Zipper Codes: High-Rate Spatially-Coupled Codes with Algebraic Component Codes

by

Alvin Yonathan Sukmadji

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical & Computer Engineering
University of Toronto

Abstract

Zipper Codes: High-Rate Spatially-Coupled Codes with Algebraic Component Codes

Alvin Yonathan Sukmadji
Master of Applied Science
Graduate Department of Electrical & Computer Engineering
University of Toronto
2020

Zipper codes, a new framework for describing spatially-coupled product-like codes, are introduced. This framework encompasses many types of codes such as staircase codes and braided block codes. New types of codes such as tiled and delayed diagonal zipper codes are also introduced. Simulation results show that these new type of codes achieve comparable performance to staircase codes while requiring less memory. This thesis also analyzes the types of stall patterns that can arise in zipper codes and how they affect the error floor. Finally, the impact of error-and-erasure decoding in zipper codes is also studied. Software simulation results show that adding erasure symbols improves the coding gain by around 0.1 dB with only a small increase in memory overhead and decoding complexity.

Acknowledgements

I would like to thank Prof. Frank Kschischang for his guidance and supervision. His comments and insights are invaluable in this research. His passion and enthusiasm are infectious and always rekindle my interest in the subject area. This thesis would not have been possible without his patience and dedication.

I would also like to thank Umberto Martínez-Peñas for his collaboration in formulating zipper codes. His expertise and advice in mathematical writing were very helpful in developing formal definitions and proofs of theorems presented in this thesis.

I was very fortunate to have the opportunity to visit the Technical University of Munich for three months. I would like to express my gratitude to Prof. Dr.-Ing. Antonia Wachter-Zeh for hosting my visit there. I would also like to thank Lukas Holzbaur and Andreas Lenz for the discussions that we had about error-and-erasure decoding.

I would like to thank all other residents of BA4165: Reza Rafie Borujeny, Mohannad Shehadeh, Saber Rahbarfam, Qun Zhang, Amir Tasbihi, Masoud Barakatain, Bo Lian, and Susanna Rumsey, for the fun that we have had in the lab.

This research was partly funded by NSERC CGS-M, NSERC MSFSS, and Ontario Graduate Scholarship. I would like to acknowledge their financial support.

Finally, I would like to thank my family for their unconditional love and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

For a long time, the use of Forward Error Correction (FEC) in optical transport networks (OTNs) was ignored due to its high data integrity [2]. However, as data rates have increased, the effect of transmission impairments such as noise, dispersion, and nonlinear distortions have caused FEC to become a vital component of every optical communication system [3]. Since then, FEC systems for OTNs have been extensively researched and standardized in order to accomodate higher data rates.

The first standardized FEC for an OTN was in the ITU-T recommendation G.975. Released in 2000, the recommendation specifies a $(255, 239)$ Reed-Solomon (RS) code operating at 2.5 Gb/s throughput. The code is 3.77 dB away from capacity at 10^{-15} bit error rate (BER). In 2004, ITU-T recommendation G975.1 was released with nine proposed coding schemes, some of which are summarized in Table 1.1. Note that all schemes listed in Table 1.1 use hard-decision, algebraic codes. Further, they also use concatenated scheme, where two codes are used in tandem.

Table 1.1: Proposed Coding Schemes from ITU-T Recommendation G975.1

FEC Scheme	Gap to Capacity (dB)
$(3860, 3824)$ BCH outer, $(2040, 1930)$ BCH inner	0.98
$(1023, 1007)$ RS outer, $(2047, 1952)$ BCH inner	1.3
$(1901, 1855)$ RS outer, $(512, 502)\times$ $(510, 500)$ Hamming product inner	1.47

Then, the third-generation FEC such as turbo codes [4] and Low Density Parity Check (LDPC) codes [5,6] are also considered for optical networks. Those codes leverage iterative decoding to obtain realizable receivers and achieve high coding gains [7]. In

2012, staircase codes [8] were formulated. The recent 400ZR standard uses a concatenated staircase code and Hamming code [9] operating at 400Gb/s.

In almost two decades, the bit-rates of standardized OTNs have gone up 160-fold and 1 Tbit/s bit-rate is expected to be standardized within the next few years [10]. With high-throughput applications, such as internet protocol traffic and data centers, as well as very low bit error rate (BER) constraint of 10^{-15}, it is vital that the decoders of FECs can maintain data integrity while keeping the complexity of the system low.

In the past few years, there has been considerable interest in concatenated code systems, such as [11, 12]. These concatenated schemes achieve state-of-the-art error-correction performance while maintaining extremely low decoding complexity. The idea of concatenated code system for OTNs is that a very high-rate outer code is used in tandem with a lower-rate inner code. The inner code is used for correcting a portion of the errors up to some threshold, then the outer code will correct it further down to the 10^{-15} BER requirement. However, a very high-rate outer staircase code cannot be used for practical reasons such as the required memory for encoding and decoding. Figure 1.1 shows the staircase block size as a function of the code rate. As the code rate approaches one, the memory size grows explosively. Hence, we need to formulate code families that are memory-efficient to harness the full potential of the concatenated scheme.

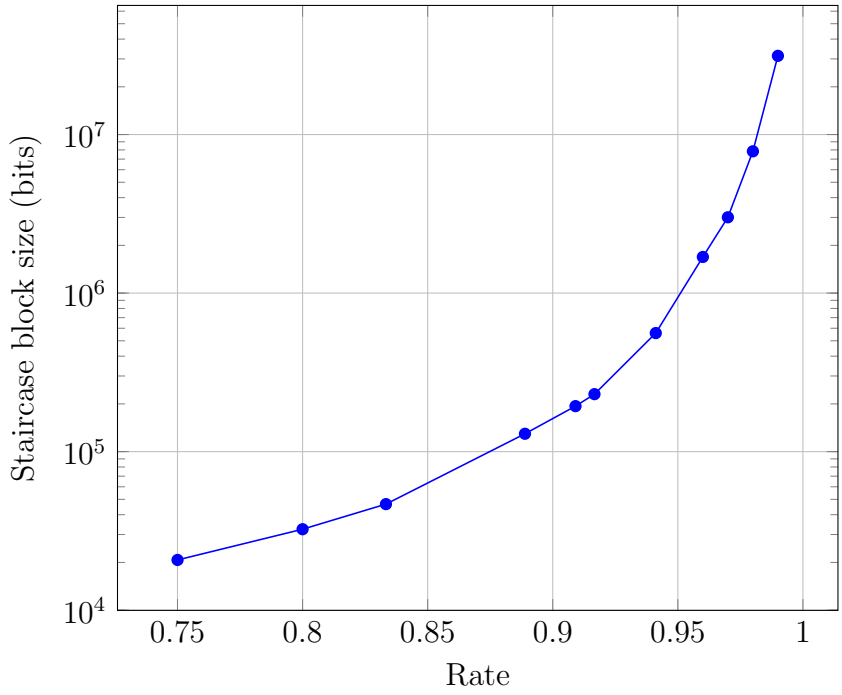

Figure 1.1: Staircase block size for quadruple-error-correcting constituent code, data points for rate < 0.95 taken from [1].

Since OTNs deal with a very high data throughput in the order of hundreds of gigabits per second, the FEC used must have low complexity. Often, the decoders for such FEC codes have a hard-decision decoding [13] as soft-decision decoding is shown to consume more than an order of magnitude more power than hard-decision decoding [14–16]. In addition, the codes often have a product-like structure akin to Elias's construction [17] or Wyner and Ash's recurrent code construction [18] (a form of spatial coupling), or a combination of both. Spatial coupling provide efficient encoding and decoding via locality, and they have universality property. This means that a single code construction performs well for various channel conditions and parameters [19, 20]. Indeed, spatially-coupled systems yield better performance than non-coupled ones.

In this thesis, we introduce zipper codes, a framework for describing spatially-coupled codes with iterative algebraic decoding. This framework encompasses many well-known codes such as staircase codes [8], braided block codes [21], and swizzle codes [22]. We also introduce new types of code family called tiled diagonal zipper codes and delayed diagonal zipper codes, that are more memory-efficient than staircase codes while having comparable error-correcting performance.

1.2 Literature Review

We provide brief descriptions of some families of spatially-coupled and product-like codes below.

- Recurrent codes [18] are a type of code inspired by convolutional codes. They are described by a parity check matrix

$$A = \begin{pmatrix} B_0 & B_m & B_{2m} & \dots \end{pmatrix},$$

  where B_0 is a semi-infinite matrix with infinitely many rows and b columns, m is a positive integer, and B_k is B_0 shifted down by k rows with the first k rows filled with zeros for all positive integer k. The codes are shown to be effective for burst or random error correction.

- LDPC convolutional codes [23, 24] are a type of LDPC codes where the parity check matrix is banded similar to recurrent codes. The codes are decoded using a sliding window decoder which allows pipelined implementation. The performance is substantially better than convolutional codes while having same complexity. The code has also been shown to be asymptotically capacity-achieving [25,26]. However,

they are using soft-decision codes, which are significantly more complex than hard-decision.

- 'Half'-product codes are discussed in [27] as a variant of product codes by removing the lower triangular part of the regular product code matrix. They have shorter overall block length, only half of that of regular product code, but they have worse error-correcting performance.

- Staircase codes [8] are characterized by having an infinite sequence of matrices of size $m \times m$: $B_0, B_1, \ldots$ such that each row of $\left(B_i \; B_{i+1}^T\right)$ is a codeword of some constituent code of length $2m$. The error-correcting performance of staircase codes surpassed the performance of the codes listed in the Appendix I of the ITU-T recommendation G975.1 when it was introduced. The codes are 0.56 dB away from the Shannon limit. As mentioned above, the codes have been adopted as part of the 400ZR standard.

- Braided block codes (BBC) [21] operate on continuous data stream and are composed of an interconnection between two constituent codewords, namely vertical and horizontal codewords, in an infinite, two-dimensional array. Braided BCH codes, a variant of BBC with Bose-Chaudhuri-Hocquengham (BCH) component codes, have exhibited coding gain of 9.35 dB at 7% redundancy [19].

More review of historical and recent development of FECs for OTNs can be seen in [28–30].

1.3 Outline of the Thesis

The organization of the thesis is as follows:

1. Chapter 2 starts with a mathematical description of zipper codes along with some properties that the codes have. We then provide examples of some well-known codes that zipper codes subsume as well as introduce new types of codes: tiled diagonal and delayed diagonal zipper codes. We describe the decoding procedure of zipper codes and some assumptions with regards to decoding properties. We then provide a representation of zipper codes as a graph. Finally, we conclude the chapter with the simulation results of high-rate codes.

2. Chapter 3 includes a description of a "stall pattern" as well as some patterns that can arise from particular zipper code structures. We describe in detail the

consequences of some code properties described in Chapter 2 in the context of stall patterns that they produce. We also examine in detail the stall patterns of the new types of codes that are introduced in Chapter 2.

3. Chapter 4 introduces the concept of error-and-erasure decoding for zipper codes. We show that declaring erasure symbols helps the decoding procedure via simulation results. The data flow and decoding complexity that arise from error-erasure decoding will be examined. In particular, we show that decoding performance can be improved with small additional complexity. An analysis of error-and-erasure stall patterns is briefly discussed.

The author acknowledges Umberto Martínez-Peñas for his collaboration in formulating some parts of the thesis listed below:

- Examples of zipper code formulation of well-known codes in Chapter 2.
- Examples of graph representation of "stall patterns" in Chapter 3.

Chapter 2

Zipper Codes

In this chapter, we introduce zipper codes, a new framework for describing spatially-coupled product-like codes. Zipper codes are a member of the class of Generalized Low-Density Parity-Check Codes [31]; however they focus on product-like codes where every variable node has degree two. Such codes seems to be the case of most practical interest in the high-rate regime [32]. This framework includes codes described in [8, 21, 22, 33–35].

2.1 Code Structure

In this section, we will describe the two major components of a zipper code: a *zipping pair* and an *interleaver map*.

2.1.1 Zipping Pair

A zipper code is composed of a sequence of codewords

$$c_0, c_1, \ldots, \tag{2.1}$$

where each c_i is a codeword of a constituent code $\mathcal{C}(n, k, d)$. Here, n, k, d respectively denote the block length, dimension, and minimum Hamming distance of $\mathcal{C}$. We also define the sequence (2.1) to be a *buffer*. Denote the (i, j)-th entry of a buffer to be the j-th entry of the i-th codeword, where $(i, j) \in \mathbb{Z} \times [n]$ and

$$[n] = \{0, 1, \ldots, n-1\}.$$

Now, let $m_0, m_1, \ldots$ be a sequence of integers such that for all i,

$$0 \leq m_i \leq k.$$

Denote $A_i = \{(i,j) : j \in [m_i]\}$, $B_i = \{(i,j) : j \in [n] \setminus [m_i]\}$, and

$$A = \bigcup_i A_i, \quad B = \bigcup_i B_i.$$

We call A the *virtual set* and B the *real set*. Together, we refer to the pair (A, B) as a *zipping pair*. The parameter m_i is the width of virtual buffer at the i-th codeword.

For a buffer with zipping pair (A, B), we denote its entries by

$$c_{A_i} = (c_{i,0}, \ldots, c_{i,m_i-1}) \quad \text{and} \quad c_{B_i} = (c_{i,m_i}, \ldots, c_{i,n-1}).$$

Together, we have $c_i = (c_{A_i}, c_{B_i}) \in \mathcal{C}$ to be the i-th codeword, and denote the virtual and real buffer to be the sequences $\{c_{A_i}\}$ and $\{c_{B_i}\}$, respectively.

A visualization of a zipper code is shown in Figure 2.1. Note that from the visualization, we will also sometimes denote the i-th constituent codeword as the i-th row. Further, we also sometimes call the (i,j)-th entry of the buffer to be the entry at the i-th row and j-th column.

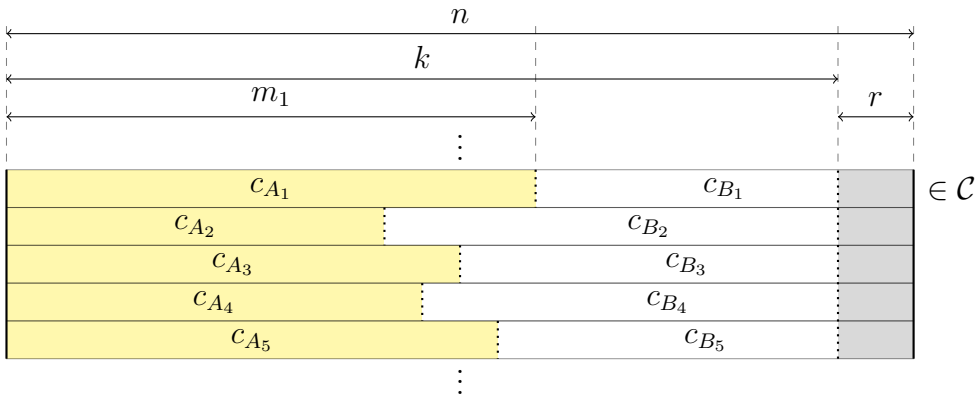

Figure 2.1: Zipper code with $c_i = (c_{A_i}, c_{B_i})$ being a codeword of $\mathcal{C}$ for all i. The first m_i entries of each constituent codeword belong to the virtual buffer and the remaining $n - m_i$ entries (including the r parity symbols) belong to the real buffer.

2.1.2 Interleaver Map

An interleaver map ϕ is defined as a function

$$\phi : A \to B$$

associating each bit in the virtual set with a bit in the real set. We require that every symbol in the virtual buffer be a direct copy of its associated symbol in the real buffer. In other words, in a valid codeword, for every $(i, j) \in A$, we have

$$c_{i,j} = c_{\phi(i,j)}. \tag{2.2}$$

Encoding Procedure

The encoding procedure of the i-th constituent code of a zipper code is as follows:

1. Fill in $c_{i,m_i}, c_{i,m_i+1}, \ldots, c_{i,n-r-1}$ with new message symbols.
2. Fill in c_{A_i} by duplicating the symbols from the locations prescribed by the interleaver map, that is, for each $j \in A_i$, $c_{i,j} = c_{\phi(i,j)}$.
3. Compute the parity symbols $c_{i,n-r}, c_{i,n-r+1}, \ldots, c_{i,n-1}$ with $(c_{i,0}, c_{i,1}, \ldots, c_{i,n-r-1})$ as the information symbols.

Properties of Interleaver Maps

We define a projection map

$$\begin{aligned} \pi_1 : A \cup B &\to \mathbb{Z} \\ (i, j) &\mapsto i, \end{aligned}$$

and for all $a \in A$, define $\phi_1(a)$ to be a shorthand notation of $\pi_1(\phi(a))$. For practical purposes, we will only consider interleaver maps ϕ that are *bijective*, *periodic*, and *causal*. The precise definition of each property is described below.

Definition 1. An interleaver map ϕ is *bijective* if there exists a map

$$\phi^{-1} : B \to A$$

such that for any arbitrary $a \in A$ and $b \in B$, we have $\phi^{-1}(\phi(a)) = a$ and $\phi(\phi^{-1}(b)) = b$. Such a ϕ^{-1} is called the *inverse map* of ϕ.

Definition 2. Let ϕ be an interleaver map such that $\phi(i,j) = (i',j')$. We say that ϕ is periodic with period ν if

$$\phi(i+\nu, j) = (i'+\nu, j)$$

for all $(i,j) \in A$.

Observe that the periodicity of an interleaver map implies periodicity of the code structure and zipping pair. If an interleaver map is periodic with period ν, then $m_{i+\nu} = m_i$ for all $i \in \mathbb{Z}_{\geq 0}$.

Definition 3. An interleaver map ϕ is *causal* if $\phi_1(i,j) < i$ for all $(i,j) \in A$. Conversely, ϕ is *anti-causal* if $\phi_1(i,j) > i$ for all $(i,j) \in A$.

Our running assumption from this point on is that all interleaver maps ϕ are bijective. We will now introduce a few shorthand notations. For all $A' \subseteq A$, denote

$$\phi(A') = \{\phi(a) : a \in A'\},$$

and similarly, for all $B' \subseteq B$, denote

$$\phi^{-1}(B') = \left\{\phi^{-1}(b) : b \in B'\right\}.$$

Finally, for all $X \subseteq A \cup B$, denote

$$\pi_1(X) = \{\pi_1(x) : x \in X\}.$$

Before we define a "scattering" interleaver map, observe that for all row i, the constituent code c_i is composed of real symbols in positions $B_i \cup \phi(A_i)$. For two distinct rows i, j define the positions of the symbols in the real set that are shared between the two codewords to be the intersections between rows i and j, denoted by

$$\chi(i,j) = (B_i \cup \phi(A_i)) \cap (B_j \cup \phi(A_j)).$$

If $\chi(i,j)$ is nonempty, we say that that rows i and j intersect. The expression above can be manipulated as follows:

$$\begin{aligned}\chi(i,j) &= (B_i \cup \phi(A_i)) \cap (B_j \cup \phi(A_j)) \\ &= (B_i \cap B_j) \cup (B_i \cap \phi(A_j)) \cup (B_j \cap \phi(A_i)) \cup (\phi(A_i) \cap \phi(A_j))\end{aligned}$$

Note that A_i, B_i, A_j, and B_j are disjoint. Also, $\phi(A_i) \cap \phi(A_j) = \varnothing$ since ϕ is bijective. Thus, the resulting expression that we are left with is

$$\chi(i,j) = (B_i \cap \phi(A_j)) \cup (B_j \cap \phi(A_i)) \,. \tag{2.3}$$

The expression above implies that the intersections between two distinct rows i, j can only happen at rows i and j. Also, $(B_i \cap \phi(A_j))$ and $(B_j \cap \phi(A_i))$ are disjoint since B_i and B_j are disjoint. Thus, the cardinality of the intersection between rows i and j is

$$|\chi(i,j)| = |B_i \cap \phi(A_j)| + |B_j \cap \phi(A_i)|.$$

Remark 1. If ϕ is causal and $i < j$, $\phi_1(a) < i$ for all $a \in A_i$, and so $B_j \cap \phi(A_i)$ is empty. Hence, we have $\chi(i,j) = B_i \cap \phi(A_j)$.

We will now state the definition of a scattering interleaver map.

Definition 4. Let ϕ be a bijective interleaver map. We say that ϕ *scatters* if for all $i \in \mathbb{Z}_{\geq 0}$, $i \notin \phi_1(A_i)$, and for all $j \neq i$, $|\chi(i,j)| \leq 1$.

In other words, any two constituent codewords have at most one bit in common. Also, no entry in the virtual component of any row is a direct copy of a real component in the same row, and if two codewords from distinct rows i, j intersect, they intersect at exactly one point in either B_i or B_j. Finally, define the set of rows reachable from row i, or the neighbourhood of i, to be

$$\rho(i) = \phi_1(A_i) \cup \phi_1^{-1}(B_i). \tag{2.4}$$

We define row j to be reachable from row i, or j to be a neighbour of i, if $j \in \rho(i)$.

Remark 2. Observe that for $j \neq i$, it immediately follows from (2.3) and (2.4) that $j \in \rho(i)$ if and only if $\chi(i,j) \neq \varnothing$. Also, $j \in \rho(i)$ if and only if $i \in \rho(j)$ due to the bijectivity of the interleaver map.

We now have three equivalent formulations of a scattering interleaver map

Theorem 1. *Let ϕ be a bijective interleaver map. The following are equivalent.*

1. *ϕ scatters.*
2. *For all $i \in \mathbb{Z}_{\geq 0}$, $i \notin \rho(i)$ and $|\rho(i)| = n$.*
3. *All of the following conditions are satisfied for all $i \in \mathbb{Z}_{\geq 0}$:*

(a) For all $a \in A_i$, $\phi_1(a) \neq i$,

(b) For all $a \in A_i$ and $b \in B_i$, $\phi_1(a) \neq \phi_1^{-1}(b)$,

(c) For all distinct $a, a' \in A_i$ and $b, b' \in B_i$, $\phi_1(a) \neq \phi_1(a')$ and $\phi_1^{-1}(b) \neq \phi_1^{-1}(b')$.

Proof. First, note that due to the bijectivity of ϕ, for all $i \in \mathbb{Z}_{\geq 0}$, $i \in \phi_1(A_i)$ if and only if $i \in \phi^{-1}(B_i)$. It immediately follows from the definition of ρ that

$$i \notin \rho(i) \Leftrightarrow i \notin \phi_1(A_i) \Leftrightarrow \phi_1(a) \neq i \text{ for all } a \in A_i.$$

For the remainder of the proof, we will assume that $i \notin \rho(i)$. Observe that the number of neighbours that row i has is given by

$$\begin{aligned} |\rho(i)| &= |\phi_1(A_i) \cup \phi_1^{-1}(B_i)| && (2.5)\\ &\leq |\phi_1(A_i)| + |\phi_1^{-1}(B_i)| && (2.6)\\ &\leq m_i + (n - m_i) && (2.7)\\ &= n. && (2.8) \end{aligned}$$

(1) $\Leftrightarrow$ (2): If $|\rho(i)| < n$, then by pigeonhole principle there exists a $j \in \rho(i)$ such that $|\chi(i,j)| > 1$. Conversely, if $|\chi(i,j)| > 1$ for some $j \in \rho(i)$, then at least one of (2.6) and (2.7) will not achieve equality. Thus, $|\rho(i)| = n$ if and only if ϕ scatters.
(2) $\Leftrightarrow$ (3): The equality (2.6) is achieved if and only if $\phi_1(A_i)$ and $\phi_1^{-1}(B_i)$ are disjoint. Similarly, the equality (2.7) is achieved if and only if for every distinct $a, a' \in A_i$, $\phi_1(a) \neq \phi_1(a')$ and for every distinct $b, b' \in B_i$, $\phi_1^{-1}(b) \neq \phi_1^{-1}(b')$. □

The scattering property is important for an interleaver map design and will be discussed in detail in Chapter 3.

2.2 Code Properties

This section describes the properties of zipper codes. Given the construction in Section 2.1, we consider the *encoder memory size, rate,* and *initialization.*

2.2.1 Encoder Memory Size

All symbols in the virtual buffer are copies of some earlier symbols in the real buffer. We need to store some old, previously encoded symbols in order to encode future rows. We define the encoder memory size to be the number of old real symbols that the encoder

stores in order to encode future rows. In order to encode row i, we require that the interleaver map can recover every entry of c_{A_i} by looking it up in the encoder memory.

For all $i \in \mathbb{Z}_{\geq 0}$ and $j \in [m_i]$, denote $\delta_{i,j} = i - \phi_1(i,j)$ as the number of row lookbacks needed for the (i,j)-th entry. Now denote $m_{\max} = \max_i\{m_i\}$ as the maximum virtual buffer width. We will state a theorem that describes the minimum size of an encoder memory.

Theorem 2. *Suppose that ϕ is a bijective causal scattering interleaver map. Then the minimum encoder memory size is*

$$\frac{m_{\max}(m_{\max}+1)}{2}.$$

Proof. Let $i \in \mathbb{Z}_{\geq 0}$ be fixed. The encoder memory for row i is given by

$$M_i = \sum_{j=0}^{m_i-1} \delta_{i,j}.$$

Without loss of generality, let $0 < \delta_{i,0} < \delta_{i,1} < \ldots < \delta_{i,m_i-1}$. Then, observe that

$$\begin{aligned}
\delta_{i,0} &\geq 1, \\
\delta_{i,1} &\geq \delta_{i,0} + 1 \geq 2, \\
&\vdots \\
\delta_{i,m_i-1} &\geq \delta_{i,m_i-2} + 1 \geq m_i.
\end{aligned}$$

Hence, the encoder memory size of row i is

$$M_i = \sum_{j=0}^{m_i-1} \delta_{i,j} \geq \sum_{j=1}^{m_i} j = \frac{m_i(m_i+1)}{2}.$$

Finally, the minimum encoder memory size of the overall zipper code has to be at least as large as the encoder memory of the row with the largest virtual buffer width, i.e.,

$$\frac{m_{\max}(m_{\max}+1)}{2}.$$

□

For simplicity in the calculation of the encoder memory, we can assume that the encoder stores old (real) rows, not symbol-per-symbol. In addition to c_{B_i}, the encoder memory stores $c_{B_{i-1}}, c_{B_{i-2}}, \ldots, c_{B_{i-\lambda}}$, where λ is defined to be the maximum lookback parameter, i.e.,

$$\lambda = \max_{i,j} \delta_{i,j}.$$

We require the encoder memory to store as many as $(\lambda + 1)m_{\max}$ symbols.

2.2.2 Code Rate

Suppose that the interleaver map of the zipper code we are interested in is bijective and periodic with period ν. The rate of zipper code is given by the fraction of the number of information symbols in the real buffer over one period:

$$R = \frac{\sum_{i=0}^{\nu-1}(n - m_i - r_i)}{\sum_{i=0}^{\nu-1}(n - m_i)} = 1 - \frac{\sum_{i=0}^{\nu-1} r_i}{\sum_{i=0}^{\nu-1}(n - m_i)} = 1 - \frac{\bar{r}}{n - \bar{m}}, \tag{2.9}$$

where

$$\bar{r} = \frac{1}{\nu}\sum_{i=0}^{\nu-1} r_i \quad \text{and} \quad \bar{m} = \frac{1}{\nu}\sum_{i=0}^{\nu-1} m_i.$$

Now since the interleaver map is assumed to be bijective, the sizes of virtual and real components have to be equal, and so we have

$$\sum_{i=0}^{\nu-1} m_i = \sum_{i=0}^{\nu-1}(n - m_i), \tag{2.10}$$

which implies $n = 2\bar{m}$. Hence, (2.9) simplifies to

$$R = 1 - \frac{2\bar{r}}{n} = 1 - \frac{\bar{r}}{\bar{m}}.$$

Remark 3. Observe that the number of symbols in the buffer over one period is νn. Also, from (2.10), we have

$$\nu n = 2\sum_{i=0}^{\nu-1} m_i.$$

This implies that νn is even, and thus we require at least one of ν or n to be even.

2.2.3 Initialization

Assuming that the information symbols start to fill in at time $i = 0$, we will initialize by setting $c_i = 0$ for $i < 0$.

2.3 Examples

This section describes some examples of related spatially-coupled codes that can be described in the zipper code framework.

2.3.1 Staircase Codes

Staircase codes [8] are characterized by having an infinite sequence of matrices of size $m \times m$: $B_0, B_1, \ldots$ such that each row of $\left(B_i \; B_{i+1}^T\right)$ is a codeword of some constituent code of length $2m$.

In our formulation of staircase codes, the width of both virtual and real buffers is m for all rows and so $n = 2m$. The interleaver map for staircase codes is periodic with period m and it is characterized by the transposition of the i-th staircase block, i.e.,

$$\phi(mi + r, j) = (m(i-1) + j, m + r)\,,$$

where $r \in [m]$. To encode a row of the current staircase block, the encoder memory also needs to store one previous staircase block. Thus, the encoder needs to store $m^2 + m$ symbols.

A graphical representation of staircase codes in the form of a zipper code is shown in Figure 2.2.

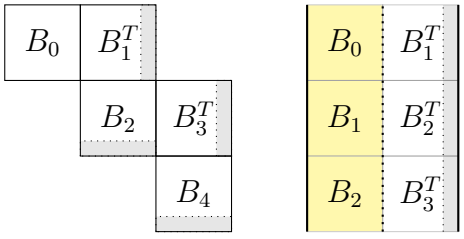

Figure 2.2: Staircase code (left) and its graphical representation in the form of a zipper code (right).

2.3.2 Braided Block Codes

Braided block codes (BBC) [21] operate on continuous data stream and are composed of an interconnection between two constituent codewords, namely vertical and horizontal codewords, in an infinite, two-dimensional array. To illustrate the concepts of braided block codes, Figure 2.3 shows a side-by-side comparison on how the rate 1/7 tightly braided code from [21] can be represented in our formulation. Without loss of generality, we assume that for a given information symbol, we encode its row (even-numbered) constituent codeword first, then its column (odd-numbered) codeword. The width of the virtual buffer alternates between 3 and 4 depending on which constituent codeword is being encoded. Also, observe that the even-numbered rows are the only rows containing information symbols, while the odd-numbered rows only contain parity symbols. The information symbol in the even-numbered row is copied directly below it in the virtual buffer. Hence, the interleaver map is defined as

$$\phi(i,j) = \begin{cases} (i+2j-5, 6-j) & \text{for } i \text{ even}, \\ (i-2j-3, 4+j) & \text{for } i \text{ odd}, \ j \neq 3, \\ (i-1, 3) & \text{for } i \text{ odd}, \ j = 3. \end{cases}$$

To encode a constituent codeword, we need to look back as far as the rows containing the three preceding information symbols. Thus, $\lambda = 7$, and the encoder memory has to store up to $(3 \times 7) + 4 + 3 = 28$ symbols.

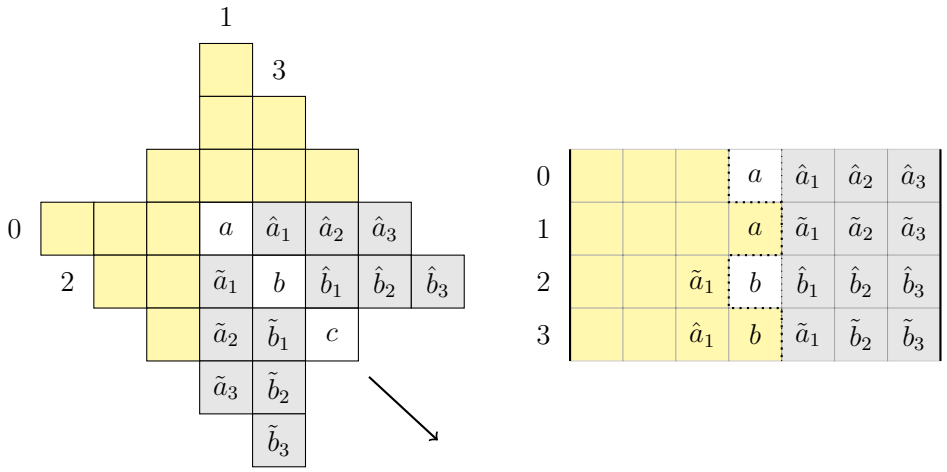

Figure 2.3: Graphical representation of tightly braided code with $(7, 4)$ Hamming component code (left) and its representation in the form of a zipper code (right). The numbers on the side of each diagram represent the constituent codeword number.

2.3.3 Diamond Codes

Diamond codes [34] are also a subfamily of zipper codes. Let $\mathcal{C}_1, \mathcal{C}_2$ be two constituent codes of length n. Define $n = 2m$ and choose the zipping pair (A, B) given by $A = \mathbb{Z} \times [m]$ and $B = \mathbb{Z} \times [n] \setminus [m]$. Next define

$$\phi(i, j) = (-i - s, m + j),$$

where the shifting parameter $s \in \mathbb{Z}_{>0}$ can be chosen freely. The map ϕ scatters and is causal and periodic with period $\nu = m + s - 1$. The zipper code with these parameters and constituent code $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2$ forms a general diamond code. See also Figure 4 of [34].

2.3.4 Swizzle Codes

Swizzle codes [22] are a family of spatially-coupled block code comprised of an information array I_t of size $w \times k/2$, a parity array Z_t of size $w \times (n-k)$, and a parity-on-parity array Z'_t of size $p \times \ell$. We select the parameters such that $w > k/2$ and $w(n-k) \leq p(n-\ell)$.

Figure 2.4 shows the formulation of swizzle codes in our scheme. The period of the buffer is $\nu = w + p$. The width of the virtual buffer is $k/2$ for the first w rows and $n - \ell$ for the last p rows. Also, observe the irregularity in the number of parity symbols. Each of the first w rows has $n - k$ parity symbols and each of the last p rows has ℓ parity symbols.

The interleaver map for I_t^* is defined as a diagonal-like mapping in I_t, skipping the current row as shown in Figure 2.4. On the other hand, the "inverse" interleaver map is designed so that Z_t^* contains a rearrangement of Z_t. If $w(n-k) < p(n-\ell)$, then we fill in the remaining symbols in Z_t^* with zero.

Note that since a swizzle code is a type of block code, the notion of the interleaver map differs from the previously discussed codes. First, the interleaver map is not bijective since the symbols in Z'_t are not replicated anywhere else. Also, the interleaver map for I_t^* is not causal since we are reading from "future" rows.

To make the interleaver map bijective, we can impose another constraint to the code parameters. Suppose that $w(n-k) + \ell p \leq p(n-\ell)$, then we can fit the symbols in the previous parity-on-parity array Z'_{t-1} into Z_t^*, and so the lookback that we need to do is at most $w + 2p - 1$.

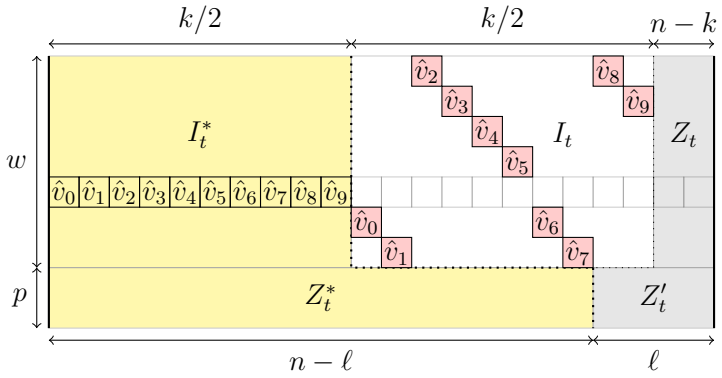

Figure 2.4: The formulation of swizzle codes into zipper codes. The shaded tiles in I_t denote the corresponding symbol in the real buffer for the marked tiles in I_t^*.

2.3.5 Tiled Diagonal Zipper Codes

Tiled diagonal zipper codes are characterized by their "tile-like" structure. We first fix positive integers L, w, then let $m = wL$ and $n = 2m = wL$. Now, we represent the row and column coordinates in the form of $wq + i$ and $ws + j$ for $q \in \mathbb{Z}$, $s \in [2L]$, and $i, j \in [w]$. Denote a submatrix of C,

$$T_{q,s} = \begin{pmatrix} c_{0,0}^{(q,s)} & \cdots & c_{0,w-1}^{(q,s)} \\ \vdots & \ddots & \vdots \\ c_{w-1,0}^{(q,s)} & \cdots & c_{w-1,w-1}^{(q,s)} \end{pmatrix}$$

of size $w \times w$, where $c_{i,j}^{(q,s)}$ is a compact notation of $c_{wq+i,ws+j}$. We call $T_{q,s}$ a *tile* in the q-th row and s-th column. If $s < L$, then we call such tile a *virtual tile*. Otherwise, we call it a *real tile*. For $s \in [L]$, we would like to have $T_{q,s} = T_{q-s-1,L+s}^T$, and so the interleaver map is defined as

$$\phi(wq + i, ws + j) = (w(q - s - 1) + j, w(L + s) + i). \tag{2.11}$$

Figure 2.5 shows an example of a tiled diagonal zipper code with $L = 3$.

Note that if $w = 1$, the formulation is identical to continuously interleaved BCH (CI-BCH) codes [35]. On the other hand, if $L = 1$, then the scheme is identical to staircase codes.

The encoder has to store

$$w^2 + 2w^2 + \ldots + Lw^2 = L(L + 1)w^2/2$$

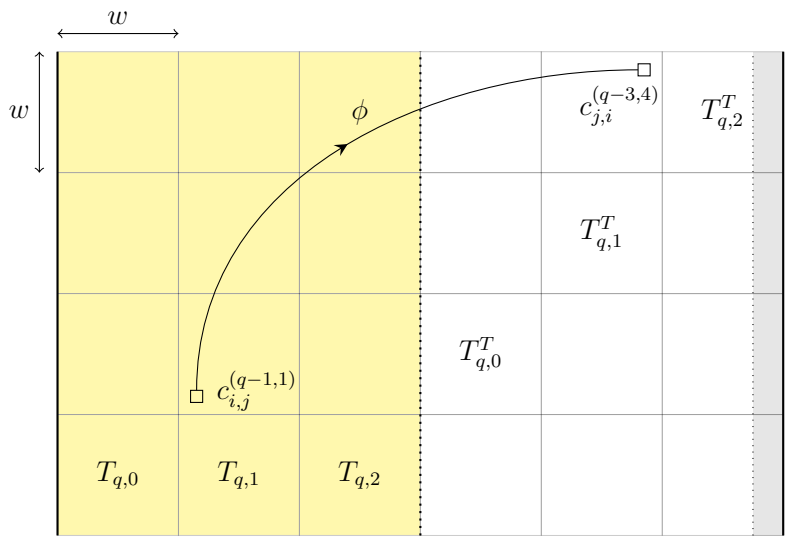

Figure 2.5: Tiled diagonal zipper code with $L = 3$, tile size $w \times w$, and interleaver map as described in (2.11).

symbols for lookback plus additional wL symbols for the current row. Observe that if $w = 1$, the encoding memory size lower bound is achieved.

2.3.6 Delayed Diagonal Zipper Codes

Delayed diagonal zipper codes are variants of tiled diagonal zipper codes with $w = 1$ and with added 'delay' to the interleaver map. More specifically, the interleaver map is given by

$$\phi_\delta(i, j) = (i - j - \delta, j + m),$$

where $\delta \in \mathbb{Z}_{>0}$ is the delay parameter. Observe that if $\delta = 1$, the interleaver map is identical to the interleaver map of CI-BCH codes. The encoder memory of delayed diagonal zipper codes is

$$\delta + (\delta + 1) + \ldots + (\delta + m - 1) = \delta m + \frac{m(m-1)}{2}.$$

Similar to tiled diagonal zipper codes, the encoding memory size lower bound is achieved if $\delta = 1$. Figure 2.6 shows an example of delayed diagonal zipper code with $m = 8$.

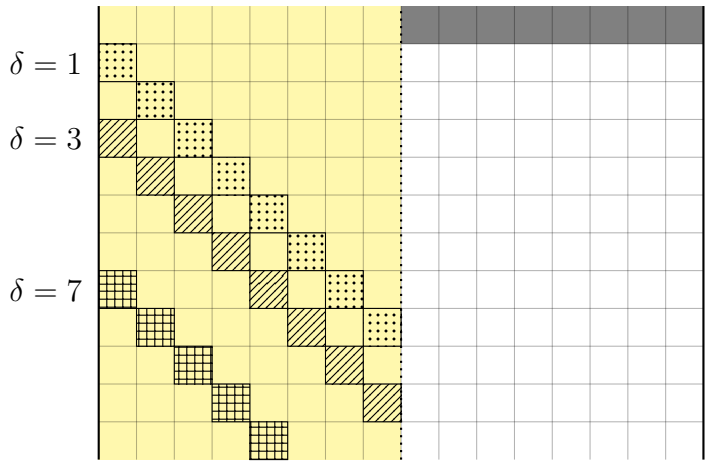

Figure 2.6: Delayed diagonal zipper code with $m = 8$ and different delay values δ.

2.4 Decoding

2.4.1 Transmission Procedure and Channel Model

We transmit only the real symbols since all symbols in the virtual buffer are copies of previously transmitted symbols. The symbols are transmitted in the order they are encoded, i.e., if $i < j$, then c_{B_i} gets transmitted before c_{B_j}. However, we can also similarly transmit the symbols in terms of chunks. In other words, for some positive integer μ, we transmit μ rows of real symbols,

$$c_{B_i}, c_{B_{i+1}}, \ldots, c_{B_{i+\mu-1}}$$

all at once instead of row-per-row. We transmit the symbols over binary memoryless symmetric channel with some crossover probability, independently and identically distributed for all transmitted (real) symbols.

2.4.2 Decoding Procedure

We decode using a sliding-window decoder on M consecutive received rows of (possibly erroneous) symbols. Now suppose that the constituent code $\mathcal{C}_1$ can correct up to t_i errors. The decoding procedure for an incoming row of data c'_{B_i} is described as follows:

1. Compute and store the syndrome of $\left(c'_{A_i}, c'_{B_i}\right)$, where c'_{A_i} is obtained by looking up the decoding window in the same manner as (2.2). If the syndrome of c'_i is zero, then we set the flag for that syndrome to 'stale.' Otherwise, set the flag to 'fresh.'

2. Decode each 'fresh' syndrome in the decoding window using the decoder for $\mathcal{C}$.

(a) If the decoding is successful, the decoder will output $t_i' \leq t_i$ error locations. For each of the t_i' locations, update the codeword and syndrome for both affected codewords. If the new syndrome of a particular codeword becomes zero, then we set the flag to 'stale.' Otherwise, we set it to 'fresh.'

(b) If the decoding fails, we set the flag to 'stale.'

3. If there is any syndrome with 'fresh' flag and if maximum allowed number of iteration is not exceeded yet, return to step 2.

We assume for now that the decoder is serial, i.e., only one codeword-syndrome pair is updated at any given time, in order to prevent conflicts and race conditions during read/write operations on the codewords and syndromes. Note that we can also precompute the syndromes for future rows. Suppose that $\phi^{-1}(i,j) = (i^*, j^*)$. If $c'_{i,j} \neq 0$, we update the syndromes for row i^* with the j^*-th column of the parity check matrix of $\mathcal{C}$. This way, when a future row of data c_{B_ℓ} $(\ell > i)$ arrives, we only need to update the syndrome of row ℓ with the syndrome of $(0, c_{B_\ell})$.

The decoding subroutine can potentially be computationally expensive, so in the implementation we can add a scheduler so that the iterative decoding subroutine will be invoked every time the receiver receive μ rows of symbols instead of every row.

We will now determine the memory size for the decoder. The number of symbols in the decoding window is Mm_B^*. However, the decoder also has to store some older symbols for decoding. The number of such additional symbols is λm_B^*, where λ is the maximum lookback parameter defined in Section 2.2.1. Therefore, the decoder must be able to contain as many as $(M + \lambda)m_B^*$ symbols. The decoder must also be able to store additional $(M + \lambda)r$ symbols to store syndromes for the codewords in the decoding window and future rows.

The values of M for some of the codes mentioned in Section 2.3 are also discussed. For staircase codes with staircase block of size $m \times m$ and a decoding window that fits L staircase blocks, we can let $M = mL$. Then, for braided block codes with I horizontal decoders and I vertical decoders, we let $M = 2I$. Finally, for swizzle codes, we let $M = w + p$. However, a larger M can be used assuming we modify the formulation of swizzle code as described in Section 2.3.4.

The whole structure of a zipper decoder is summarized in Figure 2.7.

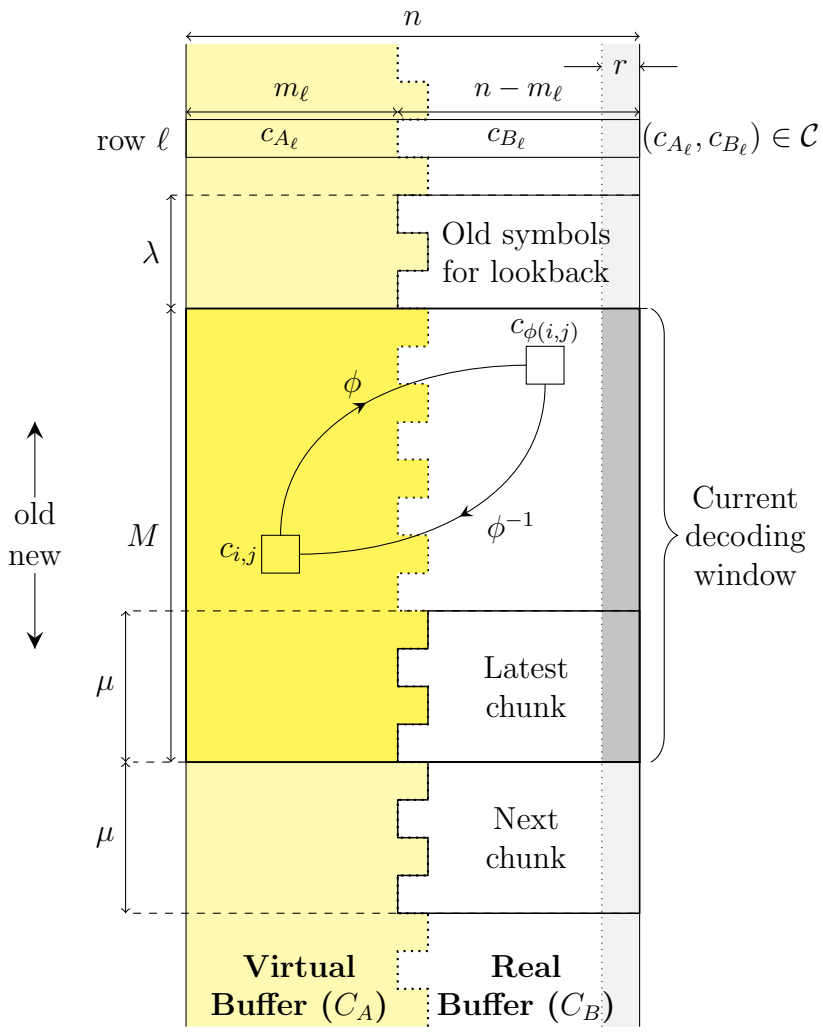

Figure 2.7: Diagram of a zipper decoder with periodic virtual buffer width (period $\nu = 2$).

2.4.3 Decoding Complexity

A few factors that can determine decoding complexity are as follows:

1. **Hard or soft decision constituent codes:** In general, soft decision codes can achieve better error correcting performance than hard decision codes, but their decoding procedure is more complex with high in-chip data flows [36]. Thus, soft-decision constituent codes are not ideal for high-throughput applications such as optical communications.

2. **Decoding frequency:** Figure 2.8 shows the cumulative frequency of constituent decoding as a function of row index of the decoding window for tiled diagonal zipper and staircase codes (obtained experimentally with $m = \mu = 254$ and BER of 10^{-7}). The constituent decoder is used only when it detects a row with 'fresh' flag. The figure shows that most decoding activities take place around the newest rows, while there are significantly less decoding activities on older rows. Around 80.7% and 54.7% of the decoding take place in the newest 254 rows for the diagonal zipper and staircase codes, respectively. In addition, the average number of decoding per row is 4.93 for diagonal zipper and 4.21 for staircase codes.

3. **Error correcting capability of the constituent codes:** For syndrome-based constituent decoders, having stronger codes means that the syndromes are longer and the decoder will have to process more symbols.

2.5 Graph Representation

2.5.1 Periodic Graph

Another way to represent periodic zipper codes is in the form of a periodic graph, an infinite graph with repetitive structure. We will use similar notation introduced in [37] to describe the periodic graph representation of zipper codes.

Definition 5. Let $G = (V, E, f)$ be a finite weighted directed graph, where V, E respectively denote the set of vertices and the set of edges. Also, let $f : E \to \mathbb{Z}$ denote a weight function of the edges. The periodic graph $G^* = (V^*, E^*)$ is the undirected graph where

$$V^* = \left\{ r^{(i)} : r \in V, i \in \mathbb{Z} \right\},$$
$$E^* = \left\{ \left(r^{(i)}, v^{(i+f(u,v))} \right) : (r, v) \in E, i \in \mathbb{Z} \right\}.$$

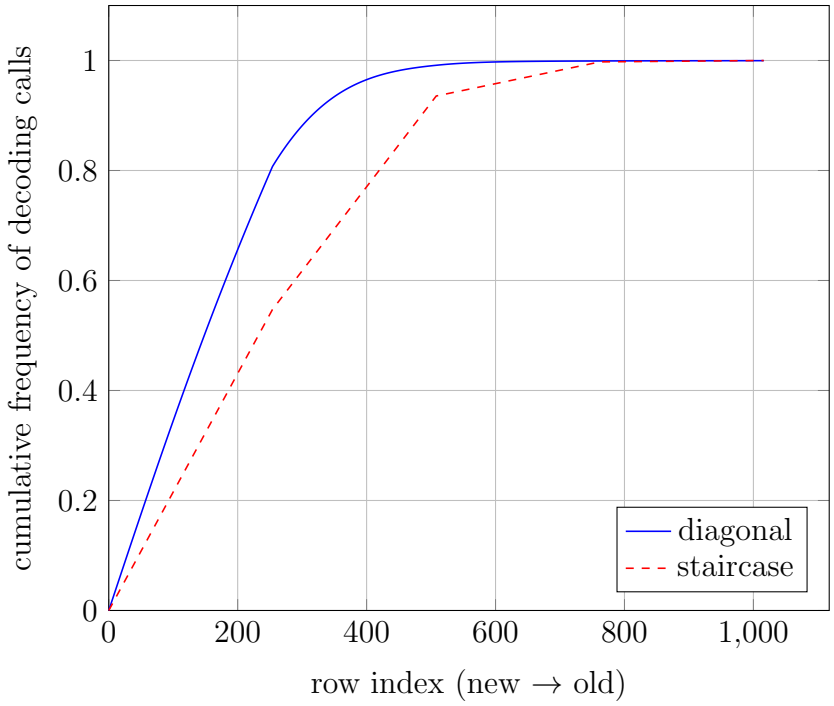

Figure 2.8: Cumulative frequency of decoding subroutine executed for each index in the decoding window for tiled diagonal zipper codes and staircase codes with $m = \mu = 254$ and BER of 10^{-7}.

We call G a static graph and G^* the periodic graph generated by the static graph.

Example 1. Figure 2.9 shows a static graph and periodic graph generated by the static graph. The weight of the edge connecting r_0 and r_1 is 1 in the static graph, so we connect $r_0^{(i)}$ with $r_1^{(i+1)}$ in the periodic graph. In similar vein, we connect $r_0^{(i)}$ with $r_2^{(i-1)}$ and $r_2^{(i)}$ with $r_2^{(i+2)}$. Observe that if we reverse any edge and flipping the sign of the weight of the edges that we flip, we obtain the exact same periodic graph.

We will now describe how to formulate a zipper code structure as a periodic graph. Let ϕ be a periodic interleaver map with period ν. In the static graph representation, we will have ν vertices, denoted as

$$V = \{r_0, r_1, \ldots, r_{\nu-1}\}.$$

The vertex i represents the row index i modulo ν. Furthermore, for each $i \in [\nu], j \in [m_i]$, define

$$f(i,j) = \left\lfloor \frac{i}{\nu} \right\rfloor - \left\lfloor \frac{\phi_1(i,j)}{\nu} \right\rfloor \quad \text{and} \quad \psi(i,j) = \phi_1(i,j) \pmod{\nu}.$$

We form an edge $\left(r_i, r_{\psi(i,j)}\right)$ with weight $f(i,j)$ in the static graph. The number of

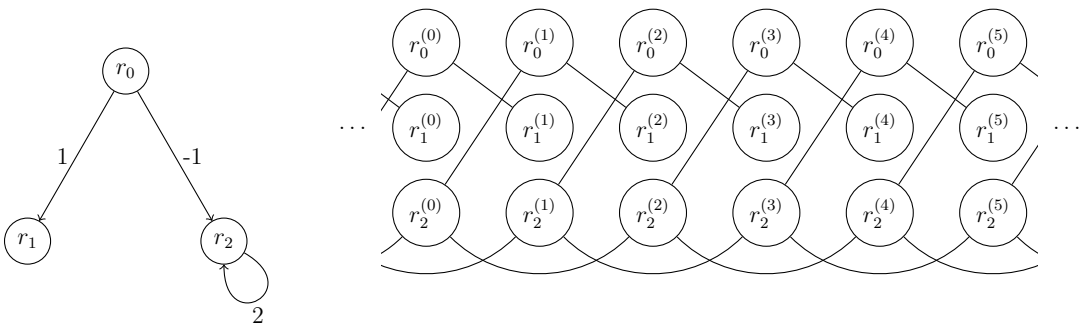

Figure 2.9: A simple example of static graph (left) and periodic graph (right).

outgoing edges for r_i is m_i and the number of incoming edges is $n_i - m_i$. Note that if $m_i > \nu$, then there will be parallel edges and/or self-loops in the static graph since there are more outgoing edges than the number of vertices. Also, if $i > \phi_1(i,j)$, then $f(i,j) \geq 0$. Thus, we have proved the following lemma.

Lemma 1. *If ϕ is causal, then $f(i,j) \geq 0$ for all $i, \in [\nu], j \in [m_i]$.*

The periodic graph generated by the static graph will then have vertices

$$V^* = \left\{ r_0^{(\ell)}, r_1^{(\ell)}, \ldots, r_{\nu-1}^{(\ell)} : \ell \in \mathbb{Z} \right\}.$$

The vertex $r_i^{(\ell)}$ corresponds to the $(\nu\ell+i)$-th row of the code. Then, for all $i \in [\nu], j \in [m_i]$, for all edge $\left(r_i, r_{\psi(i,j)}\right)$ with weight $f(i,j)$ in the static graph, we will form an edge

$$\left(r_i^{(\ell)}, r_{\psi(i,j)}^{(\ell - f(i,j))}\right)$$

in the periodic graph for all $\ell \in \mathbb{Z}$.

Lemma 2. *A bijective periodic interleaver map ϕ scatters if and only if the periodic graph G^* is simple.*

Proof. If G^* has a loop, then $f(i,j) = 0$ and $\psi(i,j) = i$ for some $i \in [\nu], j \in [m_i]$, which implies that $\phi_1(i,j) = i$. Conversely, if $\phi_1(i,j) = i$, then $f(i,j) = 0$ and $\psi(i,j) = i$, which means that for all $\ell \in \mathbb{Z}$, we form a loop at the vertex $r_i^{(\ell)}$ in the periodic graph.

If G^* has an parallel edge, then there exists some i, j, j' such that $\psi(i,j) = \psi(i,j')$ and $f(i,j) = f(i,j')$. This implies that $\phi_1(i,j) = \phi_1(i,j')$. Conversely, if $\phi_1(i,j) = \phi_1(i,j')$, then we have $\psi(i,j) = \psi(i,j')$ and $f(i,j) = f(i,j')$. This implies that there exist parallel edges between vertices $r_i^{(\ell)}$ and $r_{\psi(i,j)}^{(\ell - f(i,j))}$ in G^* for all $\ell \in \mathbb{Z}$. □

Figures 2.10, 2.11, and 2.12 show examples of the periodic and static graph representations of tiled diagonal zipper codes and staircase codes.

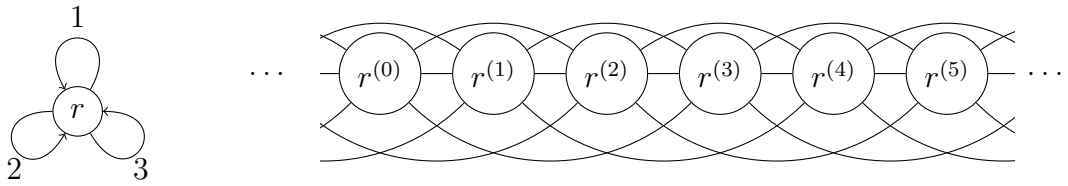

Figure 2.10: Static and periodic graph representation of tiled diagonal zipper codes, $w = 1$, $m = 3$.

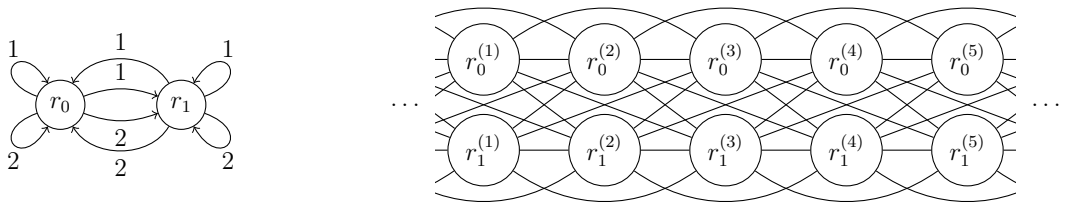

Figure 2.11: Static and periodic graph representation of tiled diagonal zipper codes, $w = 2$, $m = 4$.

2.5.2 Factor Graph

Similarly, we can also show the factor graph representation of a zipper code as shown in Figure 2.13. Each symbol is represented as a variable node (denoted as a circle in Figure 2.13), which can either represent a real symbol or a virtual symbol. Then, each variable node is connected to two check nodes. One represents the constituent code constraint, labeled as '$\mathcal{C}$' and the other represents the interleaver map constraint, labeled as '='. Each constituent code $\mathcal{C}$ check node connects to the n real and virtual symbols of the corresponding codeword, and each interleaver map check node connects a virtual variable node with a real variable node. Note that if the zipper code is periodic, then so is the structure of the factor graph. Also, observe that by replacing all variable and '=' nodes with edges, we transform the factor graph into a periodic graph.

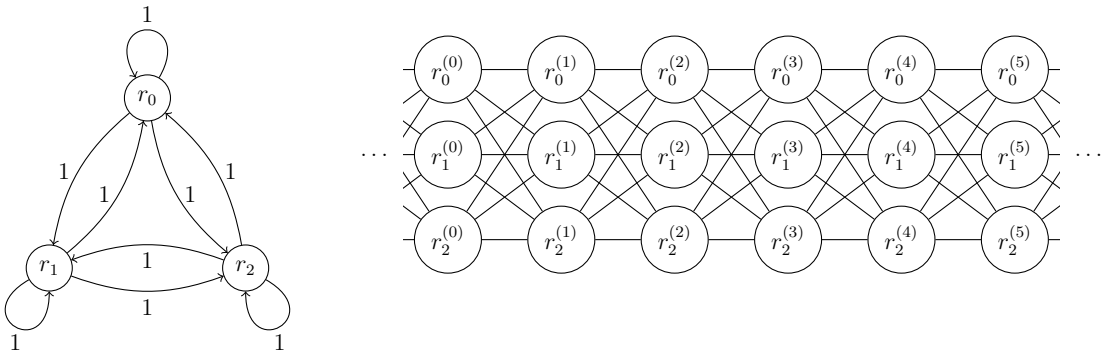

Figure 2.12: Static and periodic graph representation of staircase codes, $m = 3$.

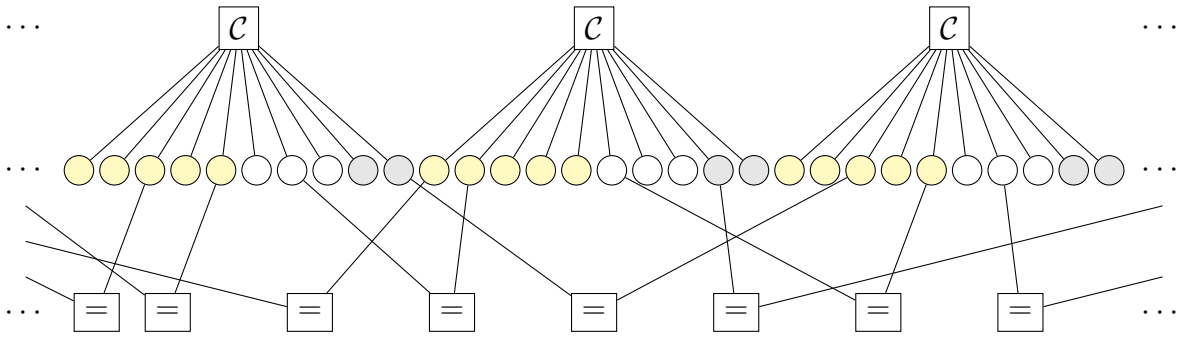

Figure 2.13: Factor graph representation of zipper codes

2.6 Irregular Zipper Codes

So far we have only discussed "regular" zipper codes with identical constituent codes for each row as well as bijective interleaver map. This section describes a few generalization that zipper codes can have by introducing a few "irregularities" such as non-uniform constituent codes as well as non-bijective interleaver map.

2.6.1 Non-uniform Constituent Codes

In the non-uniform constituent codes scheme, each row i is now described by constituent code $\mathcal{C}_i(n_i, k_i, d_i)$ of block length n_i, dimension k_i, and minimum Hamming distance d_i. The choice of parameter in this scheme is similar to the regular case, e.g., for each row i, we reserve the first $m_i \leq k_i$ symbols for the virtual symbols and the last $r_i \leq n_i - m_i$ symbols for the parity symbols. Also, we define

$$A_i = \{(i, j) : j \in [m_i]\} \quad \text{and} \quad B_i = \{(i, j) : j \in [n_i] \setminus [m_i]\},$$

and respectively define the virtual and real set to be

$$A = \bigcup_i A_i \quad \text{and} \quad B = \bigcup_i B_i.$$

The definition of a "buffer" remains the same, although the graphical representation of the buffer changes since we use a different code for each row, as shown in Figure 2.14.

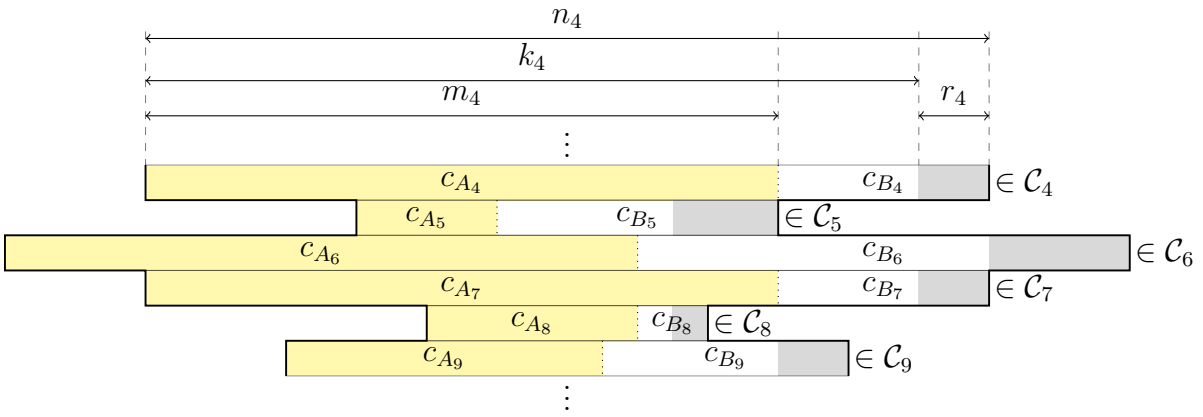

Figure 2.14: Buffer of an irregular zipper code, each row i is a constituent code of $\mathcal{C}_i$ with the first m_i symbols being virtual symbols.

Further, the definitions of periodicity and causality in the irregular case are identical to the regular case. However, if an interleaver map is periodic with period ν, it implies that $m_{i+\nu} = m_i$ and $n_{i+\nu} = n_i$.

2.6.2 Non-bijective Interleaver Maps

In the non-bijective scheme, we still require that each real symbol is copied in the virtual buffer. However, in this scheme we allow each real symbol to have more than one copy in the virtual buffer. In other words, we relax the constraint of the interleaver map to be surjective instead of bijective.

Definition 6. An interleaver map ϕ is *surjective* if for any $(i', j') \in B$, there exists $(i, j) \in A$ such that $\phi(i, j) = (i', j')$.

For all $b \in B$ define the preimage of b to be

$$\phi^{-1}(b) = \{a \in A : \phi(a) = b\}.$$

We define the degree of b, denoted as $\gamma(b)$, to be the cardinality of $\phi^{-1}(b)$. In other words, $\gamma(b)$ denotes the number of copies of c_b in the virtual buffer. If $|\gamma(b)| = 1$ for all $b \in B$, then the interleaver map is bijective.

Suppose that the code is periodic with period ν, we also define the average degree of each entry in the virtual set to be

$$\bar{\gamma} = \frac{\sum_{b \in B_0 \cup \cdots \cup B_{\nu-1}} \gamma(b)}{\sum_{i=1}^{\nu}(n_i - m_i)} = \frac{\sum_{i=1}^{\nu} m_i}{\sum_{i=1}^{\nu}(n_i - m_i)} = \frac{\bar{m}}{\bar{n} - \bar{m}},$$

where

$$\bar{n} = \frac{1}{\nu}\sum_{i=1}^{\nu} n_i.$$

Hence, the rate is given by

$$R = \frac{\sum_{i=1}^{\nu}(n_i - m_i - r_i)}{\sum_{i=1}^{\nu}(n_i - m_i)} = 1 - \frac{\bar{r}}{\bar{n} - \bar{m}} = 1 - (\bar{\gamma} + 1)\frac{\bar{r}}{\bar{n}}.$$

The implication of surjective maps with $\bar{\gamma} > 1$ is that the code rate is lower than the bijective case. Indeed,

$$1 - (\bar{\gamma} + 1)\frac{\bar{r}}{\bar{n}} < 1 - \frac{2\bar{r}}{\bar{n}},$$

where the right-hand side of the inequality denotes the rate of zipper code with bijective interleaver map.

2.7 Simulation Results

We present our software simulation results of tiled diagonal zipper codes with $\delta = 1$ and triple-error-correcting BCH component codes of various rates and tile sizes. Figure 2.15 shows the simulation results of the code with $w = 10$, $m = 1200$ (rate 0.97) over various values of M and μ. We also show how the performance varies when we modify the rate of the code. Figure 2.16 shows the performance of the code over various rates listed in Table 2.1.

Table 2.1 shows p^*, the binary symmetric channel crossover probability such that the bit error rate (BER) is 10^{-15}. We cannot bring the BER down to 10^{-15} in the software simulation, so we use a least-squares fit linear extrapolation in the log-log scale. The table also shows the gap to capacity for each parameter. The gap to capacity in a binary symmetric channel for a code with rate R is defined as

$$\text{Gap (dB)} = 20 \log_{10} \left(\frac{\operatorname{erfc}^{-1}(2p^*)}{\operatorname{erfc}^{-1}\left(2\mathcal{H}^{-1}(1-R)\right)} \right), \tag{2.12}$$

where erfc denotes the complementary error function,

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty \exp\left(-t^2\right) \mathrm{d}t,$$

and erfc^{-1} denotes its inverse. Moreover, $\mathcal{H}$ and $\mathcal{H}^{-1}$ respectively denote the binary entropy function and its "inverse". Note that since $\mathcal{H}(p) = \mathcal{H}(1-p)$ for $p \in [0, 1]$, the "inverse" entropy function $\mathcal{H}^{-1}(q)$ will yield two values for $q \neq 1$. We will only consider the lower of the two to be the "inverse" in (2.12) above.

The simulation results show that the gap to capacity decreases as we increase the code rate. However, it comes at the cost of having to use longer constituent block length, and consequently, the memory size also grows very quickly.

Concluding Remarks

In this chapter, we have introduced zipper codes as a framework to formulate spatially-coupled product-like codes. We have also described some properties of zipper codes that we deem practical and have given examples of codes that can be described in our framework. We have introduced tiled and delayed diagonal zipper codes as a type of code that is more memory efficient. Finally, simulation results show that tiled diagonal zipper codes have comparable performance to staircase codes while requiring less memory.

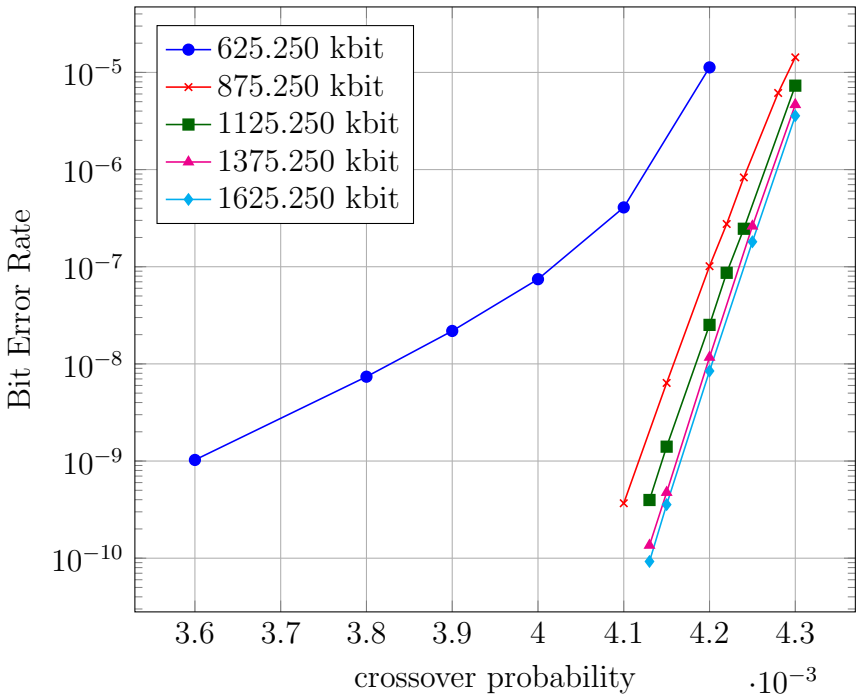

Figure 2.15: Simulation results on a binary memoryless symmetric channel of diagonal zipper codes with $m = 500$ using a triple-error correcting $(1000, 970)$ shortened BCH constituent code. The values on the legend indicate the sizes of the decoder memory used.

Table 2.1: Parameters, p^*, and Gap to Capacity of Diagonal Zipper Code and Staircase Codes.

Type	Rate	m	Chunk Size (kbit)	Window Size (Mbit)	p^*	Gap (dB)
diagonal	0.96	825	339.9	3.744	2.67×10^{-3}	0.506
staircase	0.96	825	680.6	4.084	2.72×10^{-3}	0.489
diagonal	0.97	1200	720.0	7.921	2.01×10^{-3}	0.418
staircase	0.97	1200	1440.0	8.640	2.07×10^{-3}	0.392
diagonal	0.98	1800	1620.0	17.821	1.23×10^{-3}	0.397
staircase	0.98	1800	3240.0	19.440	1.25×10^{-3}	0.384
staircase	0.98375	2400	5760.0	34.560	1.02×10^{-3}	0.338
staircase	0.985	2600	6760.0	40.560	9.27×10^{-4}	0.333

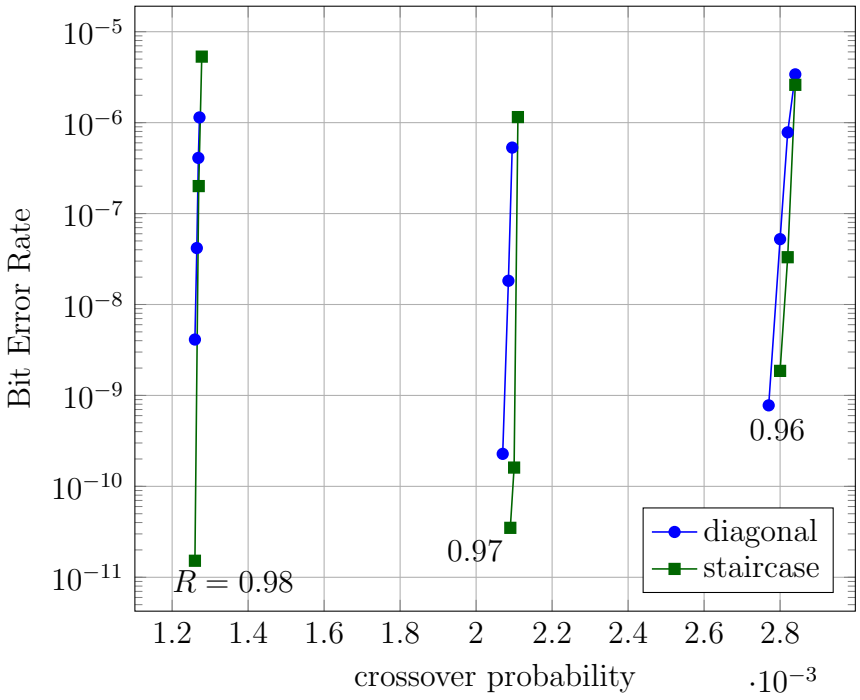

Figure 2.16: Simulation results of diagonal zipper codes and staircase codes with $t = 3$ based on the parameters listed in Table 2.1.

Chapter 3

Stall Pattern Analysis

This chapter describes the analysis "stall patterns" of zipper codes. In simple terms, a stall pattern is a set of errors in the received symbols which cannot be corrected using iterative decoding. The exact definition of a stall pattern will be provided in Section 3.1. For simplicity, we analyze zipper codes with identical t-error-correcting constituent code and virtual buffer width m throughout this chapter. Also, we use a genie-aided, miscorrection-free constituent decoder: given a received word, the decoder "knows" exactly how many erroneous symbols there are and their locations. If there are fewer than t errors, the decoder will correct those symbols. Otherwise, the decoder will do nothing.

3.1 Error Patterns, Stall Patterns, and Decoding Procedure

An error pattern S is a nonempty subset of the real set B. Although S can be arbitrarily large, we will only consider finite-sized S for error pattern analysis.

First, for all $S \subseteq B$, denote

$$S^* = S \cup \phi^{-1}(S)$$

to be the error pattern with its copy in the virtual buffer. We define $\pi_1(S^*)$ to be the *affected rows* of S^* and we say that row $i \in \pi_1(S^*)$ is correctable if

$$|S^* \cap (A_i \cup B_i)| \leq t,$$

where t denotes the error-correcting capability of the constituent code. Also, denote

$$\kappa(S) = \{i \in \pi_1(S^*) : |S^* \cap (A_i \cup B_i)| \leq t\}$$

to be the set of correctable rows of S. We call a nonempty set S a *stall pattern* if no rows of S are correctable.

We will now define the decoding function as follows:

$$\begin{aligned} &\mathcal{D} : \mathcal{P}(B) \to \mathcal{P}(B) \\ &\qquad S \mapsto \begin{cases} S & \text{if } \kappa(S) = \varnothing, \\ S \setminus (B_{\kappa^*} \cup \phi(A_{\kappa^*})) & \text{otherwise,} \end{cases} \end{aligned} \tag{3.1}$$

where $\mathcal{P}(B)$ denotes the power set of B and $\kappa^* = \min \kappa(S)$. In other words, if S is nonempty, $\mathcal{D}$ will remove the errors in the row of S^* with lowest affected row index as well as their duplicates.

Given an error pattern S, we have $|\mathcal{D}(S)| \leq |S|$, so the correctability of a stall pattern S can be determined by repeatedly applying $\mathcal{D}$ up to $|\pi_1(S^*)|$ times, i.e., we call S to be *correctable* if

$$\mathcal{D}^{|\pi_1(S^*)|}(S) = \underbrace{\mathcal{D}(\mathcal{D}(\cdots \mathcal{D}(\mathcal{D}(S)) \cdots)}_{|\pi_1(S^*)| \text{ times}} = \varnothing.$$

We will now define the minimality of a stall pattern.

Definition 7. A stall pattern S is *minimal* if for all nonempty $T \subsetneq S$, T is correctable.

Definition 8. A stall pattern S is *minimum-sized* if for all stall patterns T, we have $|T| \geq |S|$.

Remark 4. A minimum-sized stall pattern is always minimal, but the converse is not always true. For example, in staircase codes with $t = 2$, the minimum-sized stall patterns have size $(t+1)^2 = 9$, but we can also have a minimal, but not minimum-sized, stall pattern of size 12 as shown in Figure 3.1. Removing any one error from the second diagram will cause the pattern to be correctable.

3.1.1 Periodic Graph Representation

We can also describe an error pattern S using graphs. The graph representation of error pattern S is a nonempty subgraph G of the periodic graph representation of the zipper code we are interested in. The vertices of G represent the affected rows and for all $b \in S$,

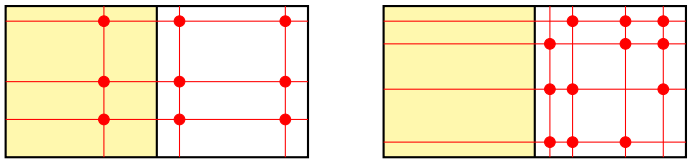

Figure 3.1: Minimum-sized (left) and minimal-but-not-minimum-sized (right) stall patterns of a staircase code with double-error-correcting constituent code.

we connect vertices that correspond to rows $\pi_1(b)$ and $\phi_1^{-1}(b)$ with an edge. The number of edges represent the number of errors in an error pattern. Assuming the interleaver map is scattering, G will be a simple graph. In addition, a vertex is decodable if the degree is t or smaller, and an error pattern G is a stall pattern if all vertices have degree of at least $t+1$. The decoding function will therefore act as if we delete decodable vertices as well as the edges that are connected to them. We will repeatedly remove vertices that have at most t edges connected to them until either:

- the resulting graph is empty, in which case the error pattern is correctable, or
- the remaining vertices have degree $\geq t+1$, in which case we converge to a stall pattern.

We define the $(t+1)$-core of G to be the maximum induced subgraph of G such that the degree each vertex is at least $t+1$, as defined in [38]. If G converges to a stall pattern subgraph after repeatedly removing vertices that have degree t or smaller, such subgraph is the $(t+1)$-core of G.

Remark 5. The decoding function (3.1) corrects the correctable row with the lowest row index. However, the order in which we correct the correctable rows does not matter as the $(t+1)$-core of G is unique.

Examples of the graph representation of stall patterns of tiled diagonal zipper codes are shown in Figures 3.2, 3.3, and 3.4.

3.2 Zipper Codes with Scattering Interleaver Map

Recall that an interleaver map scatters if the duplicates of all n entries in row i are all in pairwise distinct rows and not in row i, for all i. For example, the interleaver maps for staircase codes, braided block codes, and tiled diagonal zipper codes introduced in Section 2.3 are scattering. We will discuss the importance of scattering interleaver maps in this section.

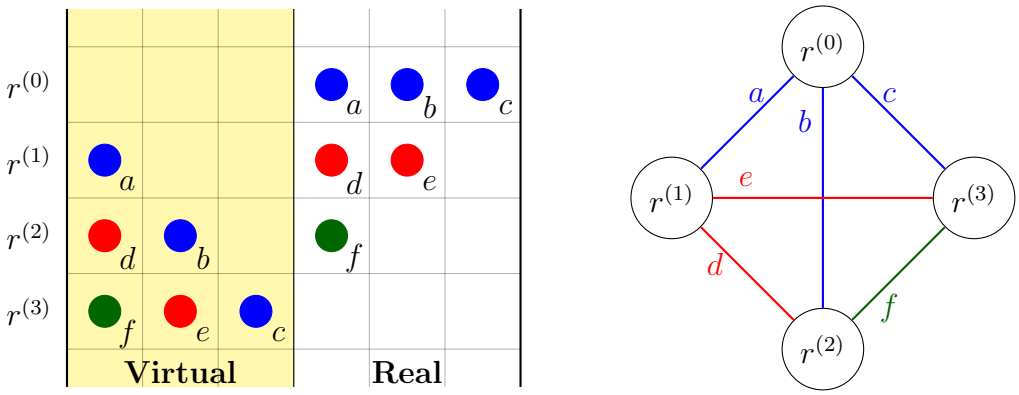

Figure 3.2: Graph representation of a stall pattern of size 6 of tiled diagonal zipper code, $w = 1$, $m = 3$, $t = 2$.

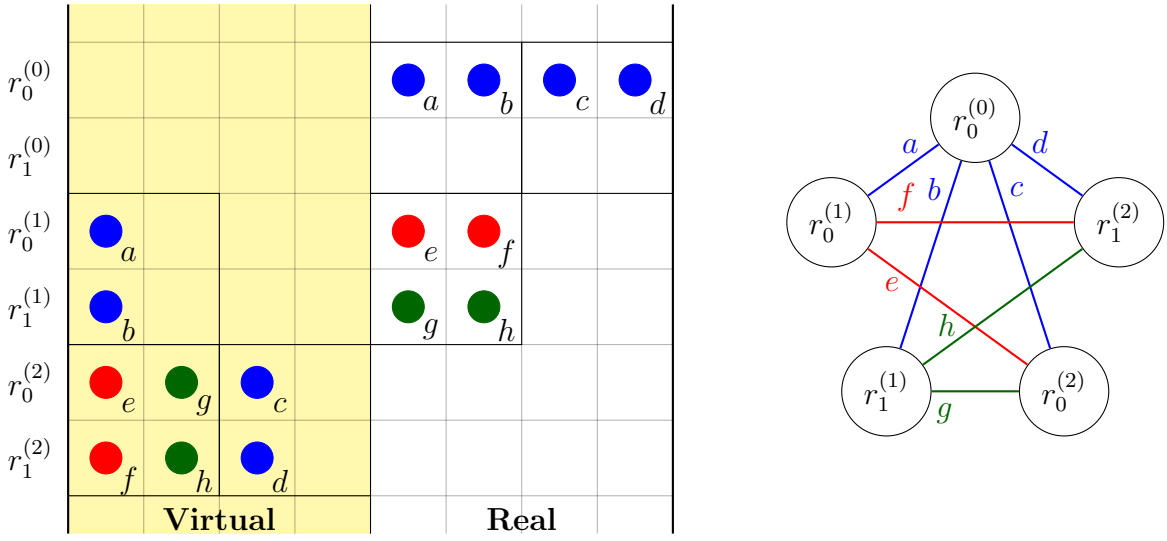

Figure 3.3: Graph representation of a stall pattern of size 8 of tiled diagonal zipper code, $w = 1$, $m = 4$, $t = 2$.

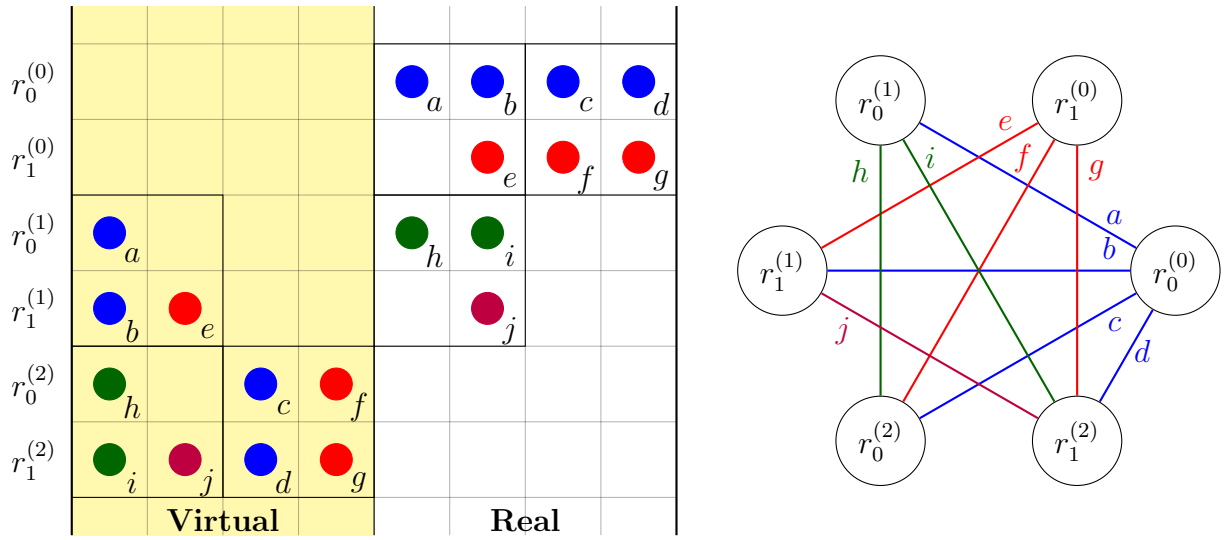

Figure 3.4: Graph representation of a stall pattern of size 10 of tiled diagonal zipper code, $w = 2$, $m = 4$, $t = 2$.

For brevity, we will sometimes combine ϕ and ϕ^{-1} into one function defined as follows.

$$\hat{\phi} : A \cup B \to A \cup B$$
$$x \mapsto \begin{cases} \phi(x) & \text{if } x \in A \\ \phi^{-1}(x) & \text{if } x \in B. \end{cases}$$

Such a function is sometimes called a union of functions $\hat{\phi} = \phi \cup \phi^{-1}$. It is a well-defined concept since the domains are disjoint. We also define a shorthand notation

$$\hat{\phi}(X) = \left\{ \hat{\phi}(x) : x \in X \right\}$$

for all $X \subseteq A \cup B$.

We will first begin by showing a simple property of a minimal stall pattern in the theorem below.

Theorem 3. *Let S be a minimal stall pattern of a zipper code with a bijective scattering interleaver map. Suppose that there exists a row index $i \in \pi_1(S^*)$ such that $|S^* \cap (A_i \cup B_i)| > t + 1$. Then for all $j \in \hat{\phi}_1(S^* \cap (A_i \cup B_i))$,*

$$|S^* \cap (A_j \cup B_j)| = t + 1.$$

In other words, if any row in a minimal stall pattern has more than $t+1$ errors, then all of its neighbouring rows in the stall pattern have exactly $t+1$ errors. An equivalent periodic graph formulation of the theorem is that the neighbours of a node in a minimal stall pattern with degree larger than $t+1$ all have degree exactly $t+1$.

Proof. Removing an error location shared between two rows having more than $t+1$ errors in a stall pattern gives a stall pattern of smaller size. Thus, such a configuration cannot exist in a minimal stall pattern. □

Note that Theorem 3 implies the existence of an affected row with exactly $t+1$ errors in every minimal stall pattern, provided that the interleaver map is bijective and scattering. Figures 3.2 and 3.3 are examples of minimal stall patterns. On the other hand, Figure 3.4 is not minimal since removing the edge connecting $r_0^{(0)}$ and $r_1^{(2)}$ will result in a stall pattern of smaller size.

Now we will show a theorem that describes the lower bound on the size of a stall patterns with scattering interleaver map.

Theorem 4. *Let S be a stall pattern of a zipper code with a bijective scattering interleaver map. Suppose that the constituent codes can correct up to t errors. Then,*

$$|S| \geq \frac{(t+1)(t+2)}{2}.$$

Proof. Let i be an arbitrary element of $\pi_1(S^*)$, and suppose that

$$S^* \cap (A_i \cup B_i) = \{u_1, u_2, \ldots, u_v\},$$

i.e., $\{u_1, u_2, \ldots, u_v\}$ are the error locations in row i. Since S is a stall pattern we have $v \geq t+1$. Furthermore, since ϕ scatters,

$$\hat{\phi}(u_1), \hat{\phi}(u_2), \ldots, \hat{\phi}_1(u_v)$$

are distinct and not equal to i. Thus, there are at least $t+1$ errors in each of those affected v rows, and so the total number of errors is

$$\begin{aligned} |S^*| &\geq v(t+1) + v \\ &= v(t+1+1) \\ &\geq (t+1)(t+2). \end{aligned}$$

Finally, since ϕ is bijective, we have

$$|S| = \frac{|S^*|}{2} \geq \frac{(t+1)(t+2)}{2}.$$

□

Now recall that

$$\rho(i) = \phi_1(A_i) \cup \phi_1^{-1}(B_i)$$

is the set of rows reachable from row i (or neighbours of row i) defined in Section 2.1.2. We will now describe the condition to produce stall patterns of size $\frac{1}{2}(t+1)(t+2)$.

Theorem 5. *Let ϕ be bijective and scattering. Suppose that the consituent code can correct up to t errors. Then there exists a stall pattern of size $\frac{1}{2}(t+1)(t+2)$ if and only if there exists $I \subseteq \mathbb{Z}$ such that $|I| = t+2$ and $I \setminus \{i\} \subseteq \rho(i)$ for all $i \in I$.*

Equivalently, a stall pattern of size $\frac{1}{2}(t+1)(t+2)$ exists if and only if there exists a $(t+2)$-clique in the periodic graph.

Proof. **(⇐)** Since ϕ scatters, for each $i \in I$, there are exactly $t+1$ distinct coordinates $\alpha_1^{(i)}, \ldots, \alpha_{t+1}^{(i)} \in A_i \cup B_i$ such that

$$\left\{\hat{\phi}_1\left(\alpha_1^{(i)}\right), \ldots, \hat{\phi}_1\left(\alpha_{t+1}^{(i)}\right)\right\} = I \setminus \{i\}.$$

We then let

$$S^* = \left\{\alpha_j^{(i)} : i \in I, j = 1, \ldots, t+1\right\},$$

which contains $(t+1)(t+2)$ entries, and since ϕ is bijective, we have

$$|S| = \frac{|S^*|}{2} = \frac{1}{2}(t+1)(t+2).$$

(⇒) Suppose that S is a stall pattern of size $\frac{1}{2}(t+1)(t+2)$, then $|S^*| = (t+1)(t+2)$ since ϕ is bijective. Now suppose that there are fewer than $t+2$ affected rows in S^*. Then on average there will be more than $t+1$ errors per row, and so there exists a row that contains more than $t+1$ errors. This implies that there are more than $t+2$ affected rows since ϕ is dispersive, which contradicts our previous assumption. On the other hand, if there are more than $t+2$ affected rows, at least one row in S^* will contain fewer than $t+1$ errors, which cannot happen if S is a stall pattern. Hence, S^* is forced to contain exactly $t+2$ affected rows, each having exactly $t+1$ errors.

Now let $I = \pi_1(S^*)$. Observe that $|I| = t + 2$ and for each row $i \in I$,

$$\phi_1(S^* \cap A_i) \cup \phi_1^{-1}(S^* \cap B_i) = I \setminus \{i\},$$

i.e., the neighbours of row i in the stall pattern are all rows in I other than i. Thus,

$$\begin{aligned} I \setminus \{i\} &= \phi_1(S^* \cap A_i) \cup \phi_1^{-1}(S^* \cap B_i) \\ &\subseteq \phi_1(A_i) \cup \phi_1^{-1}(B_i) \\ &= \rho(i). \end{aligned}$$

□

For example, Figure 3.2 shows a stall pattern of a tiled diagonal zipper code with $w = 1, t = 2$. The stall pattern size is $\frac{1}{2}(t+1)(t+2) = 6$ and the periodic graph representation is a 4-clique.

Note that Theorems 4 and 5 imply that the minimum number of affected rows of a stall pattern is at least $t + 2$. If we know the number of affected rows of a stall pattern, we can find a bound on the number of errors that the stall pattern has.

Theorem 6. *Let S be a stall pattern of a zipper code with a bijective scattering interleaver map and t-error-correcting constituent code. Suppose that S^* has ℓ affected rows, then*

$$\left\lceil \frac{\ell(t+1)}{2} \right\rceil \leq |S| \leq \frac{\ell(\ell-1)}{2}.$$

Proof. There have to be at least $t + 1$ errors in each affected row of S^*, so

$$|S| = \frac{|S^*|}{2} \geq \left\lceil \frac{\ell(t+1)}{2} \right\rceil.$$

On the other hand, the maximum number of edges in a simple graph with ℓ vertices is precisely $\frac{1}{2}\ell(\ell-1)$, i.e., when an ℓ-clique is formed. □

3.3 Zipper Codes with Causal Interleaver Map

In this section, we will describe a few properties of the stall patterns that can occur from a causal interleaver map.

Theorem 7. *Let ϕ be bijective, and let S be a stall pattern. Denote $\ell = \min \pi_1(S^*)$ and $h = \max \pi_1(S^*)$ as the first and last rows of the affected rows of S^*, respectively. If ϕ is causal, then $S^* \cap A_\ell = \varnothing$ and $S^* \cap B_h = \varnothing$, i.e., the errors in the first row must all be in the real buffer, while the errors in the last row must all be in the virtual buffer.*

Proof. Suppose that $S^* \cap A_\ell \neq \varnothing$, then for all $a \in S^* \cap A_\ell$, we have $\phi_1(a) \geq \ell$. Similarly, if $S^* \cap B_h \neq \varnothing$, then for all $b \in S^* \cap B_h$, we have $\phi^{-1}(b) \leq h$. In either case, ϕ is not causal. □

If the interleaver map is both scattering and causal, we also have the following corollary that follows from Theorems 5 and 7.

Corollary 1. *Let ϕ be bijective, scattering, and causal. Then there exists a stall pattern S of size $\frac{1}{2}(t+1)(t+2)$ if and only if there exists $I = \{i_1, \ldots, i_{t+2}\} \subseteq \mathbb{Z}$ with $i_i < i_2 < \ldots < i_{t+2}$ such that for $j = 1, \ldots, t+2$,*

$$\phi_1^{-1}(S \cap B_{i_j}) = \{i_{j+1}, i_{j+2}, \ldots, i_{t+2}\}.$$

Proof. **(⇒)** By Theorem 5, we have $I = \{i_1, i_2, \ldots, i_{t+2}\} \subseteq \mathbb{Z}$ (without loss of generality, let $i_1 < i_2 < \ldots < i_{t+2}$) and there are $t+1$ errors in each affected row. Now by Theorem 7, all errors in row i_1 are contained in B_{i_1}, and since ϕ scatters, by Theorem 5 we have

$$\phi^{-1}(S \cap B_{i_1}) = \{i_2, i_3, \ldots, i_{t+2}\} \quad \text{and} \quad \phi^{-1}(S \cap B_{i_{t+2}}) = \varnothing.$$

For the remaining rows i_j, $j = 2, \ldots, t+1$, observe that since ϕ is causal, $\phi_1^{-1}(S \cap B_{i_j}) \subseteq \{i_{j+1}, \ldots, i_{t+2}\}$. Now suppose that $\phi_1^{-1}(S \cap B_{i_j}) \subsetneq \{i_{j+1}, \ldots, i_{t+2}\}$, then there will be at least one of $i_{j+1}, \ldots, i_{t+2}$ contained in $\phi_1(S^* \cap A_{i_j})$, which contradicts our assumption that ϕ is causal. Hence,

$$\phi_1^{-1}(S \cap B_{i_j}) = \{i_{j+1}, i_{j+2}, \ldots, i_{t+2}\}.$$

(⇐) Since ϕ scatters, we have $|\phi_1^{-1}(S \cap B_i)| = |S \cap B_i|$ for all $i \in I$, and so,

$$|S| = \sum_{j=1}^{t+2} |S \cap B_{i_j}| = \sum_{j=1}^{t+2} (t+2-j) = \sum_{k=0}^{t+1} k = \frac{(t+1)(t+2)}{2}$$

□

3.4 Diagonal Zipper Codes

In this section, we analyze the stall patterns of tiled diagonal and delayed diagonal zipper codes introduced in Sections 2.3.5 and 2.3.6, respectively.

3.4.1 Tiled Diagonal Zipper Codes

Recall that the interleaver map of tiled diagonal zipper codes is given by

$$\phi(wq+i, ws+j) = (w(q-s-1)+j, w(L+s)+i),$$

where w and L respectively denote the tile size and the number of tiles in a row of virtual (or real) buffer, $s \in [L]$ $i \in [w]$, $j \in [w]$. Assuming that we use a constituent code that can correct up to t errors, there are three small-sized pattern sizes that can be easily constructed for this type of code: $\frac{1}{2}(t+1)(t+2)$, t^2+2t, and $(t+1)^2$. Examples of the first two stall patterns are shown in Figures 3.2 and 3.3. For each of the three sizes listed above, we will now determine the number of stall patterns that can be completely contained inside a decoding window with $m = wL$ and $M = wK$ $(K > L)$.

1. *Case* $\frac{1}{2}(t+1)(t+2)$: We assume that $L \geq t+1$. First, we pick $t+1$ real tiles from the same row. Then, we select a row index that we are going to fill with errors. For each of the $t+1$ tiles, pick one column index and we will place an error there. The location of the other $t(t+1)/2$ errors in future real tiles can then be obtained deterministically. Thus, the number of patterns of such type is

$$\sum_{s=t}^{L-1} \binom{s-1}{t-1} (K-s)(L-s) w^{t+2}.$$

   This is the exact number of stall patterns of this size in a decoding window of size $M \times m$ due to Theorem 5.

2. *Case* t^2+2t: We assume that $w \geq t$ and $L \geq 2$. We first select two real tiles in the same row. Select a row index, and for each of those two tiles, select t column indices for the errors. Finally, the other t^2 errors are contained in a future real tile which coordinates can be obtained deterministically. Note that there exist stall patterns of this size of this size even if $w < t$ or $L < 2$, but the exact number and

shape cannot be determined easily. Hence, there are at least

$$\sum_{s=1}^{L} \binom{w}{t}^2 (K-s)(L-s)w$$

ways to choose a pattern.

3. *Case* $(t+1)^2$: Here, we assume that $w \geq t+1$. Select any real tile from the decoding window, then fill it in with $(t+1)^2$ errors in a similar fashion to the minimum stall patterns of a product code of size w^2. Similar to the previous case, we can form stall patterns of this size even if $w < t+1$, but the exact number and shape cannot be determined easily. Therefore, there are at least

$$\binom{w}{t+1}^2 KL$$

ways to obtain a stall pattern of this type.

Remark 6. There exist other stall patterns with size between $\frac{1}{2}(t+1)(t+2)$ and $(t+1)^2$ as well as configurations other than the ones described above. However, they are harder to construct and describe, e.g. stall patterns of size $2(t+1) + \frac{1}{2}t(t+1)$. There are also stall pattern with sizes that are impossible to construct due to Theorem 6, such as stall patterns of size $\frac{1}{2}(t+1)(t+2)+1$ for $t \geq 2$.

3.4.2 Delayed Diagonal Zipper Codes

Recall that the interleaver map of the delayed diagonal zipper codes with delay δ is given by

$$\phi_\delta(i,j) = (i-j-\delta, j+m).$$

The delay parameter δ controls the occurrences and size of the minimum-sized stall patterns. Having larger delay suppresses the occurrences of stall patterns of size $\frac{1}{2}(t+1)(t+2)$, which exist when $\delta = 1$ (or tiled diagonal zipper code with $w = 1$). We will state a theorem below regarding the existence of stall patterns of size $\frac{1}{2}(t+1)(t+2)$ in delayed diagonal zipper codes.

Theorem 8. *For a delayed diagonal zipper code with delay δ, there exists a stall pattern of size $\frac{1}{2}(t+1)(t+2)$ if and only if*

$$\delta \leq \frac{m-1}{t}.$$

Proof. Without loss of generality, let the first affected row of the stall pattern be row zero.
($\Leftarrow$) We claim picking $I = \{0, \delta, \ldots, (t+1)\delta\}$ yields a stall pattern of size $\frac{1}{2}(t+1)(t+2)$. To see this, observe that

$$(t+1)\delta = t\delta + \delta \leq m + \delta - 1,$$

thus, for $i = 0, \ldots, t$,

$$\{(i+1)\delta, (i+2)\delta, \ldots, (t+1)\delta\} \subseteq \phi^{-1}(B_{i\delta}).$$

So we now let

$$\phi^{-1}(S \cap B_{i\delta}) = \{(i+1)\delta, (i+2)\delta, \ldots, (t+1)\delta\}.$$

Hence, by Corollary 1, there exists a stall pattern of size $\frac{1}{2}(t+1)(t+2)$.
($\Rightarrow$) Let $I = \{0, i_1, \ldots, i_{t+1}\}$ and $0 < i_1 < \ldots < i_{t+1}$. By Corollary 1, for $j = 0, \ldots, t+1$, we require

$$\phi^{-1}(S \cap B_{i_j}) = \{i_{j+1}, \ldots, i_{t+1}\}.$$

Also, observe that

$$i_1 \geq \min \phi^{-1}(S \cap B_0) \geq \min \phi^{-1}(B_0) = \delta,$$

and so,

$$\begin{aligned}
i_2 &\geq \min \phi^{-1}(S \cap B_{i_1}) \geq \min \phi^{-1}(B_{i_1}) = i_1 + \delta \geq 2\delta \\
i_3 &\geq \min \phi^{-1}(S \cap B_{i_2}) \geq \min \phi^{-1}(B_{i_2}) = i_2 + \delta \geq 3\delta \\
&\vdots \\
i_{t+1} &\geq \min \phi^{-1}(S \cap B_{i_t}) \geq \min \phi^{-1}(B_{i_t}) = i_t + \delta \geq (t+1)\delta.
\end{aligned}$$

However, we also require row i_{t+1} to be reachable from row zero, i.e., $i_{t+1} \leq m + \delta - 1$. Hence,

$$\begin{aligned}
(t+1)\delta &\leq m + \delta - 1 \\
t\delta &\leq m - 1 \\
\delta &\leq \frac{m-1}{t}.
\end{aligned}$$

□

Theorem 8 states that for a fixed m, the required delay to eliminate stall patterns of size $\frac{1}{2}(t+1)(t+2)$ is inversely proportional to the error correcting capability of the constituent code. Thus, the encoder memory,

$$\frac{m(m-1)}{2} + \delta m \approx m^2 \left(\frac{1}{2} + \frac{1}{t}\right)$$

is also smaller for higher t. For example, the encoder memory for $t = 2$ is comparable to that of staircase codes, while for $t = 4$, the memory is about 3/4 of that of staircase codes.

Suppose that $\delta \leq \frac{m-1}{t}$ and a stall pattern S has size $\frac{1}{2}(t+1)(t+2)$. The set of row indices $I = \{0, i_1, \ldots, i_{t+1}\}$ of the first row of S must satisfy the following inequalities:

$$\begin{aligned}
\delta \leq & i_1 \leq m - (t-1)\delta - 1 \\
i_1 + \delta \leq & i_2 \leq m - (t-2)\delta - 1 \\
i_2 + \delta \leq & i_3 \leq m - (t-3)\delta - 1 \\
& \vdots \\
i_t + \delta \leq & i_{t+1} \leq m + \delta - 1.
\end{aligned}$$

The number of possible configurations is the number of paths from source to sink of the trellis network in Figure 3.5, which is

$$\binom{m - t\delta + t}{t+1}.$$

The expression above can be approximated as

$$\frac{(m - t\delta)^{t+1}}{(t+1)!}$$

for $m - t\delta \gg 1$.

Example 2. Figure 3.6 shows the number of stall patterns of size $\frac{1}{2}(t+1)(t+2)$ in a delayed diagonal zipper codes for $m = 1000$ and $t = 3, 4, 5$. Observe that when $\delta = 1$ and $t = 3$, there are

$$\binom{1000}{4} \approx 4.14 \times 10^{10}$$

possible configurations in the first row. However, there is only one possible configuration for $\delta = 333$ and stall patterns of size $\frac{1}{2}(t+1)(t+2)$ do not exist for $\delta \geq 334$.

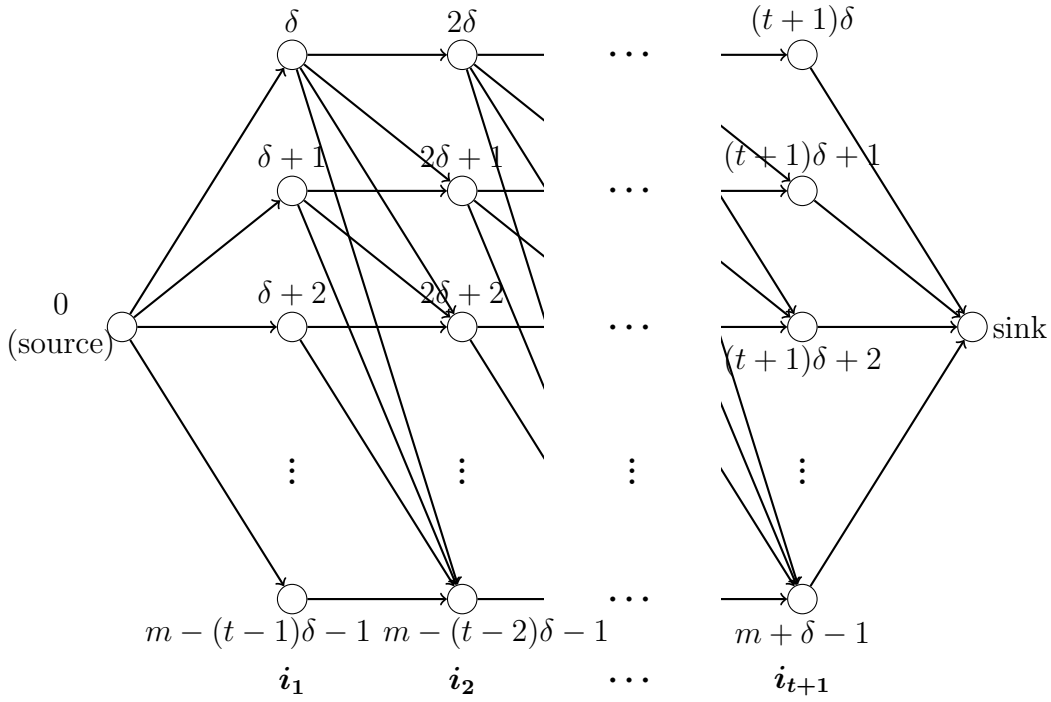

Figure 3.5: Trellis diagram for selecting $i_1, \ldots, i_{t+1}$. The number of possible patterns are given by the number of paths from source to sink.

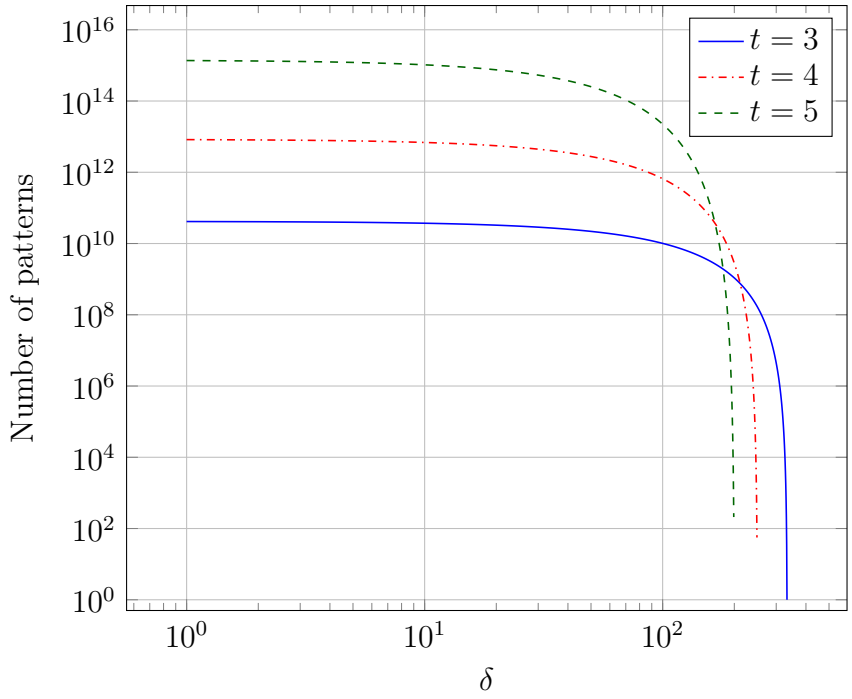

Figure 3.6: Number of stall patterns of size $\frac{1}{2}(t+1)(t+2)$ in a delayed diagonal zipper codes with $m = 1000$ and varying t, δ.

3.5 Error Floor Approximation

We approximate the error floor using the union bound technique similar to [8]. We consider a decoding window of size $M \times m$, and denote the set of all stall patterns in the decoding window to be $\mathcal{S}$. We determine the error floor estimate to be the enumeration of $\mathcal{S}$ and evaluating the probability of error with p denoting the probability that a symbol is in error.

$$\text{BER}_{\text{floor}} \leq \frac{1}{M \times m} \sum_{S \in \mathcal{S}} |S| p^{|S|}. \tag{3.2}$$

Suppose that we know exactly the sizes of stall pattern that can occur in a decoding window. We can also express (3.2) to be

$$\text{BER}_{\text{floor}} \leq \frac{1}{M \times m} \sum_{\ell \in \mathcal{L}} N_\ell p^\ell,$$

where $\mathcal{L}$ denotes the set of all stall pattern sizes that can occur in the decoding window and N_ℓ denotes the number of occurrences of stall patterns of size ℓ. Note that since we only consider stall patterns that can fit in the decoding window, we have $\ell \leq M \times m$.

The possible sizes and the number of occurrences of stall patterns of certain size depends on the interleaver map. For example, the sizes and occurrences of some stall patterns of diagonal zipper codes are described above in Section 3.4. Also, [8] and [39] describe the sizes and occurrences of stall patterns of staircase codes. We will now define dominant stall patterns of zipper codes.

Definition 9. Given a decoding window and a set $\mathcal{L}$ of possible stall pattern sizes that can fit in the window, we call the stall pattern of size $\ell^* \in \mathcal{L}$ to be the dominant if

$$N_{\ell^*} p^{\ell^*} \geq N_\ell p^\ell \quad \text{for all} \quad \ell \in \mathcal{L}.$$

In general, the dominant stall pattern size is not always minimum-sized since there are some cases where the number of occurrences of a larger stall pattern dominate the $N_\ell p^\ell$ term. However, if p is sufficiently small, we can assume that stall patterns of minimum size is the dominant stall pattern. The error floor can be then further approximated as the contribution of just the minimum-sized stall pattern.

Figure 3.7 shows the error floor contributions from stall patterns of sizes $\ell = 10, 15, 16$ for tiled diagonal zipper codes with $m = 1200, M = 6000, t = 3$ and tile sizes $w = 5, 300$. From the figure, we can infer that the stall patterns of size 10 are the dominant stall

patterns and the error floor contributions are below 10^{-15} in both cases at $p^* \approx 2 \times 10^{-3}$.

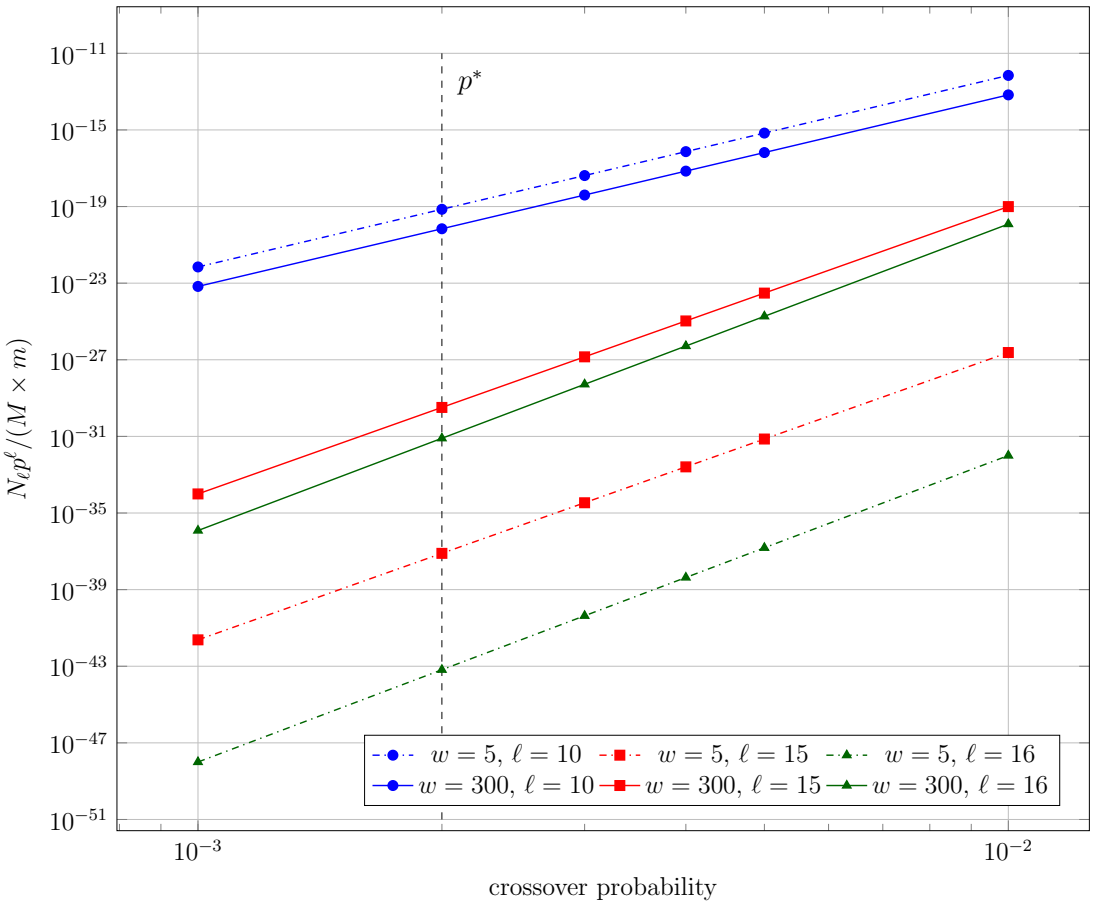

Figure 3.7: Error floor contributions from stall patterns of sizes $\ell = 10, 15, 16$ of tiled diagonal zipper codes with $m = 1200, M = 6000, t = 3$ and tile sizes $w = 5, 300$.

Concluding Remarks

In this chapter, we have introduced the concept of a stall pattern in zipper codes. The types of stall patterns that can occur depend on the interleaver map. We have determined the lower bound of the size of the stall pattern for zipper codes with scattering interleaver map. We also have shown that for delayed diagonal zipper codes, the stall patterns of small size vanish with sufficiently large "delay." Finally, we have shown the error floor analysis of zipper codes.

Chapter 4

Error-and-Erasure Decoding

In this chapter, we investigate the use of error-and-erasure decoding in zipper codes. In particular, we study the error-and-erasure staircase decoder. Whereas the conventional interface to a noisy channel expected by staircase code is binary $\{0, 1\}$-valued, we wish to investigate a ternary $\{0, 1, ?\}$-valued interface. Here the '?' symbol denotes a so-called "erasure," appropriate for symbols where the decoder cannot reliably decide between 0 and 1. It is well-known that error-and-erasure decoding can achieve improved performance with small increase in decoding complexity [40].

We have researched whether employing erasure declaration on staircase codes improves its decoding performance. This can be done by concatenating staircase code with a soft-decision inner code with ternary output: $0, 1$, and ?. This also implies that the staircase decoder needs to be modified so that it can handle erasure symbols.

We answer the following research questions:

- How should the inner code declare erasures?
- What is the appropriate fraction of erasures to declare?
- How should we modify the staircase decoder to handle erasures?
- Can the new system achieve a better performance and/or complexity than the existing system?

4.1 Related Works

Some related works on improving the decoding performance of staircase codes are listed below:

1. Holzbaur et al. [39] locate stall patterns by intersecting non-zero syndromes and flipping corresponding bits. The error floor estimate is shown to improve from $\sim 2 \times 10^{-10}$ to $\sim 9 \times 10^{-15}$. However, determining the error locations of a stall pattern increases the decoder complexity.

2. Häger and Pfister [41] propose an anchor-based bit decoding method, which performance outperforms [8]. However, this method suffers from increased complexity as the anchors need to be tracked during iterative decoding.

3. Lei et al. [42] include marked bits as part of the decoding procedure. A fraction of the received bits are marked to be either "highly reliable" or "highly unreliable" by setting a threshold on the log-likelihood ratios (LLRs) of the channel output. The decoder detects miscorrection if any of the highly reliable bits is flipped while decoding. The decoder then flips the unreliable bits and repeats the decoding procedure. Experimental verification shows that the algorithm can achieve up to 0.30 dB SNR gain [43]. However, additional bitflow into the chip is required to describe the locations of the highly reliable and unreliable bits.

In addition, it has been shown that erasure declaration yields some improvement to the decoding performance in other coding schemes, e.g. Hermitian codes in [44] and Reed-Solomon codes in [45]. Further, [46] shows that there exists an optimal erasure threshold for block codes, convolutional codes, and LDPC codes. Also, [47] lists some approximations of the optimum erasure threshold in the context of generalized minimum distance decoding.

Our proposed method is similar to [42] and [43]. First, we mark some bits to be erased by setting a threshold on the LLRs of the channel output. Then, we perform error-and-erasure decoding for each constituent codeword in the iterative decoding method. The details of the implementation are described in the following sections.

4.2 Channel Model

For simplicity, we use a Binary Phase Shift Keying (BPSK) constellation over AWGN channel for declaring erasures. Let $\eta \geq 0$ be the erasure threshold, and let x be the transmitted symbol, which has value in $\{+1, -1\}$. Also, given noise variance σ^2, let y be the received value, i.e., $y = x + \nu$, where $\nu \sim \mathcal{N}(0, \sigma^2)$. We then assign $\hat{x}$ as the quantized

value of y as follows:

$$\hat{x} = \begin{cases} 0 & \text{if } y > \eta \\ 1 & \text{if } y < -\eta \\ ? & \text{otherwise} \end{cases}$$

Figure 4.1 shows the decision region of a BPSK constellation. As shown in the diagram, when y is close to zero we will declare erasure. On the other hand, if y is highly biased we will deem it "reliable" and declare 0 or 1 based on which bin it falls into. Note that setting $\eta = 0$ is equivalent to allowing errors only. Also, observe that the error and erasure probabilities depend on η and σ^2. The error and erasure probabilities, given by $e(\eta, \sigma^2)$ and $\epsilon(\eta, \sigma^2)$, respectively, are

$$\begin{aligned} e(\eta, \sigma^2) =& \frac{1}{2} \operatorname{erfc}\left(\frac{1+\eta}{\sqrt{2\sigma^2}}\right), \\ \epsilon(\eta, \sigma^2) =& \frac{1}{2} \operatorname{erfc}\left(\frac{1-\eta}{\sqrt{2\sigma^2}}\right) - \frac{1}{2} \operatorname{erfc}\left(\frac{1+\eta}{\sqrt{2\sigma^2}}\right) = \frac{1}{2} \operatorname{erfc}\left(\frac{1-\eta}{\sqrt{2\sigma^2}}\right) - e(\eta, \sigma^2). \end{aligned} \tag{4.1}$$

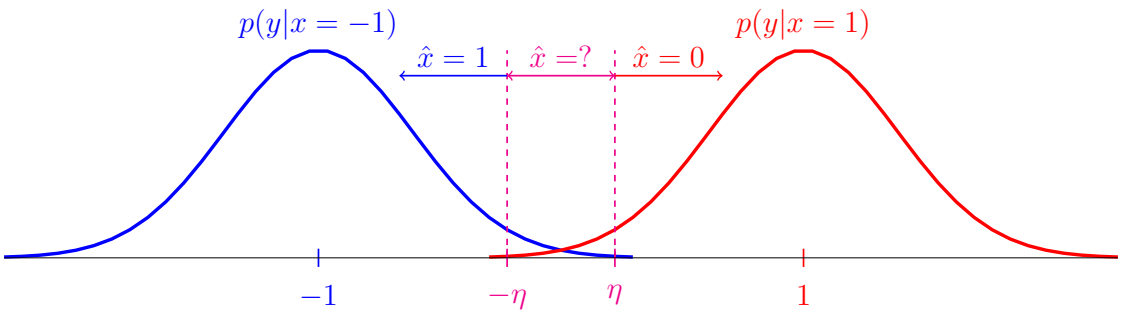

Figure 4.1: Decision regions for BPSK constellation over the AWGN channel

Now suppose that we have a code of length n with minimum distance d. A codeword of such a code can be corrected using a bounded distance decoder if and only if

$$s + 2t < d \tag{4.2}$$

where s and t respectively denote the number of erasures and errors. Therefore, the probability of receiving a codeword that cannot be corrected is

$$\mathcal{P}(n, d, \eta, \sigma^2) = \sum_{\substack{s,t \\ s+2t \geq d}} \binom{n}{s} \binom{n-s}{t} e(\eta, \sigma^2)^t \epsilon(\eta, \sigma^2)^s \left(1 - e(\eta, \sigma^2) - \epsilon(\eta, \sigma^2)\right)^{n-s-t} \tag{4.3}$$

We are looking for the optimum erasure threshold η^* for any given noise variance σ^2, i.e.

$$\eta^* = \arg\min_{\eta} \mathcal{P}(n, d, \eta, \sigma^2) \tag{4.4}$$

Figure 4.2 shows $\mathcal{P}(950, 8, \eta, \sigma^2)$ for different values of SNR. The optimum η for each SNR value lies around 0.1. However, considerable gain can be seen in the high SNR regime as shown in Figure 4.3.

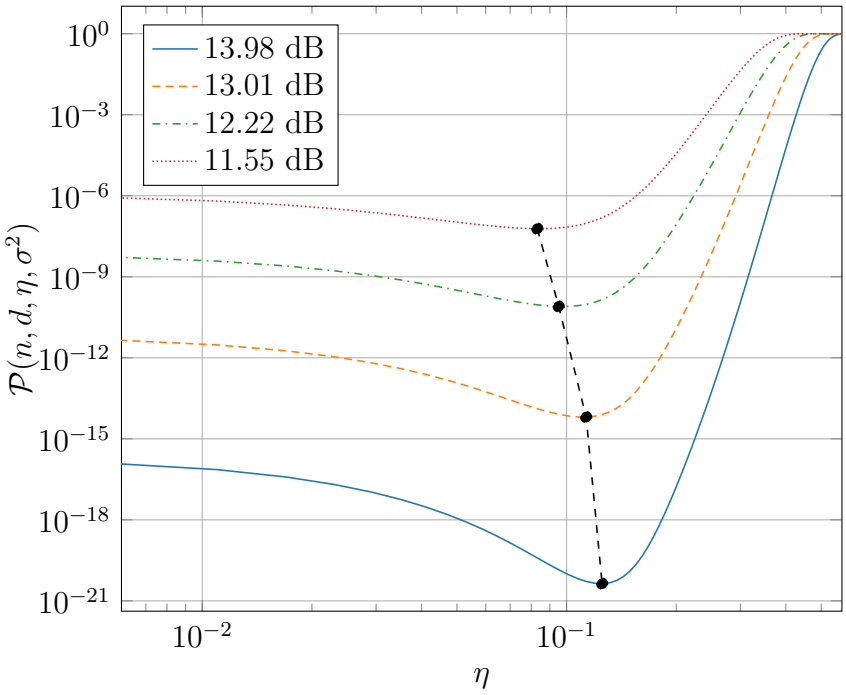

Figure 4.2: $\mathcal{P}(n, d, \eta, \sigma^2)$ versus η for $n = 950$ and $d = 8$ over different SNR values. The optimum η for each SNR is also marked.

4.2.1 Threshold on Log-Likelihood Ratios

Equivalently, we can also set the threshold on the LLR of the channel outputs. The LLR of a channel output y is defined as

$$\mathrm{LLR}(y) = \ln \frac{p_{Y|X}(y|x = 1)}{p_{Y|X}(y|x = -1)}.$$

In similar fashion to the binning method introduced in Section 4.2, if the LLR is highly positive, then we will declare the value of that symbol to be 0. If it is highly negative, then we will declare 1. Otherwise, we will declare an erasure symbol.

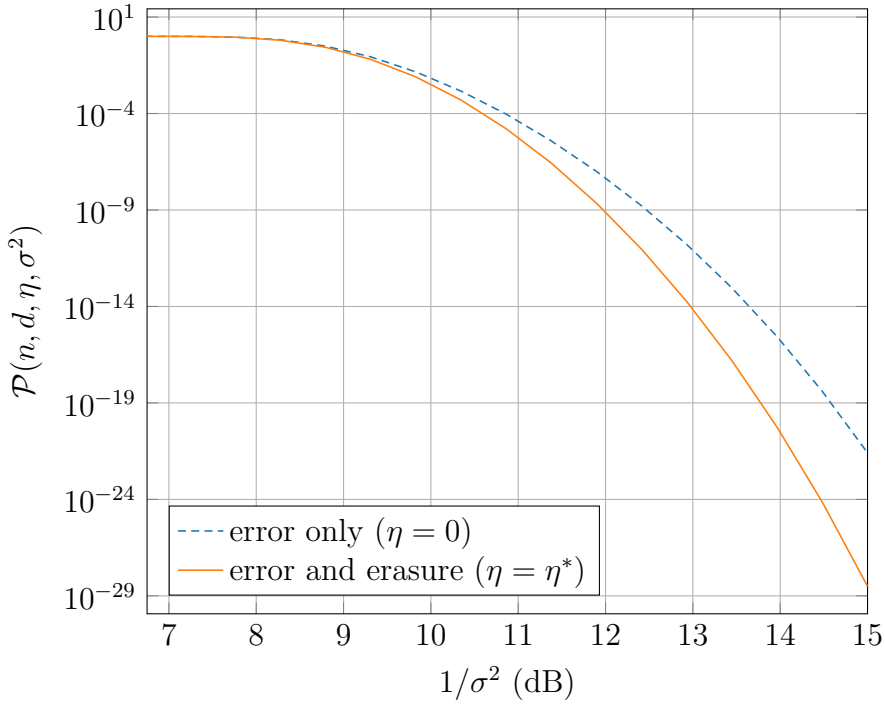

Figure 4.3: $\mathcal{P}(n, d, \eta, \sigma^2)$ versus SNR for $n = 950$ and $d = 8$ with error only and error-and-erasure.

For a binary AWGN channel with zero mean and noise variance σ^2, the density function of the resulting LLR is

$$L(y, \sigma^2) = \sqrt{\frac{\sigma^2}{8\pi}} \exp\left(-\frac{\left(y - \frac{2}{\sigma^2}\right)^2}{8/\sigma^2}\right).$$

The function $L(y, \sigma^2)$ is a Gaussian with mean $\frac{2}{\sigma^2}$ and variance $\frac{4}{\sigma^2}$ [48]. Thus, the corresponding error and erasure probabilities are respectively

$$\begin{aligned} e_{\mathrm{L}}(\eta, \sigma^2) =& \frac{1}{2} \operatorname{erfc}\left(\frac{\frac{2}{\sigma^2} + \eta}{\sqrt{8/\sigma^2}}\right), \\ \epsilon_{\mathrm{L}}(\eta, \sigma^2) =& \frac{1}{2} \operatorname{erfc}\left(\frac{\frac{2}{\sigma^2} - \eta}{\sqrt{8/\sigma^2}}\right) - \frac{1}{2} \operatorname{erfc}\left(\frac{\frac{2}{\sigma^2} + \eta}{\sqrt{8/\sigma^2}}\right) = \frac{1}{2} \operatorname{erfc}\left(\frac{\frac{2}{\sigma^2} - \eta}{\sqrt{8/\sigma^2}}\right) - e_{\mathrm{L}}(\eta, \sigma^2). \end{aligned} \tag{4.5}$$

It is also worth noting that from (4.1) and (4.5), if $\eta_2 = \frac{2}{\sigma^2}\eta_1$, then

$$e(\eta_1, \sigma^2) = e_L(\eta_2, \sigma^2) \quad \text{and} \quad \epsilon(\eta_1, \sigma^2) = \epsilon_L(\eta_2, \sigma^2).$$

Thus, setting the threshold on channel outputs and LLRs are equivalent with proper

scaling of the threshold.

The optimum threshold, however, cannot be obtained analytically in product and staircase codes due to their structures. Instead, we rely on simulation results. The next section will describe the simulation setups and its results.

4.3 Simulation Results

We simulated the error-and-erasure decoding of product codes and staircase codes. For product codes, we use a genie-aided, miscorrection-free constituent decoder. For each constituent codeword, if the number of errors and erasures satisfies (4.2), then the decoder will correct the corresponding constituent code. Otherwise, the decoder will do nothing. We use a triple-error-correcting BCH decoder for staircase codes with the "decode twice" method described in [49]: if a constituent codeword is affected by an erasure, we decode twice: first time with all erased symbols replaced with 0's, second time with 1's. The codeword with fewer number of errors corrected outside the erased symbols is chosen to be the 'correct' codeword.

4.3.1 Product Codes

We simulated the error-and-erasure decoding of product codes of sizes 500×500 and 1000×1000 with minimum distance 8 and maximum decoding iteration of 10. We set the thresholds on the outputs of the binary AWGN channel. Figures 4.4 and 4.5 show the effect of erasure declaration on the decoding performance. We also compare the decoding performance of error-and-erasure with the error-only case on Figure 4.6.

4.3.2 Staircase Codes

We simulated the error-and-erasure decoding of staircase codes of staircase block sizes 125×125 (rate 0.808), 250×250 (rate 0.892), and 400×400 (rate 0.925) and a triple-error-correcting constituent code, which can correct up to $d = 8$. Unlike the simulations for product codes above, we set the threshold on the LLRs of the output of the channel. Figures 4.7, 4.8, and 4.9 show the BER performance against the erasure threshold η. We then compare the performance against the error-only case, shown on Figure 4.10. By picking the appropriate value for the erasure threshold, we can achieve around $0.10 \sim 0.15$ dB gain compared to the error-only case.

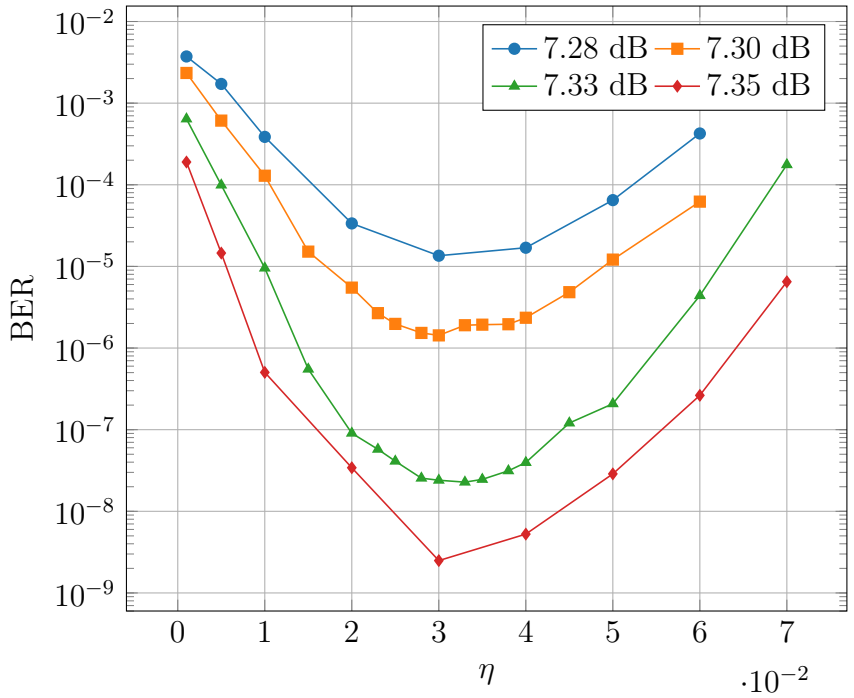

Figure 4.4: BER versus erasure threshold for product code of size 500×500 over different SNR values.

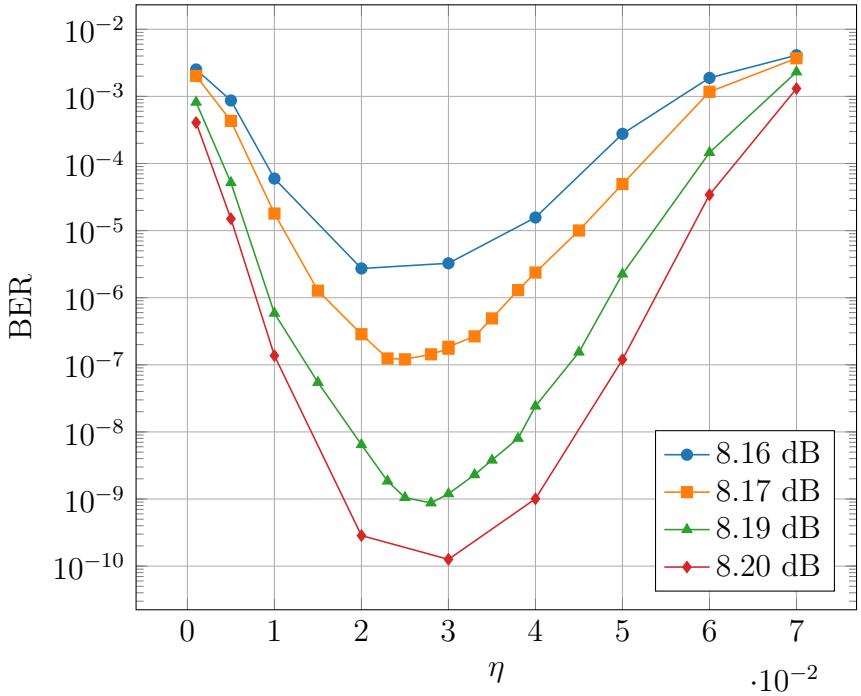

Figure 4.5: BER versus erasure threshold for product code of size 1000×1000 over different SNR values.

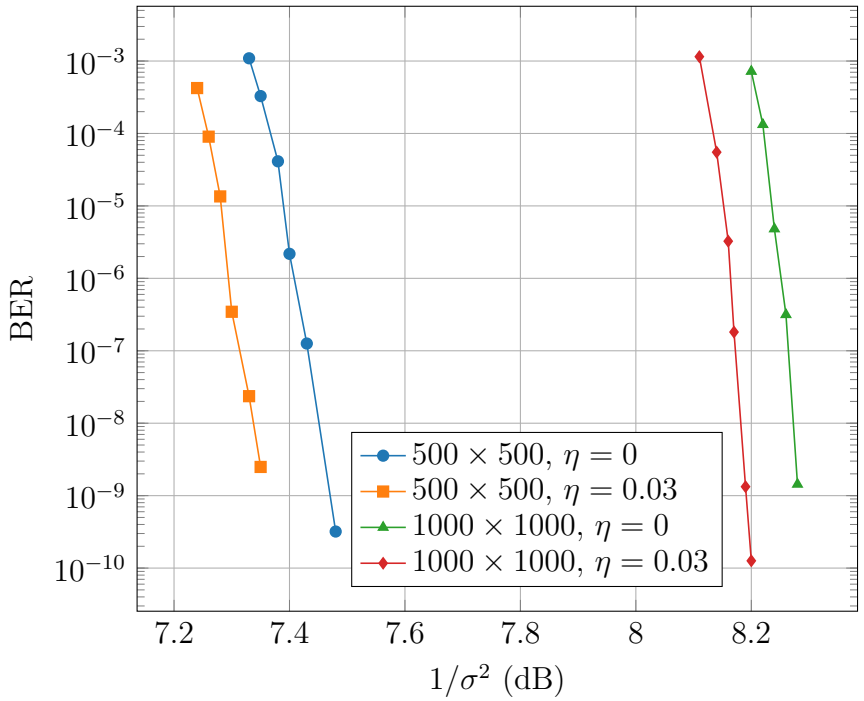

Figure 4.6: BER versus SNR for product codes of sizes 500×500 and 1000×1000 with error only and error-and-erasure.

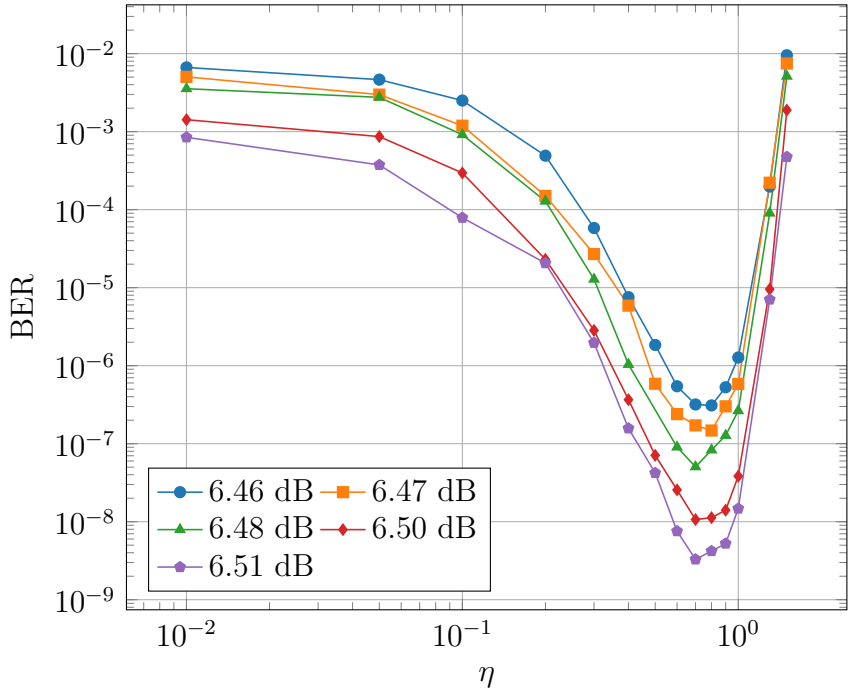

Figure 4.7: BER versus erasure threshold for staircase codes with staircase block of size 125×125 over different SNR values.

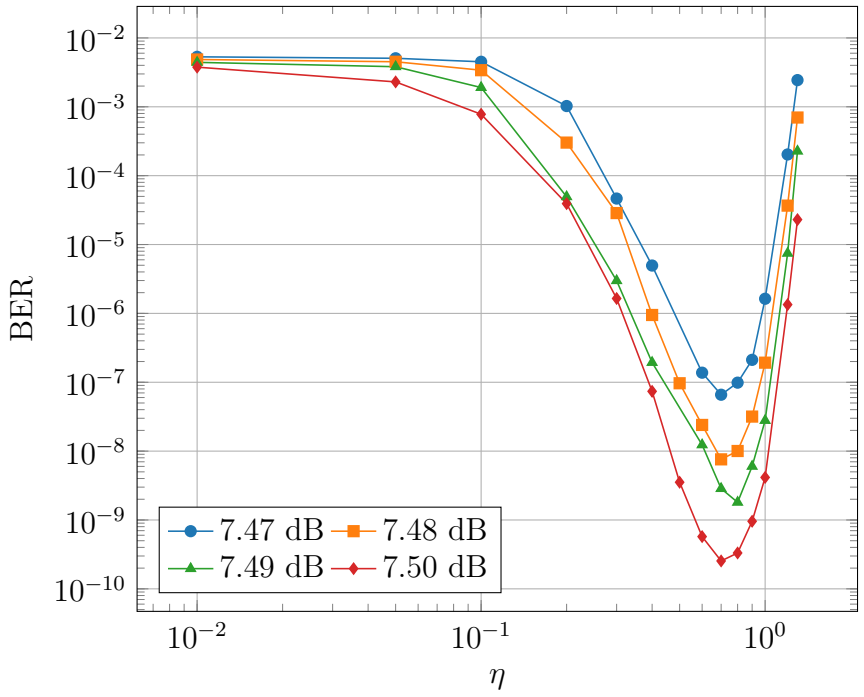

Figure 4.8: BER versus erasure threshold for staircase codes with staircase block of size 250×250 over different SNR values.

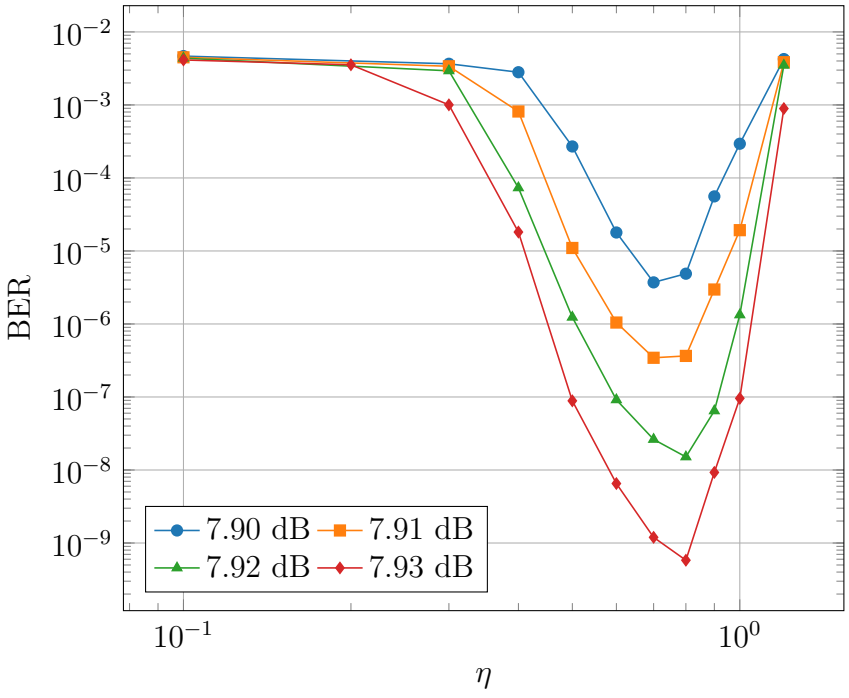

Figure 4.9: BER versus erasure threshold for staircase codes with staircase block of size 400×400 over different SNR values.

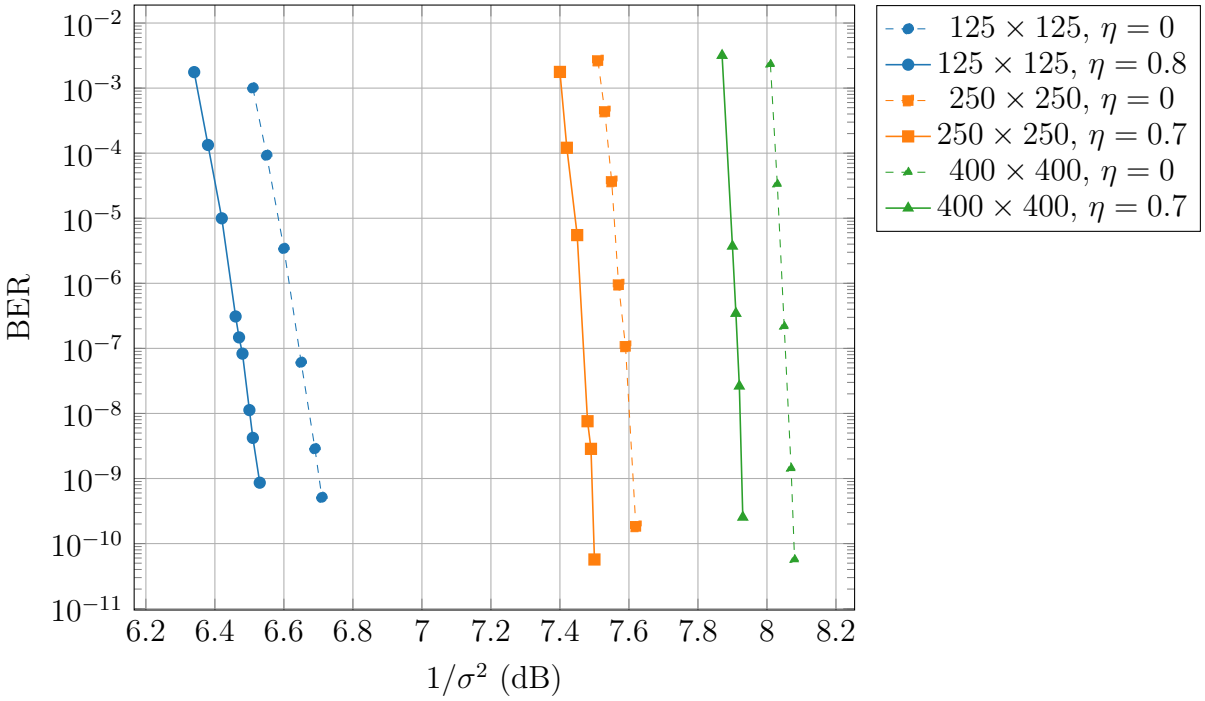

Figure 4.10: BER versus SNR for staircase codes with staircase blocks of sizes 125×125, 250×250, and 400×400 with error only and error-and-erasure.

4.4 Data Flow and Decoding Complexity

We modified the existing staircase decoder so that the outer decoder can decode errors and erasures. For each symbol, the LDPC decoder outputs the corresponding LLR, which will be then be quantized to zero or one depending on the LLR value. In addition, if the LLR is between $-\eta$ and η, then we deem the symbol to be unreliable and also output the coordinate of the symbol. Thus, the modified staircase decoder will take the quantized LLRs as well as the coordinates of the 'erased' symbols. The high-level decoding procedure is summarized in Figure 4.11.

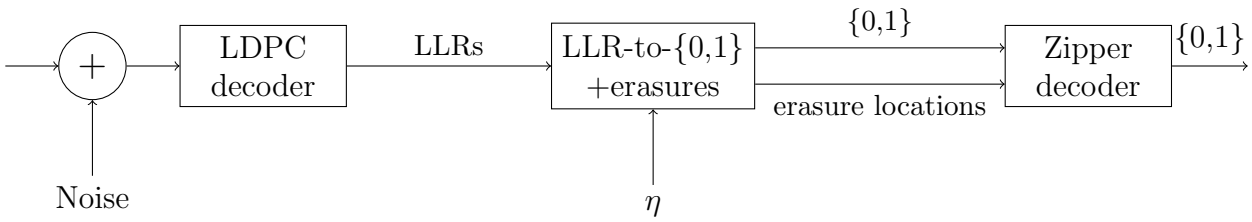

Figure 4.11: Block diagram of the inner and outer decoders showing data flow.

The decoding complexity consideration described in Section 2.4.3 also applies here, but we have additional criteria for error-and-erasure decoding. For example, the number

of additional bits to describe the erasure locations depends on the size of a staircase block, the operating SNR, as well as η. Suppose that the staircase block is of size $m \times m$, then we will require $2 \times \lceil \log_2 m \rceil$ to describe the row and column indices of an erasure. Hence, the number of additional bits required per staircase block on average is $2 \times \lceil \log_2 m \rceil \times m^2 \times \epsilon_L(\eta, \sigma^2)$. The relative increase of the data flow per staircase block on average is therefore given by

$$\frac{2 \times \lceil \log_2 m \rceil \times m^2 \times \epsilon(\eta, \sigma^2)}{m^2} = 2 \times \lceil \log_2 m \rceil \times \epsilon_L(\eta, \sigma^2)$$

Table 4.1 shows the average relative increase in the data flow over various parameters. The data flow increases by about $26 \sim 27\%$ for staircase codes of rate 0.808. However, the data flow only increases by about $13 \sim 15\%$ for rate 0.892 and $10 \sim 11\%$ for rate 0.925.

Table 4.1: Increase of number of bits for error-and-erasure staircase decoding.

Size	SNR (dB)	η^*	$e_L(\eta^*, \sigma^2)$	$\epsilon_L(\eta^*, \sigma^2)$	Relative increase
125×125	6.47	0.8	1.083×10^{-2}	1.683×10^{-2}	0.269
125×125	6.50	0.8	1.064×10^{-2}	1.652×10^{-2}	0.264
125×125	6.51	0.8	1.058×10^{-2}	1.641×10^{-2}	0.263
250×250	7.48	0.7	5.971×10^{-3}	7.301×10^{-3}	0.131
250×250	7.49	0.8	5.582×10^{-3}	8.329×10^{-3}	0.150
250×250	7.50	0.7	5.885×10^{-3}	7.197×10^{-3}	0.130
400×400	7.90	0.7	4.344×10^{-3}	5.242×10^{-3}	0.105
400×400	7.91	0.7	4.310×10^{-3}	5.199×10^{-3}	0.104
400×400	7.92	0.8	4.030×10^{-3}	5.923×10^{-3}	0.118

Further, the "decode twice" method increases the number of decoding operations required, with the worst-case scenario being each constituent codeword having at least one erasure, effectively doubling the number of decoding operations for the first iteration of the decoding window. The subsequent iterations, however, will have some erasures resolved from the previous iteration so the "decode twice" operation will not take place as often. Figure 4.12 shows the cumulative frequency of the "decode twice" operations taking place for each row index for staircase codes with triple-error-correcting BCH constituent code and block sizes 125×125 and 400×400. In both cases, it shows that about 94% of the "decode twice" operations take place in the newest staircase block. In addition, Figure 4.13 shows the fraction of the "decode twice" operation over different values of η. For $\eta = 0.7$, the constituent decoder sees erasures, and therefore the decoder "decodes twice," about 40% of the time.

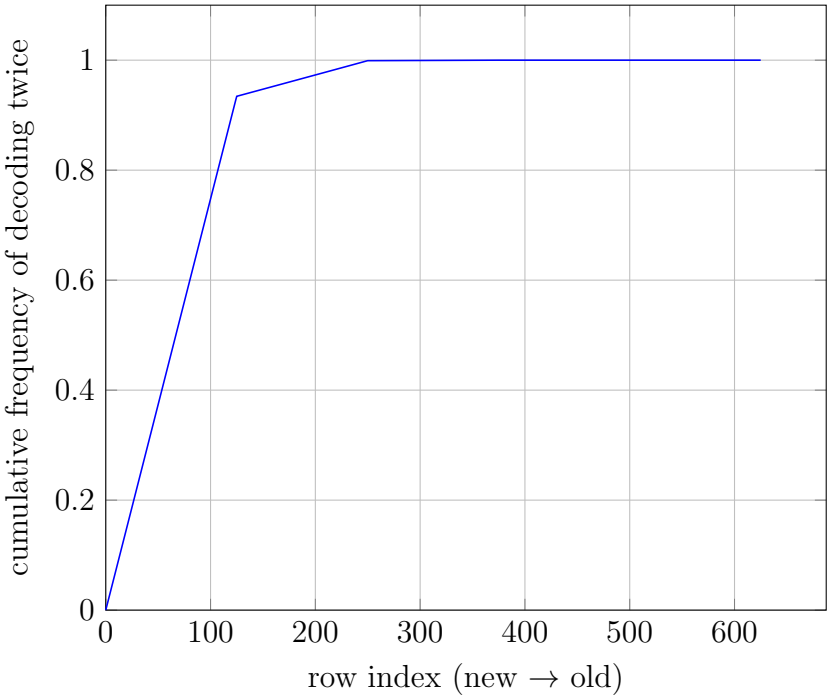

Figure 4.12: Cumulative frequency of "decoding twice" over all row indices for staircase codes with $m = 125$ and triple-error-correcting constituent codes.

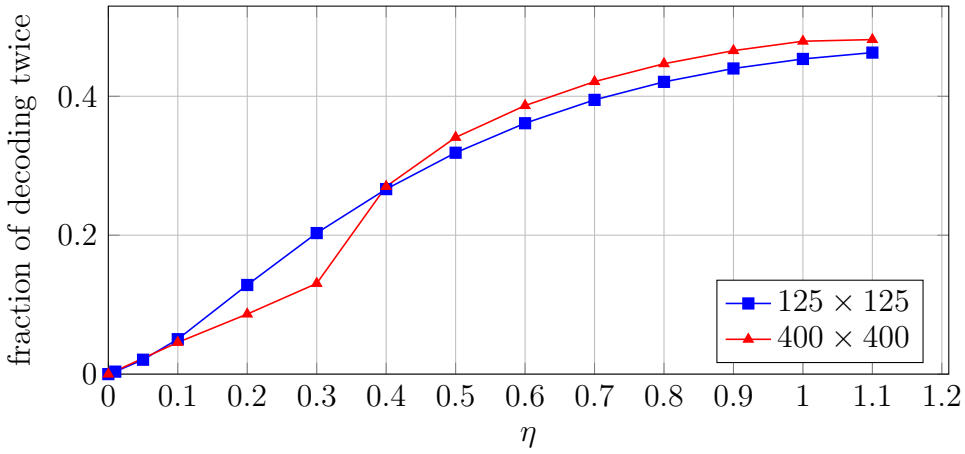

Figure 4.13: Fraction of "decode twice" in an error-and-erasure staircase decoding.

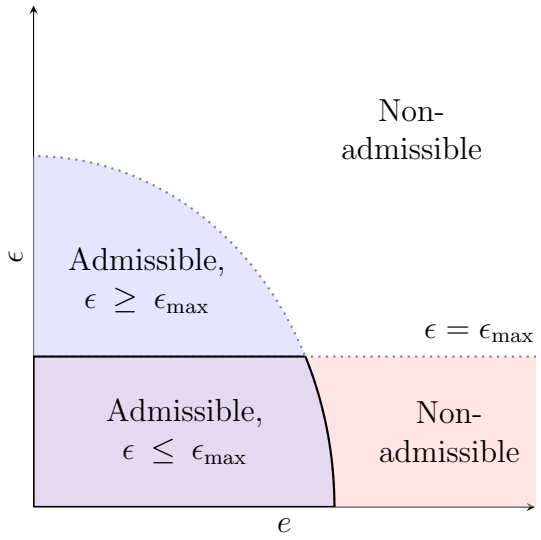

Figure 4.14: Diagram showing admissible and non-admissible sets as well as the $\epsilon_{\max}$ constraint.

4.5 Concatenation with Inner Codes

So far we have shown the performance of the channel with binary input and ternary output, which is equivalent to having a "rate-1" inner code. We can extend the analysis with an actual inner code, such as an LDPC code.

Suppose that we know the distribution of the LLRs of the inner code. For example, the LLR distribution at the output of the LDPC decoder is a symmetric Gaussian assuming AWGN channel model. We would like to have an error-and-erasure probability (e, ϵ) such that the outer staircase decoder corrects to at most 10^{-15} bit error rate. We call such error-erasure pair to be *admissible*. Let $\mathcal{A}$ denote the set of all admissible error-erasure pairs. In addition, suppose that we do not want to have too many erasure symbols as having more erasure symbols will increase the data flow and decoding complexity. This implies that we impose another constraint such as having a maximum allowable erasure probability $\epsilon_{\max}$. Figure 4.14 shows a diagram of the admissible set as well as the $\epsilon_{\max}$ constraint. We require the operating point of a code to be at the intersection of the admissible and $\epsilon \leq \epsilon_{\max}$ regions.

4.6 Stall Patterns and Error Floor Approximation

The definition of a stall pattern in error-and-erasure decoding is similar to the error-only case as defined in Chapter 3. The difference lies on the condition of the correctability of a row or constituent codeword. Recall that a row is correctable if it contains s erasures

and t errors such that

$$s + 2t < d,$$

where d denotes the constituent code minimum distance.

The set of error-and-erasure stall patterns of zipper codes with consituent code of minimum distance d contains the set of stall patterns of the error-only case where the constituent code is $\left\lfloor \frac{d-1}{2} \right\rfloor$-error-correcting. The difference here is that some stall patterns include erasures. Figure 4.15 shows some examples of stall patterns in error-and-erasure scheme of a staircase code with $d = 8$.

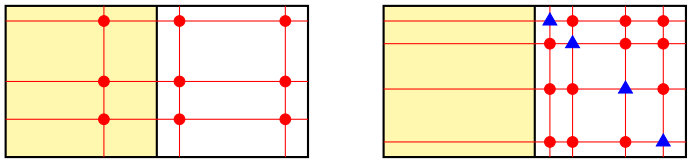

Figure 4.15: Examples of stall patterns in error-and-erasure scheme of a staircase code with $d = 8$. The dots and triangles respectively denote errors and erasures.

Similar to the error-only case, we approximate the error floor by considering the enumeration of the set of stall patterns $\mathcal{S}$ in a decoding window of M rows and m columns. Let e, ϵ respectively denote the error and erasure probability. Given a stall pattern S, partition it into S_e and S_ϵ as the subsets containing only errors and only erasures, respectively. Thus,

$$S_e \cap S_\epsilon = \varnothing \quad \text{and} \quad S_e \cup S_\epsilon = S.$$

Then by using union bound, the error floor is approximated to be

$$\text{BER}_{\text{floor}} \leq \frac{1}{M \times m} \sum_{S \in \mathcal{S}} |S| \epsilon^{|S_\epsilon|} e^{|S_e|}.$$

Likewise, we can also enumerate over the number of errors and erasure in the stall patterns that occur,

$$\text{BER}_{\text{floor}} \leq \frac{1}{M \times m} \sum_{(s,t) \in \mathcal{L}} N_{s,t} \epsilon^s e^t,$$

where

$$\mathcal{L} = \{(s,t) : \text{there exists a stall pattern in } \mathcal{S} \text{ with } s \text{ erasures and } t \text{ errors}\},$$

and $N_{s,t}$ denotes the number of stall patterns in $\mathcal{S}$ with s erasures and t errors.

We will also define dominant stall patterns of error-and-erasure code analogous to

Definition 9 of the error-only case. We say that a stall pattern of size (s^*, t^*) is dominant if

$$N_{s^*,t^*}\epsilon^{s^*}e^{t^*} \geq N_{s,t}\epsilon^{s}e^{t} \quad \text{for all} \quad (s,t) \in \mathcal{L}.$$

If ϵ and e are sufficiently small, we can assume that the dominant stall patterns are the minimum-sized one. Therefore, the error floor estimate can be simplified to

$$\frac{1}{M \times m} N_{s^*,t^*}\epsilon^{s^*}e^{t^*},$$

where (s^*, t^*) corresponds to the number of erasures and errors in the minimum-sized error-and-erasure stall patterns.

Remark 7. Theorem 4 also applies in the error-and-erasure case, i.e., the size of the minimum-sized stall patterns is lower bounded by $\frac{1}{2}(t+1)(t+2)$, where in this case $t = \left\lfloor \frac{d-1}{2} \right\rfloor$. However, the number of errors and erasures in the minimum-sized stall patterns depend on the value of d. If d is even, the minimum-sized stall patterns will contain only errors and therefore will be equivalent to the error-only case. To see this, note that the 'adversarial power' of an error is twice that of an erasure. By Theorem 3, any replacement of an error with erasure also affects a row that contains exactly $d/2$ errors, and so we will need to introduce at least another error or erasure for such rows to keep them undecodable. The replacement effectively increases the stall pattern size. However, if d is odd, we may be able to form minimum-sized stall patterns that contain some erasures as shown in Figure 4.16. Replacing erasures with errors in such cases will not increase the stall pattern size.

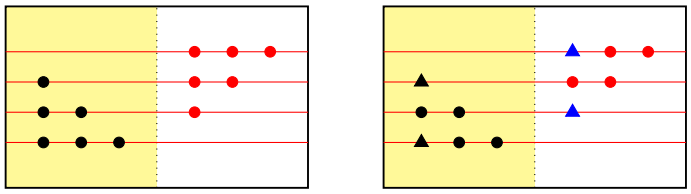

Figure 4.16: Examples of minimum-sized error-and-erasure stall patterns of diagonal zipper code with $d = 6$ (left) and $d = 5$ (right). The circles denote errors and triangles denote erasures.

Also, there are some cases where a stall pattern can be corrected even if $s + 2t \geq d$. To illustrate this, we will show an error-and-erasure stall pattern of a staircase code with $d = 6$ in Figure 4.17. Suppose that we replace each erasure symbol with either a correct symbol or erroneous symbol. Then observe that the pattern becomes correctable if, for

example, we replace all erasure symbols with the correct ones. By exhaustive search, we have determined that there are 151 correctable patterns in this case.

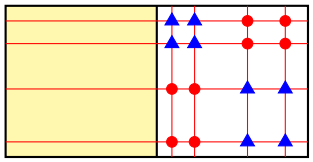

Figure 4.17: Error-and-erasure stall pattern of staircase code with $d = 6$. The errors are denoted with circles and the erasures are denoted with triangles.

Suppose that we replace each erasure symbol with a correct or erroneous symbol with equal probability in an i.i.d. manner. The probability that the stall pattern is correctable is given by

$$\frac{151}{2^8} \approx 0.590.$$

Figure 4.18 shows the graph representation of the example above. The solid edges represent the errors and the dotted edges represent the erasures. Removing a dotted edge is equivalent to replacing the corresponding erasure with the correct symbol. Similarly, replacing a dotted edge with an solid edge is equivalent to replacing the corresponding erasure with an erroneous symbol. We can determine if the new, error-only graph is correctable by repeatedly deleting nodes with degree $\leq t$ and if the resulting graph is empty, then the new error-only graph is correctable. Otherwise, the resulting graph converges to a stall pattern that is at most as big as the original error-and-erasure stall pattern.

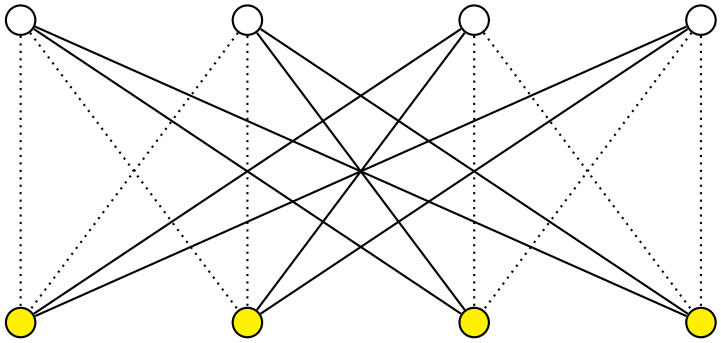

Figure 4.18: Graph representation of the stall pattern shown in Figure 4.17. Solid edges denote errors and dotted edges denote erasures.

Observe that if we remove all erasures in the stall pattern above, we are left with an error pattern that is correctable in the error-only scheme with double-error-correcting constituent code. We call such stall pattern *conditionally correctable*.

Theorem 9. *Let S be an error-and-erasure stall pattern of a zipper code with a bijective scattering interleaver map. Then S is conditionally correctable if and only if S_e is correctable.*

Proof. If S_e is not correctable, then it contains a stall pattern of size at most $|S_e|$. This implies that S is not a minimal stall pattern and therefore not correctable. On the other hand, if S_e is correctable, then S can be corrected if we replace all erasure symbols with the correct ones. □

Concluding Remarks

In this chapter, we have demonstrated that declaring erasures in zipper decoding yield coding gain of around 0.1 dB compared to the error-only case. Adding erasures increases the data flow into the chip by around 26 percent for staircase codes with block size 125×125 and around $10 \sim 11$ percent for 400×400.

Chapter 5

Conclusion and Future Work

In this thesis, we have described zipper codes as a framework of spatially-coupled product-like codes. We have compared and formulated some well-known codes in our zipper codes framework. In addition, we have also introduced tiled and delayed diagonal zipper codes as a new type of code that are memory-efficient in the encoding and decoding procedure, hence allowing us to use high-rate codes. We have also analyzed the types and characteristics of stall patterns that arise from certain types of zipper codes and how they translate to the error floor analysis. Finally, we have also implemented an error-and-erasure decoding procedures for staircase codes. Software simulation results have shown that declaring erasures yield considerable coding gain when compared to the error-only case.

A possible future direction may include examining the performance of zipper codes in concatenated schemes. We are interested in using a very-high rate outer zipper code with lower rate inner LDPC codes. Also, most analysis that we have performed for zipper codes concerns the case with identical constituent codes. It may be worth to investigate the cases where more than one type of constituent code is used.

In addition, all numerical results presented in this thesis are software-based, which may not reflect the actual performance, especially since we cannot get the BER down to 10^{-15}. Hence, we are also interested in examining the performance of the proposed scheme when implemented in hardware. We anticipate that it yields comparable performance to state-of-the-art while using comparatively less memory.

Similarly, we may analyze the performance of hardware implementation of our proposed error-and-erasure decoding of zipper codes. We would like to determine the actual optimum erasure threshold of the LLRs with different type of decoder implementation, e.g., Berlekamp-Welch-styled or syndrome-based decoding. In addition, we are also interested in the hardware implementation of the concatenated LDPC-zipper error-and-

erasure decoding scheme.

Finally, we have assumed that we decode each constituent codeword in a decoding window in a serial manner. This assumption is made so that we do not have codeword or syndrome conflicts due to race conditions and other read/write conflicts. We are interested in investigating whether parallel iterative decoding is possible in a zipper decoder. More specifically, we would like to investigate the criteria of interleaver maps that can give rise to parallel zipper decoding.

Bibliography

[1] L. M. Zhang and F. R. Kschischang, "Staircase codes with 6% to 33% overhead," *J. Lightw. Technol.*, vol. 32, no. 10, pp. 1999–2002, May 2014.

[2] T. Mizuochi, "Recent progress in forward error correction and its interplay with transmission impairments," *IEEE J. Sel. Topics Quantum Electron.*, vol. 12, no. 4, pp. 544–554, Jul. 2006.

[3] E. Agrell, M. Karlsson, A. R. Chraplyvy, D. J. Richardson, P. M. Krummrich, P. Winzer, K. Roberts, J. K. Fischer, S. J. Savory, B. J. Eggleton, M. Secondini, F. R. Kschischang, A. Lord, J. Prat, I. Tomkos, J. E. Bowers, S. Srinivasan, M. Brandt-Pearce, and N. Gisin, "Roadmap of optical communications," *J. Optics*, vol. 18, no. 6, pp. 23–24, 2016.

[4] D. Ouchi, K. Kubo, T. Mizuochi, Y. Miyata, H. Yoshida, H. Tagami, K. Shimizu, T. Kobayashi, K. Shimomura, K. Onohara, and K. Motoshima, "A fully integrated block turbo code FEC for 10 Gb/s optical communication systems," in *Proc. 2006 OFC*, pp. 1–3.

[5] I. B. Djordjevic, L. Xu, and T. Wang, "Optical LDPC decoders for beyond 100 Gbits/s optical transmission," *Opt. Lett.*, vol. 34, no. 9, pp. 1420–1422, May 2009.

[6] Y. Miyata, K. Kubo, H. Yoshida, and T. Mizuochi, "Proposal for frame structure of optical channel transport unit employing LDPC codes for 100 Gb/s FEC," in *Proc. 2009 OFC*, pp. 1–3.

[7] A. Tychopoulos, O. Koufopaulou, and I. Tomkos, "FEC in optical communications - a tutorial overview on the evolution of architectures and the future prospects of outband and inband FEC for optical communications," *IEEE Circuits Devices Mag.*, vol. 22, pp. 79 – 86, 12 2006.

[8] B. P. Smith, A. Farhood, A. Hunt, F. R. Kschischang, and J. Lodge, "Staircase codes: FEC for 100 Gb/s OTN," *J. Lightw. Technol.*, vol. 30, no. 1, pp. 110–117, 2012.

[9] B. Smith, I. Lyubomirsky, and S. Bhoja. (2017) Leveraging 400G ZR FEC technology. IEEE 802.3 Beyond 10km Optical PHYs Study Group. [Online]. Available: http://www.ieee802.org/3/B10K/public/17_11/lyubomirsky_b10k_01_1117.pdf

[10] I. Kaminow, T. Li, and A. E. Willner, *Optical Fiber Telecommunications Volume VIA, Sixth Edition: Components and Subsystems*, 6th ed. Orlando, FL, USA: Academic Press, Inc., 2013.

[11] M. Barakatain and F. R. Kschischang, "Low-complexity concatenated LDPC-staircase codes," *J. Lightw. Technol.*, vol. 36, no. 12, pp. 2443–2449, Jun. 2018.

[12] L. M. Zhang and F. R. Kschischang, "Low-complexity soft-decision concatenated LDGM-staircase FEC for high-bit-rate fiber-optic communication," *J. Lightw. Technol.*, vol. 35, no. 18, pp. 3991–3999, Sep. 2017.

[13] J. Justesen, K. J. Larsen, and L. A. Pedersen, "Error correcting coding for OTN," *IEEE Commun. Mag.*, vol. 48, no. 9, pp. 70–75, Sep. 2010.

[14] M. Weiner, M. Blagojevic, S. Skotnikov, A. Burg, P. Flatresse, and B. Nikolic, "27.7 A scalable 1.5-to-6Gb/s 6.2-to-38.1mW LDPC decoder for 60GHz wireless networks in 28nm UTBB FDSOI," in *2014 IEEE Int. Solid-State Circuits Conf.*, pp. 464–465.

[15] T. Ou, Z. Zhang, and M. C. Papaefthymiou, "27.6 an 821MHz 7.9Gb/s 7.3pJ/b/iteration charge-recovery LDPC decoder," in *2014 IEEE Int. Solid-State Circuits Conf.*, pp. 462–463.

[16] Y. Lee, H. Yoo, J. Jung, J. Jo, and I. Park, "A 2.74-pJ/bit, 17.7-Gb/s iterative concatenated-BCH decoder in 65-nm CMOS for NAND flash memory," *IEEE J. Solid-State Circuits*, vol. 48, no. 10, pp. 2531–2540, Oct. 2013.

[17] P. Elias, "Error-free coding," *IRE Trans. Inf. Theory*, vol. 4, no. 4, pp. 29–37, Sep. 1954.

[18] A. Wyner and R. Ash, "Analysis of recurrent codes," *IEEE Trans. Inf. Theory*, vol. 9, no. 3, pp. 143–156, Jul. 1963.

[19] Y. Jian, H. D. Pfister, and K. R. Narayanan, "Approaching capacity at high rates with iterative hard-decision decoding," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5752–5773, Sep. 2017.

[20] M. Lentmaier, I. Andriyanova, N. Hassan, and G. Fettweis, "Spatial coupling - a way to improve the performance and robustness of iterative decoding," in *Proc. 2015 Eur. Conf. Netw. and Comm.*, pp. 1–2.

[21] A. J. Feltström, D. Truhachev, M. Lentmaier, and K. S. Zigangirov, "Braided block codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 6, pp. 2640–2658, Jun. 2009.

[22] P. Northcott, "Cyclically interleaved dual BCH, with simultaneous decode and per-codeword maximum likelihood reconciliation," U.S. Patent 8 479 084, Jul., 2013.

[23] A. J. Feltström and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2181–2191, Sep. 1999.

[24] M. Lentmaier, A. Sridharan, D. J. Costello, and K. S. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5274–5289, Oct. 2010.

[25] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 803–834, Feb. 2011.

[26] L. Schmalen, D. Suikat, D. Rsener, V. Aref, A. Leven, and S. ten Brink, "Spatially coupled codes and optical fiber communications: An ideal match?" in *Proc. IEEE 16th Int. Workshop Signal Process. Adv. Wireless Commun.*, Jun. 2015, pp. 460–464.

[27] J. Justesen, "Performance of product codes and related structures with iterated decoding," *IEEE Trans. Commun.*, vol. 59, no. 2, pp. 407–415, Feb. 2011.

[28] F. Chang, K. Onohara, and T. Mizuochi, "Forward error correction for 100 G transport networks," *IEEE Commun. Mag.*, vol. 48, no. 3, pp. S48–S55, Mar. 2010.

[29] A. Leven and L. Schmalen, "Status and recent advances on forward error correction technologies for lightwave systems," *J. Lightw. Technol.*, vol. 32, no. 16, pp. 2735–2750, Aug. 2014.

[30] G. Tzimpragos, C. Kachris, I. B. Djordjevic, M. Cvijetic, D. Soudris, and I. Tomkos, "A survey on FEC codes for 100 G and beyond optical networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 209–221, Oct. 2016.

[31] M. Lentmaier and K. S. Zigangirov, "On generalized low-density parity-check codes based on Hamming component codes," *IEEE Commun. Lett.*, vol. 3, no. 8, pp. 248–250, Aug. 1999.

[32] *Forward error correction for high bit-rate DWDM submarine systems*, ITU-T Std. G.975.1, 2004.

[33] L. M. Zhang, D. Truhachev, and F. R. Kschischang, "Spatially coupled split-component codes with iterative algebraic decoding," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 205–224, Jan. 2018.

[34] C. P. M. J. Baggen and L. M. G. M. Tolhuizen, "On diamond codes," *IEEE Trans. Inf. Theory*, vol. 43, no. 5, pp. 1400–1411, Sep. 1997.

[35] T. Coe, "Continuously interleaved error correction," U.S. Patent 8 276 047, Sep., 2012.

[36] A. Sheikh, A. Graell i Amat, G. Liva, and F. R. Steiner, "Probabilistic amplitude shaping with hard decision decoding and staircase codes," *J. Lightw. Technol.*, vol. PP, no. 99, 2017.

[37] E. Cohen and N. Megiddo, "Recognizing properties of periodic graphs," in *Proc. Appl. Geom. and Discr. Math., Proc. DIMACS Workshop, Providence, RI, USA*, 1990, pp. 135–146.

[38] A. Bickle, "The k-cores of a graph," Ph.D. dissertation, Western Michigan University, 2010.

[39] L. Holzbaur, H. Bartz, and A. Wachter-Zeh, "Improved decoding and error floor analysis of staircase codes," *Designs, Codes and Cryptography*, vol. 87, no. 2, pp. 647–664, Mar. 2019.

[40] R. Roth, *Introduction to Coding Theory*. New York, NY, USA: Cambridge University Press, 2006.

[41] C. Häger and H. D. Pfister, "Miscorrection-free decoding of staircase codes," in *Proc. 2017 ECOC*, pp. 1–3.

[42] Y. Lei, A. Alvarado, B. Chen, X. Deng, Z. Cao, J. Li, and K. Xu, "Decoding staircase codes with marked bits," in *Proc. 2018 IEEE 10th ISTC*, pp. 1–5.

[43] B. Chen, Y. Lei, S. van der Heide, J. van Weerdenburg, A. Alvarado, and C. Okonkwo, "First experimental verification of improved decoding of staircase codes using marked bits," in *Proc. 2019 OFC*, pp. 1–3.

[44] M. Johnston and R. A. Carrasco, "Performance of hermitian codes using combined error and erasure decoding," *IEE Proc.-Commun.*, vol. 153, no. 1, pp. 21–30, Feb. 2006.

[45] S. W. Lee and B. V. K. V. Kumar, "Soft-decision decoding of Reed-Solomon codes using successive error-and-erasure decoding," in *Proc. IEEE GLOBECOM*, Nov. 2008, pp. 1–5.

[46] Y. H. Kwon, M. K. Oh, and D. J. Park, "Optimal erasure selection of M-ary PAM signaling for errors and erasures decoding algorithms," *IEEE Trans. Commun.*, vol. 56, no. 12, pp. 2071–2079, Dec. 2008.

[47] J. H. Weber, V. R. Sidorenko, C. Senger, and K. A. S. Abdel-Ghaffar, "Asymptotic single-trial strategies for GMD decoding with arbitrary error-erasure tradeoff," *Problems of Information Transmission*, vol. 48, no. 4, pp. 324–333, Oct. 2012.

[48] T. Richardson and R. Urbanke, *Modern Coding Theory*. New York, NY, USA: Cambridge University Press, 2008.

[49] S. Lin and D. J. Costello, *Error Control Coding, Second Edition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2004.