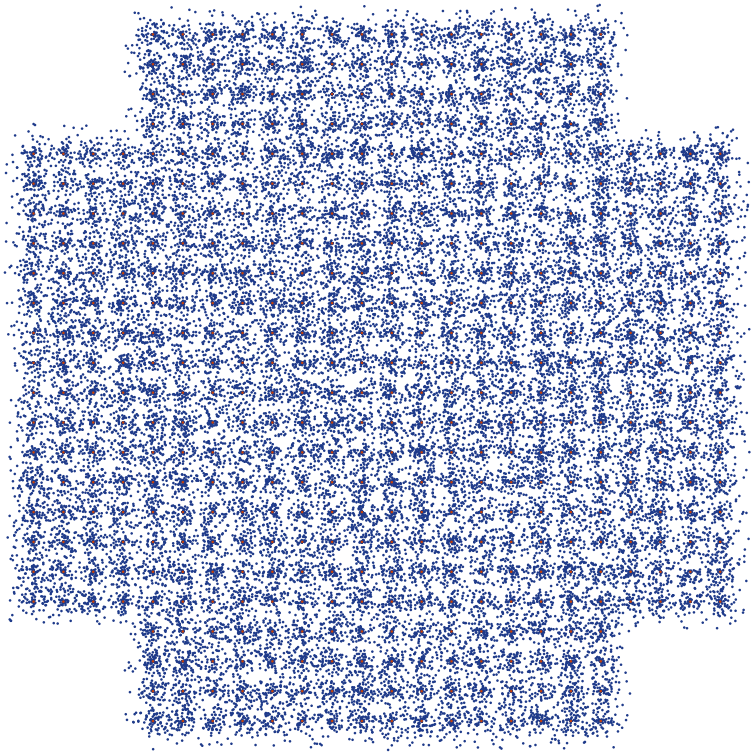

Impact of Non-Square Modulation Formats on Multi-Layer Error-Correction Systems

Master's Thesis in Embedded Electronic System Design

HAMPUS LANG
GALO SANCHEZ

Department of Microtechnology and Nanoscience
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025

Master's Thesis 2025

Impact of Non-Square Modulation Formats on Multi-Layer Error-Correction Systems

HAMPUS LANG
GALO SANCHEZ

Department of Microtechnology and Nanoscience
Chalmers University of Technology
Gothenburg, Sweden 2025

Impact of Non-Square Modulation Formats on Multi-Layer Error-Correction Systems
HAMPUS LANG
GALO SANCHEZ

Supervisor: Erik Börjeson, Department of Microtechnology and Nanoscience
Company advisors: Ali Mirani and Martin Malmström, Ericsson AB
Examiner: Per Larsson-Edefors, Department of Microtechnology and Nanoscience

Master's Thesis 2025
Department of Microtechnology and Nanoscience
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: X 512-QAM constellation with additive white Gaussian noise.

Typeset in LaTeX
Gothenburg, Sweden 2025

Impact of Non-Square Modulation Formats on Multi-Layer Error-Correction Systems
HAMPUS LANG
GALO SANCHEZ
Department of Microtechnology and Nanoscience
Chalmers University of Technology

Abstract

This work presents a set of algorithms for integrating non-square modulation formats with multi-layer forward error-correction (FEC) systems. This was done to improve the adaptive coding and modulation (ACM) capabilities of Ericsson's next-generation digital signal processing (DSP) product. Specifically, support for cross- (X) quadrature amplitude modulation (QAM) formats with 9 to 19 bits per symbol was implemented. This was achieved while respecting limitations imposed by the multi-layer FEC, consisting of low-density parity check and Reed Solomon error-correcting codes. The main contributions were made to the symbol mapper and demapper as well as the most significant bit decoders.

The algorithms were implemented in hardware using high-level synthesis, with the target platform being a field-programmable gate array (FPGA). Testing on the target FPGA was performed using a digital loop and an internal noise generator to produce bit error rate (BER) vs. signal-to-noise ratio (SNR) curves.

The results displayed a 1 dB SNR gain when switching from a higher-order square QAM format to a lower-order X QAM format with higher code rate. Since these two options provide the same capacity, it implies that it is more beneficial to reduce modulation order to improve BER performance compared to increasing FEC overhead when channel conditions degrade. Hence, the implementation of X QAM formats has the potential to improve the ACM flexibilty in Ericsson's DSP product, with only a small increase in hardware usage.

Keywords: Quadrature amplitude modulation, QAM, cross QAM, Forward error correction, Adaptive coding and modulation, FPGA, High-level synthesis.

Acknowledgements

First, we would like to thank our company advisors, Ali and Martin. Not only for their immense technical expertise, which they are more than happy to share, but also for welcoming us to Ericsson and making us feel like a part of the team.

Second, we want to extend a thanks to Erik, our Chalmers supervisor. His suggestions for ways of working and for improving this report have greatly improved the quality of our work. Furthermore, his contributions as a TA in the early courses of the master's program have been appreciated by many, including us.

Last, but of course not least, we want to give a special thanks to Per. Partly for his efforts in the role as examiner for this project, but mainly for coordinating the excellent master's program from which we are now graduating. The skills we have obtained during these last two years have benefited us greatly in producing this work, and will continue to do so for the rest of our careers.

Hampus Lang and Galo Sanchez, Gothenburg, June 2025

Abbreviations

ACM	Adaptive coding and modulation
AI	Artificial intelligence
ASIC	Application-specific integrated circuit
AWGN	Additive white Gaussian noise
BEP	Bit error probability
BER	Bit error rate
DSP	Digital signal processing
FEC	Forward error correction
FPGA	Field-programmable gate array
HDL	Hardware description language
HLS	High-level synthesis
I	In-phase
LDPC	Low-density parity check
LSB	Least significant bit
ML	Maximum-likelihood
MSB	Most significant bit
NS	Non-square
PDF	Probability density function
PSD	Power spectral density
Q	Quadrature
QAM	Quadrature amplitude modulation
RS	Reed Solomon
RTL	Register transfer level
RX	Receiver
S	Square
SINR	Signal-to-interference-plus-noise ratio
SNR	Signal-to-noise ratio
TX	Transmitter
X	Cross

Contents

1

Introduction

Quadrature amplitude modulation (QAM) has become an industry standard in modern telecommunications. The QAM format allows transmission at higher information rates with greater spectral efficiency compared to other common formats. However, when transmission of an odd number of bits per symbol is desired, we are faced with rectangular constellation diagrams. Since some symbols are placed far from the origin, these constellations provide suboptimal performance in terms of power efficiency. It can be shown that there are benefits to reorganizing the symbols to create a symmetric structure. For this purpose, cross (X) QAM formats were introduced as a means to reduce peak and average power requirements for rectangular constellations, with only a small penalty in implementation complexity [1]. However, these formats present other challenges when it comes to system integration, such as the need to handle two-dimensional decision boundaries between symbols.

This thesis work aims to extend the functionality of Ericsson's microwave and millimeter-wave digital signal processing (DSP) system by implementing support for X QAM formats. The main focus is on integrating these formats with the existing multi-layer forward error-correction (FEC) system, consisting of low-density parity check (LDPC) and Reed Solomon (RS) error correcting codes. This will improve the adaptive coding and modulation (ACM) capabilities of the system. In the current implementation, if the system is operating using, e.g., 1024-QAM and the channel conditions become increasingly noisy, the modulation order has to be dropped to 256-QAM since this is the closest format with an even number of bits per symbol. If the option of using an odd number of bits per symbol is viable, we could potentially instead use 512-QAM in the hypothetical scenario and thus maintain higher throughput. Ultimately, a successful implementation makes it possible to achieve higher information rates and greater power efficiency under varying channel conditions, which in turn provides an improved experience for the end user of Ericsson's products.

Throughout this report, we refer to modulation formats that use an even number of bits per symbol (e.g., 16-QAM, 64-QAM, or 256-QAM) as square (S). Conversely, we refer to modulation formats that use an odd number of bits per symbol (e.g., 512-QAM, 2048-QAM, or 8192-QAM) as non-square (NS). Additionally, we distinguish between rectangular QAM formats and cross-shaped QAM formats, by referring to them as NS QAM and X QAM. Higher-order formats, like S 65536-QAM or S 262144-QAM, we will refer to as S $64k$-QAM and S $256k$-QAM where $k = 1024$.

1.1 Related Work

The concept of X QAM was originally introduced by Smith in [1]. Also introduced in the paper is the concept of Gray penalty (G_p), which is used as a metric to compare Gray-coded and impurely Gray-coded constellations in terms of bit-error rate (BER). Here, impure Gray coding refers to the case where two or more adjacent symbols differ in more than one bit position, which is always the case for X QAM.

In [2], derivations of exact BER expressions are presented for the different decision regions of X QAM constellations with standard symbol mapping of order $M \geq 32$. The derivations are carried out for additive white Gaussian noise (AWGN) and Rayleigh fading channels. Special treatment of certain decision regions is introduced, as some of the decision boundaries in X QAM constellations are two-dimensional. Furthermore, the derivations assume pseudo-Gray coding, meaning that adjacent symbols differ in at most two bit positions. The results in [2] serve as a good baseline for evaluating the performance of our system in terms of BER.

Finally, [3] presents how X QAM modulation can be used in practice to improve performance compared to other QAM formats in the presence of non-ideal behavior introduced by optical modulators.

1.2 Goals

The main goal of the thesis is to implement support for X M-QAM formats, where $M \in [512, 2048, 8192, 32k, 128k, 512k]$, in a way that is compatible with Ericsson's multi-layer FEC system. This will involve making changes to the current implementation of the symbol mapper and demapper as well as the parts of the multi-layer FEC which handle correction of the most significant bits (MSBs). The implementation will account for the cases where the decision boundaries between symbols in the X QAM constellation are two-dimensional. Furthermore, the implementation needs to be feasible to realize hardware in order to run on the target field-programmable gate array (FPGA). Ideally, the number of bits encoded using RS should be configurable from software to 0, 2, 4, 6 or 8 bits. The implementation will be deemed successful once the desired BER vs. signal-to-noise ratio (SNR) curves are produced from running tests on the target platform. This means that we want to see the same performance in terms of BER for the same code rate as for the already implemented S QAM cases, but for SNR values halfway between the S QAM curves.

1.3 Ethical and Societal Impact

Since our design will be used to communicate private and potentially sensitive data, security is an important consideration. Care needs to be taken such that our implementation does not introduce vulnerabilities in Ericsson's system that could potentially be exploited. The design also needs to be robust in order to be used in a safety critical real-time system, e.g. in automotive applications.

The use of large-language models and other artificial intelligence (AI)-based tools

during this thesis work was limited to providing suggestions to improve grammar and structure in the report. At no point was AI-compiled text used as a source, nor did recommendations made by an AI engine influence any design choices.

1.4 Thesis Outline

In Chapter 2 we provide a general introduction to digital communication, including QAM and signal space analysis, as well as X QAM and FEC. We also describe the parts of Ericsson's multi-layer FEC relevant to this project. Chapter 4 presents the modifications made to the symbol mapper, symbol demapper, and MSB decoders in order to support X QAM and what remains for future work. The performance of our design is summarized in Chapter 5. Finally, we conclude our work in Chapter 6.

2

Technical Background

This chapter provides the theoretical background needed to follow the discussions in subsequent chapters. We start by introducing the concept of QAM in Section 2.1. We then provide a summary of the tools used for analyzing and comparing the performance of different constellations on a system level in Section 2.2. In Section 2.3 we extend our analysis to constellations where the number of bits per symbol is odd. Section 2.4 gives a brief introduction to the broad topic of FEC and, finally, we give an overview of the current implementation of Ericsson's multi-layer FEC in Section 2.5.

2.1 Quadrature Amplitude Modulation

QAM is a modulation format based on the concept of orthogonal signaling. A QAM signal is constructed from two orthogonal signals, i.e., signals with a relative phase of 90°, often referred to as the in-phase and quadrature or I and Q components. These signals form what is known as an orthonormal basis, which we will discuss in further detail in Section 2.2. Each of the I and Q components consists of a baseband pulse modulated onto a carrier wave. Here, a baseband signal refers to a signal with an amplitude spectrum centered on 0 Hz. The carrier wave serves to shift the spectrum of the baseband pulse to higher frequencies. We cannot simply transmit a baseband pulse as this would require unreasonably large antennas due to the antenna size being proportional to wavelength [4]. Furthermore, we would not utilize the available spectrum in an efficient way. The modulated baseband pulse generates what is known as a passband signal. The passband spectrum will have the same bandwidth as in the baseband case, but will now be centered around the frequency of the carrier wave. Generally, we can write a passband signal as [5]

$$s(t) = A(t)\cos\left[\omega_0 t + \psi(t)\right], \tag{2.1}$$

where $A(t)$ is the amplitude modulation, $\psi(t)$ is the phase modulation, and ω_0 is the carrier frequency. We can rewrite the above expression in terms of I and Q as [5]

$$s(t) = I(t) - Q(t) = I_{BB}(t)\cos\omega_0 t - Q_{BB}(t)\sin\omega_0 t, \tag{2.2}$$

where $I_{BB}(t) = A(t)\cos\psi(t)$ and $Q_{BB}(t) = A(t)\sin\psi(t)$ are baseband signals. It is often convenient to normalize I and Q. This can be done by rewriting (2.2) to [4]

$$s(t) = \sqrt{\frac{2E_s}{T}}[I_{BB}(t)\cos\omega_0 t - Q_{BB}(t)\sin\omega_0 t], \tag{2.3}$$

where E_s is the average energy of a symbol when the symbol slot is of duration T. Using simple trigonometry we can then express the amplitude and phase modulation in (2.1) as

$$A(t) = \sqrt{\frac{2E_s}{T}}\sqrt{[I_{BB}(t)]^2 + [Q_{BB}(t)]^2} \tag{2.4}$$

$$\psi(t) = \arctan\left[\frac{Q_{BB}(t)}{I_{BB}(t)}\right]. \tag{2.5}$$

$A(t)$ and $\psi(t)$ should reflect the content of the information signal. In digital communications, $I_{BB}(t)$ and $Q_{BB}(t)$ usually consist of trains of baseband pulses that carry information. For QAM, we can define these as [4]

$$I_{BB}(t) = C_0 \sum_{n=0}^{K-1} \hat{I}_n \sqrt{T} v(t - nT) \tag{2.6}$$

$$Q_{BB}(t) = C_0 \sum_{n=0}^{K-1} \hat{Q}_n \sqrt{T} v(t - nT), \tag{2.7}$$

where K is the length of the transmitted message and $\hat{I}_n$ and $\hat{Q}_n$ are chosen from the available discrete amplitude values in the I and Q dimensions. These amplitude values are usually taken as odd integers by convention. Here, $v(t)$ is an orthogonal pulse. This means that the pulse is uncorrelated with itself at time lags that are integer multiples of T. A constant scalar C_0 is introduced to make the average symbol energy equal to E_s and the average power equal to 1. It is defined as [4]

$$C_0 = \left(\frac{1}{\frac{1}{M}\sum_{\hat{I},\hat{Q}}[\hat{I}^2 + \hat{Q}^2]}\right)^{1/2}, \tag{2.8}$$

where M is the number of symbols in the constellation (see next section), which are assumed to be equiprobable, that is, equally likely to be transmitted.

2.2 Signal Space Analysis

As mentioned in the previous section, $I(t)$ and $Q(t)$ constitute an orthonormal basis for the transmitted signals. An orthonormal basis is a set of N functions that are uncorrelated, generating an N-dimensional signal space. Using this fact, we can represent the transmitted signals as weighted sums of these basis functions. These weighted sums are what characterize a modulation format. The more combinations of basis functions to choose from, the higher the modulation order, M. Each signal can now be described using vector notation. This way, we reduce the analysis of continuous-time signals to vector analysis in Euclidean space. This is often referred to as signal space analysis [6].

2.2.1 Signal Constellations

We can visualize any M-QAM format by plotting its signal constellation. A constellation is a collection of points referred to as symbols. Each symbol is characterized

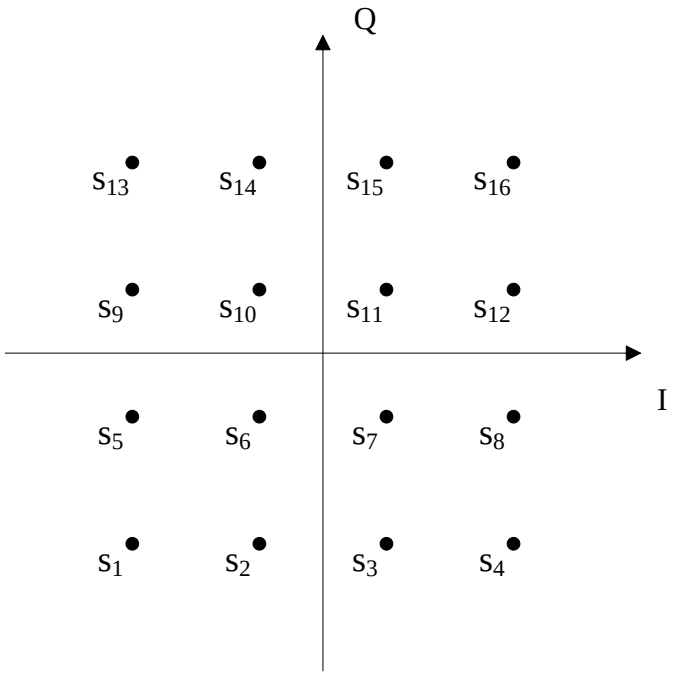

Figure 2.1: S 16-QAM signal constellation.

by an amplitude and a phase, given by the amplitude scaling of the basis functions. Displayed in Figure 2.1 is the constellation of S 16-QAM, a two-dimensional format of order 16. As can be seen, no symbol is placed directly on one of the axes. This is to avoid transmitting a symbol with zero power in the I or Q channels, as doing so would make it impossible to distinguish a correct transmission from an erroneous one.

The average symbol energy, E_s, can now be expressed as [4]

$$E_s = \sum_{i=1}^{M} \mathbb{P}[S = s_i] \; ||s_i||^2 = \frac{1}{M} \sum_{i=1}^{M} ||s_i||^2, \tag{2.9}$$

where $\mathbb{P}[S = s_i]$ denotes the likelihood of symbol s_i being transmitted and $||s_i||$ is the Euclidean distance from the origin to the symbol in the constellation. The equality holds when all symbols are equiprobable. A cursory examination of (2.9) reveals that it is the same expression as the denominator in (2.8), apart from the square root. This means, for example, that symbol s_1 will have coordinates $(-1/\sqrt{E_s}, -1/\sqrt{E_s})$.

The objective of any receiver is to find, given a received symbol, which of the symbols in the constellation is most likely to have been transmitted. To aid this objective we can define the decision regions for each symbol. These are displayed in Figure 2.2. In the figure, each of the rectangular decision regions is confined by the dashed lines and/or the I- or Q-axis. These are the decision boundaries. If the received symbol falls within the region corresponding to s_i, it will be mapped to that specific symbol.

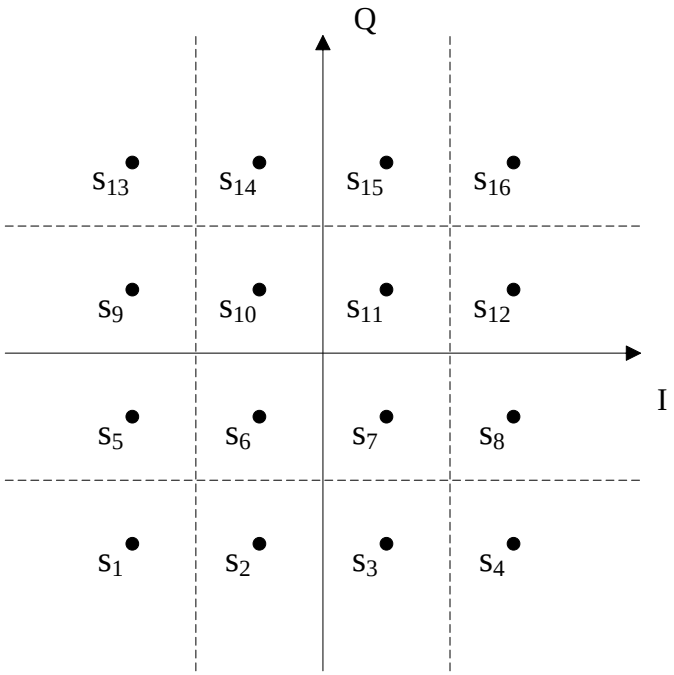

Figure 2.2: Decision regions of the S 16-QAM constellation.

The receiver objective is straightforward in the ideal case. However, the transmitted signal will suffer from non-ideal properties imposed by the channel. A channel is simply a medium through which the signal has to propagate to reach the receiver. This could be a coaxial cable, an optical fiber, or air in the wireless case. Figure 2.3 shows a common way to model a digital communication link, including the the effects of the channel. This is often referred to as Shannon's communication model [4]. The figure displays the signal path from the information source to its destination, called a sink. The encoder in the transmitter performs both source and channel encoding. Source encoding relates to data compression, which we do not deal with in this report, while channel encoding relates to FEC (see Section 2.4). The modulator shifts the signal spectrum to the desired frequency band before transmission over the channel. In the receiver, we simply apply the inverse operations to retrieve the original data.

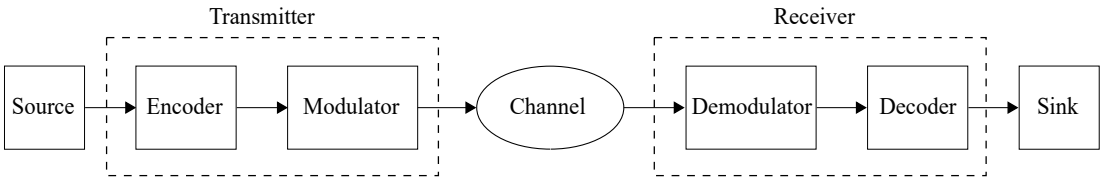

Figure 2.3: Shannon's communication model.

The channel also includes impairments in the transmitter and receiver hardware. The noise and distortion corrupting the transmitted signal will force the corresponding symbol in the constellation to move away from its original position, potentially causing errors if a decision boundary is crossed. Thus, we need an accurate channel model in order to design a receiver that accounts for this fact. The following section will examine the most common way to model a channel in a communication system and how to construct a receiver for this type of channel.

2.2.2 Additive White Gaussian Noise

So far, we have omitted the influence of noise in the discussions. The most common and, luckily, convenient type of noise in a communication system is Gaussian noise, specifically additive white Gaussian noise (AWGN) [6]. Usually, AWGN manifests itself as thermal noise in the amplifier stage of a transmitter. AWGN has the property of having a constant power spectral density (PSD) over all frequencies. This does not imply that we need to handle infinite powers as AWGN is, in practice, always bounded by the finite bandwidth of electronic devices [4].

The AWGN channel can be thought of as a white Gaussian random process, $\eta(t)$, projected onto the orthonormal basis functions. This generates a set $\{\eta_j\}$ of Gaussian random variables, one for each dimension j, with properties [4]

$$\mathbb{E}[\eta_j] = 0 \text{ for all } j \tag{2.10}$$

$$\text{cov}(\eta_j, \eta_k) = \begin{cases} 0, & j \neq k \\ N_0/2, & j = k \end{cases}, \tag{2.11}$$

where $\mathbb{E}$ is the expectation operator. This means that we have a process with 0 mean, a variance (and PSD) of $N_0/2$, and uncorrelated samples. Our received signal will now have the vector form $\mathbf{r} = \mathbf{s_i} + \boldsymbol{\eta}$. Each element in $\mathbf{s_i}$ is a transmitted symbol, $i \in [1, M]$, and each element in $\boldsymbol{\eta}$ is a random variable. We can then formulate the receiver objective as finding the probability $\mathbb{P}[\boldsymbol{\eta} = \mathbf{r} - \mathbf{s_i}]$, that is, the probability that a random variable takes the value of the difference between the received and transmitted signals. Specifically, we want to maximize this probability over all symbols. It can be shown after some derivation that for equiprobable symbols this can be expressed as [6]

$$\max_i \mathbb{P}[\boldsymbol{\eta} = \mathbf{r} - \mathbf{s_i}] \equiv \min_i ||\mathbf{r} - \mathbf{s_i}||^2, \tag{2.12}$$

which is known as the maximum-likelihood (ML) receiver. We can conclude from the above expression that calculating the probability is reduced to finding the symbol $\mathbf{s_i}$ in the constellation at minimum Euclidean distance from the received signal $\mathbf{r}$.

2.2.3 Error Probabilities

From the previous section it should be clear that the ML receiver will make an error if the received symbol falls outside of the decision region of the transmitted symbol. Thus, we need to quantify the probability of mistaking the received symbol

for a symbol in the constellation other than the transmitted one. To calculate this probability we start from the simplest case, where we only have two signals in the constellation separated by the distance D, as in Figure 2.4. The orientation in a two-dimensional signal space should be such that the line connecting $\mathbf{s_i}$ and $\mathbf{s_k}$ runs parallel to the I-axis. We actually do not need the Q-dimension in this setup as the problem is purely one-dimensional. Hence, we could simply discard the phase.

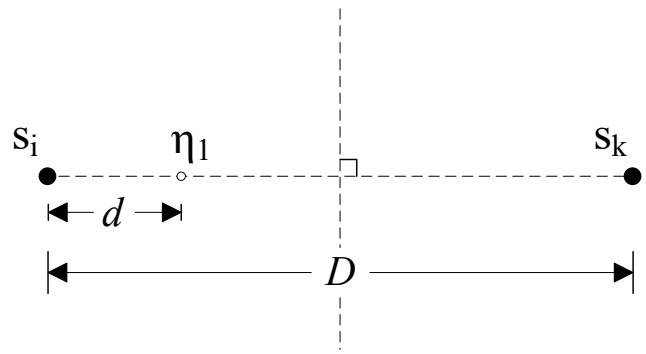

Figure 2.4: Setup for calculating two-signal error probability.

We can clearly see from the above figure that the ML receiver will mistake $\mathbf{s_i}$ for $\mathbf{s_k}$ when a Gaussian random variable η_1 in the I-dimension causes the received symbol to cross the bisector perpendicular to the axis. In the above case, this is equivalent to the symbol being moved further than a distance $D/2$ along the axis from its original position after transmission (i.e., $d > D/2$). The probability of this event occurring can be calculated by integrating the Gaussian probability-density function (PDF) [6], i.e.

$$p_2(k|i) = \frac{1}{\sqrt{\pi N_0}} \int_{D/2}^{\infty} \exp\left(-\frac{\eta_1^2}{N_0}\right) d\eta_1, \tag{2.13}$$

which is just a standard Gaussian integral with zero mean and variance $N_0/2$. The notation $p_2(k|i)$ translates to the conditional probability that we receive $\mathbf{s_k}$ given that $\mathbf{s_i}$ was transmitted. In communication engineering, the Gaussian integral is conventionally given in terms of the Q function [6]

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_{x}^{\infty} \exp\left(-\frac{1}{2}u^2\right) du. \tag{2.14}$$

Again, this is simply a Gaussian integral, however, now with zero mean and unity variance. If we wish to rewrite (2.13) in terms of (2.14) we can make the variable substitution $\eta_1 = u\sqrt{N_0/2}$. Then $du/d\eta_1 = 1/\sqrt{N_0/2} \leftrightarrow du = d\eta_1/\sqrt{N_0/2}$ and $D/2 \rightarrow D/2\sqrt{N_0/2} = D/\sqrt{2N_0}$. The two-signal error probability then becomes

$$p_2(k|i) = Q\left(\frac{D}{\sqrt{2N_0}}\right). \tag{2.15}$$

For multi-signal constellations the probability of erroneous detection can generally

be described by [4]

$$p_e = \sum_{i=1}^{M} \mathbb{P}[\mathbf{s_i}] \sum_{k \neq i} \mathbb{P}[\mathbf{r} \in R_k | \mathbf{s_i}] = \frac{1}{M} \sum_{i=1}^{M} \sum_{k \neq i} \mathbb{P}[\mathbf{r} \in R_k | \mathbf{s_i}], \tag{2.16}$$

which is the probability that the received symbol $\mathbf{r}$ falls in a decision region R_k where the transmitted symbol s_i does not reside. As before, the equality holds for equiprobable symbols. Unfortunately, the probability $\mathbb{P}[\mathbf{r} \in R_k | \mathbf{s_i}]$ is generally not known. It turns out, however, that the two-signal error probability in (2.15) can be used to create a union bound (see [7]) for the probability in (2.16). That is [4]

$$p_e \leq \frac{1}{M} \sum_{i=1}^{M} \sum_{k \neq i} p_2(k|i) = \frac{1}{M} \sum_{i=1}^{M} \sum_{k \neq i} Q\left(\frac{D_{i,k}}{\sqrt{2N_0}}\right), \tag{2.17}$$

where $D_{i,k}$ is the distance between symbols s_i and s_k. The sum will be dominated by the signal pairs separated by the minimum distance. Thus, we can write [4]

$$p_e \leq \frac{2K}{M} Q\left(\frac{D_{min}}{\sqrt{2N_0}}\right), \tag{2.18}$$

where K is the number of signal pairs lying at distance D_{min}. This is a tight bound. It can thus be used as an approximation for the multi-signal error probability for equiprobable symbols in conditions where the SNR is high.

We can also define the union bound for the probability of a single bit error, the bit-error probability (BEP), as [4]

$$b_e \leq \frac{1}{M} \sum_{i=1}^{M} \sum_{i \neq k} \frac{H_{i,j}}{m} Q\left(\frac{D_{i,k}}{\sqrt{2N_0}}\right), \tag{2.19}$$

where $m = \log_2(M)$ is the number of bits per symbol and $H_{i,k}$ is the Hamming distance between symbols s_i and s_k, that is, the number of bits that differ between the labels of the symbols. A label is nothing but the bit pattern we choose to assign to a specific symbol. This can be done in different ways as we will see in the next section. Under the same assumptions as above we can write (2.19) as [4]

$$b_e \leq \frac{2H_{min}}{Mm} Q\left(\frac{D_{min}}{\sqrt{2N_0}}\right), \tag{2.20}$$

where H_{min} is the total number of bits differing between signal pairs at minimum distance. Again, this is a good approximation of the probability for equiprobable symbols at high SNR.

Multiplying the BEP with the transmission rate (bits/s) gives an estimate of the BER [8], which is one of the most important performance metrics in a digital communication system. The transmission rate is commonly referred to as the capacity of a communication link. The key difference between BEP and BER is that the BEP is a theoretical estimate, while the BER is a measured quantity. The BER reflects the occurrence of incorrect bits in a received sequence per unit time. This is usually given at different values of SNR. Hence, BER vs. SNR curves are very useful for visualizing the performance of a communication system.

2.2.4 Gray Code

From (2.20) we can conclude that reducing the number of bits differing between symbols at minimum distance is critical for reducing the BEP and, consequently, the BER. Clearly, the best we can do is to employ a labeling where adjacent symbols differ only in one bit. Such a labeling is called a Gray code [9]. The Gray code, sometimes referred to as the reflected binary code, gets its name from Frank Gray who was a physicist and researcher at Bell Labs. Figure 2.5 shows how to construct a Gray labeling for the S 16-QAM constellation. As can be seen, the labels of all symbol pairs at distance D_{min} are indeed Gray coded. Note that there exist many ways of Gray labeling S 16-QAM and that this is only one of the possible solutions.

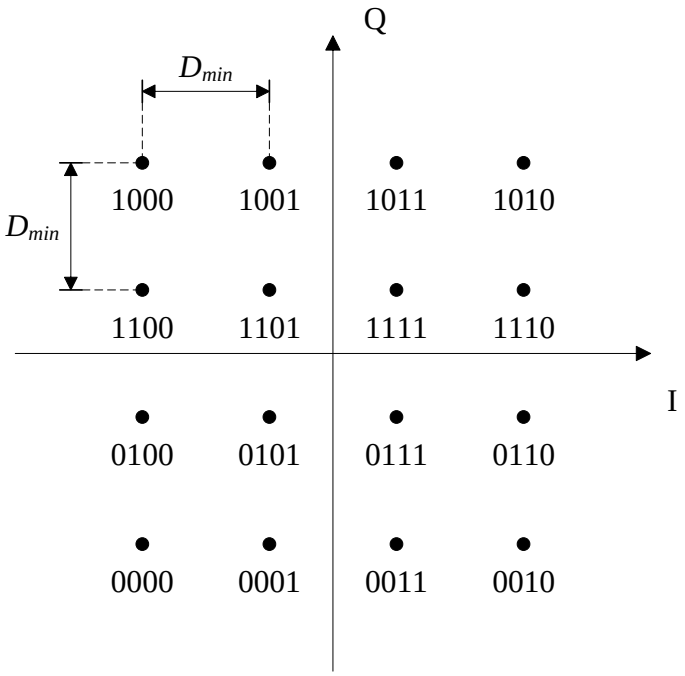

Figure 2.5: S 16-QAM constellation with Gray labeling.

2.3 Non-Square QAM

When the number of bits per symbol, m, is even, the square QAM constellations that we discussed so far provide the optimal way to organize symbols in a two-dimensional space [1]. However, for an odd number of bits per symbol we obtain rectangular constellations which provide suboptimal performance. Figure 2.6 shows the rectangular constellation for NS 32-QAM ($m = 5$). It turns out that by re-arranging the symbols we can improve certain performance metrics of the 32-QAM format. By moving a certain number of columns from both sides of the rectangular constellation to the top and bottom, we can create a symmetric structure.

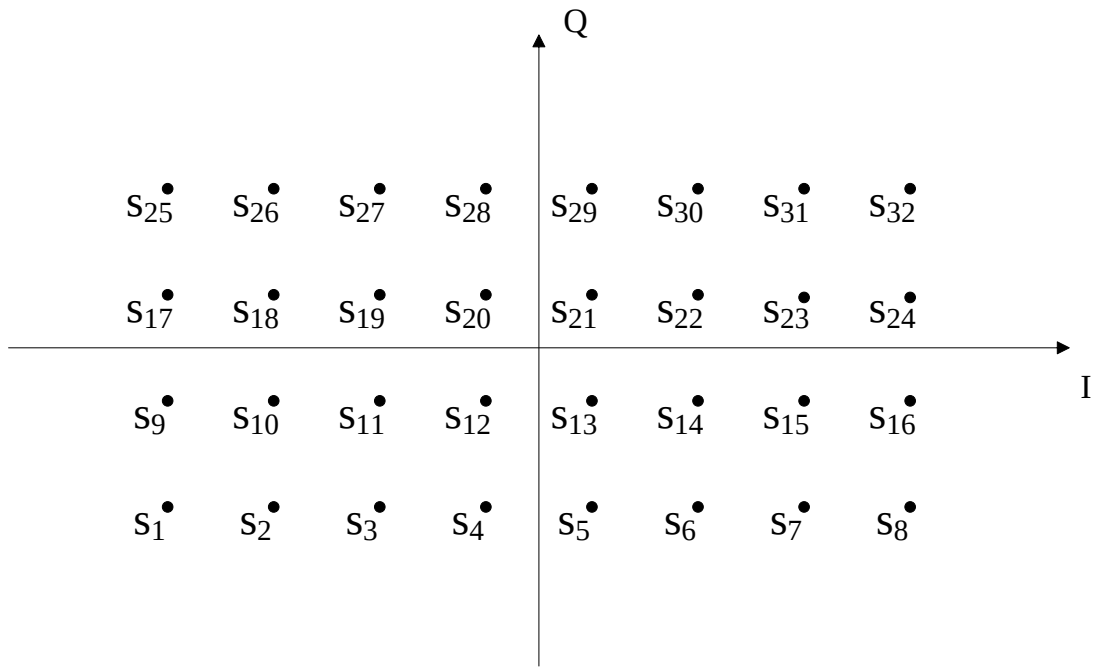

Figure 2.6: Rectangular NS 32-QAM constellation.

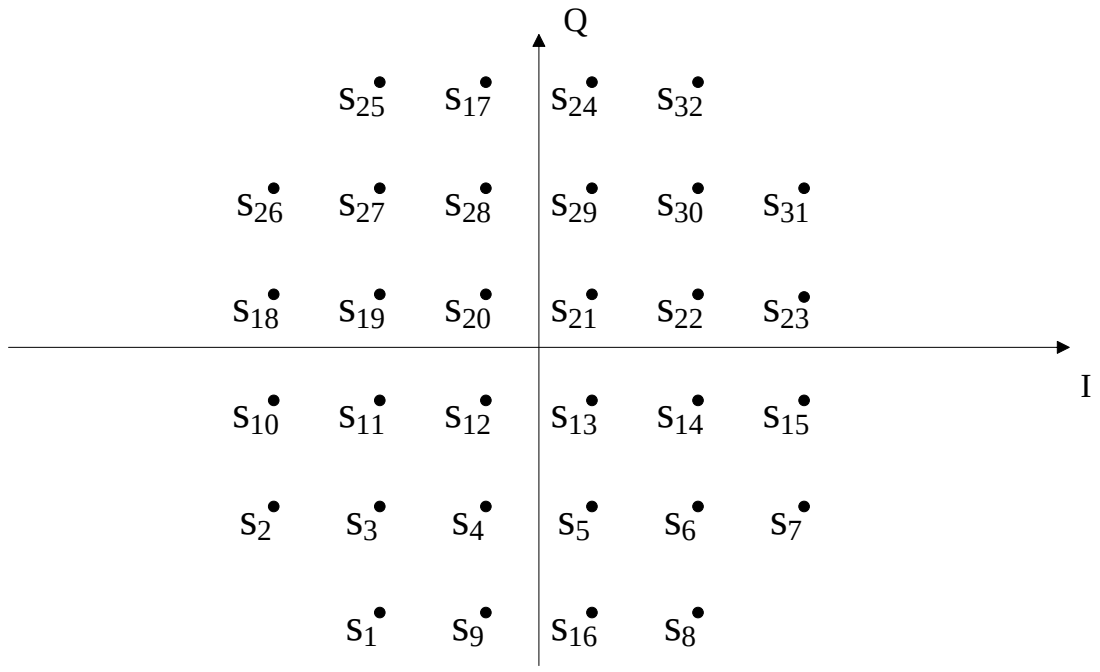

Figure 2.7: Symmetric X 32-QAM constellation.

The number of columns to be moved can be shown to be [2]

$$N_{col} = \frac{\sqrt{2M}}{8}. \tag{2.21}$$

From (2.9) we know that symbol energy, and thus power, is directly linked to the Euclidean distance from the origin to each symbol in the constellation. Hence, if we can pack the symbols closer to the origin we can reduce the power requirements. Additionally, it is convenient to have the same average energy in both dimensions. Figure 2.7 displays a symmetric version of the NS 32-QAM constellation which achieves exactly this, which we refer to as the X 32-QAM constellation.

By arranging the symbols as in Figure 2.7, we reduce the peak and average power and thus the required SNR to achieve a given BER compared to rectangular QAM. This comes at a BER penalty as well as a minor implementation penalty in terms of the extra logic required in the mapper and demapper to create the symmetric structure. The mapper is the unit that assigns a bit sequence from the data to be transmitted to a specific symbol, where as the demapper handles the inverse operation. The BER penalty stems from the fact that it is not possible to perfectly Gray code a X QAM constellation and it is therefore often referred to as a Gray penalty, G_p [1]. In high SNR conditions, the BEP can be given in terms of G_p as [1]

$$b_e \approx \frac{G_p N}{m} Q\left(\frac{D_{min}}{\sqrt{2N_0}}\right), \tag{2.22}$$

where N is the average number of neighbors (i.e. symbols a distance D_{min} away) for a symbol. We can identify the similarities to (2.20) and directly write an expression for G_p as

$$G_p = \frac{2H_{min}}{N_{tot}}, \tag{2.23}$$

where N_{tot} is the total number of neighbors for all symbols. For a perfectly Gray-coded constellation G_p will be unity. We can confirm this by taking the Gray-coded S 16-QAM constellation as an example. The total number of signal pairs at minimum distance is 24 and since their labels differ only in one bit, we get $H_{min} = 24$. Counting the nearest neighbors for all symbols in the constellation gives $N_{tot} = 2 \cdot 4 + 8 \cdot 3 + 4 \cdot 4 = 48$. Thus, we get $G_p = (2 \cdot 24)/48 = 1$.

For higher orders of NS M-QAM ($M > 32$) the process of remapping a rectangular structure to X M-QAM is very similar to the NS 32-QAM case. We can always divide the constellation into 32 square subconstellations and arrange these in the same way we did for the symbols in Figure 2.7. To do this, we introduce the block parameter [1]

$$\gamma = 2^{k-2}. \tag{2.24}$$

Here, k is the number of rows in the rectangular constellation with minimum E_s, which would then have $k+1$ symbols per row. If we define the NS QAM modulation orders as $M = 2^{2k+1}$ we can write [1]

$$M = 32\gamma^2, \tag{2.25}$$

where γ^2 is the number of symbols in each subconstellation.

One of the big advantages of the S QAM format is that all decision regions are rectangular (see Figure 2.2) which means that we have decision boundaries that are straight lines in each dimension. This allows us to handle the I and Q dimensions separately when demapping a symbol. Unfortunately, this is not the case for all symbols in the X QAM constellation. In Figure 2.8 we have drawn the decision boundary between symbols s_1 and s_2, which is also shared with symbols s_{31} and s_{32}.

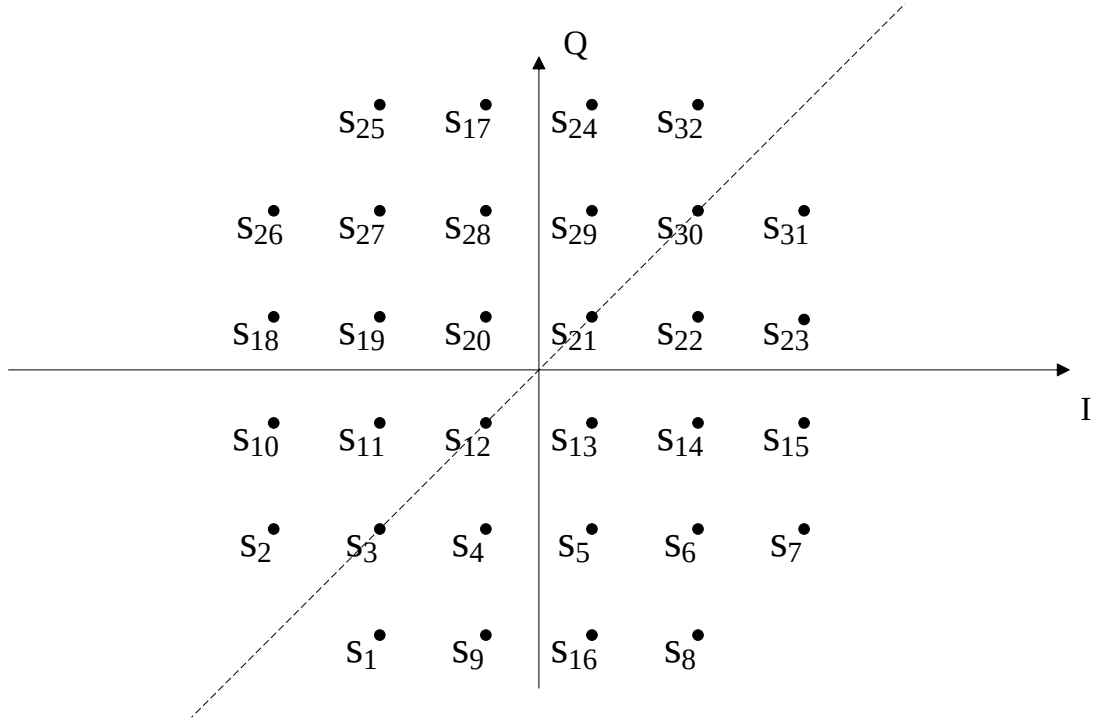

Figure 2.8: Decision boundary for symbols s_1/s_2 and s_{31}/s_{32}.

This decision boundary lies a distance $\sqrt{D_{min}^2 + D_{min}^2}/2 = D_{min}/\sqrt{2}$ from the closest symbols, which is greater than the usual $D_{min}/2$. However, noisy conditions could force us to take this boundary into consideration, which adds extra complexity to the receiver.

2.4 Forward Error Correction

The aim of this section is to give a very brief introduction to FEC. This thesis does not directly deal with FEC, but some background on the topic helps the reader understand certain design choices made in subsequent chapters.

2.4.1 Error-Correcting Codes

In order to protect binary data from being corrupted by impairments during transmission or storage, we can introduce redundancy (extra bits) in order to detect and/or correct errors [10]. This redundancy is often referred to as an error-correcting

code, an error-control code, or a channel code. The general problem of error correction revolves around finding a code that in the most efficient way encodes k information bits as n bits, called a codeword. By efficient we mean by keeping the introduced redundancy $n - k$ as small as possible. This is often quantified by the code rate [10]

$$R = \frac{k}{n}, \tag{2.26}$$

where a higher code rate equates to a more efficient code. An efficient code could also refer to a code that has an efficient hardware realization. Clearly, the best we can do is a k-to-k mapping which results in a code rate of one. This, however, provides no protection of the information bits. There is no getting around the extra redundancy if error-correcting capabilities are desired!

A straightforward approach to protecting data is to simply transmit (or store) several copies of the data [10]. An example of a code that employs this strategy is the three-times repetition code. As the name suggests, we transmit three copies of the data instead of one in order to increase the probability of detecting and correcting potential errors at the receiver. Say that we would like to transmit a single bit x_1, then we would also transmit two copies x_2 and x_3. The conditions for a clean readout, i.e. one without errors, are then

$$\begin{cases} x_2 = x_3 = 0, & \text{if } x_1 = 0 \\ x_2 = x_3 = 1, & \text{if } x_1 = 1 \end{cases}. \tag{2.27}$$

Thus, we can conclude that no error has occurred if $x_1 \oplus x_2 = 0$ and $x_1 \oplus x_3 = 0$, where $\oplus$ denotes modulo two addition. In any other case the data has been corrupted.

The three-times repetition code is a 1-error-correcting and a 2-error-detecting code with a code rate of $R = 1/3$. For the above example this means that we can correct only 1-bit errors but we can also detect 2-bit errors, but not correct them. Of course, there exist much more sophisticated codes that achieve greater performance. We will discuss a few of these in the following sections.

2.4.2 Low-Density Parity Check

An LDPC code is a specific type of parity-check code. We have already looked at one type of parity-check code in the previous section. The three-times repetition code in the above example appends the parity bits x_2 and x_3 to the information bit x_1, which gives the parity-check equations $x_1 \oplus x_2 = 0$ and $x_1 \oplus x_3 = 0$. Similarly, an LDPC code is defined by a set of parity-check equations which form a parity-check matrix, usually denoted H. The matrix H is sparse, meaning that the number of non-zero entries is low [11]. Several algorithms exist in order to construct H and they are often pseudo-random. Hence, it is often more relevant to talk about a set of possible LDPC codes with certain parameters relating to the distribution of the non-zero entries than about codes with specific parity-check matrices.

The main difference between LDPC codes and other similar codes lies in the decoding process [11]. An ML approach to decoding generally works well when the set of

codewords is relatively small. For an LDPC code, the parity-check matrix can grow arbitrarily large. This makes an exhaustive search of all possible codewords to find the most likely one to have been transmitted unfeasible. Instead, an iterative approach is employed. Such decoding algorithms are collectively known as message-passing algorithms with some examples being bit-flipping and belief propagation [11]. The sparseness of H guarantees that these algorithms converge in a reasonable time, only growing linearly with the length of the code.

2.4.3 Reed-Solomon

In contrast to LDPC codes, RS codes operate on symbols rather than individual bits. The mathematics behind RS codes is quite dense and much too involved for the purpose of this text. However, they can be explained in terms of the less complicated problem of fitting polynomials to a set of points [12]. Recall a polynomial on the form

$$y = f(x) = f_{k-1}x^{k-1} + f_{k-2}x^{k-2} + ... + f_2x^2 + f_1x + f_0. \tag{2.28}$$

Such a polynomial is said to have a degree of $k-1$. Fitting a polynomial to a set of points is the action of designing a function on the form in (2.28) with a curve that intersects all the points in a plane. We can always uniquely fit a polynomial of degree $k-1$ to a set of k points. That is, two points are uniquely fitted by a line, three points by a parabola, and so on. If a polynomial $f(x)$ is described by a set of n points which all lie on its curve, where $n > k$, then $n-k$ points are redundant. These redundant points can be used to recover $f(x)$ if some of the n points are in error. The problem then becomes to find the polynomial with a curve that fits as many points as possible. It can be shown that $f(x)$ can be recovered if the number of erroneous points is less than $(n-k)/2$ [12].

Traditionally, the coefficients $\{f_i\}_{i=0}^{k-1}$ are decimal numbers. However, in the context of RS codes, they are binary m-vectors of 2^m distinct values. We also restrict y and x to take any of these 2^m values. These values form what is known in mathematics as a finite field or Galois field [13], but this will not be important to our continued discussion. An RS codeword is made up of n symbols corresponding to binary m-vectors, where k symbols carry information and $n-k$ symbols are redundant [12]. Each of the k information symbols corresponds to the y-value of a point (x_i, y_i) to which we fit a polynomial $f(x)$ of degree $k-1$ on the form in (2.28). We encode the k information symbols by evaluating $f(x)$ at $n-k$ additional points and form a codeword from the n total y-values. We do not need to specify the x-values as they will be known to both the encoder and decoder. At the receiver, we decode the symbols by finding a polynomial of degree $k-1$ (or less) which fits the maximum number of points. Once we have found $f(x)$ we can recover the information symbols by evaluating the function at the x-values corresponding to the k information positions. As stated above, this can be done correctly if the number of errors does not exceed $(n-k)/2$.

2.5 Multi-Layer FEC

We are now equipped with the necessary knowledge to understand the basic function of Ericsson's multi-layer FEC and related blocks such as the mapper and demapper. As mentioned in Chapter 1, the multi-layer FEC consists of LDPC and RS codes. As we will see, there is also a need for Gray coding and MSB correction. Displayed in Figure 2.9 presents a structural block diagram of the FEC chain for the transmitter (TX) and receiver (RX). The supported modulation formats are S 4- to 1024k-QAM, which equates to a number of bits per symbol in the range 2 to 20 bits.

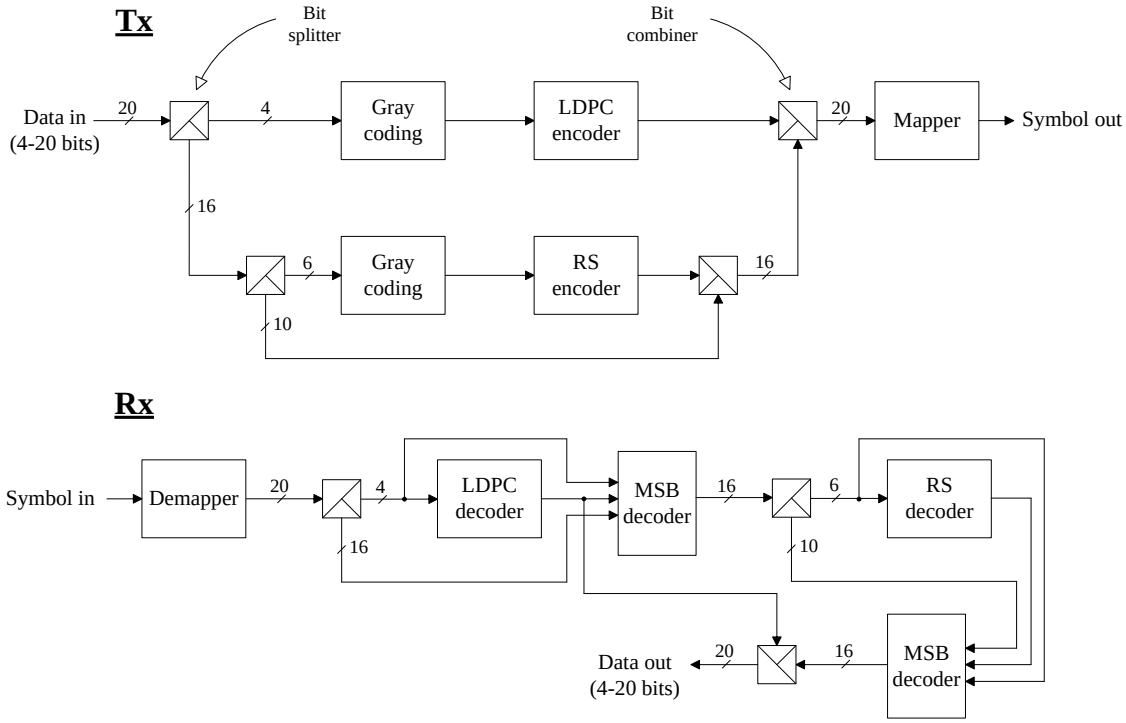

Figure 2.9: Structural block diagram of the multi-layer FEC.

For $M > 16$ the data to be transmitted is split into two parts. The four least-significant bits (LSBs), i.e. bit 3 to 0, are first Gray-coded and then sent to the LDPC encoder. Bits 9 to 4 are also Gray-coded and sent to the RS-layer. The purpose of the Gray-coding here is not to reduce the number of bit errors, but rather to spread the errors between the RS decoders in order to reduce the number of parity bytes needed in each decoder. The most probable error is to an adjacent subconstellation. Thus, if we use binary code for the RS bits, the two LSBs of those bits will always toggle if a symbol is mistaken for a neighboring one. This will then result in close to 100% of the errors ending up in the first RS decoder. If we Gray-code the bits we only get around 50% errors in the first decoder and about 25% in the other two. At most, 6 bits are RS-encoded. That is, for $M \geq 1024$ bits 9 to 4 are encoded using RS and the MSBs are left uncoded. For $M < 1024$ the number of RS bits is either 0, 2 or 4 and we have no uncoded bits.

The total code rate for the multi-layer can be calculated using

$$R_{tot} = \frac{R_{LDPC} \cdot n_{LDPC} + R_{RS} \cdot n_{RS} + R_{uncoded} \cdot n_{uncoded}}{n_{tot}}, \tag{2.29}$$

where $n_{tot} = n_{LDPC} + n_{RS} + n_{uncoded}$ (i.e., the number of bits per symbol). The RS encoders and decoders operate on bytes. Hence, the RS code rate is calculated as

$$R_{RS} = \frac{255 - b}{256}, \tag{2.30}$$

where b is the number of parity bytes. An RS frame is 255 bytes while an LDPC frame is based on multiples of 256 bytes. Thus, we have to discard byte $256 \cdot n$ from the LDPC layer in each RS frame. This leads to a code rate loss as the uncoded code rate becomes 255/256 instead of 1. As an example, if we are running an S $16k$-QAM format with an LDPC code rate of 0.5 and 8 RS parity bytes, we would get a total code rate of $R_{tot} = (0.5 \cdot 4 + ((255 - 8)/256) \cdot 6 + (255/256) \cdot 4)/14 \approx 0.841$.

The reason for encoding the LSB bits using LDPC and the higher bits using RS is that the LSB bits have a BER on the order of 10^{-2} compared to about 10^{-6} for the higher bits. Thus, the complex LDPC design is warranted for the LSB bits due to the high error rate. For the higher bits we can get away with a less complex and capable coding scheme as the BER is lower. Hence, we use RS. This saves hardware resources as the LDPC design is about ten times larger than the RS.

2.5.1 Symbol Mapping and Demapping for S QAM

After encoding, the bits are again combined to form a single data word and mapped to a symbol to be transmitted over the channel. When mapping data to an S QAM symbol, we simply take the bits in even bit positions (including 0) to form the real part of the symbol and those in odd positions to form the imaginary. As an example, for $M = 64$ the sequence 101101 would correspond to the symbol $3 + j6$ since $011_2 = 3_{10}$ and $110_2 = 6_{10}$ (subscripts 2 and 10 denote the base of the number).

The symbol demapper is responsible for the inverse operation of the mapper, that is, to convert a symbol back into a string of bits. One major difference compared to the mapper is that the received data is subject to noise introduced by the channel. Hence, before we can demap the data we need to enforce that the received symbol lies within the confines of the constellation, since we know for sure that nothing has been transmitted outside of the constellation borders. This is straightforward in the square case since we can limit each dimension individually to push a symbol left, right, up, or down. We also "remove" the noise by truncating to an integer type in both dimensions. This forces the received symbol to coincide with a valid constellation point. In order to not lose any information, we calculate the error between the received symbol and the rounded and limited symbol to be propagated to the multi-layer FEC. Finally, we can proceed with demapping, that is, placing the real bits in even bit position and the imaginary in odd bit positions of the output data word.

When the received symbol has been demapped, the acquired data is split into two parts. The four LSBs are sent to the LDPC decoder and to the first MSB decoder

while the remaining bits are passed directly to the MSB decoder. Since the LDPC only operates on the four LSBs, it can only correct within a 16-QAM subconstellation. This means that if a symbol close to the boundary of a subconstellation is transmitted and received in an adjacent subconstellation, the LDPC will push it further away from the original symbol. To mitigate this, we use an MSB decoder to correct the MSBs.

2.5.2 MSB Correction for S QAM

The MSB decoder calculates the Euclidean distance in the I- and Q-dimension between the received and LDPC-decoded symbol. If the calculated difference is $> 2D_{min}$ (that is, two symbol slots away) in either dimension, this means that there is a closer symbol in the opposite direction. A correction is then made accordingly by adding ± 1 to the MSBs in either dimension (or both), which is equivalent to moving to an adjacent subconstellation. The decoding process for S 64-QAM is displayed in Figure 2.10. The binary values in the corners are the values of the two uncoded MSBs in each subconstellation (one bit I and one bit Q). Note that any error imposed by the LDPC will be corrected as long as the noise in either dimension is $< 2D_{min}$. This corresponds to $> 99.999\%$ of the symbol errors in the system.

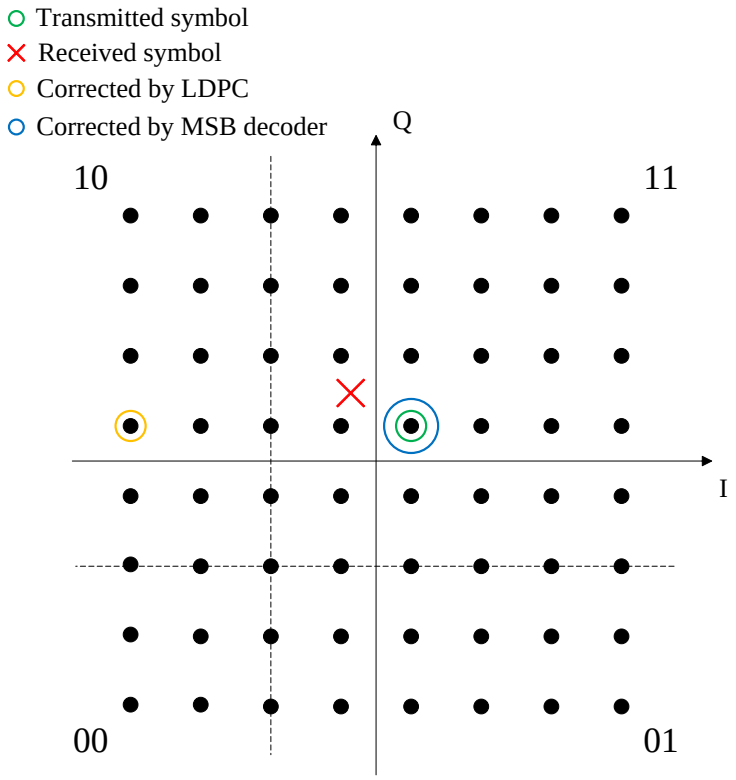

Figure 2.10: MSB correction process in second FEC layer for S 64-QAM.

The corrected MSBs are, again, split into two parts consisting of 6 and 10 bits, respectively. The two parts are passed to the second MSB decoder while the 6

bits (bits 9 to 4) are also sent to the RS decoder. Functionally, the second MSB decoder is near identical to the first one. Only now we correct based on 1024-QAM subconstellations instead of 16-QAM. This means that any symbol error $< 16D_{min}$ can be corrected. Finally, the LPDC- and RS-corrected bits are combined with the uncoded MSBs to form the output data word.

In Section 2.3 we discussed the potential impact that the Gray penalty, G_p, may have on the BER when employing X QAM in communication systems. This is less of a concern in this system, since we only Gray-code the four LSBs for the purpose of lowering BER. Symbol errors between subconstellations are generally corrected in the RS layer. Thus, moving subconstellations to create the X QAM formats will not result in a noticeable BER penalty as we do not Gray-code bits 4-20 for the purpose of reducing BER anyway. We do Gray-code the RS bits but that is for a different purpose, as described above.

2.5.3 X QAM and Multi-Layer FEC

One of the main concerns with using X QAM formats in a multi-layer FEC system is how to handle MSB correction. For S QAM and X QAM the subconstellations are arranged in different ways. Since the MSB correction is performed in the symbol domain, this must be taken into consideration. Specifically, we need logic to handle corrections in the corner constellations, where we have the diagonal decision boundaries. This is discussed in detail in Section 4.3.

3

Method

In this chapter, we outline the framework used to design and verify the desired functionality pursued in this work. We begin by introducing the concept of high-level synthesis (HLS), followed by an explanation of the verification process applied to the algorithms developed.

3.1 High-Level Synthesis

When designing hardware, whether targeting FPGAs or application-specific integrated circuits (ASICs), a hardware description language (HDL) is typically used to define the desired functionality. Common HDLs include VHDL, Verilog, and SystemVerilog. These languages are tailored for digital design, allowing engineers to precisely describe the behavior of hardware models. However, for certain applications, such as DSP related ones, algorithms that can be easily implemented in high-level languages such as C, C++, or Python, often become lengthy and complex when implemented in HDL. This increases development effort and delays time to market.

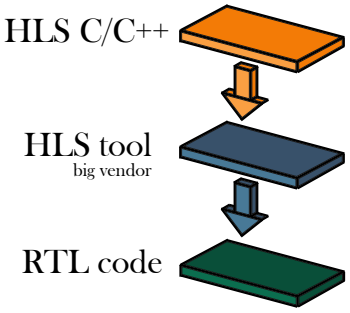

Figure 3.1: HLS process.

To address the problem of increasing development time, HLS tools have been developed by various vendors. These tools typically take a high-level description of an algorithm, written in C, C++, or Python, along with a set of directives that guide the hardware implementation. These directives may include target clock frequency, desired throughput, pipelining options, and other resource constraints. As detailed in Figure 3.1, the HLS tool then translates the high-level implementation into an HDL description, which can be further verified and integrated with other logic components of the project.

3.2 Design Verification

Given the design process described, the verification is divided into several sequential stages, starting with the functional verification of the C++ code. This is carried out by interfacing the executable code with various test blocks designed in MATLAB, which provide input stimuli to the model and verify the correctness of the output.

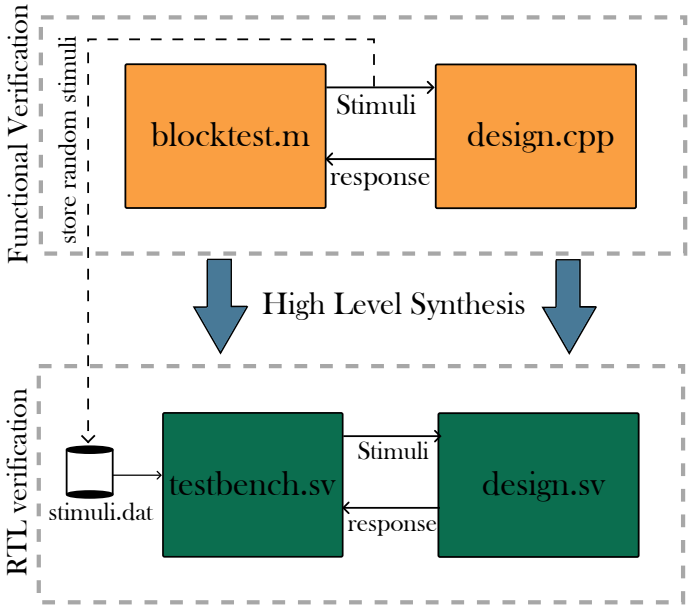

Figure 3.2: Verification process.

Once the desired behavior is achieved, the verified source code and a set of block tests, also written in C++, are passed to the HLS engine to be converted into register transfer-level (RTL) code. The block tests, now transformed into RTL testbenches, interact with the RTL design by providing the same stimuli as in the previous step and verifying both the correctness and consistency of the design behavior. This process is illustrated in Figure 3.2.

During functional verification, randomized data is generated and stored for reuse by the testbench at the RTL stage. MATLAB is particularly advantageous in this context due to its flexibility in generating randomized data with various distributions and formats.

During RTL conversion, HLS tools provide reports on resource utilization, timing, and other relevant metrics. For a synthesis result to be considered successful, the inferred design must meet all specified resource and performance constraints.

4

Design

This chapter describes the modifications made to Ericsson's multi-layer FEC to make the system compatible with X QAM formats. Our implementation mainly affects the mapper, demapper, and MSB decoder blocks in Figure 2.9. We start by describing how the symbol mapping and demapping is handled in Section 4.1 and Section 4.2. Then, we describe how the MSB decoders are modified in Section 4.3.

4.1 Symbol Mapping for X QAM

To extend the functionality of the mapper to handle X QAM formats we need some additional logic. We know from Section 2.3 that we can always rearrange a rectangular constellation to an X QAM constellation. Using the mapping scheme for S QAM, i.e. by taking every other bit to form the real and imaginary parts of the symbol, results in a rectangular NS constellation if the number of bits per symbol is odd. For example, $M = 512$ gives five bits in the real part and four bits in the imaginary part, and thus a larger range of real values compared to the imaginary. To create the symmetrical X QAM structure we thus need to move symbols from the left and right sides of the constellation to the top and bottom. As explained in Section 2.3, we need to move $\sqrt{2M}/8$ columns from each side. Or, since we can always divide the NS formats into 32 subconstellations, this is equivalent to moving four such constellations from each side.

In Figure 2.7, we show one possible way of mapping the X 32-QAM constellation. In our design, however, we have chosen an alternative mapping scheme. In Figure 4.1 we display the resulting constellation after assigning every other bit as real and imaginary. Each box represents a subconstellation of order $M \geq 16$ and the number in each box is the decimal value of the five MSBs of the symbol labels within that subconstellation. We can see that by adding ± 1 to the real and/or imaginary MSB bits we end up in a different subconstellation. For example, $25_{10} = 11001_2$ which gives the real and imaginary as $101_2 = 5_{10}$ and $10_2 = 2_{10}$. Then if we add one to the real, we get $5_{10} + 1_{10} = 6_{10} = 110_2$. Combined with the imaginary bits the five MSBs now equal $11100_2 = 28_{10}$, which corresponds to the subconstellation directly to the right.

To provide the best protection of the data, we want to encode as many bits as possible. Since the RS encoders take an even number of bits (2, 4, or 6), the best we can do for NS formats is to leave only the MSB bit uncoded. Since the MSB bit is unprotected, we need to create the largest possible separation between symbols

in the constellation that differ only in this bit. This is done to avoid a bit flip in the MSB position in the presence of noise. If we consider Figure 4.1, we see that the symbols in subconstellations that differ only in the MSB bit are separated by a

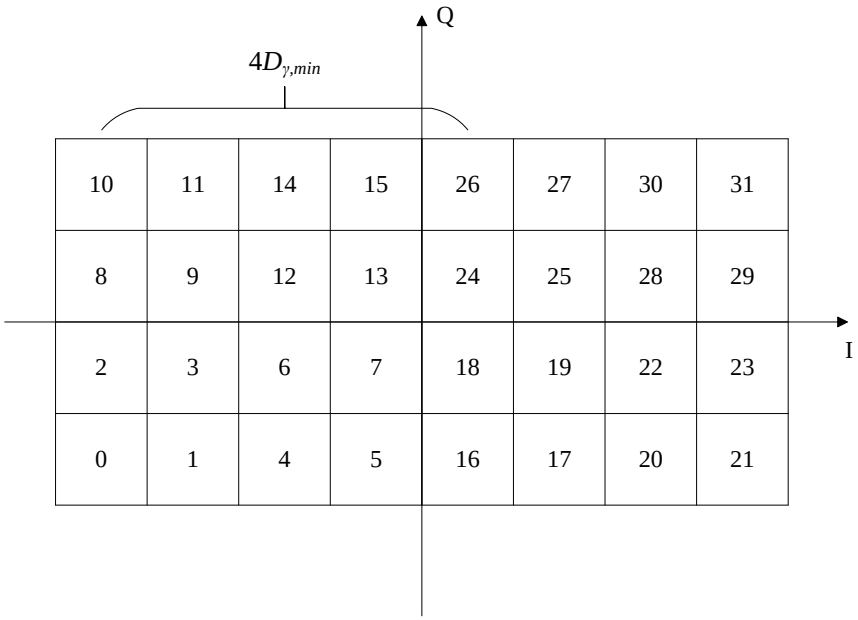

Figure 4.1: Rectangular constellation produced by mapper.

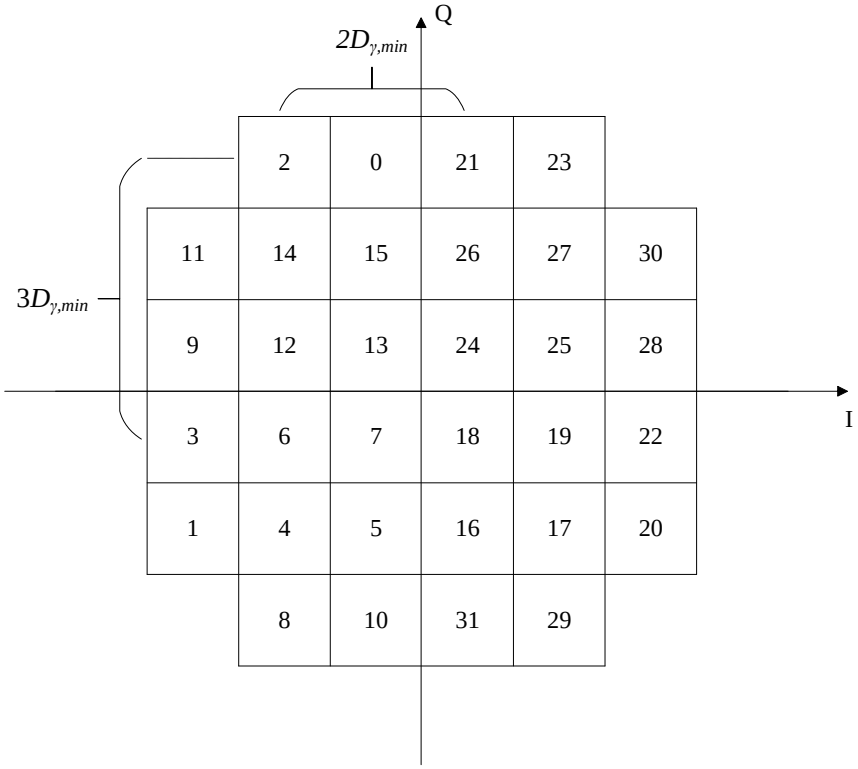

Figure 4.2: Rectangular constellation re-organized to X QAM.

distance of $4 \cdot \gamma \cdot D_{min}$, where $\gamma \cdot D_{min}$ is denoted $D_{\gamma,min}$. Recall that γ^2 represents the number of symbols per subconstellation and thus γ corresponds to the number of columns (or rows). Examples of such subconstellations are $2_{10} = 00010_2$ and $18_{10} = 10010_2$ or $12_{10} = 01100_2$ and $28_{10} = 11100_2$.

Generally, when re-organizing the subconstellations in the rectangular constellation to X QAM we need to maintain the largest distance possible to the subconstellation at a distance $4D_{\gamma,min}$ to the left or to the right, depending on which side of the imaginary axis we are. This is achieved in Figure 4.2. If we now look at subconstellations with MSBs equal to 2_{10} and 18_{10}, for example, they are separated by $\sqrt{2^2 + 3^2}D_{\gamma,min} \approx 3.6D_{\gamma,min}$.

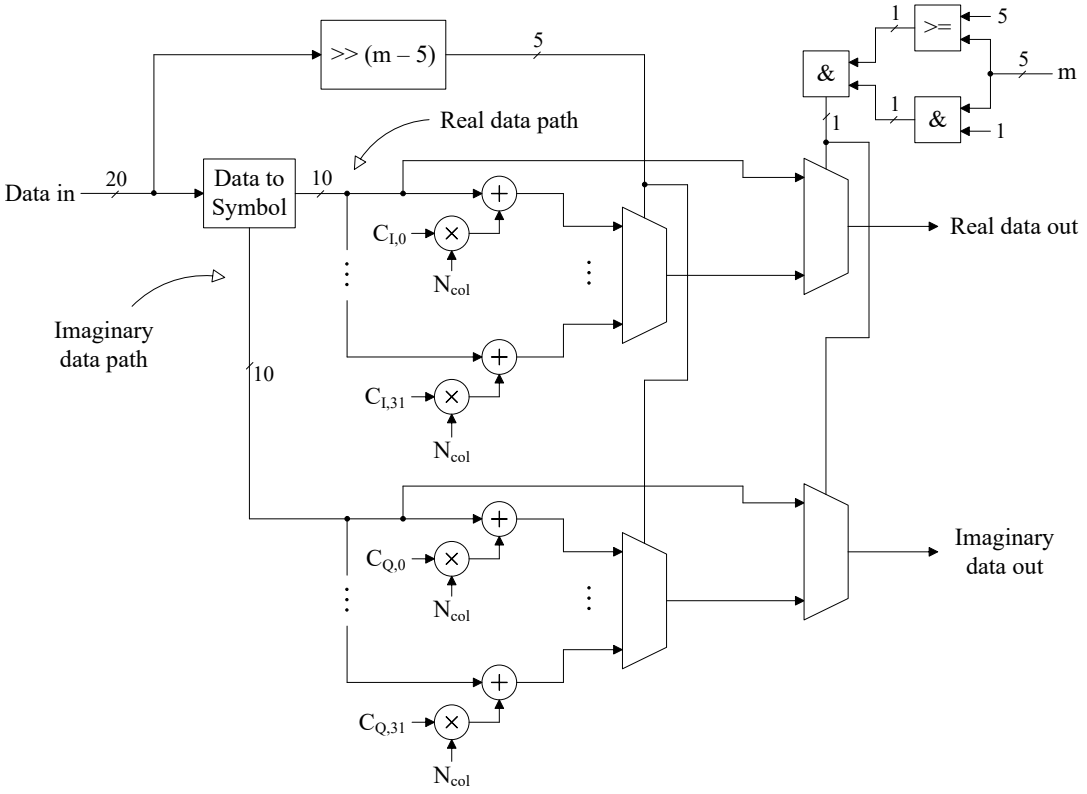

Figure 4.3: Functional block diagram of the X QAM mapper.

A functional block diagram of the X QAM mapper is displayed in Figure 4.3. In order to know in which subconstellation the data to be transmitted should be placed, we first need to create a condition on the five MSBs of the data. This is achieved by right-shifting the data by $m - 5$ where m is the number of bits per symbol. The MSB bits are used as control bits to select which bias, if any, to be added to the symbol.

The data is also mapped to a symbol in the same way as in the case of S QAM, i.e., by the taking every other bit as real and imaginary. A bias is then added to the real and imaginary data according to the subconstellation in which the symbol resides. The bias is calculated as $C_{I,k} \cdot N_{col}$ or $C_{Q,k} \cdot N_{col}$, where N_{col} is determined as shown in (2.21). The constants $\{C_{I,k}, C_{Q,k}\}$ take the values $[\{3,4\},\{2,3\},\{2,-3\},\{3,-4\},$

$\{-3,4\},\{-2,3\},\{-2,-3\},\{-3,-4\}]$ when the MSB bits are equal to 0, 2, 8, 10, 21, 23, 29, or 31, respectively. In all other cases $\{C_{I,k}, C_{Q,k}\}$ are equal to $\{0,0\}$. Note that this requires an extra bit in the quadrature to represent a larger range of values. However, this bit is not transmitted, as it carries no information.

Finally, we need to select between the real and imaginary data with the added bias and the unaltered data. If the bits per symbol are odd and ≥ 5 we select the former as the output, otherwise we select the latter.

It should be noted that the block diagram in Figure 4.3 reflects the functionality of the C++ code. There may be slight discrepancies in implementation in relation to the hardware generated by the HLS engine. However, the overall functionality is equivalent.

4.2 Symbol Demapping for X QAM

When demapping X QAM symbols we need to consider the diagonal boundary between corner constellations, e.g. 11 and 2 in Figure 4.2, to know where to push a symbol received in the vacant spot in one of the four corners. The way this situation is handled is shown in Figure 4.4. In the figure, we show the relevant boundaries for deciding where to push a symbol in the upper left corner of the X QAM constellation.

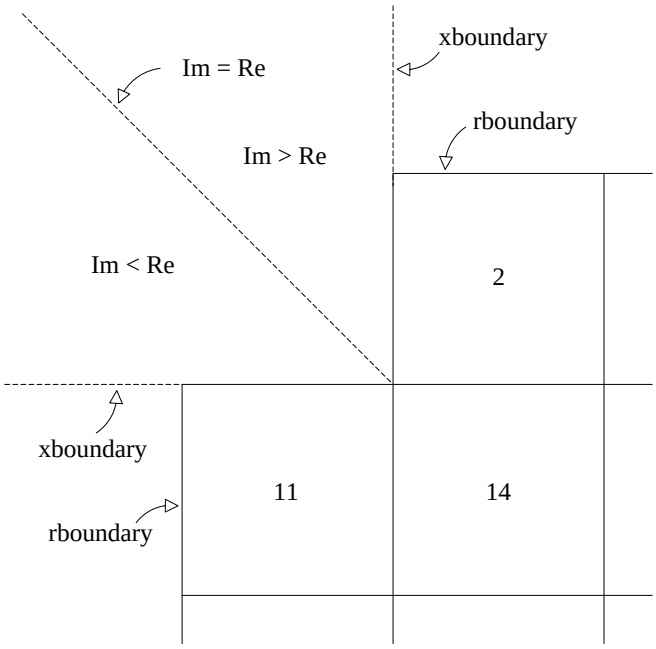

Figure 4.4: Decision boundaries for the upper left corner of the X QAM constellation.

After limiting (and rounding) we check whether the symbol, in the case shown above, is less than the *xboundary* in the real dimension and greater than the *xboundary* in the imaginary dimension. If that is the case we then need to establish if the symbol lies above or below the diagonal boundary. This boundary is inferred by comparing

the magnitudes of the real and imaginary parts. If the absolute value of the real is greater than the absolute value of the imaginary, then we are below the boundary and we set the real equal to *rboundary* and the imaginary equal to *xboundary*. If the opposite applies then we are above the boundary and we set the real equal to *xboundary* and the imaginary equal to *rboundary*. In the rare case where the real is exactly equal to the imaginary, implying that the symbol lies precisely on the diagonal boundary, it does not matter where we push it so we can just set the limits arbitrarily as there is no way of knowing from where it originated. This works similarly for all four corners.

After ensuring that the received symbol corresponds to a valid constellation point, we can proceed with demapping. Since we are now in the symbol domain, we need to resort to decision boundaries to know which subconstellation the symbol resides. First, we need to determine whether we are in one of the four subconstellations that have been moved to create the X QAM format and, if so, apply the opposite bias we did in the mapper to return to the rectangular NS QAM format. To do that, we consider the decision boundaries shown in Figure 4.5. We first check if the symbol lies above boundary 1 or below boundary 2 in the imaginary dimension. Then, we compare the symbol with boundary 4 and then boundary 3 or 5 in the real dimension. From here it is established to which subconstellation the symbol belongs and we can apply the appropriate bias. For example, if we conclude that the symbol lies in subconstellation 2 we would add -2 to the real and -3 to the imaginary.

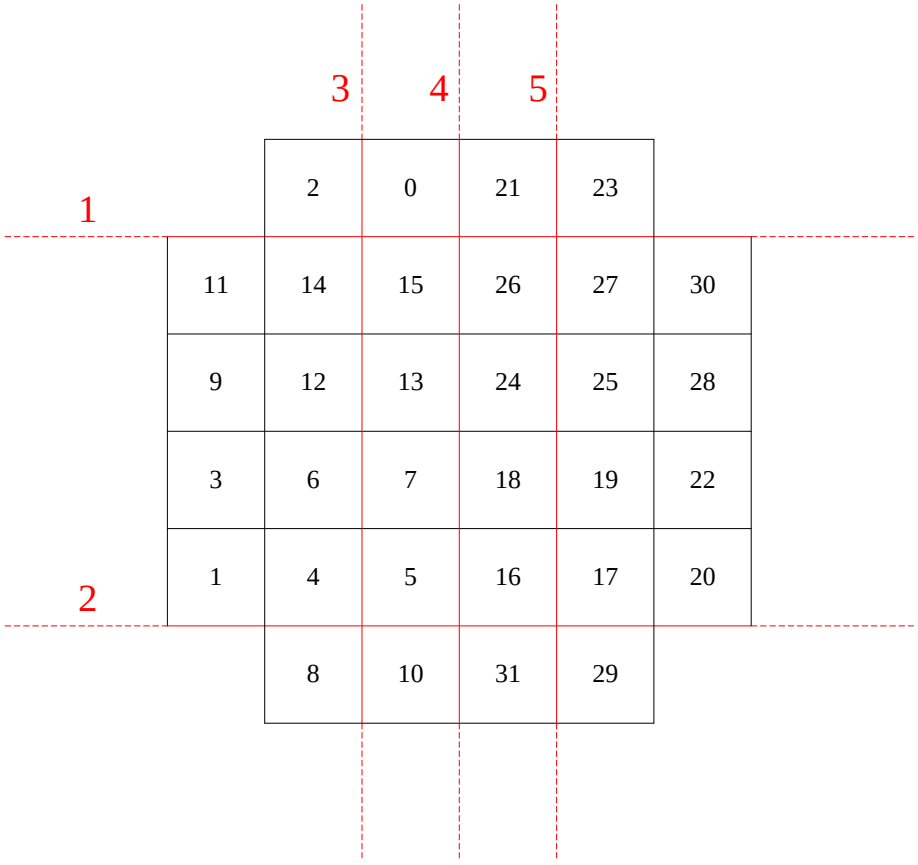

Figure 4.5: Demapper X QAM boundaries.

Once the rectangular NS QAM format is reconstructed we can convert the symbol to a single data word by placing the real bits in even bit positions and the imaginary bits on odd bit positions of the output data word.

When a symbol is received, we directly move away from the X QAM format and the symbol domain to perform operations in the data domain. However, corrections in the MSB decoders must be made in the symbol domain and they must therefore be adapted to be compatible with X QAM formats. This will be the topic of the next section.

4.3 MSB Correction for X QAM

The MSB decoders in the multi-layer FEC assume that the subconstellations are organized in a certain way for different numbers of uncoded bits. Therefore, they need to be modified in order to be compatible with X QAM formats. First, the uncoded bits are mapped to a symbol as described in Section 4.1. We then calculate the limits of the outer square contour of the X QAM constellation. The procedure after this follows what is described in Section 2.5.2. That is, if the absolute value of the error is larger than some fixed value, depending on the FEC layer, then a correction of ± 1 is made based on the sign of the error.

A correction is only made if the resulting value of the uncoded bits is strictly larger than the lower limit and strictly smaller than the upper limit. For S QAM this is straightforward. For X QAM, however, there is a case where we can correct to a non-existent subconstellation in one of the four corners. This happens since we set the limits of the X QAM constellation to trace the outer square contour. If a corrected symbol ends up in a non-existent subconstellation we push it to an existing one by limiting either the real or the imaginary dimension. The choice of which dimension to limit is arbitrary. This is because ending up in one of the four corners is always the result of an unrecoverable error in one or both dimensions. Thus, we will always get around 50% errors. We could also have a case where we have an unrecoverable error in both dimensions. However, this occurs even less frequently. Thus, having a case where we limit both dimensions will likely not have a major impact on performance. An example of the above described process is shown in Figure 4.6. Here we can see that the uncoded bits are incorrectly decoded, as we have an error $> 2D_{min}$ in the real dimension.

Another approach would be to consider the diagonal boundary, as we do in the demapper, when correcting symbols in the corners of the X QAM constellation. Doing so gives some extra margin to the decision boundary. However, an error of $2D_{min}$ in the second FEC layer or $16D_{min}$ in the fourth is already very large. That means that we will reduce the modulation order before errors of this magnitude become a problem. Thus, an implementation considering the diagonal boundary is not worth the logic overhead.

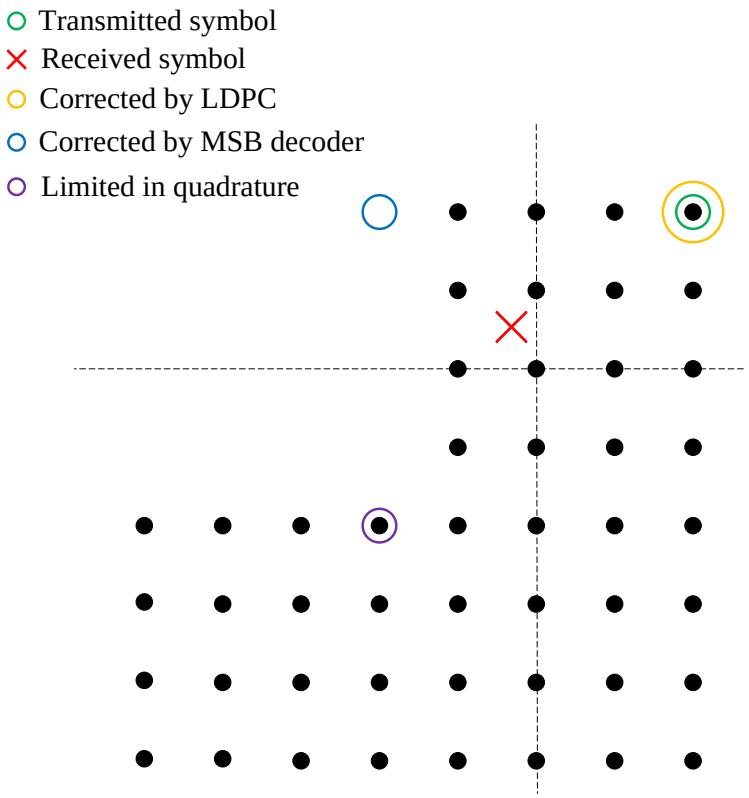

Figure 4.6: Process of decoding MSBs in the presence of an unrecoverable error.

4.4 Future Work

The design described in this chapter supports the use of X QAM formats together with LDPC and MSB decoding for bits per symbol equal to 9-19. Furthermore, it supports the use of X QAM formats with the entire multi-layer FEC (LDPC, RS and MSB decoding) for 15, 17 and 19 bits per symbol. Using RS is currently not supported when running the system with 9, 11 and 13 bits per symbol. Because of the way our symbol mapping scheme is designed, we need at least five uncoded bits in the second MSB decoder. Hence, it will not work for 9, 11 and 13 bits. A possible workaround is to limit the number of RS encoded bits for these formats. For 9 bits per symbol we could have zero RS bits, for 11 bits per symbol we could have two RS bits, and for 13 we could have four. This ensures that we always have at least five uncoded bits. However, this is not the desired functionality as we would like the number of RS bits to be programmable to 0, 2, 4, 6, or 8 (requires extra RS encoder/decoder) regardless of modulation format.

Having less than five uncoded bits also means that we need to handle diagonal decision boundaries between subconstellations in the second MSB decoder. This together with being able to map and demap less than five bits needs to be in place to run 9, 11, and 13 bits per symbol with the entire multi-layer. Supporting the use of RS with all X QAM formats as well as making the number of RS bits programmable is left as future work.

5

Results

This chapter presents the performance of our proposed design, as described in Chapter 4. After verifying the correctness of the system through simulation, an FPGA image was generated, loaded, and ran in a closed digital loop with an internal noise generator, emulating an AWGN channel. Running the design on hardware, we measured the BER produced across a fixed step sweep of different signal-to-interference-plus-noise ratio (SINR) values and obtained several BER curves for modulations ranging from S 256- to X 512k-QAM. In addition, we collected similar curves using two different LDPC code rates. In all cases where RS is active we are using 16 parity bytes. The specific number of parity bytes is not important, we only specify the value to justify the calculation of the total code rate, R_{tot}. The results and their relevance to the initially defined objectives are discussed in the following sections.

5.1 Multi-Layer FEC for Different Modulations

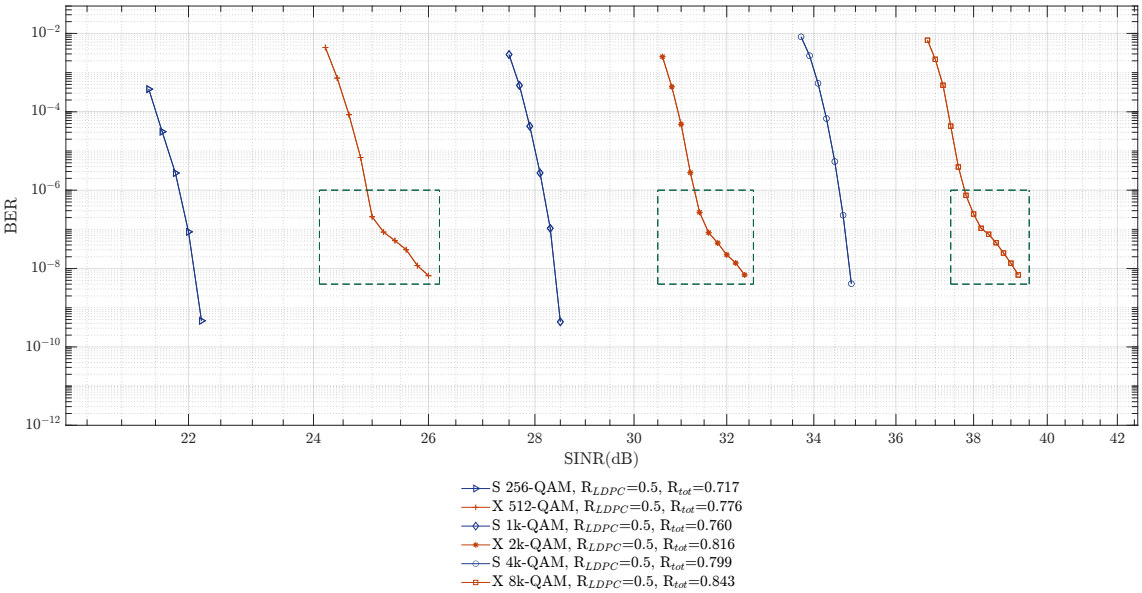

Figure 5.1: BER as a function of SINR for modulations S 256- to X 8192-QAM. Highlighted in green is the impact of RS unavailability.

As the modulation order increases, the distance between constellation points decreases; this occurs because more points must be packed into the same geometric space. Therefore, for the same noise level, higher-order modulations experience a higher BER compared to lower-order modulations under identical conditions. This principle is clearly depicted in Figure 5.1 and 5.2. Under the same code rate, different BER curves were obtained for S QAM and X QAM (shown in blue and red, respectively). It is evident that as the number of bits per symbol increases, the corresponding curves shift to the right, meaning lower performance in the face of noise.

A notable aspect in Figure 5.1, indicated by the dashed green boxes, is that for modulations of X 512-, 2048-, and 8192-QAM, where RS coding is not available, the rate of BER improvement decreases for the same incremental increase in SINR.

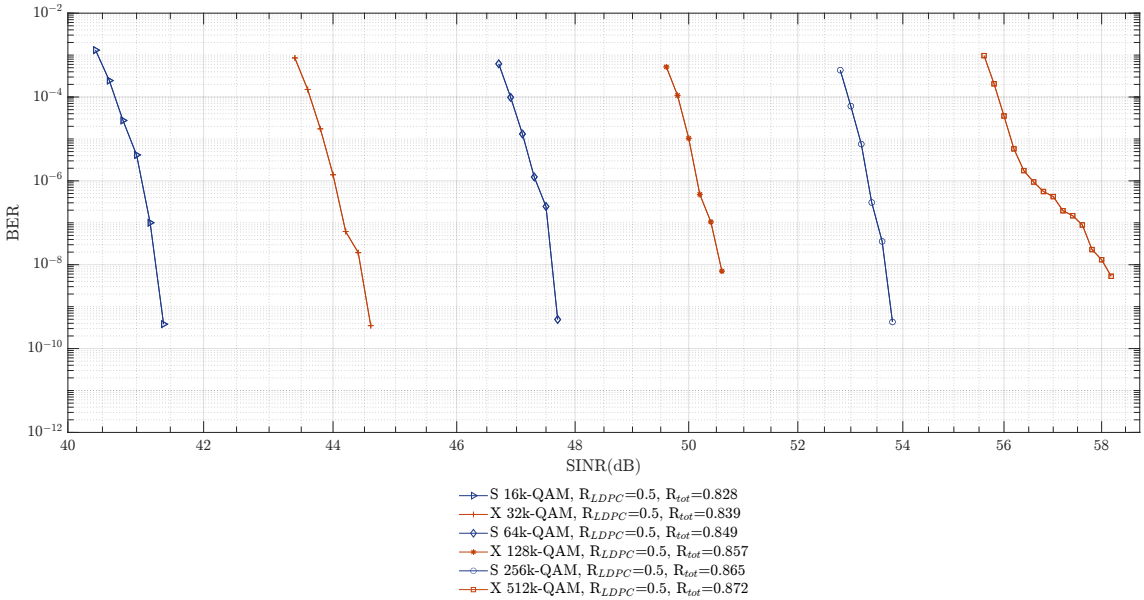

Figure 5.2: BER as a function of SINR for modulations from S 16k-QAM to X 512k-QAM, with 0.5 LDPC code rate.

The observed diminishing improvement in BER for X QAM formats does not apply to X 32k-QAM and above, for which RS coding is available. As shown in Figure 5.2, the X 32k-QAM curve behaves similarly to the adjacent S QAM curves, exhibiting a steady improvement in BER as SINR increases. We note that we experience a diminishing BER reduction for higher SINR for the X 512k-QAM format. This is because we hit the noise floor. Exactly where this noise originates from is not clear, but we know that the highest-order formats suffer from impairments unrelated to the fact that we are employing X QAM. Similar diminishing BER returns can be seen for S 1024k-QAM.

These results meet our objective of achieving higher spectral efficiency in comparison to the initial baseline. The communication link can now adapt to a noisy channel by switching to a lower, more robust modulation order with cross shape, sacrificing throughput in exchange for improved BER numbers.

5.2 Impact of Code Rate and Modulation Order

In this section, we explore the impact of the code rate on different modulation orders, as this is a key aspect of adaptive modulation. Given a channel condition with a certain noise level, the decision to switch to a lower-order modulation or to increase the code rate must balance transmission throughput against noise tolerance.

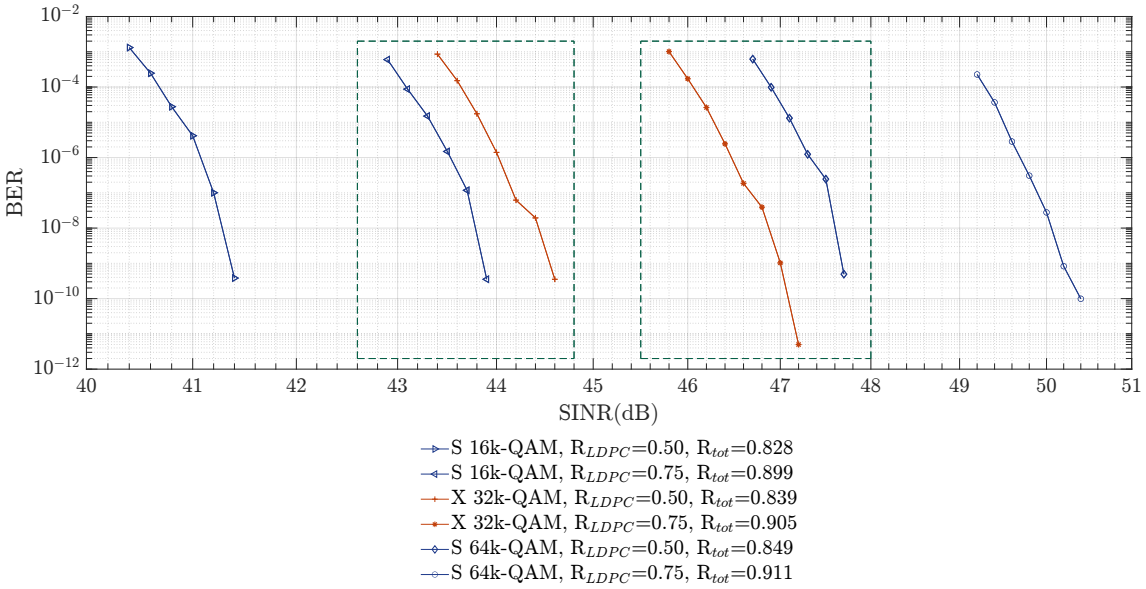

Figure 5.3: BER as a function of SINR for modulations S 16k-, X 32k-, and S 64k-QAM, with 0.5 and 0.75 LDPC code rate. Framed in green are curves with the same capacity.

As discussed in Section 2.4, the code rate represents the portion of the total block length that corresponds to the actual transmitted data, while the remaining fraction accounts for the overhead introduced by the encoding process. For example, a code rate of 0.5 means that half of the block consists of data and the other half belongs to error correction bits, while a code rate of 0.75 indicates that 75% of the block is data and 25% is allocated for FEC. This behavior is illustrated in Figure 5.3, where different BER curves were obtained for S 16k-, X 32k-, and S 64k-QAM modulations, each at LDPC code rates of 0.5 and 0.75. As expected, for the same modulation order, a higher number of coded bits (i.e. a lower code rate) results in a BER curve that shifts further to the left compared to that of a higher code rate.

The results obtained from these comparisons highlight the greater impact of increasing the spacing between constellation points (i.e. reducing the modulation order) compared to increasing FEC overhead. In a noisy channel, an adaptive modulation link can more effectively maintain throughput and improve BER performance by switching to a lower-order modulation rather than increasing the coding overhead. This is illustrated in Figure 5.3, where X 32k-QAM with an LDPC code rate of 0.75 lies approximately 1 dB to the left of S 64k-QAM with an LDPC code rate of 0.5. A similar behavior is observed when comparing X 32k-QAM with an LDPC code rate of 0.5 to S 16k-QAM with an LDPC code rate of 0.75. These formats have the

same capacity. Since we have four LDPC bits in total, increasing the code rate from 0.5 to 0.75 means that we now have 3/4 bits carrying information instead of 2/4. Thus, simultaneously reducing the modulation order from 16 to 15 bits per symbol results in exactly the same capacity. We are compensating for the fact that we have fewer bits per symbol by using less FEC overhead.

5.3 Resource Utilization

The proposed design was developed entirely using HLS as described in Chapter 3. This approach enabled faster development and verification of complex algorithms compared to conventional methods. In addition to the C++ source files that describe the design, the HLS engine also accepts specific directives that guide the synthesis process toward an implementation optimized for throughput, resources/area, or power, depending on the target requirements.

For this work, the primary concern was throughput, as the system needed to process large volumes of data at high speed. Therefore, the HLS engine was configured to prioritize performance over area. Additionally, all loops in the data path were unrolled, further increasing resource usage. Ultimately, the desired throughput was achieved and the reports showed only a small increase in resource usage, as presented in Table 5.1.

Resource	Former value [Units]	New value [Units]	Increment [%]
Total FFs	369751	382563	3.3%
Total LUTs	343784	367994	6.6%
DSPs	3450	3457	0.2%
RAMB36/FIFO	284	309	8.1%
RAMB18	254	256	0.8%

Table 5.1: Summary of resource usage, detailing former and new values corresponding to the designs without and with X QAM support.

6

Conclusion

In this work, we have presented a method to add support for X QAM formats of order 9-19 bits per symbol to Ericsson's multi-layer FEC system. The modified system is capable of running with the first two FEC layers for all X QAM formats. Furthermore, orders of 15, 17, and 19 bits per symbol are compatible with the entire multi-layer FEC.

The desired BER vs. SNR-curves were obtained, with the X QAM-curves filling the gaps between the S QAM-curves, implying improved ACM capabilities. Moreover, we display a 1 dB SNR-gain when switching to a lower-order X QAM compared to increasing FEC overhead for an S QAM format of higher order. This result is significant as these two options provide the same capacity. A similar benefit was observed when switching from an X QAM format to a lower-order S QAM format with a higher code rate, indicating higher noise resilience when lowering the modulation order compared to using a stronger code regardless of format. These results show the usefulness of X QAM formats when combined with FEC when it comes to adapting to varying channel conditions. All of this was achieved without significant increase in resource usage while maintaining high throughput.

Implementing support for running the system with RS parity for all X QAM formats, as well as making the number of RS bits programmable from software, was left as future work.

Bibliography

[1] J. Smith, "Odd-bit quadrature amplitude-shift keying," *IEEE Transactions on Communications*, vol. 23, no. 3, pp. 385–389, 1975.

[2] P. Vitthaladevuni, M.-S. Alouini, and J. Kieffer, "Exact BER computation for cross QAM constellations," *IEEE Transactions on Wireless Communications*, vol. 4, no. 6, pp. 3039–3050, 2005.

[3] A. Mirani, M. Mazur, E. Agrell, B. Foo, J. Schröder, P. A. Andrekson, and M. Karlsson, "Comparison of uniform cross QAM and probabilistically shaped QAM formats under the impact of transmitter impairments," in *45th European Conference on Optical Communication (ECOC 2019)*, 2019, pp. 1–4.

[4] J. B. Anderson, *Digital Transmission Engineering*, 2nd ed. Hoboken, NJ: Wiley-Interscience, 2005.

[5] L. Hanzo, S. X. Ng, T. Keller, and W. Webb, *Quadrature Amplitude Modulation: From Basics to Adaptive Trellis-Coded, Turbo-Equalised and Space-Time Coded OFDM, CDMA and MC-CDMA Systems*, 2nd ed. Chichester, UK / Hoboken, NJ: Wiley-IEEE Press, 2004.

[6] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*, 1st ed. New York: John Wiley & Sons, 1965.

[7] C. E. Bonferroni, "Teoria statistica delle classi e calcolo delle probabilita," (Pubbl. d. R. Ist. Super. di Sci. Econom. e Commerciali di Firenze. 8) Firenze: Libr. Internaz. Seeber. 62 S, 1936.

[8] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[9] H. M. Lucal, "Arithmetic operations for digital computers using a modified reflected binary code," *IRE Transactions on Electronic Computers*, vol. EC-8, no. 4, pp. 449–458, 1959.

[10] S. M. Moser and P.-N. Chen, *A Student's Guide to Coding and Information Theory*. Cambridge University Press, 2012.

[11] S. J. Johnson and S. R. Weller, "Regular low-density parity-check codes from oval designs," *European Transactions on Telecommunications*, vol. 14, no. 5, pp. 399–409, 2003.

[12] S. B. Wicker and V. K. Bhargava, *Reed-Solomon Codes and Their Applications*. Wiley–IEEE Press, 1994, ch. 1, pp. 1–16.

[13] I. E. Shparlinski, "Additive combinatorics over finite fields: New results and applications," in *Finite Fields and Their Applications*. Berlin, Boston: De Gruyter, 2013, pp. 233–272.