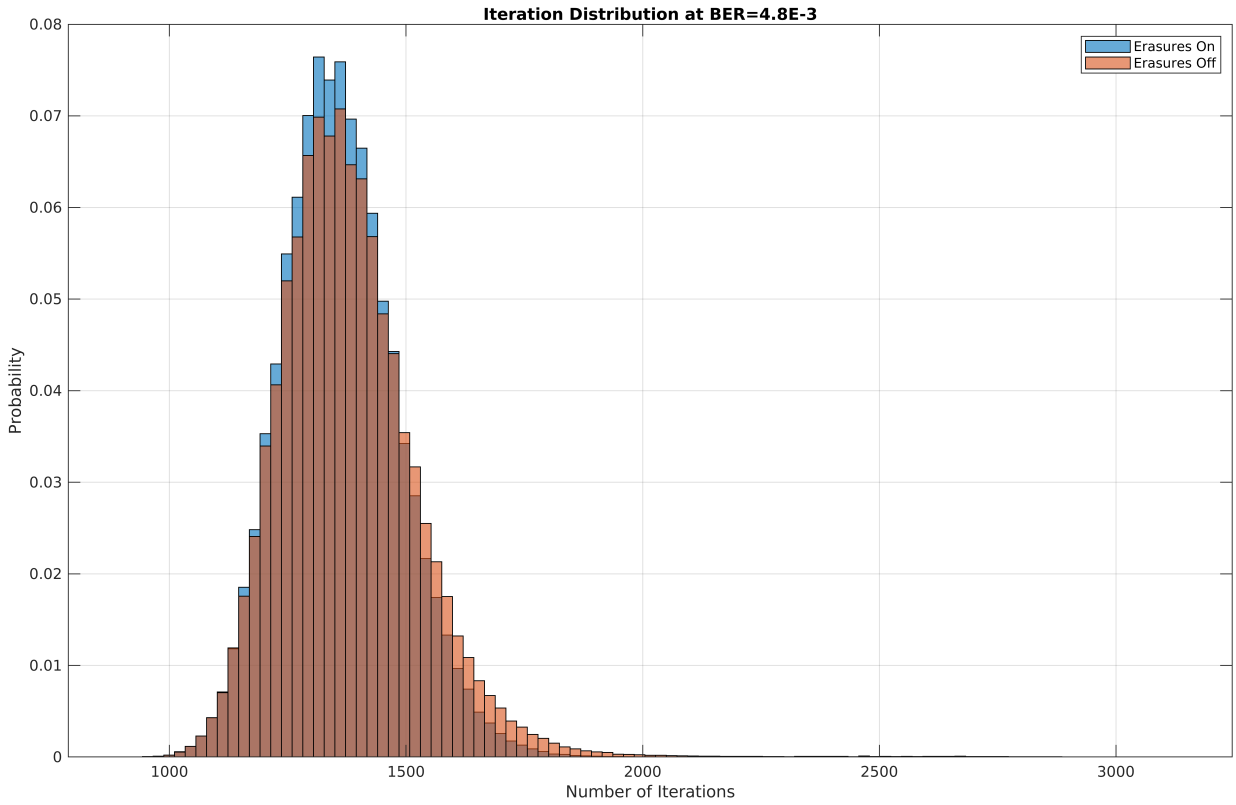

Figure 6.2: BCH Iteration Distribution at an Input BER of 4.8×10^{-3}

We can conclude that for any given input bit error rate, the upper bound on complexity (and therefore power) is reduced when using erasure decoding. Average power can also be kept constant by enabling erasure decoding on demand when the input error rate increases above 4.75×10^{-3} or the average iteration count increases above 1307.

6.2 Erasure location

In addition to erasure threshold, we can modify the performance characteristics of the erasure decoder by choosing the position within the decoding window to leverage the erasure locations. Given that we have a vector $\mathbf{F_i}$ of up to 512 erasures locations associated with the rows of block $\mathbf{B_i}$, we can choose how late in the decoding window we use $\mathbf{F_i}$ to run erasure decoding the block. The measurements were performed at an input error rate of 4.9×10^{-3} to increase the statistical significance of the results while keeping simulation runtime viable.

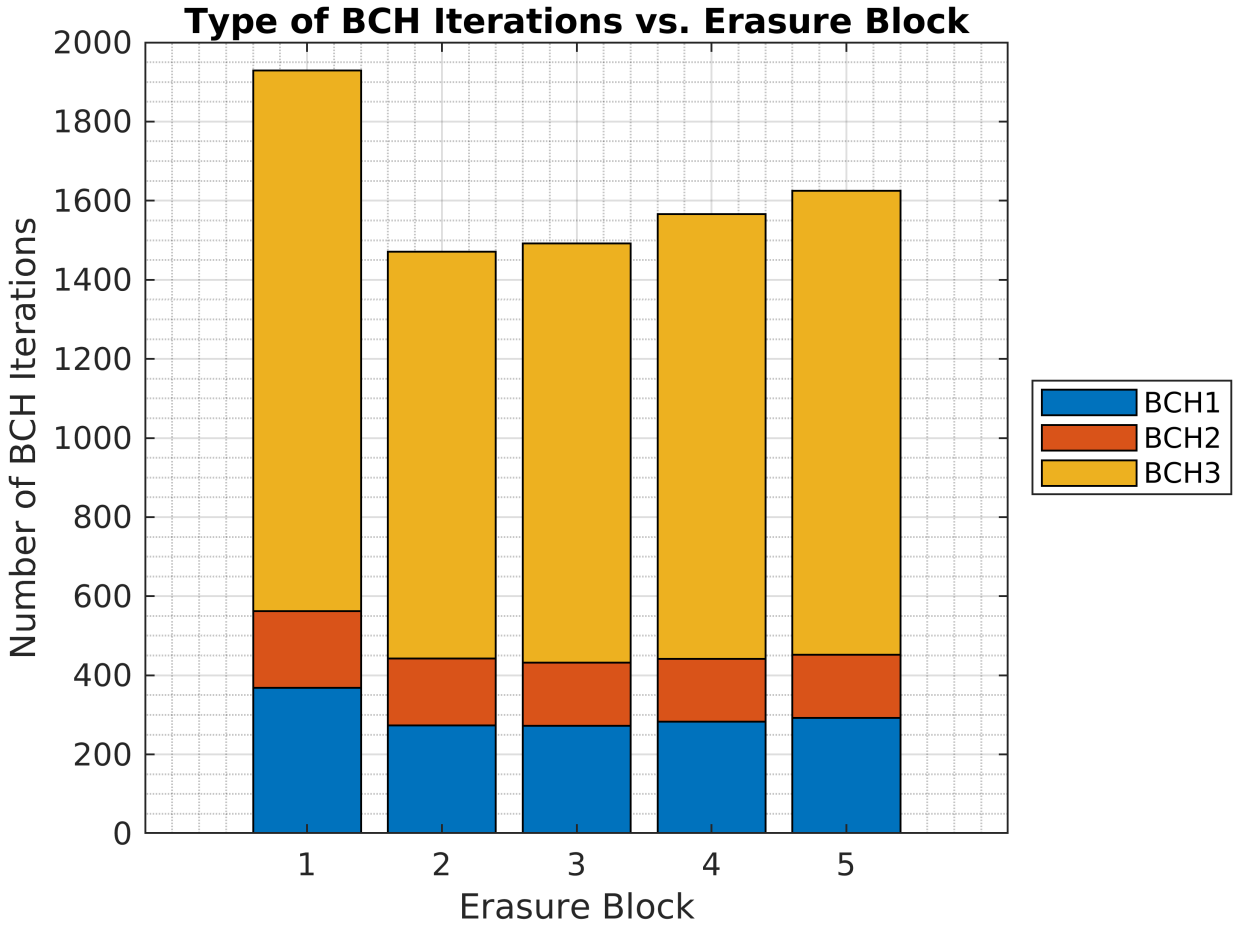

Figure 6.3: Iteration Count vs. Erasure Block Location at 4.9×10^{-3} BER

In Fig. 6.3, we show that the decoder performs the least number of algebraic decoding iterations when erasures are applied on blocks in position two of the decoding window. As seen in Fig. 6.4, applying erasures on blocks in position two also results in the best error correction performance at an input BER of 4.9×10^{-3}.

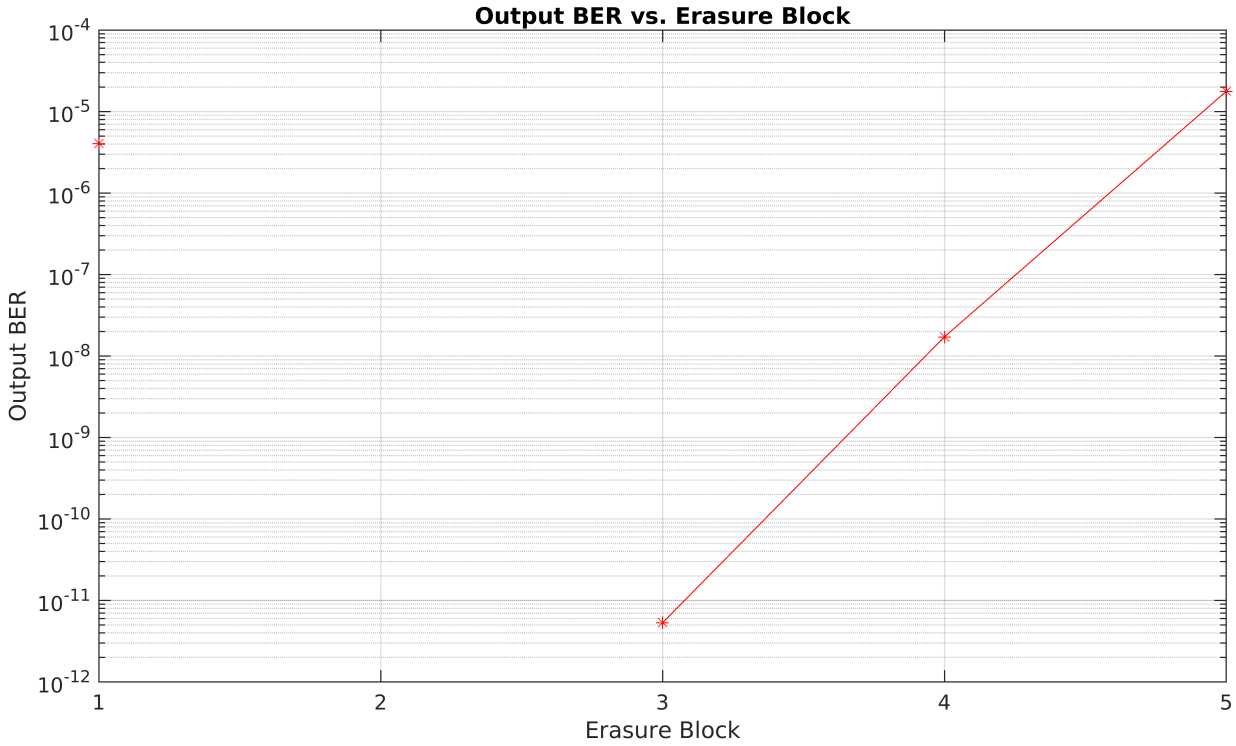

Figure 6.4: Output BER vs. Erasure Block Location at 4.9×10^{-3} BER[2]

We observe that blocks in the second window location already have a large quantity of corrections applied. When applying the erasure corrections at the optimal location, we see a reduction reduced probability of misconvergence while benefitting from the improved error correcting performance of erasure decoding.

6.3 Erasure Threshold

As detailed in Chapter 3, an optimal threshold for erasures is one that yields the lowest probability of a codeword being decoded in error. An optimal threshold was analytically

[2]An erasure block of 2 produced no errors during the simulation.

derived for block codes by Kwon *et al.*[31] and further explored for generalized block LDPC codes by Rapp *et al.*[45]. However, neither paper analyzed the performance of higher efficiency G.709 codeword at the output error rates required by optical communication links.

To reduce the complexity of the erasure circuit, we implement an erasure buffer which stores up to a single erasure per row for a maximum of 512 erasures per staircase block. In addition to reduced complexity, a single erasure decoder also reduces possibility of misconvergence. With our erasure implementation, the decoder is relatively insensitive to threshold increases above the optimum as we generate at most a single erasure in the back half of the component code. We show the performance impact as a function of erasure threshold in Fig. 6.5 using BPSK modulation $\{-1, 1\}$ noise loaded with AWGN for a channel SNR of 5.23dB (equivalent input BER of 4.85×10^{-3}).

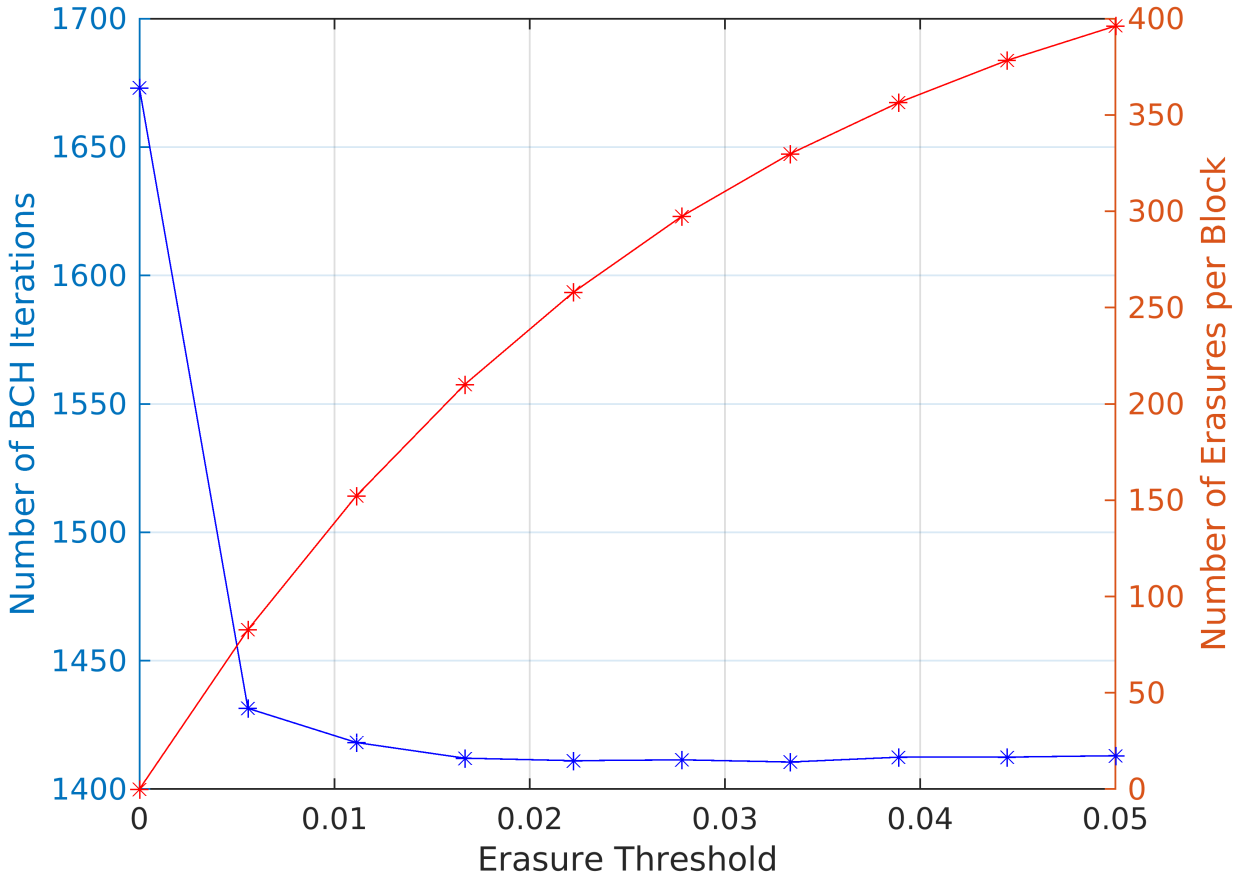

Figure 6.5: BCH Iterations and Erasure Count as a function of Erasure Threshold

We measure the work performed by the decoder at a given erasure threshold by recording the iteration statistics. An increase in the average iterations per block implies the decoder is working harder and thus closer to decoding threshold. The approach has the benefit of tractable experiment durations while also producing meaningful results at input bit error rates relevant to our SNR threshold[47].

We found a threshold of 0.033 was optimal, resulting in a reduced erasure and iteration count. However, as predicted, we see little variation in decoder performance across larger

thresholds values. This threshold differs from the predicted optimal threshold of 0.3 using Eq. 3.12. Thresholds below 0.022 show signs of degraded iteration counts, indicating unreliable bits are no longer being marked as erasures.

6.4 Performance

The performance analysis of optical transport forward error correcting codes is challenging due to the low output error rates required. Without a hardware implementation, obtaining reliable measurements near the typical output BER threshold of 10^{-15} requires a supercomputer or large scale cloud deployment. Alternatively, measurements may also be performed using programmable hardware such as an FPGA. In 2019, researchers achieved 200Gbps of throughput for a concatenated decoder including a staircase decoder[12]. Using 50 devices in parallel, they were able to obtain data points at the FEC threshold in under a day. These simulations, along with others, have shown that staircase codes are free of significant flaring using FPGA simulation, displaying roughly log-linear performance down to 3.8×10^{-21}[48].

As the staircase decoder and simulation created for the purpose of this manuscript are entirely implemented in software, we are limited in the output BER range that can be reliably measured. Our MATLAB software implementation is able to scale to 50Mbps on a single consumer desktop[3] after re-implementing the majority of the finite field routines. To measure the lowest point in Figure 6.6, a simulation was run for 11 days to sample 133 output errors.

[3] AMD Ryzen 5950X (16c/32t) with 64GB of memory running MATLAB R2023a on Red Hat Enterprise Linux 9

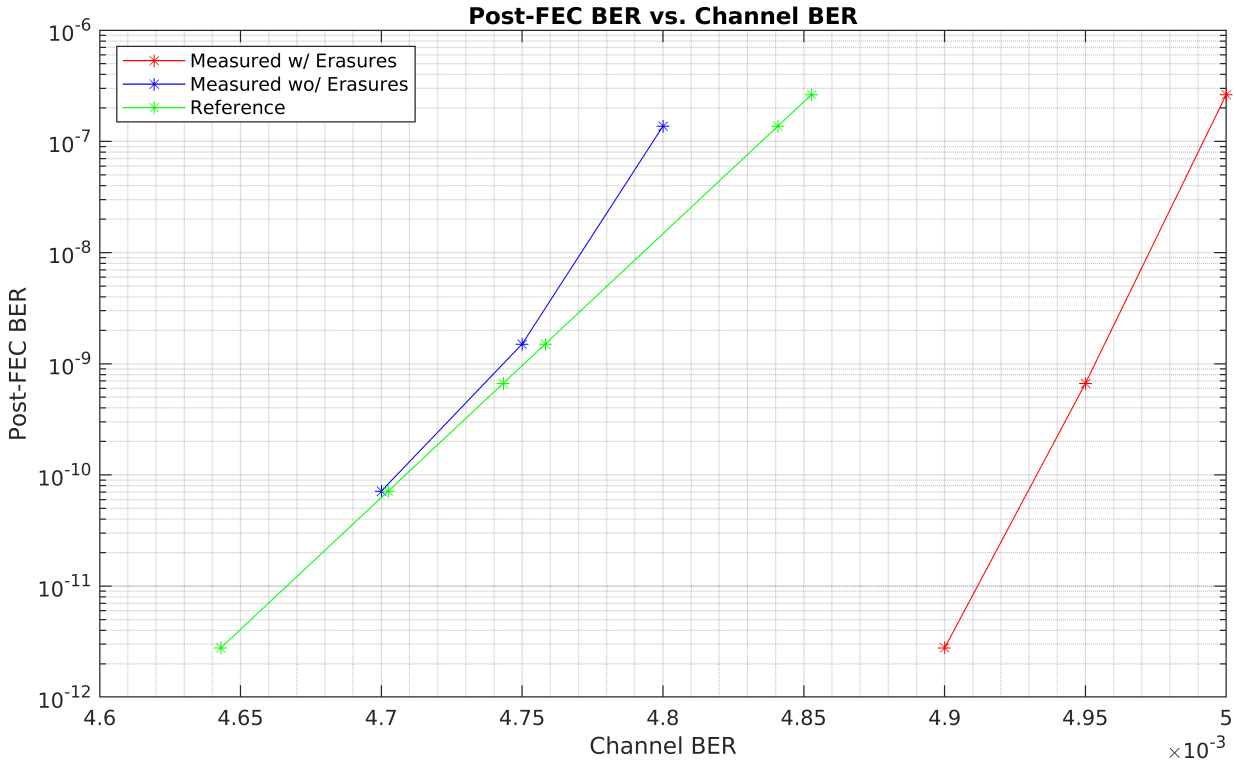

Figure 6.6: Output BER vs. Input BER

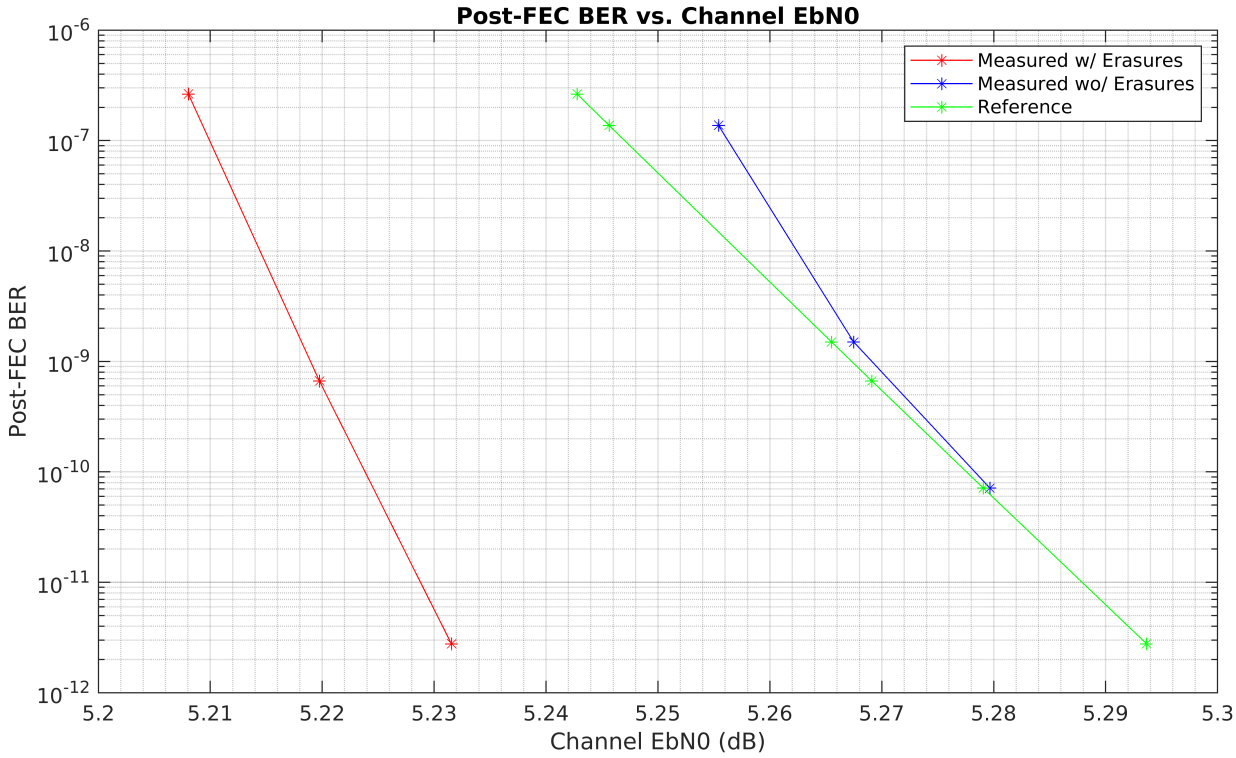

Figure 6.7: Output BER vs. Input E_bN_0

The performance improvement of the erasure decoder is measured as the change in the input BER for a given output BER. When compared to the reference G.709 implementation results, we find an erasure decoder improves coding gain by ~ 0.06dB for an output BER of 3×10^{-12}. A log-linear extrapolation shows the performance advantage of erasure decoding tends towards ~ 0.09dB at an output BER of 1×10^{-15}, shrinking the Shannon gap to 0.49dB.

Chapter 7: Conclusion

In this thesis, we presented a low complexity erasure decoder to improve the performance of a staircase decoder with no increase in complexity. The technique is particularly attractive for DCI applications, as it requires no additional power compared to an equivalent hard decision decoder. This form of erasure decoding shows a 0.06dB increase in coding gain when compared to bounded distance decoding for an output error rate of 3×10^{-12}. Using a log-linear extrapolation, we expect coding gain to increase to 0.09dB at 10^{-15}. The improved margin can extend the operational lifetime of a transceiver, or, alternatively, be used to increase the yield of transceivers by offsetting manufacturing variance in analog components.

The increase in decoding performance was achieved by finding both an optimal erasure threshold and location in the decoder window to apply erasure decoding. For the input error rate of interest, we found a threshold of 0.033 to be optimal, reducing iteration count and error rate. We found that using the erasure information to help decode blocks in the second slot of the window resulted in the best performance.

The optimization of both parameters was made tractable using the increase in decoder iterations as a metric of performance degradation. In addition, the use of iteration count as a metric allows a receiver to tune the parameters while in service.

The results were measured using a parallel software simulation achieving a real-time throughput of 50×10^6 bits per second on a consumer class computer. The high rate was achieved using efficient algorithms for Galois field arithmetic, an optimized algebraic BCH3 decoder, and a sophisticated decoding state machine. In addition to accelerating the simulation, many of the optimizations also reduce the hardware complexity of the staircase

decoder architecture.

7.1 Future Work

There are a couple of promising avenues for further research into the topic of erasure decoding of staircase codes. First, verifying that the extrapolated 0.09dB coding gain is present at the output error rate of 10^{-15}. This experiment will require considerable resources to either deploy equivalent Verilog codes to a cluster of FPGAs or obtain a prohibitive amount of time on a supercomputer to run the software implementation.

Second, a researcher may explore the effects of various correlated noise distributions on the performance of the proposed erasure decoder. With the long interleaving depth found in optical applications, the noise distribution at the decoder input is well modeled using AWGN. However, in low latency applications, the effects of correlated noise statistics need to be evaluated as the interleaver becomes shorter.

References

[1] ISO/IEC 18004:2015. Technical report, International Organization for Standardization, Geneva, CH, 2015.

[2] G.709.2: OTU4 long-reach interface. Recommendation G.709.2, International Telecomunication Union, September 2020.

[3] Implementation Agreement 400ZR. Standard, OIF, March 2020.

[4] N. H. Abel, L. Sylow, and S. Lie. Démonstration de l'impossibilité de la résolution algébrique des équations générales qui passent le quatrième degré. 2012.

[5] C. Argon and S.W. McLaughlin. An efficient Chase decoder for turbo product codes. *IEEE Transactions on Communications*, 52(6):896–898, June 2004.

[6] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate (Corresp.). *IEEE Transactions on Information Theory*, 20(2):284–287, March 1974.

[7] E. Berlekamp. Nonbinary BCH decoding (Abstr.). *IEEE Transactions on Information Theory*, 14(2):242–242, March 1968.

[8] E. R. Berlekamp, H. Rumsey, and G. Solomon. On the solution of algebraic equations over finite fields. *Information and Control*, 10(6):553–564, June 1967.

[9] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, March 1960.

[10] N. Boullis and A. Tisserand. Some optimizations of hardware multiplication by constant matrices. *IEEE Transactions on Computers*, 54(10):1271–1282, October 2005.

[11] H. Burton and E. Weldon. Cyclic product codes. *IEEE Transactions on Information Theory*, 11(3):433–439, July 1965.

[12] Yi Cai, Weiming Wang, Weifeng Qian, Jia Xing, Kai Tao, Junjie Yin, Shihua Zhang, Ming Lei, Erkun Sun, Hung-Chang Chien, Qun Liao, Ke Yang, and Huan Chen. FPGA Investigation on Error-Flare Performance of a Concatenated Staircase and Hamming FEC Code for 400G Inter-Data Center Interconnect. *Journal of Lightwave Technology*, 37(1):188–195, January 2019.

[13] SH Caldwell, J Capon, RA Silverman, Peter Elias, E Ferretti, JC Stoddard, RM Fano, EJ McCluskey Jr, FF Tung, and DA Huffman. Processing and Transmission of Information. Technical Report 37, Research Laboratory of Electronics (RLE) at the Massachusetts Institute of Techology (MIT), 1955.

[14] G. Castagnoli, S. Brauer, and M. Herrmann. Optimization of cyclic redundancy-check codes with 24 and 32 parity bits. *IEEE Transactions on Communications*, 41(6):883–892, June 1993.

[15] D. Chase. Class of algorithms for decoding block codes with channel measurement information. *IEEE Transactions on Information Theory*, 18(1):170–182, January 1972.

[16] R. Chien. Cyclic decoding procedures for Bose- Chaudhuri-Hocquenghem codes. *IEEE Transactions on Information Theory*, 10(4):357–363, October 1964.

[17] R. Chien, B. Cunningham, and I. Oldham. Hybrid methods for finding roots of a polynomial - With application to BCH decoding (Corresp.). *IEEE Transactions on Information Theory*, 15(2):329–335, March 1969.

[18] Chin-Long Chen. Formulas for the solutions of quadratic equations over GF(2m̂). *IEEE Transactions on Information Theory*, 28(5):792–794, September 1982.

[19] G. Colavolpe, G. Ferrari, and R. Raheli. Extrinsic information in iterative decoding: A unified view. *IEEE Transactions on Communications*, 49(12):2088–2094, December 2001.

[20] Baris Dogruoz, Giovanni Giobbio, Mark Nowell, Ray Nering, Anderson Tsai, Attila Aranyosi, Jeffery Maki, Hasan Ali, Chris Kapuscinski, Vivek Shah, Scott Sommers, Fadi Daou, Hani Daou, Burrell Best, and Newman Cheng. Optimizing QSFP-DD Systems to Achieve at Least 25 Watt Thermal Port Performance. White Paper, QSFP-DD MSA Group, January 2021.

[21] Michael H. Eiselt, Nicklas Eiselt, and Annika Dochhan. Direct detection solutions for 100G and beyond. In *2017 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3, March 2017.

[22] Mohammed El-Hajjar and Lajos Hanzo. EXIT Charts for System Design and Analysis. *IEEE Communications Surveys & Tutorials*, 16(1):127–153, 2014.

[23] P. Elias. Error-free Coding. *Transactions of the IRE Professional Group on Information Theory*, 4(4):29–37, September 1954.

[24] G.D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.

[25] Jürgen Freudenberger, Daniel Nicolas Bailon, and Malek Safieh. Reduced complexity hard- and soft-input BCH decoding with applications in concatenated codes. *IET Circuits, Devices & Systems*, 15(3):284–296, 2021.

[26] T. Fujiwara, T. Kasami, and S. Lin. Error detecting capabilities of the shortened Hamming codes adopted for error detection in IEEE Standard 802.3. *IEEE Transactions on Communications*, 37(9):986–989, September 1989.

[27] F. G. Gustavson. Analysis of the Berlekamp-Massey Linear Feedback Shift-Register Synthesis Algorithm. *IBM Journal of Research and Development*, 20(3):204–212, May 1976.

[28] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2:147–156, 1959.

[29] ITU-T. J.83 Digital multi-programme systems for television, sound and data services for cable distribution. Recommendation J.83, International Telecomunication Union, December 2007.

[30] Jorn Justesen, Knud J. Larsen, and Lars A. Pedersen. Error correcting coding for OTN. *IEEE Communications Magazine*, 48(9):70–75, September 2010.

[31] Yeong-hyeon Kwon, Mi-kyung Oh, and Dong-jo Park. Optimal erasure selection of M-ary PAM signaling for errors and erasures decoding algorithms. *IEEE Transactions on Communications*, 56(12):2071–2079, December 2008.

[32] Shu Lin and Daniel Costello. *Error Control Coding*. Pearson, Upper Saddle River, N.J, 2 edition edition, May 2004.

[33] Brainy Insights Pvt Ltd. Data Center Interconnect Market to Grow at CAGR of 16.03% through 2028. https://www.globenewswire.com/en/news-release/2023/04/05/2642128/0/en/Data-Center-Interconnect-Market-to-Grow-at-CAGR-of-16-03-through-2028-Rising-Demand-for-Disaster-Recovery-and-Data-Backup-Services-Propel-Growth-Says-The-Brainy-Insights.html, 4/5/2023 6:32:52 PM.

[34] Market and Markets. Data Center Interconnect Market Size, Global Industry Trends Forecast, Opportunities - 2030. Market Research SE 5486, 2022.

[35] J. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, January 1969.

[36] Eduardo F. Mateo. Cost/Bit Scaling Opportunities in Submarine Cables. In *2023 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3, March 2023.

[37] Graham H. Norton. On the complexity of finding shortest linear recurrences. 2002.

[38] Mark Nowell, Cisco Attila Aranyosi, Vu Le, Jeffery J Maki, Juniper Networks Scott Sommers, Tom Palkert, and Weiming Chen. QSFP-DD: Enabling 15 Watt Cooling Solutions. White Paper, QSFP-DD MSA Group, March 2018.

[39] W. Peterson. Encoding and error-correction procedures for the Bose-Chaudhuri codes. *IEEE Transactions on Information Theory*, 6(4):459–470, September 1960.

[40] W. Wesley Peterson and E. J. Weldon Jr. *Error-Correcting Codes, Second Edition*. The MIT Press, second edition edition, March 1972.

[41] Bipin Sankar Gopalakrishna Pillai, Behnam Sedighi, Kyle Guan, N. Prasanth Anthapadmanabhan, William Shieh, Kerry J. Hinton, and Rodney S. Tucker. End-to-End Energy Modeling and Analysis of Long-Haul Coherent Transmission Systems. *Journal of Lightwave Technology*, 32(18):3093–3111, September 2014.

[42] John G. Proakis. *Digital Communications*. McGraw-Hill Higher Education, Boston, 5th ed. edition, 2008.

[43] R.M. Pyndiah. Near-optimum decoding of product codes: Block turbo codes. *IEEE Transactions on Communications*, 46(8):1003–1010, August 1998.

[44] Chuan Qin, Binbin Guan, Kyle Edwards, Jian Kong, Ryan Morgan, Yawei Yin, Avinash Pathak, Mounika Banda, Sridharan J, Govardan Chandrababu, Jeetesh Jain, and Jamie Gaudette. Interoperable 400ZR Deployment at Cloud Scale. In *2023 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3, March 2023.

[45] Lukas Rapp and Laurent Schmalen. Error-and-Erasure Decoding of Product and Staircase Codes. *IEEE Transactions on Communications*, 70(1):32–44, January 2022.

[46] Lawrence Rennie. FEC for EFM: Introduction and Tutorial, October 2001.

[47] Andrew D. Shiner, Shahab Oveis Gharan, Michael Hubbard, and Kim B. Roberts. Assessing operating conditions of a receiver in a communication network based on forward error correction decoding properties, October 2022.

[48] Benjamin P. Smith, Arash Farhood, Andrew Hunt, Frank R. Kschischang, and John Lodge. Staircase Codes: FEC for 100 Gb/s OTN. *Journal of Lightwave Technology*, 30(1):110–117, January 2012.

[49] Atul Srivastava, Tom Williams, and David Lewis. Open ZR+ MSA. Technical Report 2.0, OpenZR+, July 2022.

[50] Jim Taylor. *Deep Space Communications*. John Wiley & Sons, Ltd, 1 edition, 2016.

[51] Stephen B. Wicker. *Error Control Systems for Digital Communication and Storage*. Prentice-Hall, Englewood Cliffs, NJ, us ed edition edition, February 1995.

[52] Stephen B. Wicker and Vijay K. Bhargava. Reed-Solomon Codes and the Exploration of the Solar System. In *Reed-Solomon Codes and Their Applications*, pages 25–40. IEEE, 1994.

[53] Kamil'Shamil'evich Zigangirov. Some sequential decoding procedures. *Rossiiskaya Akademiya Nauk. Problemy Peredachi Informatsii*, 2(4):13–25, 1966.