# Error Control Coding and Filter Banks

by

**José de Jesús Carmona Suárez, B.Eng.**

A dissertation submitted to the Department of Electronics Engineering in partial fulfilment of the requirement for the degree of

**MASTER OF SCIENCE IN ELECTRONICS**

at the

**Instituto Nacional de Astrofísica, Óptica y Electrónica**
September 2017
Tonantzintla, Puebla

Advisors:

**Gordana Jovanovic Dolecek, Ph.D.**
INAOE
**Alfonso Fernández Vázquez, Ph.D.**
ESCOM - IPN

*Para mis padres, Carolina y Jesús.*

# Acknowledgement

A big thanks to my parents, Carolina and Jesús, who have always been there for me and I am sure they will still be there.

Thanks to my advisors, Dr. Gordana Jovanovic Dolecek and Dr. Alfonso Fernández Vázquez, for their guidance, patience and support.

Thanks to the Instituto Nacional de Astrofísica, Óptica y Electrónica for accepting me as one of his students.

Thanks to the Consejo Nacional de Ciencia y Tecnología for funding my education.

Thanks to Dr. Lara Dolecek for accepting me as part of her team in the Laboratory for Robust Information Systems at the University of California, Los Angeles (UCLA) during my one month stay, from May 15th to June 15th 2017. Also many thanks to her student Homa Esfahanizadeh for helping me during this stay.

Thanks to the UCMEXUS project for supporting my stay in UCLA.

Thanks to my committee members Dr. Guillermo Espinosa Flores-Verdad, Dr. Esteban Tlelo Cuautle and Dr. Jorge Roberto Zurita Sánchez.

Thanks to my classmates at the INAOE, sharing this experience with you was awesome.

A big thanks to Dr. Saúl Martínez Díaz and Dr. Israel Marcos Santillán Méndez, they showed me the path to science and research.

A huge thanks to my sisters, Dulce and Carolina, being my sisters is a huge part of who I am and what I have achieved.

Thanks to María del Refugio Carmona, Nicolas Vega, Socorro Suárez and Jorge Canchola for helping me and supporting me while I was away from home.

Another big thanks to my cousins, Alejandro and Alberto Morales Suárez for sharing their home with me.

Finally, thanks to my friends and family, from La Paz, Puebla, Mexico City, Los Angeles and the rest of the world, meeting you and sharing time with you is what made this possible.

# Resumen

En esta tesis se propone un modelo de comunicaciones digitales para codificación de canal. El modelo utiliza técnicas de Bancos de Filtros Sobremuestreados y métodos de Verificación de Paridad de Baja Densidad (LDPC). Inicialmente se presentan los fundamentos de un sistema de comunicación y codificación. Se hace una breve descripción de varios tipos de codificación y simulaciones. En seguida se hace una introducción a la decodificación por ventanas de códigos LDPC. Después se menciona lo que es un banco de filtros sobremuestreado, sus propiedades y como diseñarlo. Posteriormente se explica cómo utilizar bancos de filtros para codificación. Finalmente se presenta el modelo de codificación de canal propuesto con simulaciones y conclusiones.

# Abstract

In this thesis a digital communications model for channel coding is proposed. The model uses Oversampled Filter Bank (OFB) techniques and Low-Density Parity-Check (LDPC) coding. First, the fundamentals of a communication channel and coding are presented. A brief description of some coding types and simulations are described. After that, an introduction to the window decoding of LDPC codes is introduced. Later is mentioned what is an OFB, its properties and how to design it. Then is explained how an OFB is used for coding. Finally, the proposed model for channel coding is presented with simulations and conclusions.

# Contents

# Chapter 1

# Introduction

*In this chapter the fundamentals of a communication channel are described. Also the fundamentals of coding are presented.*

## 1.1 Communication System

The simple model of digital communication system is presented in Figure 1.1. The principal blocks of this model are the *Source encoder*, the *Channel encoder*, and the *Modulator* in the Transmitter. In the receiver side, the principal blocks are the *Demodulator*, the *Channel decoder* and the *Source decoder*.

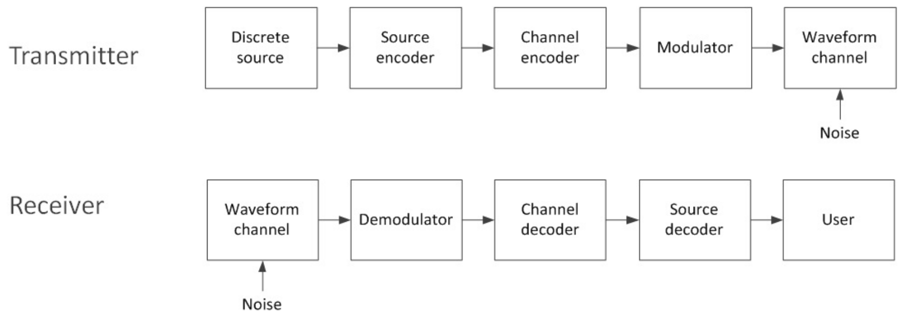


Figure 1.1: Digital communication system model.

This thesis is focused on the Channel encoder and the Channel decoder. This modules provide the capacity to protect the messages when they are exposed to noise in the channel by adding controlled extra information to the messages, also known as *redundancy*, in the Channel encoder. This redundancy is used by the Channel decoder to detect and correct errors in the corrupt messages. This techniques are applied to decrease the error probability in the transmission of messages.

## 1.2 Coding Fundamentals

### 1.2.1 Alphabet Coding

A data source must be analyzed as a random data source, because it is unknown what data is going to be delivered by the source. Still, the data's behaviour can be described by a probability distribution, assigning certain probability to each datum. An alphabet can be defined by a probability distribution. A source alphabet is defined as the collection of all possible messages [1]. An alphabet, denoted as $\mathcal{U}$, with $r$ messages, like $u_1$, $u_2$, ... , $u_r$, and with their respective probabilities $p_1$, $p_2$, ... , $p_r$ satisfy:

$$p_i \geq 0, \forall i, \tag{1.1}$$

and:

$$\sum_{i=1}^{r} p_i = 1. \tag{1.2}$$

### 1.2.2 Entropy

The amount of acquired information after the event $s_k$, with probability $p_k$, is:

$$I(s_k) = \log_2 \left( \frac{1}{p_k} \right). \tag{1.3}$$

The average information in the alphabet $\mathcal{U}$ is:

$$\begin{aligned} H(\mathcal{U}) &= E[I(s_k)] && (1.4) \\ &= \sum_{k=0}^{r} p_k I(s_k) && (1.5) \\ &= \sum_{k=0}^{r} p_k \log_2 \left( \frac{1}{p_k} \right). && (1.6) \end{aligned}$$

This amount $H(\mathcal{U})$ is called *entropy of a discrete memoryless source* with source alphabet $\mathcal{U}$. It provides us the average information per symbol in the source [2].

### 1.2.3 Discrete Memoryless Channel

A *Discrete Memoryless Channel* is a statistical model with an input $X$ and an output $Y$ which is a contaminated version of $X$; both $X$ and $Y$ are random variables [2]. The channel accepts an input symbol $X$ from the alphabet $\mathcal{X}$ and, in response, it emits a symbol $Y$ from an alphabet $\mathcal{Y}$. The channel is a discrete channel because both alphabets have finite sizes and it is memoryless because the output symbol does not depend on the input symbol.

The channel has an input alphabet:

$$\mathcal{X} = \{x_0, x_1, ..., x_{J-1}\}, \tag{1.7}$$

an output alphabet:

$$\mathcal{Y} = \{y_0, y_1, ..., x_{K-1}\}, \tag{1.8}$$

and a set of transition probabilities:

$$p(y_k|x_j) = P(Y = y_k|X = x_j). \tag{1.9}$$

### 1.2.4 Mutual Information

The mutual information denoted as $I(\mathcal{X}, \mathcal{Y})$ is defined as:

$$I(\mathcal{X}, \mathcal{Y}) = H(\mathcal{X}) - H(\mathcal{X}|\mathcal{Y}) \tag{1.10}$$

$$= H(\mathcal{Y}) - H(\mathcal{Y}|\mathcal{X}) \tag{1.11}$$

$$= I(\mathcal{Y}, \mathcal{X}), \tag{1.12}$$

where $H(\mathcal{X})$ is the input channel entropy, $H(\mathcal{Y})$ is the output channel entropy, $H(\mathcal{X}|\mathcal{Y})$ is the entropy of the input channel after knowing the output and $H(\mathcal{Y}|\mathcal{X})$ is the entropy of the output channel given certain input. The difference $H(\mathcal{X}) - H(\mathcal{X}|\mathcal{Y})$ represents the remaining uncertainty of the input after knowing the channel output [2].

The conditional entropy $H(\mathcal{X}|\mathcal{Y})$ is defined as:

$$H(\mathcal{X}|\mathcal{Y}) = \sum_{k=0}^{K-1} \sum_{j=0}^{J-1} p(x_j, y_k) \log_2 \left[\frac{1}{p(x_j|y_k)}\right], \tag{1.13}$$

where

$$p(x_j, y_k) = p(x_j|y_k)p(y_k). \tag{1.14}$$

### 1.2.5 Channel Capacity

The channel capacity of a discrete memoryless channel is the maximum mutual information, finding the distribution probability at the input that maximizes the mutual information [2]. The channel capacity is denoted as $C$ and defined as:

$$C = \max_{p(x_j)} I(\mathcal{X}, \mathcal{Y}). \tag{1.15}$$

### 1.2.6 Shannon's Channel Coding Theorem

For a discrete memoryless channel, is possible to transmit messages with a small error probability if communication rate $r$ is below or equal the channel capacity $C$, that is:

$$r \leq C. \tag{1.16}$$

The communication rate is defined as:

$$r = \frac{k}{n}, \tag{1.17}$$

where $k$ is the length of the message and $n$ is the length of transmitted bits [1].

# Chapter 2

# Code Classification

*In this chapter is given a classification of coding techniques. Also, a brief description of some coding types and simulations is presented.*

The coding techniques may be classified as linear and nonlinear codes. The linear codes may be subclassified in block codes and convolutional codes. This classification is shown in Figure 2.1.

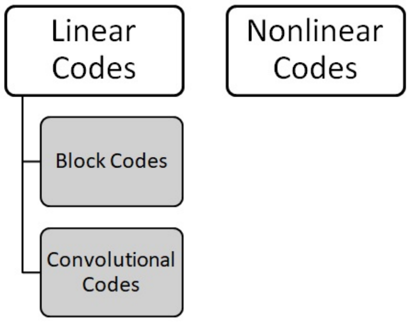


Figure 2.1: Coding classification.

This thesis only considers linear codes. Some of them are described in the next sections.

## 2.1 Linear Codes

A code is linear if the sum of any two *codewords* (encoded messages) generates a third codeword. This sum is performed in modulo-2 arithmetic. In a $(n, k)$ linear code, where $n$ is the length of the codeword and $k$ is the length of the message, the $n - k$ bits are generated from the message bits according to the coding rule that determines the code mathematical structure. This $n - k$ bits are called the parity bits [2].

The system can be defined in a matrix form like this:

$$\boldsymbol{m} = [m_0, m_1, ..., m_{k-1}] \tag{2.1}$$
$$\boldsymbol{b} = [b_0, b_1, ..., b_{n-k-1}] \tag{2.2}$$

$$\boldsymbol{c} = [c_0, c_1, ..., c_{n-1}] = [\boldsymbol{b} \vdots \boldsymbol{m}], \tag{2.3}$$

where $\boldsymbol{m}$ is the message vector, $\boldsymbol{b}$ is the parity vector, and $\boldsymbol{c}$ is the codeword.

The equation system that define the parity bits is:

$$\boldsymbol{b} = \boldsymbol{mP}, \tag{2.4}$$

where $\boldsymbol{P}$ is the *coefficient* matrix:

$$\boldsymbol{P} = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,n-k-1} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n-k-1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} \end{bmatrix}, \tag{2.5}$$

and $p_{i,j}$ is 0 or 1 and is set in a way that the rows of the generator matrix are linearly independent and the parity equations are unique.

From the equations (2.1)-(2.5) it can be noted that:

$$\boldsymbol{c} = [\boldsymbol{b} \vdots \boldsymbol{m}] \tag{2.6}$$

$$= [\boldsymbol{mP} \vdots \boldsymbol{m}] \tag{2.7}$$

$$= \boldsymbol{m}[\boldsymbol{P} \vdots \boldsymbol{I}_k], \tag{2.8}$$

where $\boldsymbol{I}_k$ is the $k$-by-$k$ identity matrix, and the generator matrix $\boldsymbol{G}$ is defined as:

$$\boldsymbol{G} = [\boldsymbol{P} \vdots \boldsymbol{I}_k]. \tag{2.9}$$

The codeword can be redefined as:

$$\boldsymbol{c} = \boldsymbol{mG}. \tag{2.10}$$

Other way to express the relation between the message bits and the parity matrix is with the parity check matrix $\boldsymbol{H}$, defined as:

$$\boldsymbol{H} = [\boldsymbol{I}_{n-k} \vdots \boldsymbol{P}^T]. \tag{2.11}$$

It can be shown that:

$$\begin{aligned} \boldsymbol{HG}^T &= 0 \\ &= \boldsymbol{GH}^T. \end{aligned} \tag{2.12}$$

Then:

$$\begin{aligned} \boldsymbol{c}\boldsymbol{H}^T &= \boldsymbol{m}\boldsymbol{G}\boldsymbol{H}^T \\ &= 0. \end{aligned} \tag{2.13}$$

The generator matrix and the parity check matrix are basic for the description and operation of a linear block code. This two equations are described in Figure 2.2 as blocks.

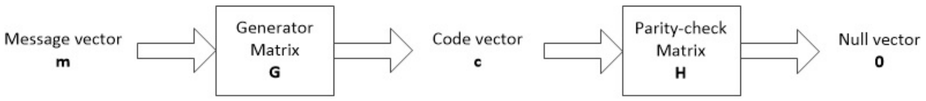


Figure 2.2: Linear coding [2].

During the decoding process, we define $\boldsymbol{r}$ as the resulting vector after sending the codeword $\boldsymbol{c}$ through a noisy channel like:

$$\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}, \tag{2.14}$$

where $\boldsymbol{e}$ is the error vector.

The $i$th element of $\boldsymbol{e}$ is 0 if the corresponding element of $\boldsymbol{r}$ is equal to $\boldsymbol{c}$, otherwise is 1. The decoding algorithm begins with the vector syndrome $\boldsymbol{s}$ defined as:

$$\boldsymbol{s} = \boldsymbol{r}\boldsymbol{H}^T, \tag{2.15}$$

which depends only on the error pattern $\boldsymbol{e}$ defined as:

$$\boldsymbol{s} = \boldsymbol{r}\boldsymbol{H}^T \tag{2.16}$$

$$= (\boldsymbol{c} + \boldsymbol{e})\boldsymbol{H}^T \tag{2.17}$$

$$= \boldsymbol{c}\boldsymbol{H}^T + \boldsymbol{e}\boldsymbol{H}^T \tag{2.18}$$

$$= \boldsymbol{e}\boldsymbol{H}^T. \tag{2.19}$$

Erroneous bits can be detected with the syndrome $\boldsymbol{s}$ from the received vector $\boldsymbol{r}$ and get the transmitted codeword $\boldsymbol{c}$. This is accomplished with the *Decoding table* defined as:

$$\boldsymbol{d} = \left[ \begin{array}{cc} \boldsymbol{0}_{1\times(n-k)} & \boldsymbol{0}_{1\times n} \\ \boldsymbol{H}^T & \boldsymbol{I}_n \end{array} \right]. \tag{2.20}$$

The first $n-k$ columns is the syndrome, and the last $n$ columns is where the error is located in the codeword. Flipping the bit in error is enough to correct the message.

### Minimum Distance

The *Hamming distance* between two code vectors is the number of different elements on the same position of the vectors and the *Hamming weight* is the number of "1" in a code vector. The *minimum distance* $d_{min}$ of a linear block code is defined by the minimum number of rows of the matrix $\boldsymbol{H}^T$ whose sum is equal to the zero vector [2]. The minimum distance defines the error-correcting capabilities of a code. An $(n, k)$ linear block code can correct up to $t$ errors if, and only if:

$$t \leq \left\lfloor \frac{1}{2}(d_{min} - 1) \right\rfloor, \tag{2.21}$$

where $\lfloor\rfloor$ is the largest integer function.

## Simulation Results

For a (7,4) block code the coefficient matrix is equal to [2]:

$$\boldsymbol{P} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}. \tag{2.22}$$

The generation matrix and the parity-check matrix are given in the matrices:

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.23}$$

and:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \tag{2.24}$$

The decoding table is presented as:

$$\boldsymbol{d} = \begin{bmatrix} 000 & 0000000 \\ 100 & 1000000 \\ 010 & 0100000 \\ 001 & 0010000 \\ 110 & 0001000 \\ 011 & 0000100 \\ 111 & 0000010 \\ 101 & 0000001 \end{bmatrix}. \tag{2.25}$$

If the message vector is $\boldsymbol{m} = [1000]$, the codeword vector will be $\boldsymbol{c} = [1101000]$ after the encoding process. Assuming one error produced by the channel, then the received vector could be $\boldsymbol{r} = [1101100]$ This means that the fifth bit is erroneous. When the received vector $\boldsymbol{r}$ is processed with the parity-check matrix, we get:

$$\begin{aligned} \boldsymbol{s} &= \boldsymbol{r}\boldsymbol{H}^T \\ &= [011]. \end{aligned}$$

After checking the decoding table (2.25), we note that the syndrome $\boldsymbol{s} = [011]$ corresponds to the vector [0000100], so we conclude that the fifth bit is in error, and we only flip the bit value to correct the message.

## 2.2 Cyclic Codes

A binary code is cyclic if it has two properties [2]:

1. Linear property: The sum of any two codeword is a codeword.
2. Cyclic property: Any shifted codeword is a codeword.

If the $n$-tuple $(c_0, c_1, ..., c_{n-1})$ is a linear block codeword $(n, k)$, then the code is cyclic if the $n$-tuples:

$$\begin{array}{l} (c_{n-1}, c_0, ..., c_{n-2}) \\ (c_{n-2}, c_{n-1}, ..., c_{n-3}) \\ \vdots \\ (c_1, c_2, ..., c_{n-1}, c_0) \end{array}$$

are also codewords. These coefficients define de *codeword polynomial* like:

$$c(X) = c_0 + c_1X + c_2X^2 ... + c_{n-1}X^{n-1}, \tag{2.26}$$

where $X$ represents a time shift [2].

### Generator Polynomial

If $g(X)$ is a polynomial of $(n-k)$ degree that is a factor of $X^n + 1$, generally $g(X)$ can be expanded like:

$$g(X) = 1 + \sum_{i=1}^{n-k-1} g_iX^i + X^{n-k}, \tag{2.27}$$

where the coefficients $g_i$ are 0 or 1.

The $g(X)$ polynomial is called the *generator polynomial* of the cyclic code. The cyclic code is uniquely determined by the generator polynomial. The coefficients of the generator polynomial shifts may define the generator matrix $\boldsymbol{G}$ in a linear code. Each codeword can be expressed as:

$$c(X) = a(X)g(X), \tag{2.28}$$

where $a(X)$ is a polynomial of $k-1$ degree.

Assuming that we have a generator polynomial and if we require to encode a message $(m_0, m_1, ..., m_{k-1})$ in a cyclic code like:

$$(b_0, b_1, ..., b_{n-k-1}, m_0, m_1, ..., m_{k-1}), \tag{2.29}$$

where $b_n$ are the $n-k$ parity bits, the *message polynomial* is defined like:

$$m(X) = m_0 + m_1X + ... + m_{k-1}X^{k-1}. \tag{2.30}$$

Similarly, the *parity polynomial* is equal to:

$$b(X) = b_0 + b_1X + ... + b_{n-k-1}X^{n-k-1}. \tag{2.31}$$

The codeword polynomial would be:

$$c(X) = b(X) + X^{n-k}m(X) \tag{2.32}$$
$$= a(X)g(X). \tag{2.33}$$

After applying some algebraic properties we get:

$$\frac{X^{n-k}m(X)}{g(X)} = a(X) + \frac{b(X)}{g(X)}. \tag{2.34}$$

This equation implies that $b(X)$ is the residual from dividing $X^{n-k}m(X)$ by $g(X)$.

The summary steps to achieve the codeword are these:

1. Multiply the message polynomial $m(X)$ by $X^{n-k}$.
2. Divide $X^{n-k}m(X)$ by $g(X)$ and get the residual $b(X)$.
3. Sum $b(X)$ to $X^{n-k}m(X)$ and obtain the codeword polynomial.

### Syndrome Polynomial

When the codeword is transmitted through a channel [2], the received polynomial is obtained:

$$r(X) = r_0 + r_1X + ... + r_{n-1}X^{n-1}. \tag{2.35}$$

The remainder of dividing $r(X)$ by the generator polynomial $g(X)$ (a polynomial of degree $(n-k-1)$ or less) is called the *syndrome polynomial* $s(X)$, and its coefficients define the syndrome vector. After the syndrome is calculated, the correcting procedure applied to a linear code codeword is also applied to correct a cyclic code codeword.

### Parity-Check Polynomial

A cyclic code is uniquely specified by its generator polynomial $g(X)$ but there is other polynomial of $k$ degree that can specify the code. This is called the *parity-check polynomial* [2], which is:

$$h(X) = 1 + \sum_{i=1}^{k-1} h_iX^i + X^k. \tag{2.36}$$

This polynomial is related to $g(X)$ as:

$$g(X)h(X) \mod (X^n+1) = 0. \tag{2.37}$$

Both polynomials $g(X)$ and $h(X)$ are factors of $X^n+1$:

$$g(X)h(X) = X^n + 1. \tag{2.38}$$

The coefficients of the parity-check polynomial $h(X)$ have to be reversed and shifted to define de parity-check matrix $\boldsymbol{H}$. Knowing this, we can define the reciprocal of the parity-check polynomial as:

$$X^k h(X^{-1}) = X^k \left(1 + \sum_{i=1}^{k-1} h_i X^{-i} + X^{-k}\right) \tag{2.39}$$

$$= 1 + \sum_{i=1}^{k-1} h_{k-i} X^i + X^k. \tag{2.40}$$

The coefficient $n$-tuples of this polynomials now can be used for the parity-check matrix $\boldsymbol{H}$.

### BCH Coding

The Bose-Chaudhuri-Hocquenghem (BCH) code is a special type of cyclic code where:

| | |
|---|---|
| Block length: | $n = 2^m - 1,$ |
| Number of message bits: | $k \geq n - mt,$ |
| Minimum distance: | $d_{min} \geq 2t + 1,$ |

where $m$ is an integer greater or equal to 3.

Each BCH code is a $t$-error correcting code in that it can detect and correct up to $t$ random errors per codeword.

### Simulation Results

With a (7,4) Hamming Code and the generator polynomial [2]:

$$\boldsymbol{g} = 1 + X + X^3, \tag{2.41}$$

we assume a message vector $\boldsymbol{m} = [0100]$. After following the steps to encode a message described in this section, we will obtain the codeword $\boldsymbol{c} = [0110100]$. Assuming one error produced by the channel, the received vector could be $\boldsymbol{r} = [0110101]$. This means that the seventh bit is in error. When $r(X)$ polynomial is divided by $g(X)$ we get the syndrome $\boldsymbol{s} = [101]$. To construct the parity-check matrix $\boldsymbol{H}$, we take the reciprocal of $h(X)$ and two shifted versions:

$$X^4 h(X^{-1}) = 1 + X^2 + X^3 + X^4 \tag{2.42}$$

$$X^5 h(X^{-1}) = X + X^3 + X^4 + X^5 \tag{2.43}$$

$$X^6 h(X^{-1}) = X^2 + X^4 + X^5 + X^6. \tag{2.44}$$

Using the coefficients of this polynomials we define $\boldsymbol{H'}$ as:

$$\boldsymbol{H'} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \tag{2.45}$$

This matrix is not in a *systematic form.* A systematic form means that the original message is embedded in the codeword. To achieve a systematic form we perform some row arithmetic operations and we obtain $\boldsymbol{H}$:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}, \tag{2.46}$$

which is equal to the matrix (2.24). So we can consult the syndrome in the decoding table (2.25) to check that the seventh bit is the one that has to be flipped.

## 2.3 Convolutional Codes

A convolutional code generates redundant bits applying modulo-2 convolutions [2]. It has $1/n$ code rate, and can be described by a finite state machine that consists on $M$-stage shift registers, $n$ modulo-2 adders, and multiplexers that serialize the system output.

An $L$ bits message produces an output of $n(L+M)$ length bits. The coding rate is:

$$r = \frac{L}{n(L+M)}. \tag{2.47}$$

Typically, $L >> M$ then the code rate is simplified to:

$$r \simeq \frac{1}{n}. \tag{2.48}$$

The convolutional code length is defined by the number which shows how one bit shifts of the message can influence the encoder output. In a coder with $M$-stage registers, the coder memory is equal to the $M$ message bits, and $K = M + 1$ shifts are required to the message bit to exit the system.

An $k = 2$ registers, and $n = 2$ adders convolutional encoder is depicted in Figure 2.3.

Each path between the encoder input and output can be expressed by the impulse response, defined as the response to a single bit 1 through the path and a zero initial stage. Also we can characterize each path by a generator polynomial, defined as the *unit-delay transform* of the impulse response, like:

$$g^{(i)}(D) = g_0^{(i)} + g_1^{(i)}D + g_2^{(i)}D^2 + ... + g_M^{(i)}D^M, \tag{2.49}$$

where $D$ denotes the unit delay variable, and $g_j^{(i)}$ are the coefficients that describe the impulse response through the $i$ path.

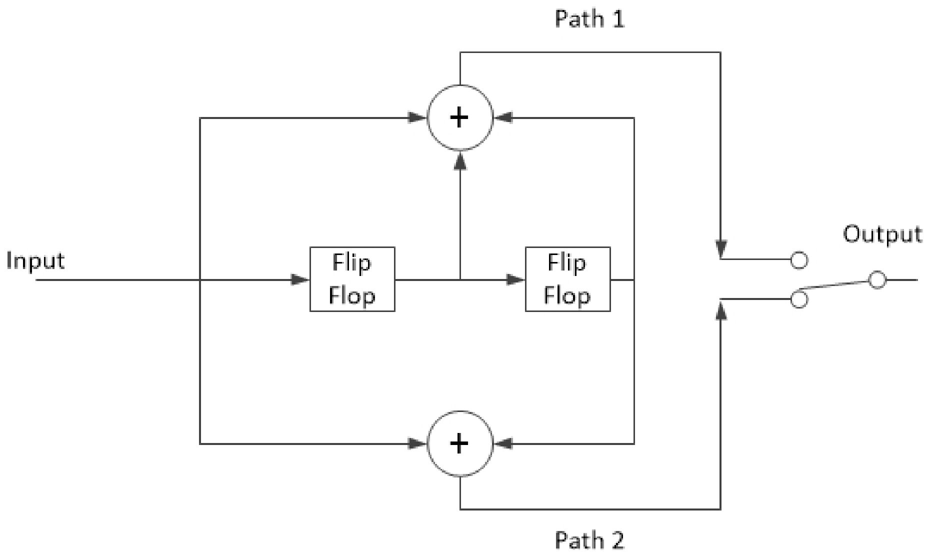


Figure 2.3: Convolutional encoder with 1/2 code rate [2].

The convolutional encoder shown in Figure 2.3 is described by the next set of generator polynomials:

$$g^{(1)}(D) = 1 + D + D^2 \tag{2.50}$$
$$g^{(2)}(D) = 1 + D^2. \tag{2.51}$$

If we have the message $\boldsymbol{m} = [10011]$, its message polynomial is:

$$m(D) = 1 + D^3 + D^4. \tag{2.52}$$

Like the Fourier transform, the output of the time domain convolution, is a product in the $D$ domain, then:

$$\begin{aligned} c^{(1)} &= g^{(1)}(D)m(D) && (2.53)\\ &= (1 + D + D^2)(1 + D^3 + D^4) && (2.54)\\ &= 1 + D + D^2 + D^3 + D^6, && (2.55) \end{aligned}$$

$$\begin{aligned} c^{(2)} &= g^{(2)}(D)m(D) && (2.56)\\ &= (1 + D^2)(1 + D^3 + D^4) && (2.57)\\ &= 1 + D^2 + D^3 + D^4 + D^5 + D^6 && (2.58) \end{aligned}$$

and the output through the path 1 is [1111001] and through the path 2 is [1011111]. Finally, with the multiplexers, the two output paths generate the codeword:

$$\boldsymbol{c} = [11, 10, 11, 11, 01, 01, 11]. \tag{2.59}$$

A trellis or a stage machine is a graphic way to describe a convolutional code. Figure 2.4 is a trellis of the Figure 2.3 coder, we can see the stages and the outputs when a 0 (solid line) and a 1 (dashed line) inputs the system. The stage diagram in Figure 2.5 provides the same information but in a compact way.

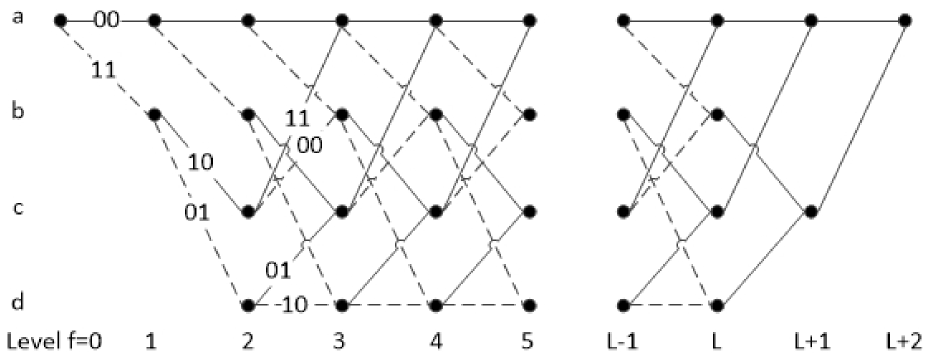


Figure 2.4: Trellis [2].

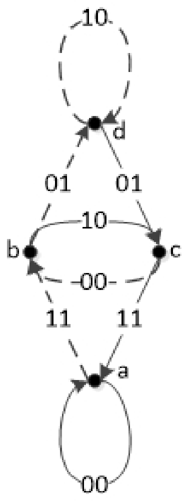


Figure 2.5: Stage diagram [2].

To decode a message is necessary to apply maximum likelihood techniques, which uses the fact that the correct message is the one that minimizes the Hamming distance between the received vector $\boldsymbol{r}$ and the transmitted vector $\boldsymbol{c}$. To achieve this we can use the Viterbi algorithm described next.

**Viterbi Algorithm [2]**

1. Initialization
   - Label the most left stage of the trellis as 0.
2. Computation step $j+1$
   - Let $j = 0, 1, 2, ...,$ and consider that in the last step of $j$ two things were done:
     - All the survivor paths where identified, *i.e.*, of all the roads ending in a single state, only the ones with the lower Hamming distance are saved.
     - All of the survivor paths and their Hamming distance to each stage are saved.
3. Final Step
   - Until the algorithm reaches the last node, the lower Hamming distance path is picked and that is the decoded message.

**Simulation Result**

Following the example explained before in this section, if the message is $\boldsymbol{m} =$ [10011] and the generator polynomials are (2.50) and (2.51). The codeword is $\boldsymbol{c} =$ [11101111010111]. When the codeword goes through the channel, an error is generated through the channel, so we will assume that the received vector is $\boldsymbol{r} =$ [11101111011111]. After applying the Viterbi algorithm, the decoded vector is equal to the original message.

## 2.4 Low-Density Parity-Check Codes

The Low-Density Parity-Check (LDPC) codes are specified by a parity-check matrix $\boldsymbol{A}$ which is *sparse* [2], *i.e.*, it consists mainly of "0" and a small number of "1". A $(n, t_c, t_r)$ LDPC code, consists of $n$ bits length, $t_c$ is the weight (the number of 1s) on each column of $\boldsymbol{A}$ and $t_r$ is the weight of each row with the constraint $t_r > t_c$. The code rate of an LDPC code is equal to:

$$r = 1 - \frac{t_c}{t_r}. \tag{2.60}$$

The rows in matrix $\boldsymbol{A}$ must be linearly independent.

An LDPC code structure is described by a bipartite graph that is shown in Figure 2.6 where an (10, 3, 5) LDPC code is presented. The left side nodes are called *variable nodes* that represent the codeword elements. The right side nodes are called *check nodes* that correspond to a set of parity-check constraints.

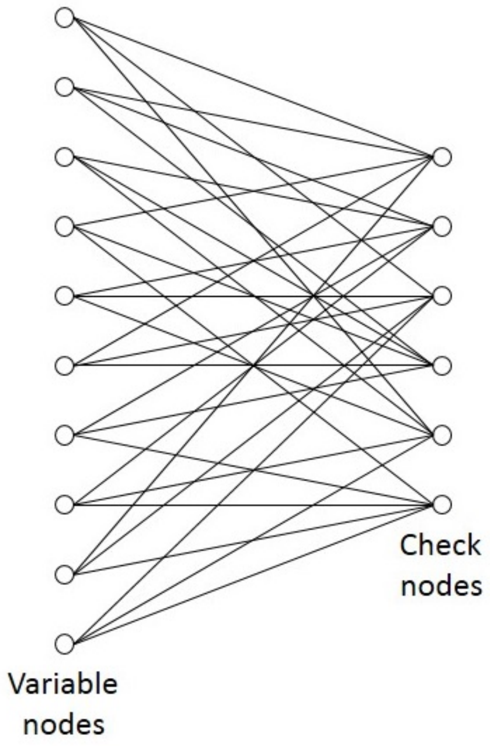


Figure 2.6: Bipartite graph of the (10, 3, 5) LDPC code [2].

The parity-check matrix $\boldsymbol{A}$ is defined as:

$$\boldsymbol{A}^T = \begin{bmatrix} \boldsymbol{A}_1 \\ \dots \\ \boldsymbol{A}_2 \end{bmatrix}, \tag{2.61}$$

where $\boldsymbol{A}_1$ is the $(n-k) \times (n-k)$ square matrix and $\boldsymbol{A}_2$ is the $k \times (n-k)$ rectangular matrix.

Using the variable definitions in Section 2.1 we have:

$$\begin{aligned} \boldsymbol{c}\boldsymbol{A}^T &= 0 && (2.62)\\ &= [\boldsymbol{b} \vdots \boldsymbol{m}] \begin{bmatrix} \boldsymbol{A}_1 \\ \dots \\ \boldsymbol{A}_2 \end{bmatrix} && (2.63)\\ &= \boldsymbol{b}\boldsymbol{A}_1 + \boldsymbol{m}\boldsymbol{A}_2 && (2.64)\\ &= \boldsymbol{m}\boldsymbol{P}\boldsymbol{A}_1 + \boldsymbol{m}\boldsymbol{A}_2 && (2.65)\\ &= \boldsymbol{P}\boldsymbol{A}_1 + \boldsymbol{A}_2, && (2.66) \end{aligned}$$

and solving for $\boldsymbol{P}$:

$$\boldsymbol{P} = \boldsymbol{A}_2\boldsymbol{A}_1^{-1}, \tag{2.67}$$

the generator matrix is defined as:

$$\boldsymbol{G} = [\boldsymbol{P} \vdots \boldsymbol{I}_k] \tag{2.68}$$

$$= [\boldsymbol{A}_2\boldsymbol{A}_1^{-1} \vdots \boldsymbol{I}_k]. \tag{2.69}$$

### Sum-Product Decoding Algorithm

An LDPC decoding method is the *Sum-Product Algorithm*, which is a message passing algorithm between the check nodes and the variable nodes of the decoding matrix $\boldsymbol{A}$. It has an horizontal step and a vertical step, which operate with the rows and columns of matrix $\boldsymbol{A}$, respectively. A bit refers to an element of the received vector and a check refers to a row of a matrix $\boldsymbol{A}$. Let $\mathfrak{J}(i)$ denote the set of bits in check $i$ and $\mathfrak{I}(j)$ denote the set of checks in which bit $j$ participates. The expression $a(k) \setminus b$ means the exclusion of element $b$ from the set $a(k)$. Next, the algorithm is described [2]:

- **Initialization** The variables $P_{ij}^0$ and $P_{ij}^1$ are set equal to the *a priori* probabilities $p_j^0$ and $p_j^1$ of symbols 0 and 1, respectively, with $p_j^0 + p_j^1 = 1$. $P_{ij}^x$ defines the probability that the bit in position $j$ is symbol $x$ (either a 0 or a 1), given the data from the checks performed in the horizontal step, except for check $i$.

- **Horizontal step** The runs on checks $i$ define:

  $$\Delta P_{ij} = P_{ij}^0 - P_{ij}^1.$$

  For each pair $(i, j)$ we define:

  $$\Delta Q_{ij} = \prod_{j' \in \mathfrak{J}(i) \setminus j} \Delta P_{ij'}$$

  and:

  $$Q_{ij}^0 = \frac{1}{2}(1 + \Delta Q_{ij})$$
  $$Q_{ij}^0 = \frac{1}{2}(1 + \Delta Q_{ij})$$

  where $Q_{ij}^x$ represents the probability that check $i$ is satisfied, given that bit $j$ is fixed at the value $x$ and the other bits have probabilities $P_{ij'} : j' \in \mathfrak{J}(i) \setminus j$.

- **Vertical step** For each bit $j$ compute:

  $$P_{ij}^0 = \alpha_{ij} p_j^0 \prod_{i' \in \mathfrak{I}(j) \setminus i} Q_{i'j}^0$$
  $$P_{ij}^1 = \alpha_{ij} p_j^1 \prod_{i' \in \mathfrak{I}(j) \setminus i} Q_{i'j}^1$$

where $\alpha_{ij}$ is chosen to make:

$$P_{ij}^0 + P_{ij}^1 = 1 \tag{2.70}$$

also:

$$\begin{aligned} P_j^0 &= \alpha_j p_j^0 \prod_{i \in \Im(j)} Q_{ij}^0 \\ P_j^1 &= \alpha_j p_j^1 \prod_{i \in \Im(j)} Q_{ij}^1 \end{aligned}$$

where $\alpha_j$ is chosen to make:

$$P_j^0 + P_j^1 = 1.$$

The results of the vertical step define a tentative vector $\hat{\boldsymbol{c}}$. If $\hat{\boldsymbol{c}}\boldsymbol{A}^T = 0$ is satisfied, the decoding is successfully finished. Otherwise, the algorithm returns to the horizontal step. If a defined number of iterations is performed and the decoding is not successful, the decoding is declared a failure.

### Simulation Result

With a (6, 2, 3) LDPC code and the parity-check matrix [3]:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}, \tag{2.71}$$

we define de generator matrix:

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \tag{2.72}$$

If the message vector is $\boldsymbol{m} = [11]$, the codeword vector will be $\boldsymbol{c} = [110011]$ after the encoding process. Assuming one error produced by the channel, then the received vector could be $\boldsymbol{r} = [110001]$. This means that the fifth bit is erroneous. When $\boldsymbol{r}$ is processed with the decoding algorithm the correct codeword is recovered.

# Chapter 3

# The Window Decoder

*First, Spatially-Coupled Codes and Window Decoder are presented. In the following, the performance of the window decoder and block decoder are compared. The comparison is made in terms of the decoding complexity, expressed with the average time needed to decode a frame of data and the average number of frames of data in error in the high Signal to Noise Ratio (SNR) region. The results are obtained by simulation.*

## 3.1 Spatially-Coupled Codes

The Spatially-Coupled (SC) codes are error correcting code techniques defined by partitioning an LDPC block $H$ to a number of component matrices $H_i$, and coupling $L$ replicas of these components together, as shown in Fig. 3.1, where $i \in \{0, 1, ..., m\}$, and $m$ is the memory parameter.

The SC codes have superior performance over memoryless binary-input channel and they approach to the Shannon limit. However, there is a large latency during the decoding due to large blocklenghts [4].

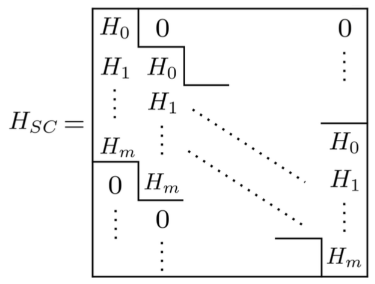


Figure 3.1: A SC-code parity check matrix with parameters $m$ and $L$ [5].

We are considering Circulant-Based (CB) LDPC codes to define the SC-coupled matrix. CB codes are a class of structured regular $(\gamma, \kappa)$ LDPC codes, where $\gamma$ is the column weight of the parity-check matrix, and $\kappa$ is the row weight.

The parity-check matrix $H$ of a CB code is constructed as follows [5]:

$$H = \begin{bmatrix} \sigma^{f_{0,0}} & \sigma^{f_{0,1}} & \cdots & \sigma^{f_{0,\kappa-1}} \\ \sigma^{f_{1,0}} & \sigma^{f_{1,1}} & \cdots & \sigma^{f_{1,\kappa-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{f_{\gamma-1,0}} & \sigma^{f_{\gamma-1,1}} & \cdots & \sigma^{f_{\gamma-1,\kappa-1}} \end{bmatrix},$$

where $\sigma$ is the $p \times p$ circulant matrix obtained by cyclically shifting the columns of an identity matrix the value in the exponent. In the parity-check matrix $H$, let $i, 0 \leq i \leq \gamma - 1$, be the row group index and $j, 0 \leq j \leq \kappa - 1$, be the column group index. The circulant exponents are non-negative integer values, for the simulations $f(i,j) = ij$ and $\kappa = p$.

A circulant-based SC code is defined by partitioning the $\kappa\gamma$ circulants in the parity-check matrix $H$, into component matrices $H_i, 0 \leq i \leq m$, where $m$ is the memory parameter. All $H_i$ components have the same size as $H$. They contain a subset of circulants in $H$, and the rest of the matrix elements is zero. Each circulant in $H$ is assigned to exactly one of the components, where $\sum_{i=0}^{m} H_i = H$.

## 3.2 Window Decoding

The Windowed Decoder (WD) breaks down the Belief Propagation (BP) algorithm decoding into a series of decoding steps. When using a window of size W, the WD performs BP over the subcode consisting of the first W sections of the variable nodes and their neighboring check nodes, and attempts to decode a subset of symbols (target symbols). Upon successful decoding of the target symbols, the window slides over one section and performs the BP algorithm [6].

There are two different approaches depending on what information the WD keeps after the BP algorithm finished decoding each window configuration:

1. Retain only the target nodes information and discard the nontarget nodes information.

2. Retain all the information. In this way, the decoding information of the nontarget nodes will be available for the decoding of the next window section.

Discarding some information between two window configurations can only perform worse than retaining all the information [6]. The WD aims to retain the features of BP, while reducing the complexity. Also, we obtain a reduced latency during the decoding.

Fig. 3.2 shows the simulation results for the SC code with block decoder and WD [6]. It shows that the performance of the WD increases as the window size increases, but a WD never has better performance than a block decoder.

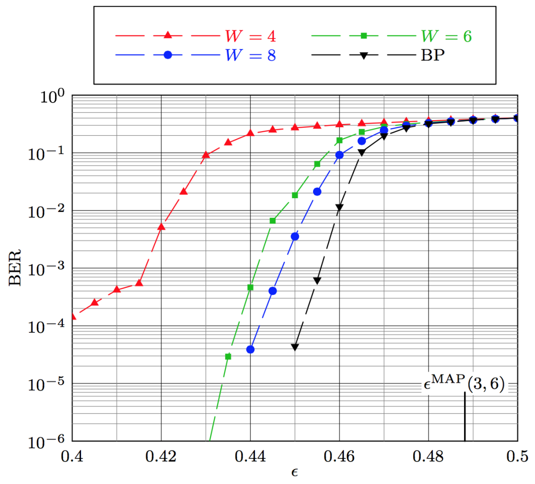


Figure 3.2: BER of the SC code with blocklengh 65,536 and window sized $W = 4, 6$ and 8 [6].

## 3.3 Comparison of Window Decoder and Block Decoder Using Simulation

This simulations compare the performance of the window decoder and block decoder in terms of the decoding complexity (the average time needed to decode a frame of data in both cases) and the error-floor (the average number of frames of data in error in the high SNR region). The simulations were coded in C++ and executed in the Hoffman server in UCLA.

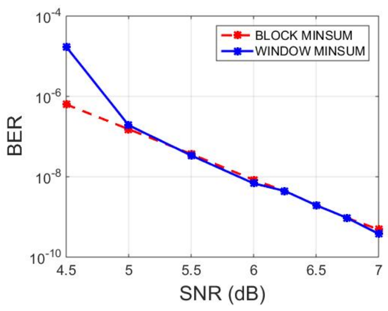
### Simulation Results

In this section, we present the results obtained by simulating WD and block decoding on a SC code, and a comparison between both of them.

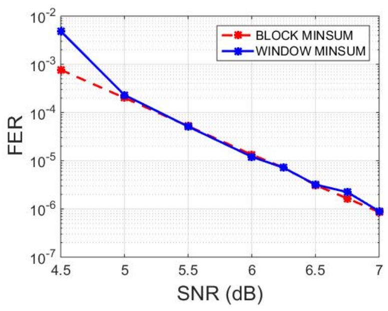
The SC code used for the simulation is an Array Based-SC code, where the size of the circulant is $p = 17$, the column weight is $\gamma = 3$, the row weight is $\kappa = 17$, the memory parameter is $m = 1$, and the number of coupled replicas is $L = 30$. The block length is 8670 bits. The simulation channel noise is Additive Gaussian White Noise (AWGN), with different SNRs and the number of iterations is 50. The BP algorithm implemented on the decoding is the MinSum algorithm.

For the WD, we used a window size $W = 3$. The implemented algorithm assigns the initial data from the channel for every window configuration at the beginning of ever WD, so it is equivalent to the first approach on how the WD keeps the information, where the data from the nontarget nodes is discarded after each section of the WD is decoded.

We can see in Fig. 3.3, that the block decoder behaves like a straight line for both the Bit Error Rate (BER) and the Frame Error Rate (FER), decreasing their values as the SNR increases. In the high SNR region, the performances of both decoders are similar. Even though, we have lower BER and higher FER in the window decoder. This is an improvement in the number of bits in error for the WD for highest SNR region, but as the SNR decreases, the graph shows that both BER and FER behaves more like the block decoder. When the SNR is 4.5 we see a significant reduction in the BER and FER. The average time for decoding a frame in block decoding is 0.01 seconds and in WD is 0.64 seconds.



(a)



(b)

Figure 3.3: Simulation results for block decoder and WD with $W = 3$ and blocklength of 8670 bits. (a) Bit Error Rate, (b) Frame Error Rate.

## Summary of Simulation

In the simulation results we see that the block decoder behaves almost like a linear function, *i.e.*, as the SNR increases the BER and FER decreases. This might be explained simply by considering that the signal strength is greater than the noise strength as we increase the SNR, so the errors in the channel are lower as the SNR increases. Both decoders, the block decoder and the WD show similar performance in the high SNR region, but the are not the same. The interesting thing is that the window size is small. In [6] is shown that as the window size increases, the BER decreases, but never as good as in the block decoder. In this results we see a reduction of the BER with a small window size. If we apply [6] we may expect lower BER when we increase the window size. The WD FER is greater than the block decoder FER and it decreases as the SNR decreases. As we are not decoding the entire frame, the decoding information in each window is limited, then more errors occur. When the SNR is 4.5 we can notice that the BER and FER have a significant reduction compared to the block decoder. The amount of information variables have decreased, so with less information, its harder to decode the channel codeword.

The time for decoding a frame with the WD is 64 times the time for decoding a frame with the block decoder. The necessary is that a WD performs the MinSum decoder on every window configuration. The advantage of the WD is that after the decoding of the first target nodes, they are ready to use them. However, in the block decoder, the user must wait until the whole blocklenght of data is decoded. Still the time difference between the block decoder and the WD is to big.

As the WD algorithm discards the information of the nodes, we may expect a better performance if we change this approach. It is better if we keep the information of the nontarget nodes, because certain level of decoding is already performed in these nodes, so that can help to decrease the complexity on the next window configuration decoding.

# Chapter 4

# Filter Banks

*In this chapter basic concepts of filter banks are described. Two kind of filter banks, the Maximally Decimated Filter Banks and the Oversampled Filter Banks, are introduced. Also the polyphase representation of a filter bank is presented.*

## 4.1 Maximally Decimated Filter Bank

A filter bank is composed of two parts, an analysis and a synthesis filter banks. The analysis bank splits the input signal $x(n)$ into the subband signals $v_k(n)$. The synthesis bank combines the subband signals $\hat{v}_k(n)$ into the output signal $\hat{x}(n)$. Both filter banks are composed of digital filters, with a common input in the analysis bank, and a common output in the synthesis bank. The filtered signals in the analysis bank $v_k(n)$ are downsampled by $M$, while the input signals of the synthesis bank $\hat{v}_k(n)$ are upsampled by $M$. If the number of subbands $N$ is equal to the downsample factor $M$, the filter bank is called Maximally Decimated filter bank as shown on Fig. 4.1, taking $N = M = 4$.

## 4.2 Filter Banks Errors

The reconstructed signal $\hat{x}(n)$ is, in the best case, an approximation of the original signal. There are some errors that occur during the splitting of the signal in the analysis bank that are mentioned in this section.

### Input-Output Relations for a Maximally Decimated Filter Bank

The input-output relation in the $z$-domain is expressed as [7]:

$$Y(z) = T_0(z)X(z) + \sum_{l=1}^{M-1} T_l(z)X(ze^{-j2\pi l/M}), \tag{4.1}$$

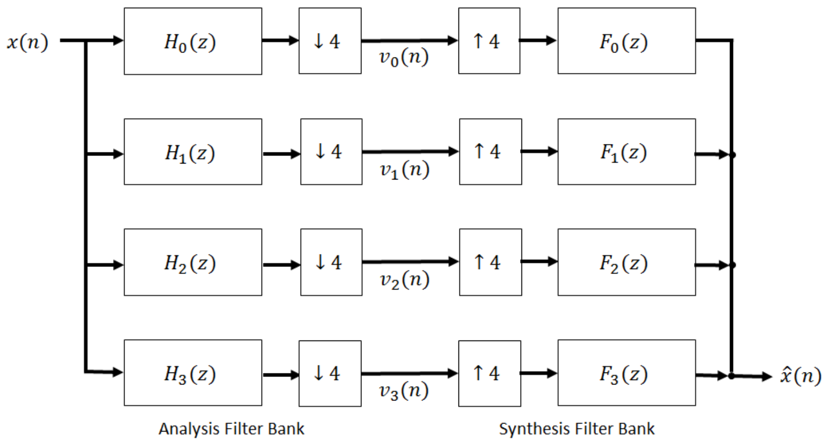


Figure 4.1: Maximally decimated filter bank.

where:

$$T_0(z) = \frac{1}{M}\sum_{k=0}^{M-1} F_k(z)H_k(z), \tag{4.2}$$

and:

$$T_l(z) = \frac{1}{M}\sum_{k=0}^{M-1} F_k(z)H_k(ze^{-j2\pi l/M}) \qquad \text{for} \qquad l = 1, 2, ..., M-1. \tag{4.3}$$

The factor $T_0(z)$ is called the distortion transfer function and determines the distortion caused by the overall system for the unaliased component $X(z)$ of the input signal. The factor $T_l(z)$ is the alias transfer function and determine how well the aliased components $X(ze^{-j2\pi l/M})$ of the input signal are attenuated [7].

## Perfect Reconstruction

If a filter bank does not present any aliasing, phase or amplitude distortion, it accomplishes with the *perfect reconstruction* (PR) property. Meaning:

$$\hat{X}(z) = cz^{-n_0}X(z), \qquad i.e., \qquad \hat{x}(n) = cx(n-n_0), \qquad c \neq 0, \tag{4.4}$$

for any signal $x(n)$, where $c$ is a real value and $n_0$ is an integer greater than zero. In other words, $\hat{x}(n)$ is a scaled and time delayed version of the $x(n)$ signal.

This is accomplished if the distortion transfer function is equal to $T_0(z) = z^{-n_0}$ and the alias transfer function is equal to zero in (4.1).

## 4.3 Polyphase Representation

A transfer function $H_k(z)$ can be expressed as [8]:

$$H_k(z) = \sum_{l=0}^{M-1} z^{-l} E_{kl}(z^M) \qquad \text{(Type 1 polyphase).} \tag{4.5}$$

That can be written as:

$$\begin{bmatrix} H_0(z) \\ \vdots \\ H_{M-1} \end{bmatrix} = \begin{bmatrix} E_{0,0}(z^M) & E_{0,1}(z^M) & \dots & E_{0,M-1}(z^M) \\ E_{1,0}(z^M) & E_{1,1}(z^M) & \dots & E_{1,M-1}(z^M) \\ \vdots & \vdots & \ddots & \vdots \\ E_{M-1,0}(z^M) & E_{M-1,1}(z^M) & \dots & E_{M-1,M-1}(z^M) \end{bmatrix} \begin{bmatrix} 1 \\ z^{-1} \\ \vdots \\ z^{-(M-1)} \end{bmatrix}, \tag{4.6}$$

or like:

$$\boldsymbol{h}(z) = \boldsymbol{E}(z^M)\boldsymbol{e}(z), \tag{4.7}$$

where:

$$\boldsymbol{E}(z) = \begin{bmatrix} E_{0,0}(z) & E_{0,1}(z) & \dots & E_{0,M-1}(z) \\ E_{1,0}(z) & E_{1,1}(z) & \dots & E_{1,M-1}(z) \\ \vdots & \vdots & \ddots & \vdots \\ E_{M-1,0}(z) & E_{M-1,1}(z) & \dots & E_{M-1,M-1}(z) \end{bmatrix}, \tag{4.8}$$

this representation is depicted in the Figure 4.2.

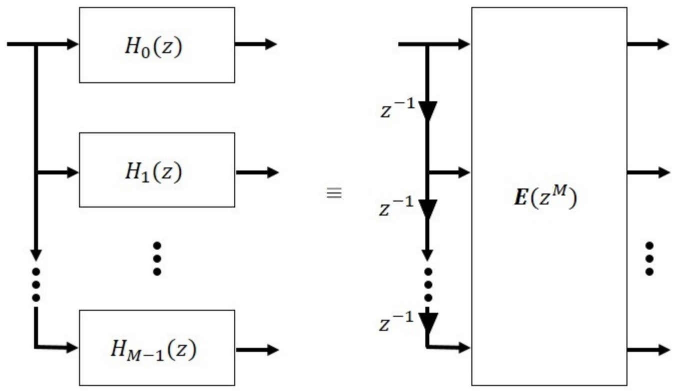


Figure 4.2: Type 1 polyphase representation of an analysis bank [8].

We can express the synthesis filters in a similar way:

$$F_k(z) = \sum_{l=0}^{M-1} z^{-(M-1-l)} R_{lk}(z^M) \qquad \text{(Type 2 polyphase).} \tag{4.9}$$

In matrix notation:

$$[F_0(z) \quad \cdots \quad F_{M-1}(z)] =$$
$$\begin{bmatrix} z^{-(M-1)} & z^{-(M-2)} & \cdots & 1 \end{bmatrix} \begin{bmatrix} R_{0,0}(z^M) & \dots & R_{0,M-1}(z^M) \\ R_{1,0}(z^M) & \dots & R_{1,M-1}(z^M) \\ \vdots & \ddots & \vdots \\ R_{M-1,0}(z^M) & \dots & R_{M-1,M-1}(z^M) \end{bmatrix}. \quad (4.10)$$

Also, in terms of $\boldsymbol{e}(z)$ and the synthesis bank vector $\boldsymbol{f}^T(z)$:

$$\boldsymbol{f}^T(z) = z^{-(M-1)}\tilde{\boldsymbol{e}}(z)\boldsymbol{R}(z^M), \quad (4.11)$$

where:

$$\boldsymbol{R}(z) = \begin{bmatrix} R_{0,0}(z) & R_{0,1}(z) & \dots & R_{0,M-1}(z) \\ R_{1,0}(z) & R_{1,1}(z) & \dots & R_{1,M-1}(z) \\ \vdots & \vdots & \ddots & \vdots \\ R_{M-1,0}(z) & R_{M-1,1}(z) & \dots & R_{M-1,M-1}(z) \end{bmatrix}. \quad (4.12)$$

and the paraconjugate of $\boldsymbol{A}$ is equal to $\tilde{\boldsymbol{A}} = \boldsymbol{A}_*^T$, where $\boldsymbol{A}^T$ is the transpose of $\boldsymbol{A}$, and $\boldsymbol{A}_*$ means the conjugation of only the coefficients of the filters of the matrix $\boldsymbol{A}$.

The synthesis bank vector $\boldsymbol{f}^T(z)$ is depicted in Figure 4.3.

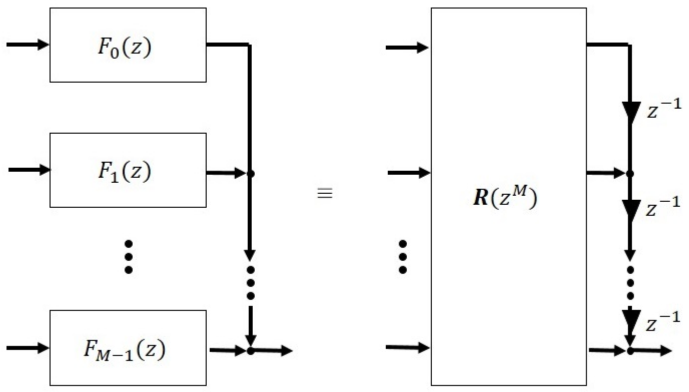


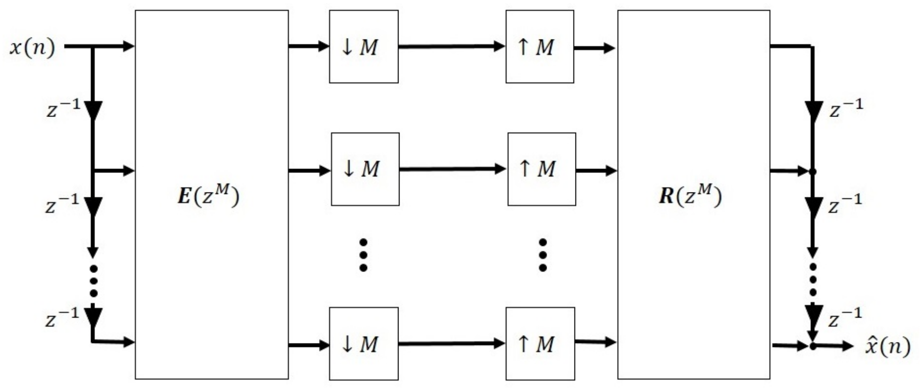
Figure 4.3: Type 2 polyphase representation of a synthesis bank [8].

Joining the analysis bank and the synthesis bank, the system depicted on Figure 4.4(a) is obtained. The Figure 4.4(b) is a simplified form when we use the noble identities.

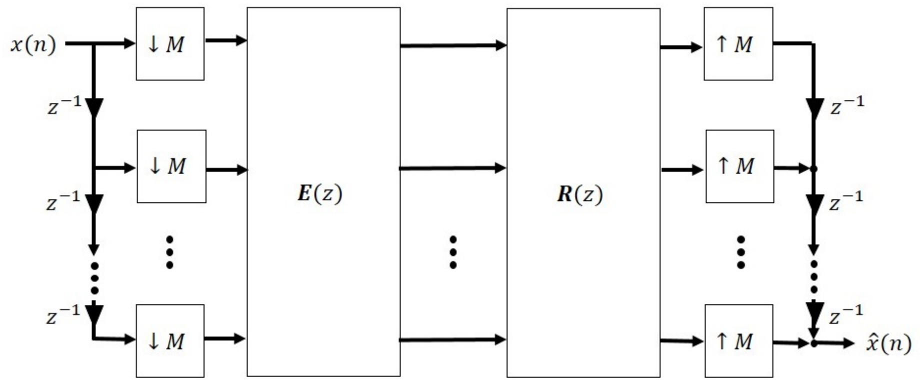
## Paraunitary Property

For rational transfer functions:

$$\tilde{\boldsymbol{H}}(z)\boldsymbol{H}(z) = d\boldsymbol{I}, \qquad \text{for all } z, \quad (4.13)$$



(a)



(b)

Figure 4.4: (a) Polyphase representation of an M-channel filter bank. (b) Rearrangement using noble identities [8].

which is called the *paraunitary* property [8].

The paraunitary property implies that:

$$\tilde{\boldsymbol{E}}(z)\boldsymbol{E}(z) = d\boldsymbol{I}, \qquad \text{for all } z \qquad d > 0. \tag{4.14}$$

So we define:

$$\boldsymbol{R}(z) = cz^{-K}\tilde{\boldsymbol{E}}(z) \tag{4.15}$$

for some $c \neq 0$, to satisfy the perfect reconstruction property. In other words, the polyphase matrix $\boldsymbol{P}(z)$ in Figure 4.5, must be $\boldsymbol{I}(z)$ to have perfect reconstruction.

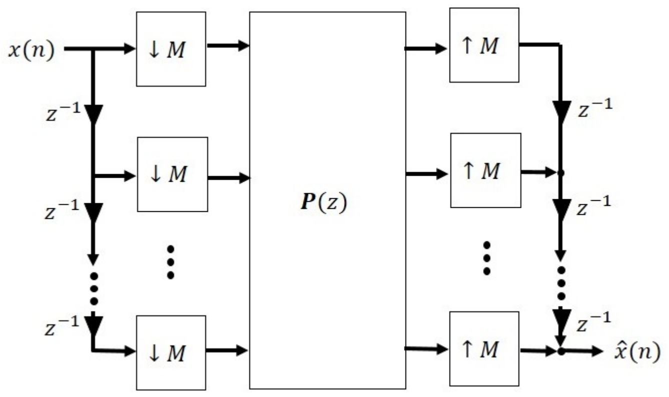


Figure 4.5: Simplification polyphase representation of an M-channel filter bank [8].

## 4.4 Oversampled Filter Banks

A filter bank, in which the number of subbands $N$ is greater than the downsampling factor $M$, $N > M$, the subband signals $v_k(n)$ are redundant. They contain more samples (per unit time) than the input signal $x(n)$. This kind of filter bank is called Oversampled Filter Bank (OFB), as shown in Figure 4.6, for $N = 6$ and $M = 6$.

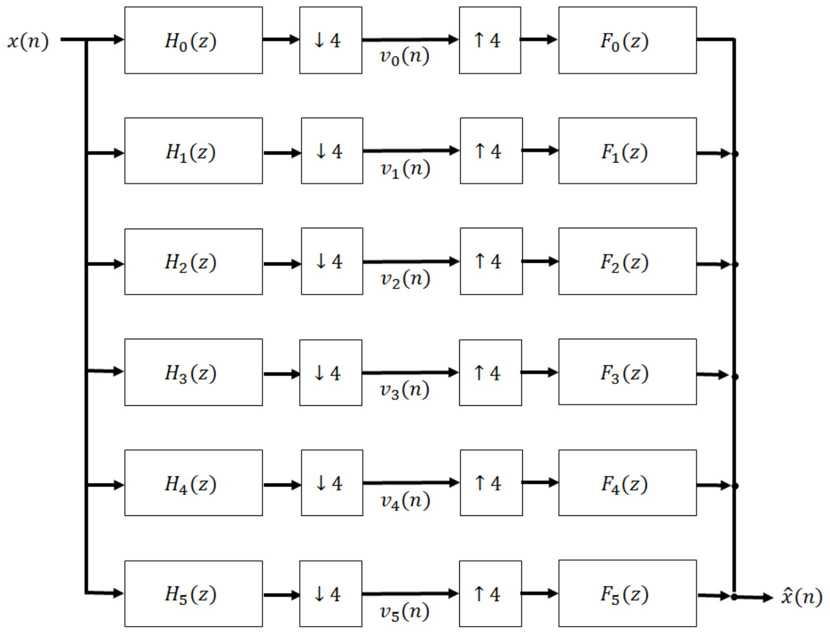


Figure 4.6: Oversampled filter bank.

Chapter 5

# Oversampled Filter Banks Design

*Although, there are many ways to design an OFB [9, 10, 11, 12, 13, 14, 15], in this thesis we will explain only the Discrete Fourier Transform Modulation technique. This chapter presents a brief description of this PR OFB design technique.*

## 5.1 Discrete Fourier Transform Modulation

The analysis bank of an OFB is designed with the DFT modulation of the prototype lowpass filter described with the impulse response $p(n)$. Denoting the causal FIR analysis filters impulse responses as $h_k(n)$, of length $L$, we have:

$$h_k(n) = \frac{1}{\sqrt{N}} p(n) \exp\left(j\frac{2\pi kn}{N}\right), \qquad n = 0, ..., L-1, \qquad k = 0, ..., N-1. \tag{5.1}$$

The $N \times M$ analysis and synthesis polyphase matrices, $\boldsymbol{H}_p(z)$ and $\boldsymbol{F}_p(z)$, respectively, have the following efficient realizations as depicted in Figure 5.1:

$$\boldsymbol{H}_p(z) = \boldsymbol{W}_N^H \boldsymbol{E}(z) \tag{5.2}$$

$$\boldsymbol{F}_p(z) = \tilde{\boldsymbol{H}}_p(z) \tag{5.3}$$

$$= \tilde{\boldsymbol{E}}(z)\boldsymbol{W}_N, \tag{5.4}$$

where $\boldsymbol{W}_N$ is the $N \times N$ orthonormal DFT matrix, $W_N = \exp\left(\frac{-2j\pi}{N}\right)$, $\boldsymbol{A}^H = (\boldsymbol{A}^*)^T$ is the hermitian of $\boldsymbol{A}$, and $\boldsymbol{E}(z)$ is the sparse matrix with the elements:

$$[\boldsymbol{E}(z)]_{i,j} = \begin{cases} z^{-l} P_{j+lN}(z^J), & \text{if } (i-j) \mod b = 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

$$i = 0, ..., N-1, \qquad j = 0, ..., M-1, \qquad (j + lM) \mod N = i, \tag{5.6}$$

where the greatest common divider of $M$ and $N$ is denoted by $g = \gcd(M, N)$, the least common multiple by $K = \mathrm{lcm}(M, N)$, $J = N/b$ [12] and $P_k(z)$ is the $z$ transform of the $k$th of $K$ type-I polyphase components of the lowpass prototype filter $p(n)$.

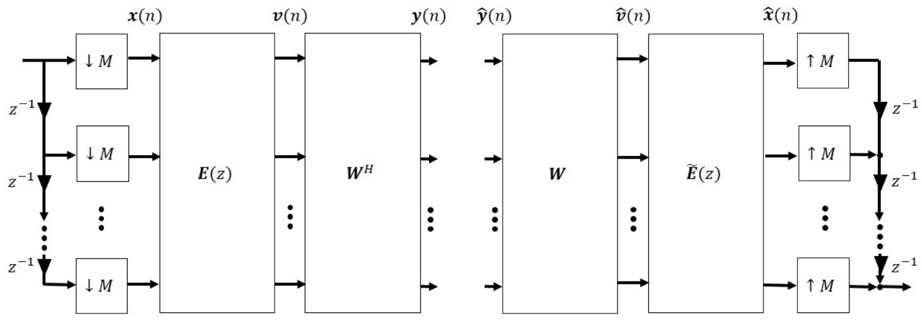


Figure 5.1: Polyphase realization of oversampled DFT-modulated filter bank [12].

The PR constraints for paraunitary oversampled DFT filter banks can be expressed as:

$$\boldsymbol{F}_p(z)\boldsymbol{H}_p(z) = \tilde{\boldsymbol{H}}_p(z)\boldsymbol{H}_p(z) = \tilde{\boldsymbol{E}}(z)\boldsymbol{E}(z) = \boldsymbol{I}_M, \tag{5.7}$$

and $\boldsymbol{E}(z)$ can be partitioned into $b$ independent set of PR matrices:

$$\begin{aligned} \tilde{\boldsymbol{E}}_l(z)\boldsymbol{E}_l(z) = \boldsymbol{I}_{M/b} \qquad &\text{with} \qquad [\boldsymbol{E}_l(z)]_{k,j} = [\boldsymbol{E}(z)]_{l+kb,l+jb}, \\ l = 0, ..., b-1, \qquad &k = 0, ..., J-1, \\ j = 0, ..., M/b - 1. & \end{aligned} \tag{5.8}$$

As $\boldsymbol{E}_l(z)$ contains delayed $J$-fold upsampled polyphase components of the prototype filter, it can be expressed as a product of $J \times J$ and $M/b \times M/b$ paraunitary diagonal matrices $\boldsymbol{\Lambda}_1(z)$ and $\boldsymbol{\Lambda}_2(z)$, respectively, containing monomials in $z$, and a matrix $\boldsymbol{E}_{l\downarrow}(z^J)$ of size $J \times M/b$ that contains polynomials in $z^J$ [12]:

$$\boldsymbol{E}_l(z) = \boldsymbol{\Lambda}_1(z)\boldsymbol{E}_{l\downarrow}(z^J)\boldsymbol{\Lambda}_2(z). \tag{5.9}$$

The $J$ times downsampled version of the PR matrices becomes:

$$\tilde{\boldsymbol{E}}_{l\downarrow}(z)\boldsymbol{E}_{l\downarrow}(z) = \boldsymbol{I}_{M/b}. \tag{5.10}$$

**Example 1**

In this simulation we generate an OFB with $N = 8$ subbands, a decimator factor of $M = 6$ and a filter length of $L = 48$. Then the greatest common divider is $g = 2$, the least common multiple is $K = 24$, $J = 4$. The prototype FIR filter $p(n)$ is a Nyquist Filter which is modulated using (5.1). Figure 5.2 shows the frequency responses of the designed filters. A random goes through the filter bank, and without channel noise, we can compare the output signal with an input signal shown in Figure 5.3. We can observe the PR property and the output signal delay.

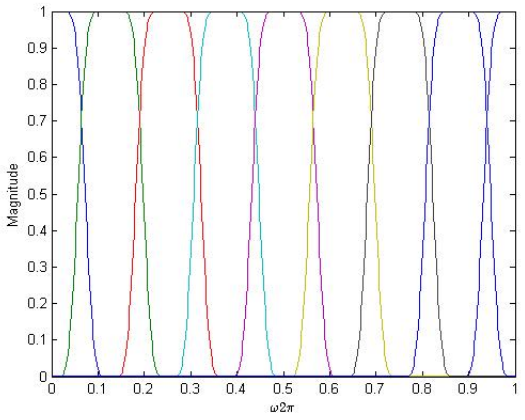


Figure 5.2: Frequency response for the filters in the example 1.

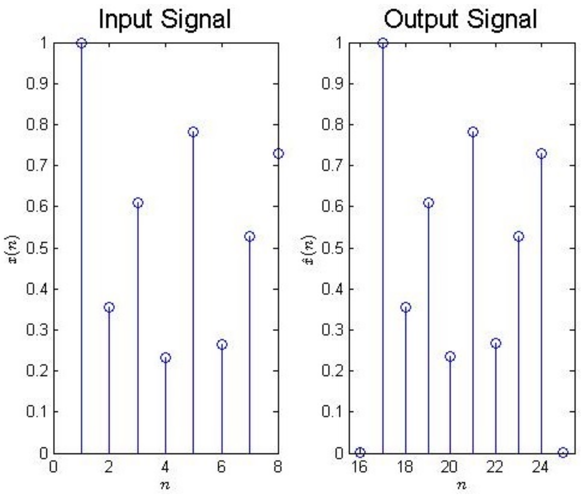


Figure 5.3: OFB input and output for the example 1.

# Chapter 6

# Oversampled Filter Banks for Coding

*In this chapter, a description of how the OFBs are used for coding and how the syndrome can be used for error correction, are presented. Also the procedure to design the prototype filter and the parity-check polynomial matrix is described.*

In an OFB the subband signals are redundant. They contain more samples (per unit time) than the input signal [16]. This redundancy can be used as a coding process for the input signal, where the $N$ subbands and the decimator factor $M < N$ provides an $M/N < 1$ code rate, ensuring the robustness to avoid the noise interference error [9]. Now, we need to define the process to generate the signal syndrome.

## 6.1 Parity-Check Polinomials

A parity-check polynomial matrix $\boldsymbol{C}^{PC}(z)$ is necessary in the receiver to calculate the $N - M$ syndrome signals from the subband signals $\hat{\boldsymbol{y}}(n)$ [12]. Since $\boldsymbol{C}^{PC}(z)$ satisfies:

$$\boldsymbol{C}^{PC}(z)\boldsymbol{H}_p(z) = \mathbf{0}_{(N-M)\times M}, \tag{6.1}$$

the syndrome signals are identical to zero if $\hat{\boldsymbol{y}}(n) = \boldsymbol{y}(n)$. To acquire an efficient implementation, we factorize the parity-check polynomial matrix as:

$$\boldsymbol{C}^{PC}(z) = z^{-v}\tilde{\boldsymbol{C}}(z)\boldsymbol{W}^H, \tag{6.2}$$

where $\tilde{\boldsymbol{C}}(z)$ is a $(N - M) \times N$ polynomial matrix that, for a given parity-check polynomial matrix $\boldsymbol{C}^{PC}(z)$, can be calculated as $\tilde{\boldsymbol{C}}(z) = z^v\boldsymbol{C}^{PC}(z)\boldsymbol{W}$.

Expressing $\boldsymbol{H}_p(z)$ according to (5.2) and $\boldsymbol{C}^{PC}(z)$ according to (6.2), the expression (6.1) simplifies to:

$$\tilde{\boldsymbol{C}}(z)\boldsymbol{E}(z) = \mathbf{0}_{(N-M)\times M}. \tag{6.3}$$

The sparseness of $\boldsymbol{E}(z)$ allows us to express (6.3) into $b$ parity-check polynomials matrices $\tilde{\boldsymbol{C}}_l(z)$ of size $(N-M)/b \times J$:

$$\tilde{\boldsymbol{C}}_l(z)\boldsymbol{E}_l(z) = \mathbf{0}_{(N-M)\times M/b}, \tag{6.4}$$
$$\text{with } [\tilde{\boldsymbol{C}}_l(z)]_{i,j} = [\tilde{\boldsymbol{C}}(z)]_{l+ib,l+jb},$$
$$0 \le l < b, \qquad 0 \le i < (N-M)/b, \qquad 0 \le j < J.$$

As $\boldsymbol{E}_l(z)$ have delayed $J$-fold upsampled polyphase components of the prototype filter, the same applies to $\tilde{\boldsymbol{C}}_l(z)$ to fulfill (6.4). A downsampled version of (6.4) is defined by expressing $\tilde{\boldsymbol{C}}_l(z)$ as a product of $(N-M)/b \times (N-M)/b$ paraunitary diagonal matrix $\tilde{\boldsymbol{\Lambda}}_3(z)$, a $(N-M)/b \times J$ matrix $\tilde{\boldsymbol{C}}_{l\downarrow}(z^J)$ that contains polynomials in $z^J$, and $\tilde{\boldsymbol{\Lambda}}_1(z)$, which is the para-conjugate of the $J \times J$ matrix $\boldsymbol{\Lambda}_1(z)$ in (5.9):

$$\tilde{\boldsymbol{C}}_l(z) = \tilde{\boldsymbol{\Lambda}}_3(z)\tilde{\boldsymbol{C}}_{l\downarrow}(z^J)\tilde{\boldsymbol{\Lambda}}_1(z), \qquad l = 0, ..., b-1. \tag{6.5}$$

Applying (5.9), (6.5) to (6.4), the downsampled version of (6.4) is obtained:

$$\tilde{\boldsymbol{C}}_{l\downarrow}(z)\boldsymbol{E}_{l\downarrow}(z) = \mathbf{0}_{(N-M)/b\times M/b}. \tag{6.6}$$

## 6.2 Syndrome

From (6.3) we realize that the analysis polyphase matrix $\boldsymbol{E}(z)$, when multiplied with the parity check matrix $\tilde{\boldsymbol{C}}(z)$, returns the zero matrix. Then, the syndrome is the output signal of the $\tilde{\boldsymbol{C}}(z)$ matrix when is fed with the $\boldsymbol{V}(z)$ signals as seen in Figure 6.1, and described in:

$$\boldsymbol{S}(z) = \tilde{\boldsymbol{C}}(z)\hat{\boldsymbol{V}}(z). \tag{6.7}$$

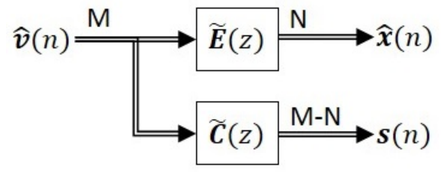


Figure 6.1: Syndrome Generation [17].

When the subband signals are corrupted by the channel noise, these can be written as $\hat{\boldsymbol{Y}}(z) = \boldsymbol{Y}(z) + \boldsymbol{Q}(z)$, where $\boldsymbol{Q}(z)$ is the z-transform of the noise. From Figure 5.1, $\hat{\boldsymbol{V}}(z)$ can be written as:

$$\begin{aligned}
\hat{\boldsymbol{V}}(z) &= \boldsymbol{W}\hat{\boldsymbol{Y}}(z) && (6.8)\\
&= \boldsymbol{W}[\boldsymbol{Y}(z) + \boldsymbol{Q}(z)] && (6.9)\\
&= \boldsymbol{W}\boldsymbol{W}^H\boldsymbol{E}(z)\boldsymbol{X}(z) + \boldsymbol{W}\boldsymbol{Q}(z) && (6.10)\\
&= \boldsymbol{E}(z)\boldsymbol{X}(z) + \boldsymbol{W}\boldsymbol{Q}(z) && (6.11)
\end{aligned}$$

and if we substitute this on (6.7) and with (6.3), we have:

$$\begin{aligned}\boldsymbol{S}(z) &= \tilde{\boldsymbol{C}}(z)\boldsymbol{E}(z)\boldsymbol{X}(z) + \tilde{\boldsymbol{C}}(z)\boldsymbol{W}\boldsymbol{Q}(z) && (6.12)\\ &= \tilde{\boldsymbol{C}}(z)\boldsymbol{W}\boldsymbol{Q}(z). && (6.13)\end{aligned}$$

If there is noise in the subband signals, the syndrome will be non-zero, and this information can be used to detect and correct the subband signals [17].

We can split the calculation of the $N-M$ syndromes into parallel calculations of $b$ parity-check polynomial matrices $\tilde{\boldsymbol{C}}_l(z)$, of size $(N-M)/b \times J$. Each calculation takes the vectors $\hat{\boldsymbol{v}}_l(n) = [\hat{v}_l(n), \hat{v}_{l+b}(n), ..., \hat{v}_{l+(J-1)b}(n)]^T$, which are subsets of the $\hat{\boldsymbol{v}}(n)$ signals, as inputs to calculate the syndrome vectors $\boldsymbol{s}_l(n) = [s_l(n), s_{l+b}(n), ..., s_{l+N-M-b}(n)]^T$, $l = 0, ..., b-1$ [12].

## 6.3 Joint Prototype Filter and Paraunitary Parity-Check Polynomial Matrix Design

The paraunitary parity-check polynomial matrices can be calculated into the prototype filter design procedure. This can be accomplished by optimizing the prototype filter with a cost function [12]. The applied cost function minimized the stopband energy. Also, $[\boldsymbol{E}_{l\downarrow}(z)\boldsymbol{C}_{l\downarrow}(z)]$ can be factored into paraunitary building blocks:

$$\begin{aligned}[\boldsymbol{E}_{l\downarrow}(z)\boldsymbol{C}_{l\downarrow}(z)] &= \prod_{i=1}^{L_e-1} \boldsymbol{V}_{l,i}(z) \prod_{j=1}^{J-1} \boldsymbol{U}_{l,j}, && (6.14)\\ \boldsymbol{V}_{l,i}(z) &= \boldsymbol{I}_J - \boldsymbol{v}_{l,i}\boldsymbol{v}_{l,i}^T + z^{-1}\boldsymbol{v}_{l,i}\boldsymbol{v}_{l,i}^T,\\ \boldsymbol{U}_{l,j} &= \boldsymbol{I}_J - 2\boldsymbol{u}_{l,j}\boldsymbol{v}_{l,j}^T,\end{aligned}$$

where $L_e = \lceil L/K \rceil$, and perform an unconstrained optimization of $\boldsymbol{v}_{l,i}$ and $\boldsymbol{u}_{l,j}$ according to our cost function.

**Example 2**

In this simulation we generate an OFB with $N = 8$ subbands, a decimator factor of $M = 6$ and a filter length of $L = 60$. Then the greatest common divider is $g = 2$, the least common multiple is $K = 24$, $J = 4$. The prototype FIR filter $p(n)$, is designed using (6.14) and is modulated using (5.1). Figure 6.2 shows us the frequency responses of the designed filters. When a random signal goes through the filter bank, without channel noise, we can compare the output signal with an input signal in Figure 6.3. We can observe the PR property and the output signal delay.

## 6.4 Signature Waveforms

When an impulse error occurs in one of the channels of the system, multiple non-zero taps are generated in the $N-M$ syndromes. Depending on the channel

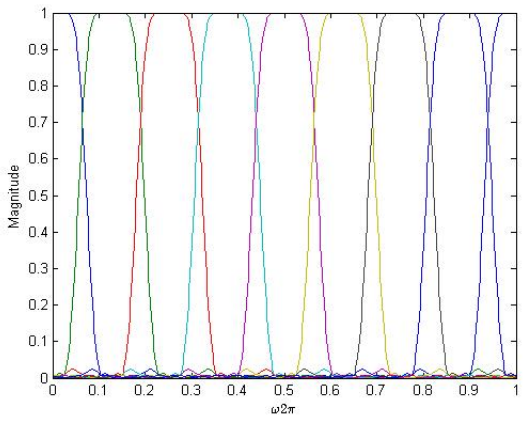


Figure 6.2: Frequency response for the filters in the example 2.

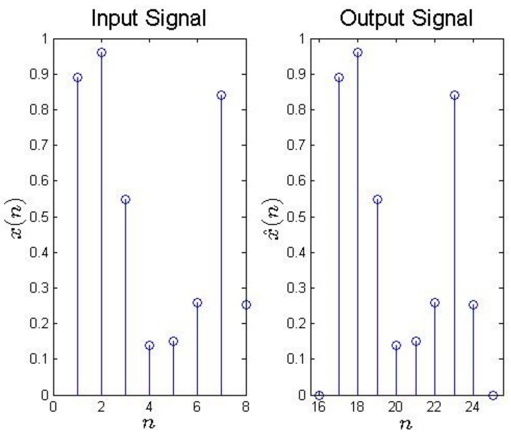


Figure 6.3: OFB input and output for the example 2.

where the impulse error occurred, the shape of the syndrome waveforms are unique. This shapes are called *signature waveforms* [17]. This information in the syndrome allows us to:

- identify the channel in error, by comparing the output syndrome with the signature waveforms.
- detect the time when the error occurs, by comparing the coefficients with certain threshold.
- calculate an approximation of the amplitude by adding the magnitude of the entire syndrome coefficient.

With this data we can correct the impulse error.

**Example 3**

With the parity-check polynomial matrices designed in Simulation 2, we obtain the signature waveforms depicted in Figure 6.4. After generating an impulse error of an amplitude equal to 2.73, in the channel 1, at time index 18, we get the syndrome depicted in Figure 6.5. We can appreciate that the shape of this syndrome is like the channel 1 signature waveform showed in Figure 6.4. When we add the magnitude of the coefficients of one of the syndromes we get the value of 2.5931, which is an approximation to the actual magnitude of the error. Also if we check the magnitude of the syndrome coefficients in Figure 6.6 we see that the first non-zero value begins at time 18, which is the time where the impulse error occurred.

## 6.5 Filter Banks and BCH Coding

The author in [17] reports a simulation where the performance of a subband coding using BCH code (a cyclic coding type) is analyzed to correct the additive noise in the channel. The signal is splitted into subbands by a filter bank. Each of the subbands is quantized by a variable number of bits defined by the next equation [8]:

$$b_k = b + \frac{1}{2}\log_2\left[\frac{\sigma_{xk}^2}{\left(\prod_{i=0}^{N-1}\sigma_{xi}^2\right)^{1/N}}\right], \tag{6.15}$$

where $b_k$ is the bit allocation in the $k$ subband, $b$ is the average bit allocation and $\sigma_{xk}^2$ is the variance of the signal in the $k$ subband.

After the quantization, the subband signals define a bit stream that is encoded with the BCH code and transmitted to the channel. In the receiver side, the inverse operations are performed.

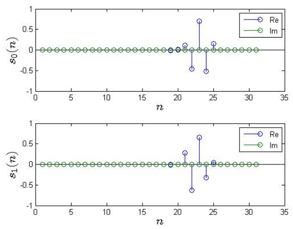
The signal in the channel is corrupted by two independent noise sources:

1. Noise by quantization.
2. A Bernoulli-Gaussian impulse noise [17] modeled as:

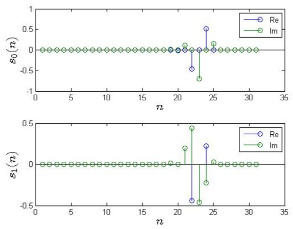
   $$n_i(n) = \xi(n)b'(n), \tag{6.16}$$

   where $\xi(n)$ stands for a Bernoulli process, and i.i.d. sequence of "0" and "1" with $\text{Prob}(\xi(n) = 1) = p$, and $b'(n)$ represents a Gaussian noise with zero mean and variance $\sigma_g^2 = 0.001$, such that $\sigma_i^2 \gg \sigma_g^2$ [18].

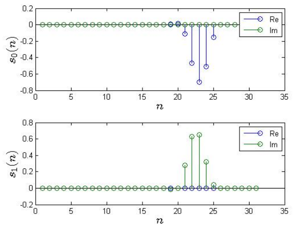
The results reported in [17] show the performance of the BCH subband coder. The results are depicted in Figure 6.7 and 6.8. Additionally of using the BCH decoder, they report the performance of the impulse error correction with OFB, based on the syndrome correction and the signature waveforms theory, explained in the last section. The correction capacity of the coding systems simulated decreases as $P_i$ increases. The syndrome based decoder cannot detect



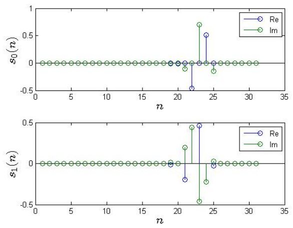
(a) Channel 0



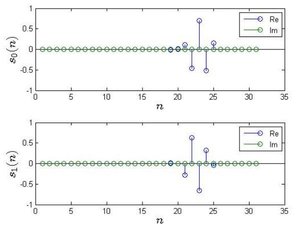
(b) Channel 1



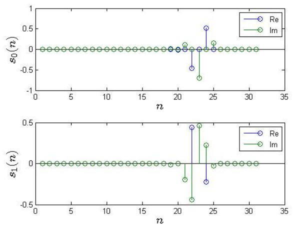
(c) Channel 2



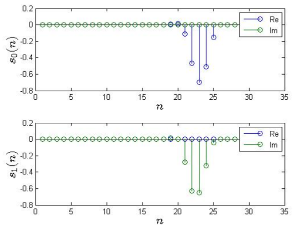
(d) Channel 3



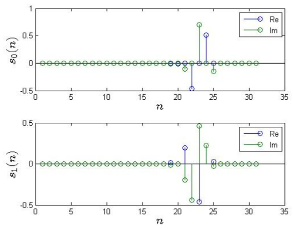
(e) Channel 4



(f) Channel 5



(g) Channel 6



(h) Channel 7

Figure 6.4: Signature Waveforms.

and correct different impulse errors that occur very closely one next to the other one. Even so, the syndrome based decoder has a better performance than the BCH decoder.

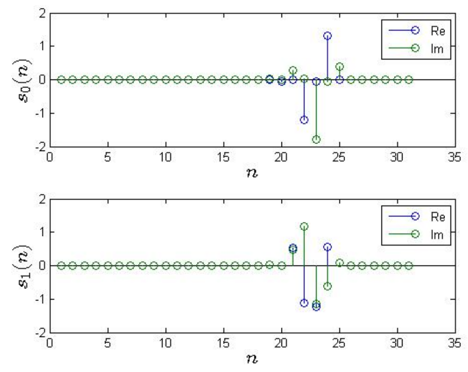
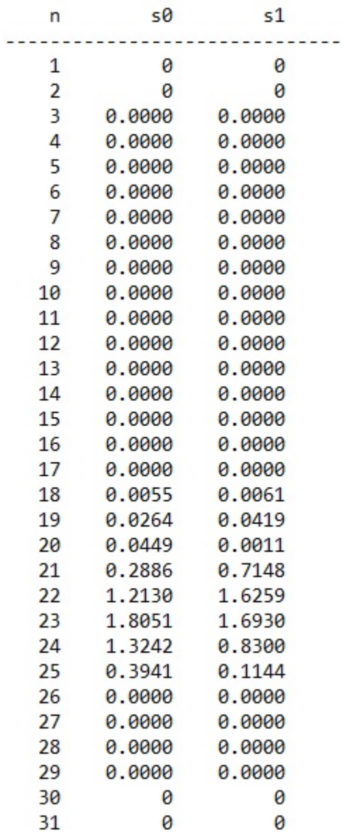


Figure 6.5: Syndrome generated in the example 3.



```
     n        s0         s1
------------------------------
     1         0          0
     2         0          0
     3    0.0000     0.0000
     4    0.0000     0.0000
     5    0.0000     0.0000
     6    0.0000     0.0000
     7    0.0000     0.0000
     8    0.0000     0.0000
     9    0.0000     0.0000
    10    0.0000     0.0000
    11    0.0000     0.0000
    12    0.0000     0.0000
    13    0.0000     0.0000
    14    0.0000     0.0000
    15    0.0000     0.0000
    16    0.0000     0.0000
    17    0.0000     0.0000
    18    0.0055     0.0061
    19    0.0264     0.0419
    20    0.0449     0.0011
    21    0.2886     0.7148
    22    1.2130     1.6259
    23    1.8051     1.6930
    24    1.3242     0.8300
    25    0.3941     0.1144
    26    0.0000     0.0000
    27    0.0000     0.0000
    28    0.0000     0.0000
    29    0.0000     0.0000
    30         0          0
    31         0          0
```

Figure 6.6: Coefficients of the syndrome in Example 3. The first column is the time index, the second and third columns are the syndrome 0 and syndrome 1 magnitude, respectively.

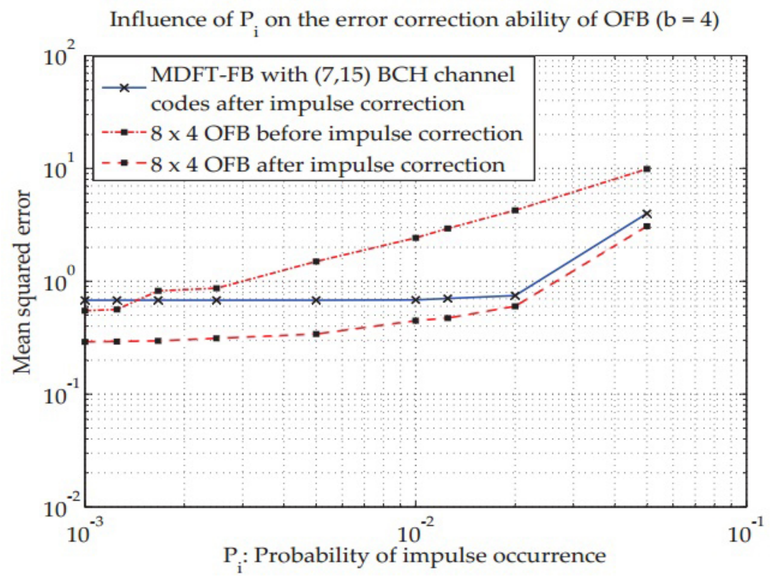


Figure 6.7: Influence of $P_i$ on the error correction ability of OFB under coarser quantization ($b = 4$) [17].

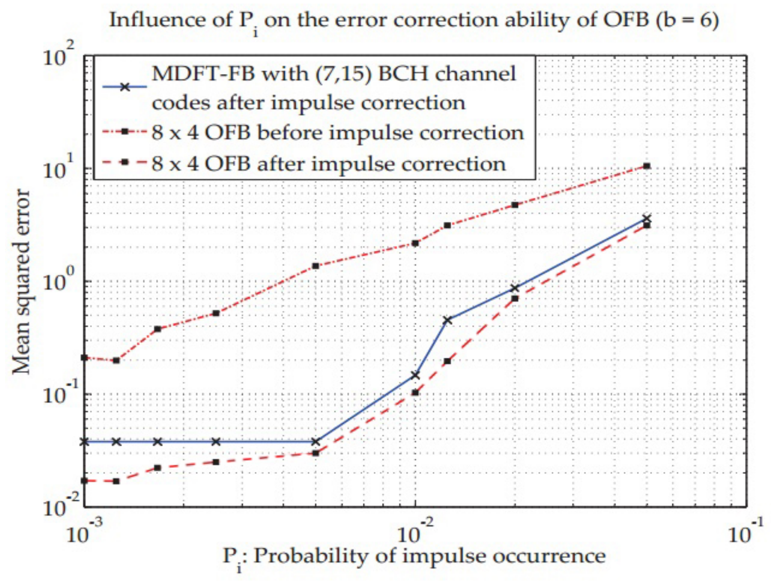


Figure 6.8: Influence of $P_i$ on the error correction ability of OFB under finer quantization ($b = 6$) [17].

# Chapter 7

# Filter Banks for LDPC Coding

*In this chapter a communication model for error correcting coding using OFB and LDPC coding is proposed.*

An OFB has shown to be usefull for coding. And the LDPC coding techniques has demonstrated that its performance approaches to the Shannon limit [19], but its decoding may be very complex [20]. A communication model using this techniques is proposed to decrease the error probability for a channel with Additive Gaussian White Noise (AWGN) and impulse noise.

The model transmitter splits the input signal into subbands with an analysis filter bank. Then the signal in each subband is quantized with an optimum bit allocation process using (6.15). The next step is defining a bit stream with the quantized data. The output bit stream may have different dimensions according to the different optimum bit allocation values assigned to each subband. The bit stream is encoded with the LDPC generation matrix defined with the parity-check matrix from [21]. After that, each binary “0” is coded as “1”, and each “1” is coded as “-1”. Then the signal is delivered to the noisy channel. The transmitter model is shown in Figure 7.1.

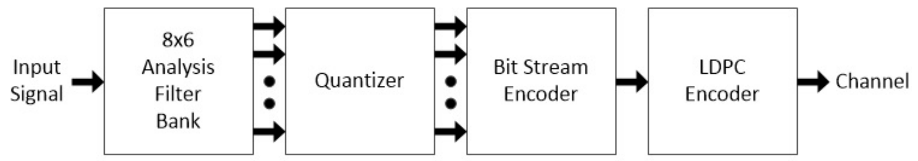


Figure 7.1: Proposed model transmitter.

In the receiver, the inverse process is performed. The corrupted signal is corrected with the LDPC decoder from [22]. Then, the bit stream is decoded and the data is delivered to each corresponding subband. After that, the inverse process of quantization is applied to the bits. Finally, the reconstruction of the message is performed with the synthesis filter bank. The receiver model is shown in Figure 7.2.

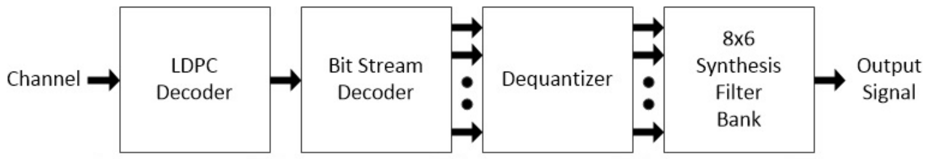


Figure 7.2: Proposed model receiver.

## Simulation Results

The model simulation used the OFB previously designed in Section 6.3, along with a (273,3,10) LDPC parity check matrix with 0.7 code rate obtained from [21]. The channel model was the same used on [17] described in Section 6.5, and also an extra source of AWGN with zero mean and identical $\sigma_w^2 = 0.001$ in all subbands is included. Different average bit allocation was used for the quantization (using 4, 6 and 7 bits). The results are depicted on figure 7.3.

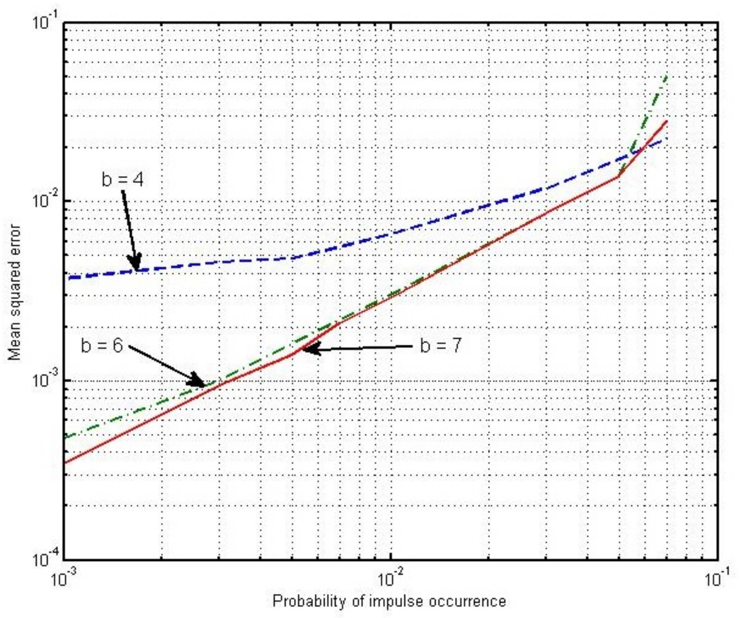


Figure 7.3: Proposed model simulation results.

Figure 7.3 shows a reduction on the Mean Square Error (MSE) from [17], depicted on Figures 6.7 and 6.8, where the worst MSE is above $10^{-2}$. The case, where $b = 4$, shows that in the proposed model worst case, in the higher probability of impulse occurrence region, is better than in [17] best case. Also, as the probability of impulse occurrence decreases, the proposed model has better correction of the errors, but the results of [17] show the same MSE from certain probability of impulse occurrence thresholds. This behavior is also presented when $b = 6$. For each case, the proposal method shows a decrease on the MSE for at least one magnitude order. The 7 bits quantization result is similar to the 6 bits quantization performance. In the high impulse occurrence probability region, the LDPC decoding cannot work correctly.

# Chapter 8

# Conclusions

It was proposed the use of DFT OFB for LDPC coding with the goal to decrease the coding complexity. This is achieved by splitting the input signal into subbands using an analysis filter bank. The simulation results are presented with the graphics of probability of error versus the probability of impulse occurrence. From the graphic it can be seen the benefits of the proposed coding in terms of the probability of error. The comparisons of the proposed coding method with method [17] show that the proposed coding presents better results regarding the probability of error.

The WD Section, studied the performance of the WD on LDPC coding. This had the objective of decrease the complexity of the LDPC decoder. The WD consists in decoding a window of the codeword. When this decoding is finished, the window slides one section and attempts to decode the next window configuration. The simulations graphics show that the WD decoder and the block decoder have similar behaviour, but the WD can decode a part of the codeword in a lower latency than the block decoder.

## Future Work

- Investigate the OFB and LDPC coding technique coding using the window decoder.
- Investigate the OFB and LDPC coding technique using different code rates.
- Investigate the OFB and LDPC coding technique with the OFB impulse syndrome correction algorithm.
- Implement on FPGA the most promising coding technique.

# Bibliography

[1] S. M. Moser and P.-N. Chen, *A Students Guide to Coding and Information Theory.* Cambridge University Press, 2012.

[2] S. Haykin, *Communication Systems*, 4th ed. John Wiley & Sons, Inc., 2001.

[3] K. Deergha-Rao, *Channel Coding Techniques for Wireless Communications.* Springer, 2015.

[4] H. Esfahanizadeh, A. Hareedy, and L. Dolecek, "Spatially coupled codes optimized for magnetic recordings applications," *IEEE Trans. Mag.*, February 2017.

[5] H. Esfahanizadeh, A. Hareedy, and L. Dolecek, "A novel combinatorial framework to construct spatially-coupled codes: Minimum overlap partitioning," *Proc. IEEE International Symposium on Information Theory (ISIT)*, June 2017.

[6] A. R. Iyengar, P. H. Siegel, R. L. Urbanke, and J. K. Wolf, "Windowed decoding of spatially coupled codes," *IEEE Trans. IT*, April 2013.

[7] G. Jovanovic-Dolecek, *Multirate Systems: Design and Applications.* Idea Group Publishing, 2002.

[8] P. P. Vaidyanathan, *Multirate Systems and Filter Banks.* Englewood Cliffs, NJ; Prentice-Hall, 1993.

[9] S. Weiss, S. Redif, T. Cooper, C. Liu, P. D. Baxter, and J. G. McWhirter, "Paraunitary oversampled filter bank design for channel coding," *EURASIP Journal on Advances in Signal Processing*, December 2006.

[10] S. Weiss, "On the design of oversampled filter banks for channel coding," *12th European Signal Processing Conference*, September 2004.

[11] T. Karp, M. Kieffer, and P. Duhamel, "Oversampled dft filter banks for error correction coding," *16th European Signal Processing conference*, August 2008.

[12] T. Karp, M. Kieffer, and P. Duhamel, “Parity-check matrix calculation for paraunitary oversampled DFT filter banks,” *IEEE Transactions on Signal Processing*, vol. 56, no. 10, October 2008.

[13] H. Bölcskei and F. Hlawatsch, “Oversampled cosine modulated filter banks with perfect reconstruction,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, pp. 1059–1071, August 1998.

[14] Z. Cvetkovic and M. Vetterli, “Oversampled filter banks,” *IEEE Transactions on Signal Processing*, vol. 46, no. 5, May 1998.

[15] K. Eneman and M. Moonen, “Dft modulated filter bank design for oversampled subband systems,” *Signal Processing*, vol. 81, no. 9, pp. 1947–1973, 2001.

[16] H. Bölcskei, F. Hlawatsch, and H. G. Feichtinger, “Frame-theoretic analysis of oversampled filter banks,” *IEEE Transactions on Signal Processing*, vol. 46, no. 12, December 1998.

[17] A. S. Lakshmi, “Oversampled filter banks with built-in error correction capabilities,” Master’s thesis, Texas Tech University, 2012.

[18] F. Labeau, J.-C. Chiang, M. Kieffer, P. Duhamel, L. Vandendorpe, and B. Macq, “Oversampled filter banks as error correcting codes: Theory and impulse noise correction,” *IEEE Transactions on Signal Processing*, vol. 53, no. 12, December 2005.

[19] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, Febraury 2001.

[20] M. K. Roberts and M. Falaq, “A low-complex min-sum decoding algorithm for irregular ldpc codes,” *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, September 2016.

[21] D. J. C. MacKay, “Encyclopedia of sparse graph codes,” (accessed September, 2017). [Online]. Available: http://www.inference.org.uk/mackay/codes/data.html

[22] A. B. J. Zhou, “Message passing algorithm for decoding binary LDPC codes,” (accessed September, 2017). [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/47383-message-passing-algorithm-for-decoding-binary-ldpc-codes