POLITECNICO DI TORINO

Master's Degree in Communications and Computer Networks Engineering

Master's Degree Thesis

Soft Decisions Multistage Decoding for Digital Radio Mondiale

Supervisors

Dr. Roberto Garello

Dr. Özgün Paker

Arie Koppelaar

Candidate

Antonio Melone

October 2020

Summary

Digital Radio Mondiale (DRM) is a digital radio broadcasting system designed as an high quality digital replacement for current analogue radio broadcasting in the AM and FM bands. In the last years it has being introduced and used in several countries, due to its advantages with respect to the analogue system. In a DRM receiver the transmission channels are received under different reception conditions. These channels are multipath channels and so they might suffer of fading. Therefore in this thesis an improved channel decoding strategy for DRM is investigated. A multilevel coding structure at the transmitter side, combined with a modified multistage decoder has been designed to improve the decoding performances in terms of bit error rate (BER). The novelty regards the exchange of soft information between the stages, provided as output from the FEC (forward error correction) decoder, by the usage of the Max-Log-Map algorithm.
In traditional DRM receivers an hard-output forward error correction decoding algorithm is adopted, where hard information is exchanged between the stages in a MSD. The hard-information regards the decoded bits in the current level. A correct decision will improve the decoding of the next level. A decoding error will not help the next level, instead it will deteriorate the demodulation and the decoding performance. By providing decoder decisions with reliability information (soft information), the demodulation and the decoding of the next level can accordingly weigh the information from a lower level and, in case of low reliability, limits the influence of this helper information.
The Max-Log-MAP algorithm was used to produce soft-output information in the form of log-likelihood ratios, used to improve the Soft Demapper in producing soft-decision information for other levels in the multistage decoder. The simulations of the modified multistage are performed. The performance of the Max-Log-MAP are improved by scaling information. Signal-to-noise ratio improvements for all the DRM channel conditions are obtained.
However these improvements obtained with the soft-output multistage decoder have to be placed in context with the required complexity increase to realize this improvement.

Acknowledgements

This document is the Master thesis of my graduation project for the M.Sc degree with a specialization in Communications and Computer Networks Engineering at the Politecnico di Torino. This thesis is based on a research project carried out at NXP Semiconductor in High Tech Campus, Eindhoven, where I spent 10 months of internship. The project started in September 2019 and ended in June 2020.

I would like to thank all the people who have supported me during these university years. I wish to thank my university supervisor Professor Roberto Garello, who have followed and helped me troughout this Master thesis.
I would especially acknowledge my company supervisors Arie Koppelaar and Dr. Özgün Paker for their patience, guidance, support and for giving me the chance to carry out this project at the NXP Semiconductor company. I would like also to thank Dr. Wim van Houtum, TUe professor, who kindly helped me in this project with his suggestions and knowledge.
Finally, I greatly appreciate the support from my parents, without whom nothing would have been possible.

Table of Contents

List of Tables

List of Figures

Acronyms

AWGN
Additive White Gaussian Noise

BCJR
named after its authors, [1]

COFDM
Coded Orthogonal Frequency-Division Multiplexing

DRM
Digital Radio Mondiale

MSD
Multistage decoding

MLC
Multilevel coding

MRR
Minimum receiver requirements

SM QAM
Standard Mapping Quadrature Amplitude Modulation

SNR
Signal-to-noise ratio

Chapter 1

Introduction

Digital Radio Mondiale is a digital radio broadcasting system designed to operate at AM and FM Bands. It is introduced to replace the analog broadcasting and, by using digital encoding, modulation and decoding, it provides better sound quality and robustness against fading and interference. Due to its features, it is used in several countries, mostly in India, but also in UK, France, Romania, Kuwait, China and New Zealand.
In this thesis an improved channel decoding strategy has been investigated for DRM. In particular, by using a multilevel coding structure at the transmitter side, combined with a modified multistage decoder, we try to improve the decoding performances in terms of BER. The novelty is the introduction of the exchange of soft information between the stages, provided as output from the FEC (forward error correction) decoder, by the usage of the Max-Log-Map algorithm. In traditional DRM receivers an hard-output forward error correction decoding algorithm is adopted, where hard information is exchanged between the stages in a MSD. To isolate the impact of the decoding approach, channel estimation and synchronization is left out in the analysis, and an ideal channel estimation and synchronization is considered.

A brief description of the organization of the thesis follows. Chapter 1 mentions the reasons behind the introduction of DRM as well as the purpose of the thesis. Chapter 2 gives an overview of the DRM standard, describing its features. In the theoretical background Chapter, theoretical concepts are described to understand the content of this work, including the structure of the encoder and decoder. The problem statement is explained in Chapter 4. Chapter 5 gives a description of the adopted decoder, the core of this work. Finally, simulation results and conclusions, including recommendations for future works, are provided respectively in Chapter 6 and 7.

Chapter 2

DRM overview

DRM has been initially designed to operate at AM frequencies, below 30 MHz, and it has been named the DRM 30 standard [2]. Later on the DRM+ standard has been standardized to be able to work on higher frequencies, between 30MHz and 174MHz. Figure 2.1 shows the DRM spectrum.

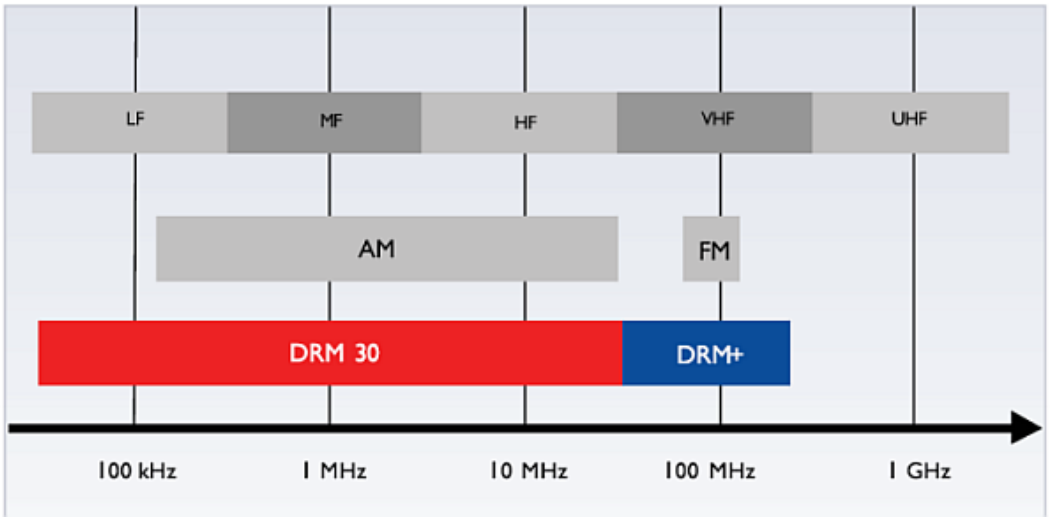

Figure 2.1: DRM frequencies [2].

In this project we work on LF, MF and HF frequencies, therefore we will always refer to DRM 30. COFDM is used, with a signal bandwidth of 9-10 kHz , but also half of this bandwidth (4.5-5 kHz) is used for simulcast transmission with AM signals or twice this bandwidth (18-20 kHz) for larger transmission capacity.

2.1 Channel profiles

LF, MF and HF broadcast radio transmission channels are multipath channels. This is due to the fact that the surface of the earth and the ionosphere are involved in electromagnetic wave propagation. Multipath channels can create fading, a significant variation in the received signal amplitude and phase over time or space. Therefore, in order to benchmark performances of DRM receivers, in the DRM standard [3], several channel models are defined, which reflect into some degree the reception conditions at the different carrier frequencies. This models are based on the following equation:

$$s(t) = \sum_{k=1}^{n} \rho_k c_k(t) e(t - \Delta_k)$$

where

- s(t) is the complex envelope of the output signal.
- ρ_k is the attenuation of the k-th path.
- e(t) is the complex envelope of the input signal.
- Δ_k is the relative delay of the k-th path.
- c_k(t) is a time-variant tap weight. It's a zero mean complex-valued Gaussian random process with Raylegh-distribuited magnitude and uniform phase.

Therefore the model consists of several stochastic processes, characterized by their variance and their power density spectrum (PDS). The relative attenuation of the path ρ_k defines the variance, which is a measure for the average signal power. The PDS defines the average speed of the variation of the path in time. The width of the PDS refers to the Doppler spread of the path. Then, whenever the centre of the PDS is not in zero, we refers to the Doppler shift .A Doppler shift occurs when the transmitter of a signal is moving in relation to the receiver. In this case the receiver perceives a frequency different from the one emitted by the transmitter. The Doppler spread refers to a difference among the Doppler shifts in signals coming from the same transmitter and using different paths. The value of these parameters is shown in Table 2.1 for each path of each channel profile.

Channel Profile	path	path n.	delay [ms]	gain	Doppler shift [Hz]	Doppler spread [Hz]
1	1	1	0	1	0	0
2	2	1	0	1	0	0
		2	1	0.5	0	0.1
3	4	1	0	1	0.1	0.1
		2	0.7	0.7	0.2	0.5
		3	1.5	0.5	0.5	1.0
		4	2.2	0.25	1.0	2.0
4	2	1	0	1	0	1
		2	2	1	0	1
5	2	1	0	1	0	2
		2	4	1	0	2
6	4	1	0	0.5	0	0.1
		2	2	1	1.2	2.4
		3	4	0.25	2.4	4.8
		4	6	0.062	3.6	7.2

Table 2.1: Channel profiles characteristics [3].

2.2 OFDM symbol parameters

In the DRM standard [3], transmission efficiency parameters are also defined to find a trade-off between capacity and rigidity to noise, multipath and Doppler. In this section, an OFDM symbol parameter overview is pointed out. In the DRM standard Robustness Modes are defined to ease the selection of transmission parameters for some typical propagation conditions. Table 2.1 associates the Robustness Modes and the typical channel properties.

Robustness mode	Typical propagation conditions
A	Gaussian channels, with minor fading
B	Time and frequency selective channels, with longer delay spread
C	As robustness mode B, but with higher Doppler spread
D	As robustness mode B, but with severe delay and Doppler spread
E	Time and frequency selective channels

Table 2.2: Robustness mode and corresponding propagation condition [3].

The signal transmitted is organized in frames. Each frame contains a number of symbols N_s. Table 2.2 defines the time-related parameters of the Robustness modes

for an OFDM transmitted symbol.

Robustness mode	Duration T_u	Carrier spacing $1/T_u$	Duration of guard interval T_g	Duration of symbol $T_s = T_u + T_g$	T_g/T_u	Number of symbols per frame N_s
A	24 ms	$41^{2/3}$ Hz	2,66 ms	26,66 ms	1/9	15
B	21,33 ms	$46^{7/8}$ Hz	5,33 ms	26,66 ms	1/4	15
C	14,66 ms	$68^{2/11}$ Hz	5,33 ms	20 ms	4/11	20
D	9,33 ms	$107^{1/7}$ Hz	7,33 ms	16,66 ms	11/14	24
E	2,25 ms	$444^{4/9}$ Hz	0,25 ms	2,5 ms	1/9	40

Table 2.3: time-related robustness mode parameters [3].

2.3 Channel Encoder

The transmitted data in DRM consist of different types contained in three different channels :

- Main Service Channel (MSC): transports encoded audio and broadcast data services.
- Fast Access Channel (FAC): provides information about

  channel parameter as well as a basic service selection information for fast scanning.
- Service Description Channel (SDC): provides audio and data coding parameters, current time and date, and an alternative frequency signaling.

The focus has been put on the MSC channel, since the key performance indicator we are focusing on is defined on this channel. However the proposed decoding works also for FAC and SDC. The coding operations of the MSC data are illustrated in Figure 2.2. It shows a multilevel coding for a Standard Mapping 64-QAM. How the multi-level coding works is explained in Chapter 3.

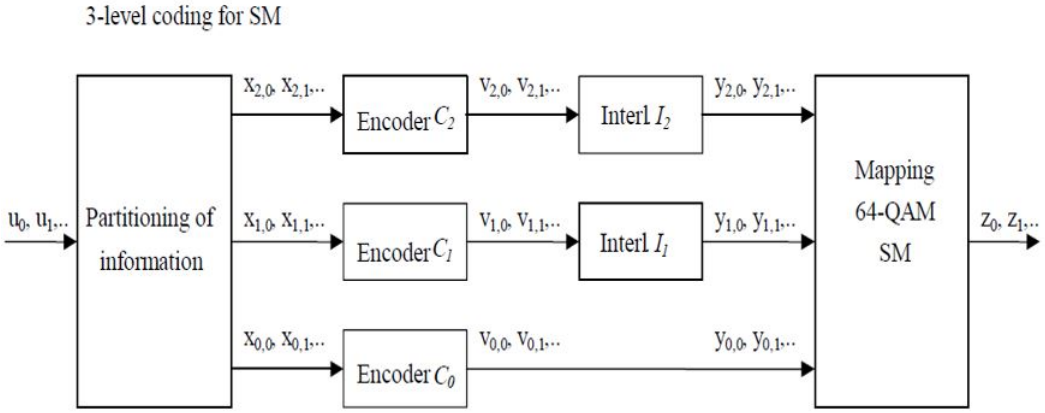

Figure 2.2: 3-level coding for SM 64-QAM [3].

The bit flow u_0, u_1 is partitioned in different levels and, for each level, the information is convolutionally encoded, adding redundant information, such that error correcting algorithms can reduce the number of bit errors that have been occurred during transmission. After that, puncturing is applied, adapting the code

rate according to the channel conditions. Next, the information is passed trough the interleaver, creating time-frequency diversity, such that the robustness will be improved in time-frequency dispersive channels.
The DRM standard allows the usage of three different constellations for data mapping: 64-QAM, 16-QAM and 4-QAM. In the analysed transmitted configuration the SM 64-QAM mapping is used for all the channel profiles except for channel 5, where SM 16-QAM is adopted. Figure 2.3 shows the SM 64-QAM, where $a = 1/\sqrt{42}$, while in Figure 2.4 the SM 16-QAM is depicted, where $a = 1/\sqrt{10}$.

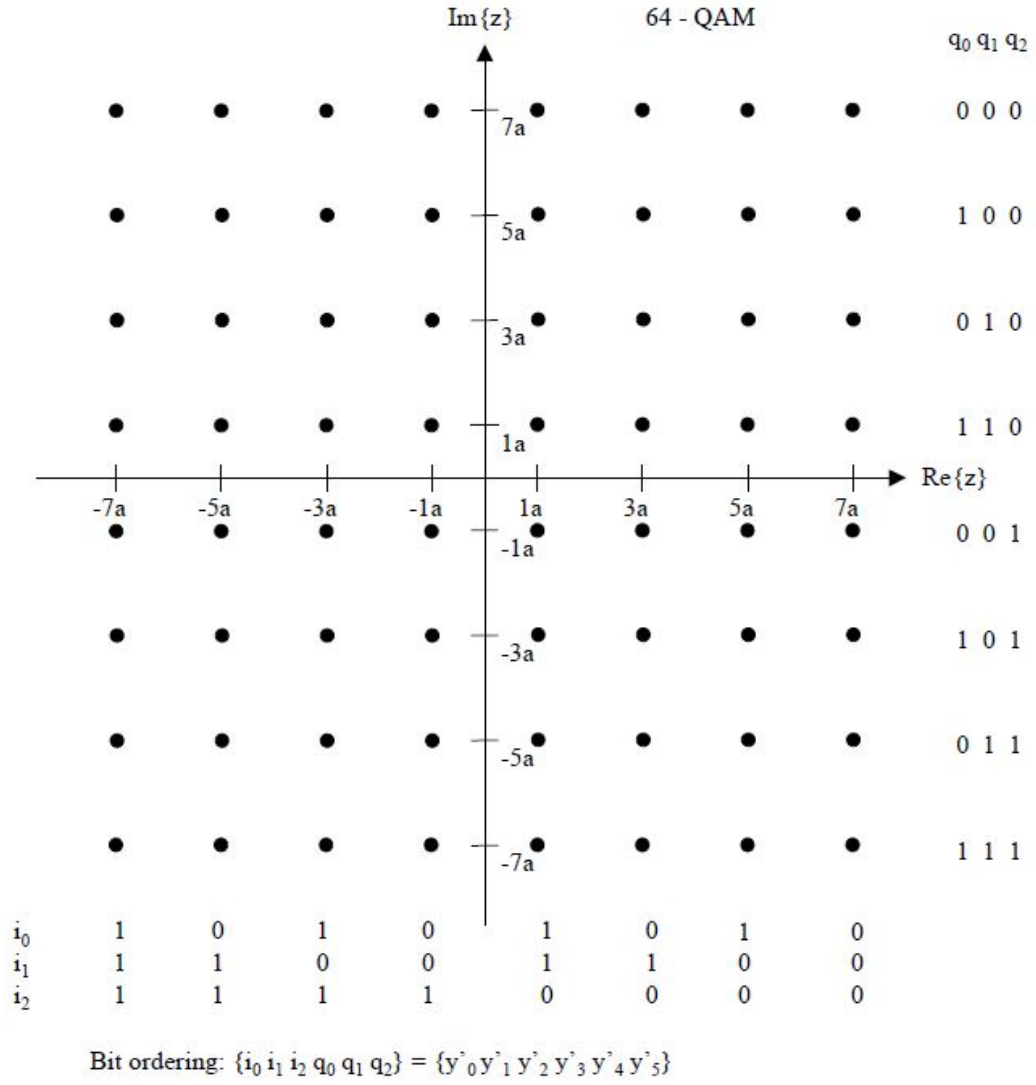

Figure 2.3: SM 64-QAM mapping [3].

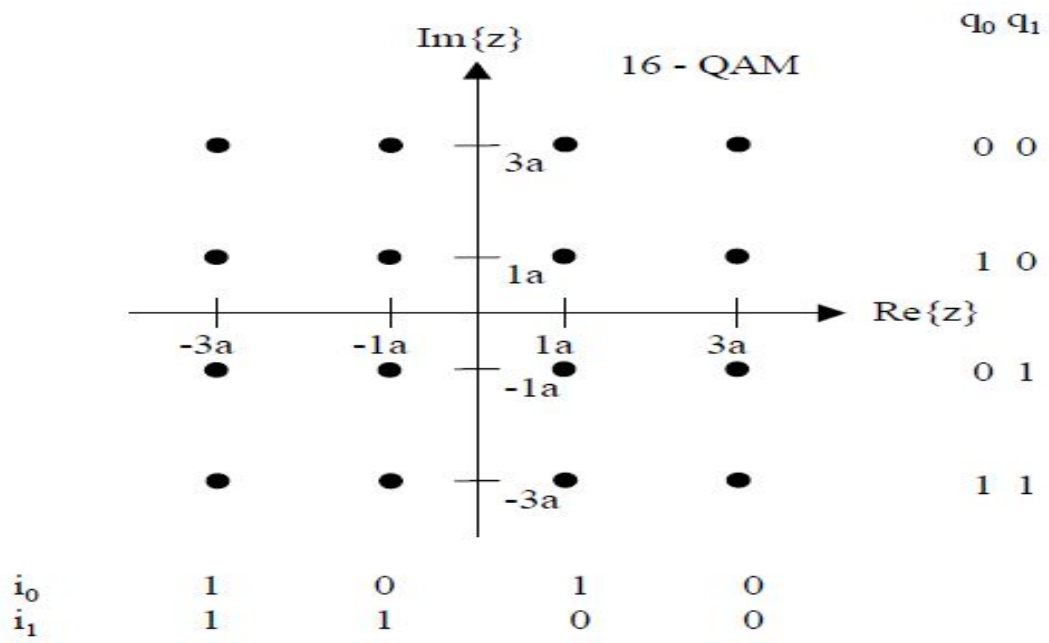

Figure 2.4: SM 16-QAM mapping [3].

2.4 Minimum Receiver Requirements

One of the strongest features of DRM is its robustness against various channel conditions. In this section minimum receiver requirements for BER performances for a proper DRM receiver are provided, used also as a benchmark to compare performances of different receiver implementations. The receiver performances are also dependent of the chosen configuration of several parameters. Table 2.4 shows the multiple combinations to measure the performances of the DRM receiver for LF, MF and HF.

Ch. Profile	Rob. Mode	QAM	Prot. level	R_{all}	R_0	R_1	R_2
1	A	64	3	0.78	2/3	4/5	8/9
2	A	64	2	0.71	1/2	3/4	7/8
3	B	64	2	0.71	1/2	3/4	7/8
4	B	64	1	0.6	1/3	2/3	4/5
5	C	16	1	0.62	1/2	3/4	NA
6	D	64	0	0.5	1/4	1/2	3/4

Table 2.4: Modulation and encoding parameters [3].

The DRM system has to meet the minimum requirements illustrated in Table 2.5, checking that the BER values are below 10^{-4} dB, value that gives a sufficiently high quality of service.

Ch. Profile	MRR Requirement	MRR Simulation
1	21	18.7
2	22	19.5
3	31	28.3
4	26	23.5
5	20	17.9
6	25	22.2

Table 2.5: MSC performance requirements [4].

The Standard gives also MRR Simulation performances obtained excluding receiver implementation losses [4].

Chapter 3

Theoretical background

In order to understand our improvements to the current architecture, it is important to put emphasis on multilevel coding and multistage decoding. In fact the current existing way of decoding in traditional DRM receivers is based on that. Therefore this Chapter shows how the multilevel encoder and the multistage decoder actually works. In particular the partitioning strategy of the information used in Figure 2.2 for the 64-QAM multilevel encoder is illustrated and how that is combined with a multistage decoder. Then, a section is reserved to the soft demapper, key operation for the understanding of the contribution of log-likelihood ratios as output of the decoding algorithm in the adopted FEC decoder.

3.1 Multilevel Coding

Multilevel coding is a method to jointly optimize coding and mapping proposed by Imai and Hikaragwa [5], in order to improve the performance of digital transmission. The idea is to maximize the minimum intra-subset Euclidean distance by using a set partitioning strategy. Figure 2.2 shows that for the 64-QAM modulation the information is partitioned in three levels and then each level is convolutionally encoded by a corresponding encoder. A short explanation about the partitioning strategy is provided here.
The approach used and proposed by Imai, is to protect the bit at level i of the bit labeling of a constellation point with a binary code C^i at level i, in such a way that the minimum distance of the Euclidean space code is maximized. Then, at the receiver side, each code C^i is decoded individually, by also using information derived from prior levels. This procedure, called multistage decoding, is illustrated in the MSD section. The address vector $x = \{x^0, x^1, ..., x^{l-1}\}$, where $x^i \in \{0,1\}$, will be mapped to a signal set $A = \{a_0, a_1, ..., a_{M-1}\}$ of a M-ary ASK modulation, that represents the in-phase or the quadrature component in a $M^2 - QAM$ scheme.

This mapping is obtained by subsequently partitioning the constellation A into subsets. Figure 3.1 shows a binary partitioning for an 8-ASK signal set.

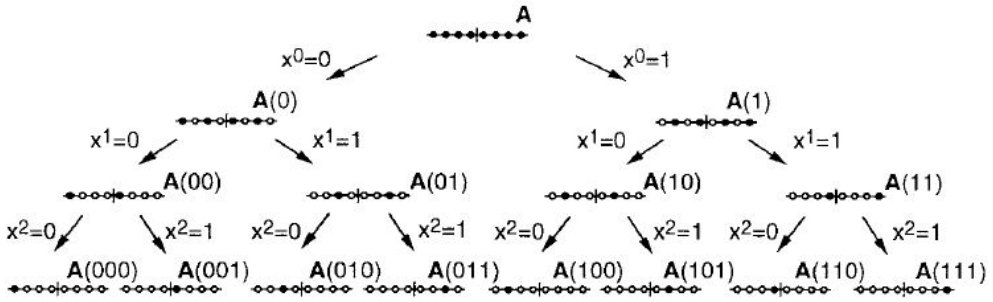

Figure 3.1: Binary partitioning of the 8-ASK signal set [6]

The signal set A is partitioned into two equiprobable subsets at each iteration i. It proceeds from the most significant bit to the least significant bit, basing the partitioning on the fact that $x^i = 0$ or $x^i = 1$. These iterations stop when just one symbol is present in the subset. In the transmitter, the real and the imaginary part of the constellation are dealt separately, therefore the 64-QAM constellation is considered as an 8-PAM x 8-PAM constellation, while the 16-QAM constellation as a 4-PAM x 4-PAM constellation. Hence, by partitioning in the way illustrated before, a 3-level structure is obtained for an 8-PAM set, while a 2-level structure for a 4-PAM.
Multilevel encoding is based upon a property of the mutual information between received signal and the transmitted signal. This mutual information $I(Y;A)$, between the transmitted signal point $a \in A$ and the received signal point $y \in Y$, where Y is the set of values received after the noisy channel, equals the mutual information $I(Y;X^0,X^1,...,X^{l-1})$ between the address vector $x \in X$ and the received signal point $y \in Y$. Applying the chain rule on mutual information we have :

$$\begin{aligned} I(Y;A) &= I(Y;X^0,X^1,...,X^{l-1}) \\ &= I(Y;X^0) + I(Y;X^1|X^0) + I(Y;X^{l-1}|X^0,X^1,...,X^{l-2}) \end{aligned} \tag{3.1}$$

Equation 3.1, as it is explained by Wachsmann in [6], symbolizes that, if the binary information $x^0, x^1, ..., x^{l-1}$ is known, the transmission of binary information x^i can be considered as the parallel transmission over l equivalent channels. Figure 3.2 shows an example of a transmission over a physical channel for an 8-ASK modulation with that label mapping and, according to the equation 3.1, its equivalent multilevel transmission in Figure 3.3.

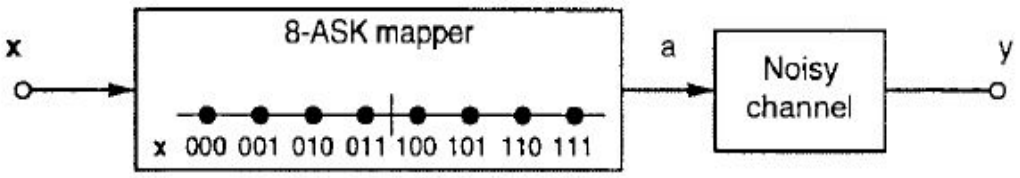

Figure 3.2: Actual channel.[6]

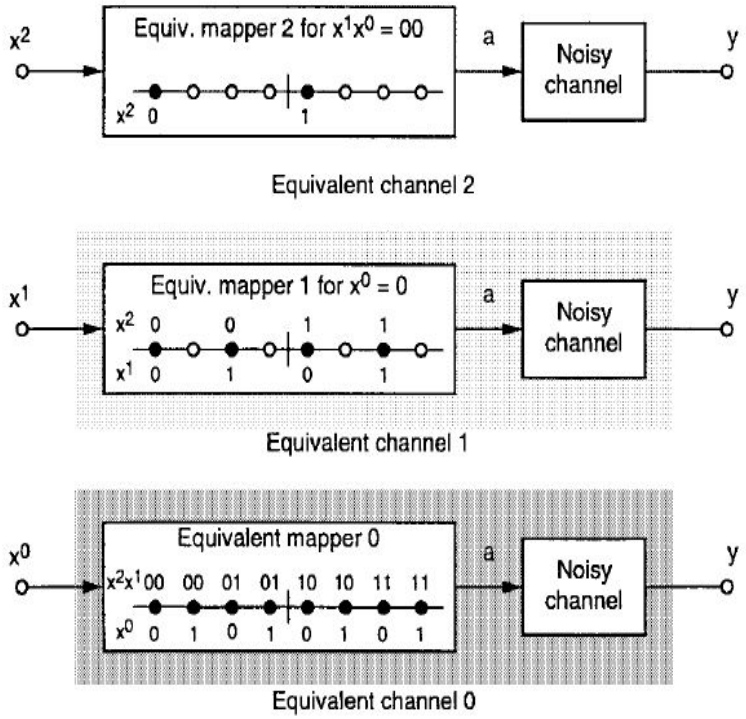

Figure 3.3: Equivalent channels for the 8-ASK mapper. [6]

In the example shown in Figure 3.3, it can be seen that the modulators of a higher level depend on the lower levels. In the equivalent channel 0 all the signal points for an 8-ASK mapping are taken into account for the binary digits x^i. In the equivalent mapper 1, due to the fact that $x^0 = 0$ in this example, half of the total signal points can be assigned. Finally , if $(x^0x^1) = (00)$ as the example shows, two different signal points can be transmitted, and the selection between these two signal points is based on the value of x^2. Looking to Figure 3.3 it is worth to notice that the Euclidean distances between the transmitted signal points increase as the level increases. Two effects can be considered: the distance between the constellation

points and the number of constellation points to select from. At moderate to high SNR, the first effect is more important, in fact the final decision will be between two competing constellation points and at a higher level the competing constellation points is always larger than at a lower level. Since, the overall error will be determined by the weakest channel, a stronger forward error correction has to be given to a lower level channel. Therefore, a convolutional code with a lower coding rate R_i is assigned to a lower level compared to the higher level as it is shown in Table 2.5.

3.2 Multistage Decoder

At this point, looking to Fig. 2.2 and having in mind the concept of equivalent channels, three different levels are encoded and QAM-64 symbols z are mapped to the channel. Next, these outputs cross the noisy channel, and distorted values $\hat{z}$ to be decoded are received. A straight forward solution to perform decoding on different levels, according to the chain rule on mutual information shown in Equation 3.1, is a multistage decoder. The multistage decoder is a leveled structure composed by a number of decoders equal to the number of levels at the transmitter side. Each decoder decodes the information of its level taking the information $\hat{u}^{(i)}$ derived from previous levels as prior information. An example of a traditional 3-levels multistage decoder is illustrated in Figure 3.4.

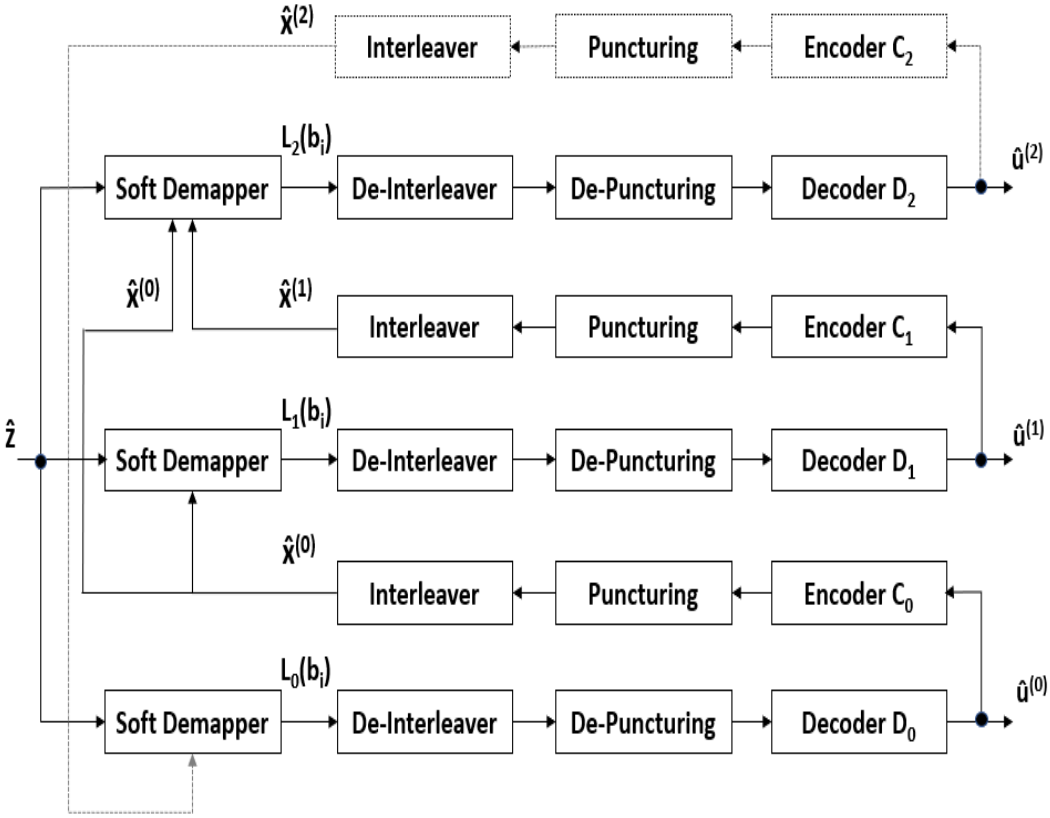

Figure 3.4: Multistage decoder.

The outputs $\hat{z}$ from the noisy channel, before being decoded by the corresponding decoder D_i, are processed by the Soft Demapper. The Soft Demapper converts the noisy signal points $\hat{z}_i$ to log likelihood ratios $L_k(b_i)$, which represent the probability

that $b_i = 0$ or $b_i = 1$. An explanation of that is given in the Soft Demapper section. Next, the $L_k(b_i)$ values are de-interleaved and de-punctured and finally they are decoded. As it is depicted in Fig. 3.4 the decoders D_i provides at their output the decoded information $\hat{u}^{(i)}$, which is used to provide a better decoding for the next stage. In that representation has been assumed that the level 0 is the first stage to be decoded and level 2 is the last. Generally speaking, the Soft Demapper at stage i, where i>1, processes not only the input $\hat{z}_i$, but also the prior decision $\hat{x}^{(j)}$ of the previous stages j. This means that each stage i could use all the prior decisions of previous stages j. As a matter of fact, in Figure 3.4 we can note that the stage at level 2 receives the prior information $\{\ \hat{x}^{(0)}, \hat{x}^{(1)}\}$ from both the previous levels.
Another important aspect to be considered in a multistage decoder is the number of stages to be performed. The number of stages should be at least equal to the number of levels encoded at the transmitter side, but after having decoded the last stage, one can by means of iterative decoding, reconsider to start decoding the first stages again as indicated by the dash boxes and dash lines in Figure 3.4 . In Chapter 6 simulation results will be provided for a different number of stages, trying to find a trade-off between performances and complexity.

3.3 Soft Demapper

The Soft Demapper is at the core of this work, therefore basic concepts are provided in this section to be able to understand its contribution. At the transmitter side, after some processing, the information is mapped into a QAM constellations and then transmitted. At the receiver side, distorted mapped signals are received due to a noisy channel and demapping from the received symbols to bits has to be performed.
The Soft Demapper demodulates the complex channels symbols R, providing n soft outputs for a M-QAM received symbol, where n = $log_2(M)$. These soft outputs can be represented as log-likelihood ratios (LLRs), values that indicate the probability that the demodulated bit b_k is a logical 0 or 1, shown in eq. 3.2.

$$LLR(b_k) = ln\frac{P(b_k = 1|r)}{P(b_k = 0|r)} \tag{3.2}$$

By developing equation 3.2 using the Bayes formula [7], the LLR of the coded bits b_k conditionated on the received symbol r can be computed as follow:

$$LLR(b_k) = ln\frac{\sum_{i=0}^{2^{n-1}-1} exp[-\frac{1}{2\sigma^2}(r - s_k(1,i))^2]}{\sum_{i=0}^{2^{n-1}-1} exp[-\frac{1}{2\sigma^2}(r - s_k(0,i))^2]} \tag{3.3}$$

where $s_k(1,i)$ and $s_k(0,i)$ are the constellation points whose k^{th} bit is respectively a 1 or a 0 and σ^2 represents the noise variance of the AWGN channel.

The LLR computation shown in Eq. 3.3 involves logarithmic and exponential functions, that are computationally complex, mostly if high order constellations are used. In order to reduce the complexity some approximations can be done. By applying $ln(\sum_j exp(-x_j)) \approx max(-x_j)) = \min(x_j)$ we can rewrite the Eq. 3.3 as:

$$LLR(b_k) = \frac{1}{2\sigma^2}[min_i(r - s_k(0,i))^2 - min_i(r - s_k(1,i))^2] \tag{3.4}$$

where i = 0,1,..., $2^{n-1} - 1$.
As it can be seen in Eq. 3.5 the complexity has been highly reduced, and this suboptimal method can be applied. Equation 3.5 also shows that this demapping method is based on Euclidean distances, searching for the two nearest constellation points to the received symbol.

Chapter 4

Challenge

In this Chapter the challenge of this Master thesis project is formulated. Car radio manufacturers differentiate in the market with reception quality. To this end, improving channel decoding performance is the main challenge tackled in this thesis. The traditional channel decoder, as explained in Chapter 3, provides sufficient performance for meeting the MRR. However to be at par or preferably ahead of the competition, NXP is always investigating how the performance can be improved and at which cost. The performance cost trade-off needs to be appropriate balanced in order to have a competitive product.

In this master thesis we intend to address that issue with soft-output decoding in the multistage decoder. First of all the objective is to quantify how much performance improvement can be obtained. Next a secondary question would be if this improvement is sufficiently large to investigate the extra complexity needed to realise this improvement. We have seen in Chapter 3 that in default multistage decoding, hard-decisions from a lower level are transferred to a higher level with the objective to improve the demodulation and decoding of that level. However, a wrong decision (decoding error) will not help the next level, instead it will deteriorate the demodulation and decoding performance. By providing decoder decisions with reliability information, the demodulation and decoding of the next level can accordingly weigh the information from a lower level and, in case of low reliability, limits the influence of this helper information.

Chapter 5

Channel decoder

5.1 Decoding algorithms

The aim of this thesis is to investigate whether it is possible to improve the current, hard-decision based, multistage decoder. For this purpose, we want to replace the hard-output FEC decoders with soft-output decoders. In case of soft-output decoding, the hard-decision decoder output is extended with reliability information. Maximum a-posteriori probability (MAP) decoders form a natural base for creating soft-decision information because they assess, based on received data and the code constraints, the probability whether a bit is a 0 or a 1. Instead of comparing which of these two probabilities is the largest for creating a hard-decision, they are combined in a so-called log-likelihood ratio in order to generate reliability information. The BCJR algorithm, so called from its authors Bahl, Cocke, Jelinev and Raviv [1], is a MAP decoding algorithm for convolutional codes. A sub-optimum, but less complex algorithm is the Max-Log-MAP algorithm, which is derived from the BCJR algorithm. Moreover, maximum a-posteriori probability algorithms provide the best performances in terms of BER. Therefore the optimum MAP decoder, the BCJR algorithm, has been initially taken into account, but due to its big complexity, we moved to the Max-Log-MAP algorithm. First, a description of the BCJR is given, next the Max-Log-Map is described. Before going into the description of these algorithms, a brief explanation of the trellis representation of convolutional codes is needed to properly understand these algorithms.

5.1.1 Trellis representation

The encoder of a convolutional code is a finite-state-machine that stores, encodes and outputs discrete valued information, e.g. binary information. The state of the encoder, which is stored in the encoder memory, is updated when new information enters the encoder. This operation is called state transition. The dynamic behavior

of the encoder, as function of time and the corresponding inputs, can perfectly be represented by a trellis diagram, as is shown in Figure 5.1.

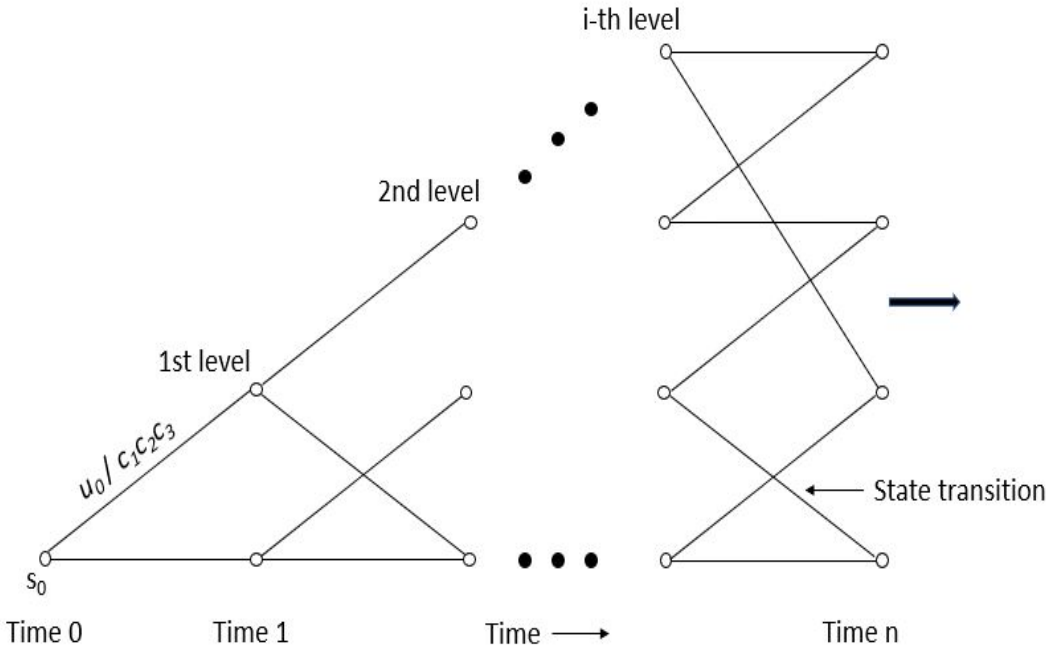

Figure 5.1: Trellis representation of an encoder [8].

Figure 5.1 illustrates a trellis representation of an encoder. The trellis shows how an encoder, as function of its current state and its input, will evolve as function of time. Every node (state) in the trellis has multiple incoming branches, except for s_0, and multiple outgoing branches, except for the final state where the encoding operation ends. In the first state transition the information input u_0 is encoded in the output bits $c_1c_2c_3$, according to a code rate = 1/3. Encoding an information sequence is equivalent to tracing a path in the trellis starting from the state s_0. Therefore a trellis structure can be used by the decoding algorithms, allowing maximum likelihood decoding with an important reduction in terms of complexity [8].

5.1.2 BCJR

Given a received sequence r, the BCJR computes the a-posteriori L-values

$$L(u_l) \equiv ln\frac{P(u_l = +1|r)}{P(u_l = -1|r)} \tag{5.1}$$

where u_l is the information bit at time l, where $l = 0,1,...,K-1$ with K=info bits + termination bits, having the following mapping: 0 to +1 and 1 to -1. The decoder outputs $\hat{u}_l = +1$ if $L(u_l)>0$, otherwise $\hat{u}_l = -1$ when $L(u_l)<0$. By using the trellis structure of the code, the $P(u_l = +1|r)$ can be expressed as:

$$P(u_l = +1|r) = \frac{P(u_l = +1, r)}{p(r)} = \frac{\sum_{(s',s)\in\sum_l^+} p(s_l = s', s_{l+1} = s, r)}{p(r)} \tag{5.2}$$

where $\sum_l^+$ is the set of all the state pairs $s_l = s'$ and $s_{l+1} = s$ corresponding to the input bit $u_l = +1$. Expressing the $P(u_l = -1|r)$ in the same way, we get:

$$L(u_l) = \ln \frac{\sum_{(s',s)\in\sum_l^+} p(s_l = s', s_{l+1} = s, r)}{\sum_{(s',s)\in\sum_l^-} p(s_l = s', s_{l+1} = s, r)} \tag{5.3}$$

Formula 5.3 expresses a ratio between the joint pdf $p(s', s, r)$ when the input bit $u_l = +1$ at the numerator and the joint pdf $p(s', s, r)$ when the input bit $u_l = -1$ at the denominator. The joint pdf $p(s', s, r)$ can be split up in the following way:

$$p(s', s, r) = p(s', s, r_{t<l}, r_l, r_{t>l}) \tag{5.4}$$

where the received sequence r is evaluated before time l, at time l and after time l. Next, by applying the Bayes formula as is shown in [8], we have:

$$p(s', s, r) = p(s', s, r_{t<l}, r_l, r_{t>l}) = p(r_{t>l}|s)p(s, r_l|s')p(s', r_{t<l}) \tag{5.5}$$

The BCJR algorithm is based on computing the $L(u_l)$ at each unit of time on the trellis. That is done by recursively computing forward and backward metrics along the trellis, as it will be shown in the sequel. To simplify Equation 5.5 we define the forward metric:

$$\alpha_l(s') \equiv p(s', r_{t<l}) \tag{5.6}$$

which represents a metric for state s' based upon received values $r_{t<l}$ and can be computed using a so called forward recursion as shown later on in this section. Similarly we define the backward metric as:

$$\beta_{l+1}(s) \equiv p(r_{t>l}|s) \tag{5.7}$$

which represents the metric of state s at time $l+1$ based on received values $r_{t>l}$, and can be calculated using a so called backward recursion, as it will be shown in the sequel. We can define a metric

$$\gamma_l(s', s) \equiv p(s, r_l|s') \tag{5.8}$$

which represents a branch metric for the transition between state s' and state s at time l and is calculated on base of the received signal r_l. This branch metric can be used both in the forward recursion and the backward recursion as shown below. In this way we can rewrite Formula 5.5 and 5.3 respectively as:

$$p(s', s, r) = \beta_{l+1}(s)\gamma_l(s', s)\alpha_l(s') \tag{5.9}$$

$$L(u_l) = \ln \frac{\sum_{(s',s)\in\sum_l^+}[\beta_{l+1}(s)\gamma_l(s', s)\alpha_l(s')]}{\sum_{(s',s)\in\sum_l^-}[\beta_{l+1}(s)\gamma_l(s', s)\alpha_l(s')]} \tag{5.10}$$

If α, β and γ are known for every state and every state transition and at every time instant l, the soft decoder output$L(u_l)$ can be determined. From a computational point of view it is therefore important to be able to compute α, β and γ in an efficient way.
The metric $\alpha_{l+1}(s)$ can be computed as function of metrics $\alpha_l(s')$ by using a forward recursion [8]:

$$\alpha_{l+1}(s) = \sum_{(s'\in\sigma_l)} \gamma_l(s', s)\alpha_l(s') \tag{5.11}$$

where σ_l is the set of all states at time l, leading to a state s. Figure 5.2 shows an example of the forward metric computation on a trellis section.

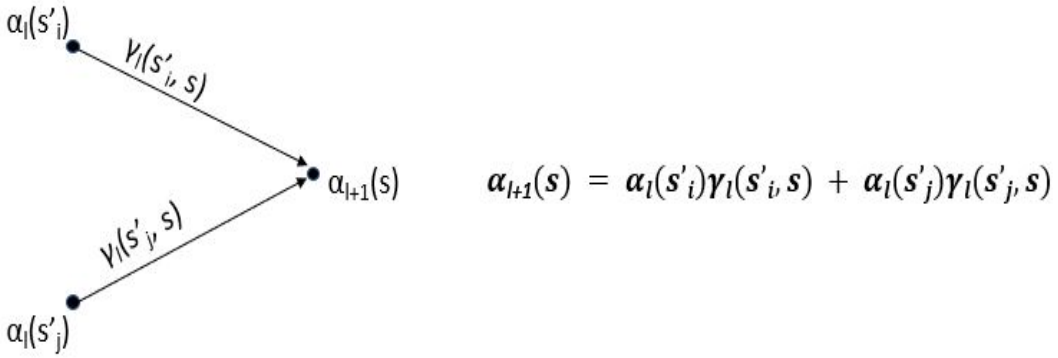

Figure 5.2: Forward recursion.

The forward recursion begins at time $l = 0$ with the following initial conditions:

$$\alpha_0(s) = \begin{cases} 1 & \text{for } s = 0 \\ 0 & \text{for } s \neq 0 \end{cases} \tag{5.12}$$

The metric $\beta_l(s')$ can be computed as function of metric $\beta_{l+1}(s)$ by using a backward recursion [8]:

$$\beta_l(s') = \sum_{(s' \in \sigma_{l+1})} \gamma_l(s', s)\beta_{l+1}(s) \tag{5.13}$$

where σ_{l+1} is the set of all states at time l+1, originating from state s'. Figure 5.3 illustrates an example of the backward metric computation on a trellis section.

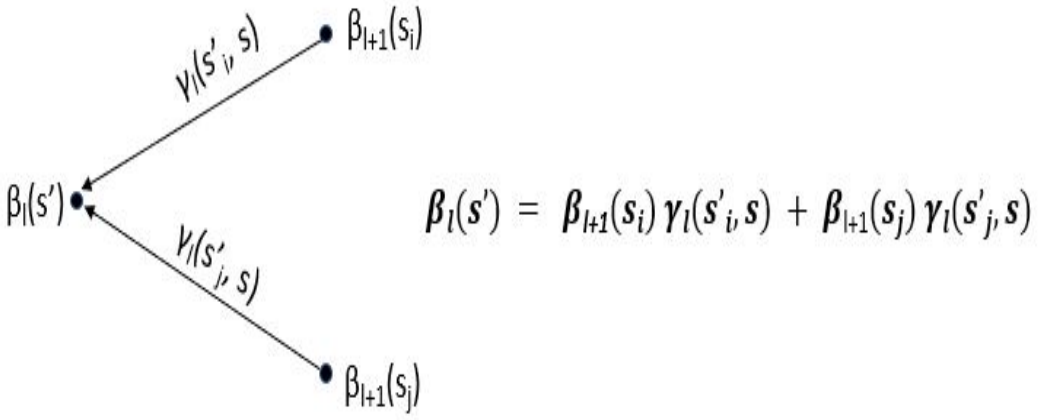

Figure 5.3: Backward recursion.

Similarly to the forward recursion, the backward recursion starts at time $l = K$ with the following initial conditions:

$$\beta_K(s) = \begin{cases} 1 & \text{for } s = 0 \\ 0 & \text{for } s \neq 0 \end{cases} \tag{5.14}$$

To compute the forward and backward recursion, the branch metric $\gamma_l(s', s)$ is needed. After applying some steps to Formula 5.8, illustrated in [8], we get:

$$\gamma_l(s', s) = P(u_l)p(r_l|c_l) \tag{5.15}$$

where c_l represents the code bits corresponding to the state transition $s' \rightarrow s$ at time l. Hence, we first compute the forward recursion from the beginning to the

end of the trellis, storing the α_l and γ_l for every state and state transition. After that we perform the backward recursion starting from the end of the trellis, and we compute on the fly the decoder outputs. In this way the β_l values don't have to be stored, saving storage space.

As shown in Figure 3.4, in a multistage decoder we need to help the next decoder with code bit data, rather then with information bit data. To avoid the encoding of information bits to code bits and the corresponding translation of reliability information, we choose to let the decoder produces directly log-likelihood ratios $L(c_i)$ on the code bits. As a matter of fact, the Decoder D_k outputs i log-likelihood ratios $L(c_i)$ of the code bits for each time l, where i is equal to n, given a code rate R_c =1/n. The computation of $L(c_i)$ is very similar to the calculation of the $L(u_l)$. The $\alpha_l, \beta_l, \gamma_l$ are calculated in an identical way but the calculation of $L(u_l)$ and $L(c_i)$ differs in the way the transitions between state s' and state s are divided into 2 groups. For $L(u_l)$ the division is based on the value of u_l, while for $L(c_i)$ the division into 2 groups of transitions is done on the base of the value of c_i. To understand better this concept we show an example of the computation of $L(c_i)$ by using a trellis branch representation illustrated in Figure 5.4.

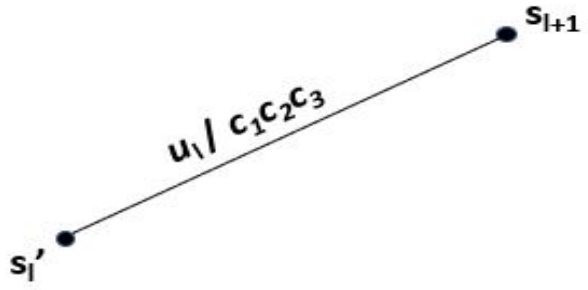

Figure 5.4: Trellis branch representation.

The code rate in the example is 1/3. The information u_l is encoded and the encoder outputs the code bits $c_1c_2c_3$. To compute the $L(c_i)$ we use Formula 5.3 as:

$$L(c_i) = \ln \frac{\sum_{(s',s)\in\sum_l^+}[\beta_{l+1}(s)\gamma_l(s',s)\alpha_l(s')]}{\sum_{(s',s)\in\sum_l^-}[\beta_{l+1}(s)\gamma_l(s',s)\alpha_l(s')]} \tag{5.16}$$

where $\sum_l^+$ and $\sum_l^-$ are the set of all the state pairs $s_l = s'$ and $s_{l+1} = s$ corresponding to the output bit $c_i = \pm 1$. As explained in this section, BCJR requires the computation of a forward and a backward recursion and is therefore also called the Forward-Backward algorithm. Moreover those computations involves many multiplications and divisions, implying a huge complexity. Hence, to provide an

efficient trade-off between decoding complexity and error performance, we moved to the Max-Log-MAP algorithm.

5.1.3 Max-Log-MAP

The Max-Log-Map decoding algorithm is an approximated version of the BCJR algorithm. It is based on the following approximation:

$$\ln(\sum_{i=1}^{q} \exp\{\delta_i\}) \approx (max_{1\leq i\leq q}\{\delta_i\}) \tag{5.17}$$

where $\{\delta_1, \delta_2, ..., \delta_q\}$ is a finite set of real numbers [8]. First of all we define:

$$A_l(s') = \ln \alpha_l(s') \tag{5.18}$$

which represents a metric for state s' and can be computed using a forward recursion as we will see later in this section.

$$B_{l+1}(s) = \ln \beta_{l+1}(s) \tag{5.19}$$

which represents the metric of state s at time $l+1$ and can be calculated using a backward recursion.

$$C_l(s', s) = \ln \gamma_l(s', s) \tag{5.20}$$

which represents a branch metric for the transition between state s' and state s at time l. In this way, applying approximation 5.17 to the Formula 5.10, we get:

$$\begin{aligned} L(u_l) = &max_{(s',s)\in\sum_l^+}\{A_l(s') + C_l(s', s) + B_{l+1}(s)\} \\ &- max_{(s',s)\in\sum_l^-}\{A_l(s') + C_l(s', s) + B_{l+1}(s)\} \end{aligned} \tag{5.21}$$

where $\sum_l^+$ and $\sum_l^-$ are the set of all the state pairs $s_l = s'$ and $s_{l+1} = s$ corresponding to the input bits $u_l = +1$ and $u_l = -1$ respectively. The logarithmic metric $A_{l+1}(s)$ can be calculated and approximated with the following forward recursion:

$$A_{l+1}(s) = \ln \alpha_{l+1}(s) = max_{s'\in\sigma_l}\{C_l(s', s) + A_l(s')\} \tag{5.22}$$

where σ_l is the set of all states at time l, leading to a state s. Figure 5.5 shows an example of the forward metric computation on a trellis section.

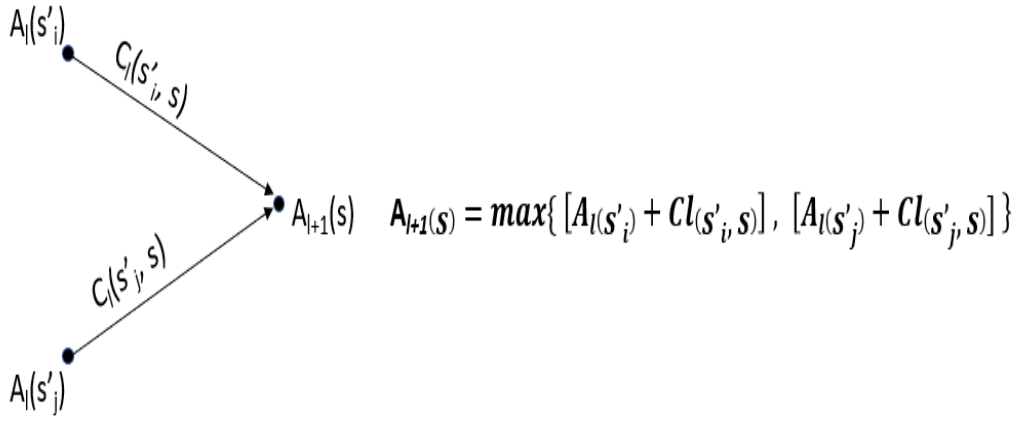

Figure 5.5: Forward recursion.

In Figure 5.5 it can be seen that the forward metric $A_{l+1}(s)$ is the max of two sums. The usage of the sums with respect to the multiplications of the forward recursion computation in the BCJR of Figure 5.2 implies a complexity reduction. As a matter of fact the Max-Log-Map forward recursion is identical to the Add-Compare-Select recursion that is used in the Viterbi algorithm [9].
As we did for the BCJR, we set the initial conditions to start the computation of the forward recursion at time l=0 :

$$A_0(s) = \ln \alpha_0(s) = \begin{cases} 0 & \text{for } s = 0 \\ -\infty & \text{for } s \neq 0 \end{cases} \tag{5.23}$$

The logarithmic metric $B_l(s')$ can be calculated and approximated with the following backward recursion:

$$B_l(s') = \ln \beta_l(s') = max_{s \in \sigma_{l+1}}\{C_l(s', s) + B_{l+1}(s)\} \tag{5.24}$$

Figure 5.6 illustrates an example of the backward metric computation on a trellis section.

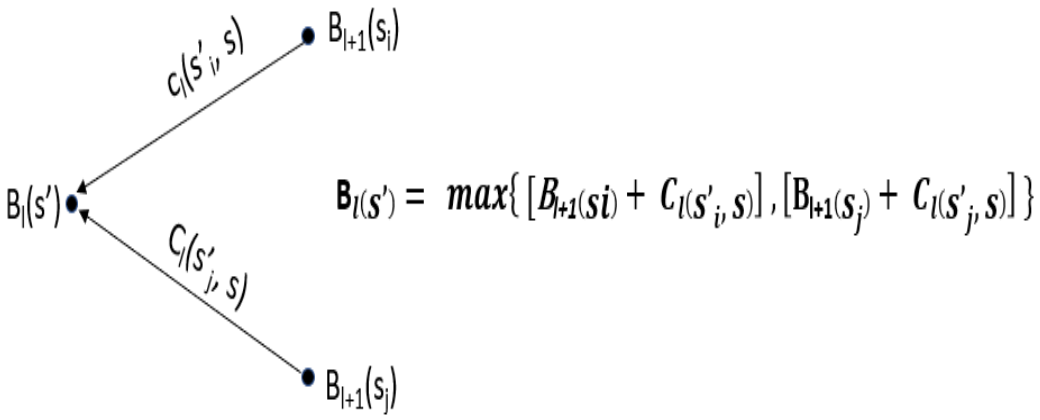

Figure 5.6: Backward recursion.

In Figure 5.6 it can be seen that the backward metric $B_l(s')$ is the max of two sums. This recursion is similar to the Add-Compare-Select recursion as present in the Viterbi algorithm, the difference is that this recursion is carried out in a time reversed way[9].

To be able to perform the backward recursion, initial conditions at time $l = K$ have to be set:

$$B_K(s) = \ln \beta_K(s) = \begin{cases} 0 & \text{for } s = 0 \\ -\infty & \text{for } s \neq 0 \end{cases} \tag{5.25}$$

Finally, the last value needed to compute the forward and backward recursion is the branch metric $C_l(s', s)$, illustrated in the following Equation[8]:

$$C_l(s', s) = \ln \gamma_l(s', s) = \ln(P(u_l)p(r_l|c_i)) \tag{5.26}$$

As we stated in the previous section, the decoder D_0 outputs LLRs $L(c_i)$ of the code bits. Making use of the example in Figure 5.4, the $L(c_i)$ are computed in the following way:

$$\begin{aligned} L(c_i) = & \, max_{(s',s)\in\sum_l^+}\{A_l(s') + C_l(s', s) + B_{l+1}(s)\} \\ & - max_{(s',s)\in\sum_l^-}\{A_l(s') + C_l(s', s) + B_{l+1}(s)\} \end{aligned} \tag{5.27}$$

where $\sum_l^+$ and $\sum_l^-$ are the set of all the state pairs $s_l = s'$ and $s_{l+1} = s$ corresponding to the output bit $c_i = \pm 1$.

5.1.4 Soft-Output Multistage Decoder

In this section a scheme for the multistage decoder with the soft decision $L(c_i)$ is provided. Since we have chosen to directly calculate log-likelihood ratios of code bits in the decoders, there is no need to encode the outputs after the Decoder, as in Figure 3.4 for the hard-output decoder, because the code rate R_c is already the one expected by the Soft Demapper. The interleaving and puncturing algorithm are the same, except that they process now soft-decision information instead of hard-decision information. The order of decoding is also the same as the hard decoder in section 3, while an important difference lies in the Soft Demapper, illustrated in the next section. The scheme of the soft output multi stage decoder is shown in Figure 5.7.

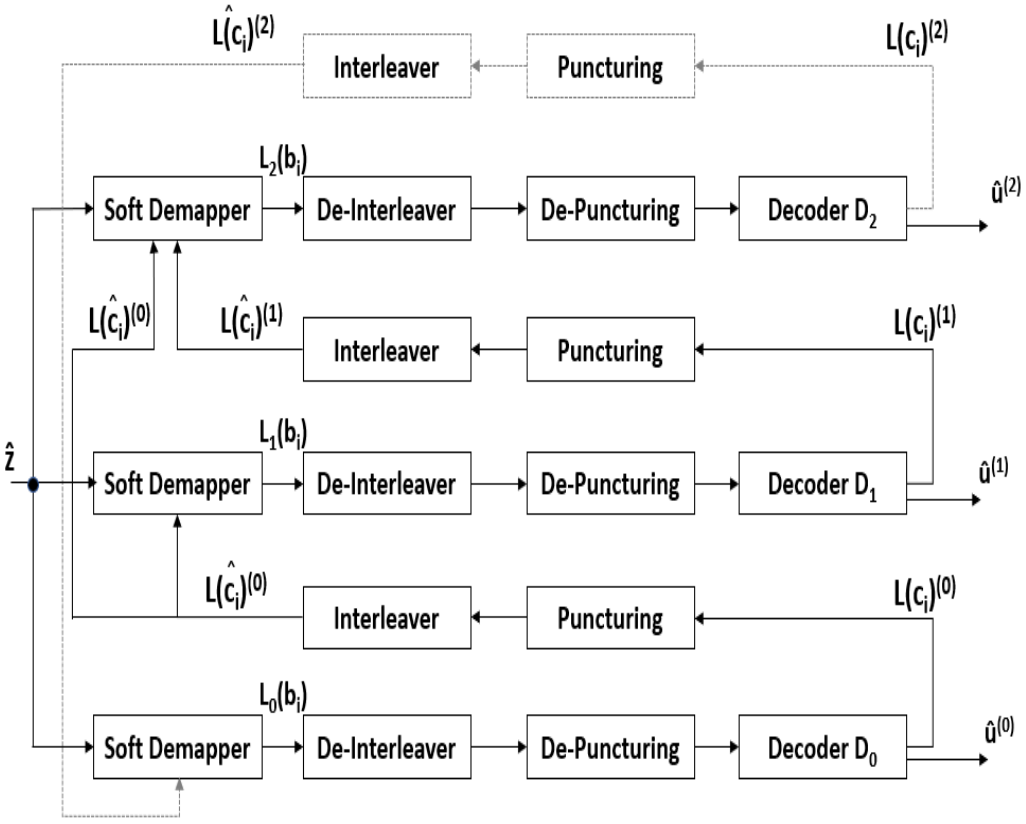

Figure 5.7: Soft-Output Multistage Decoder scheme.

5.2 Soft Demapper

In this section the Soft Demapper of the adopted decoder is explained in detail. In particular we are going to show how the LLRs from other levels can be used in calculating soft-outputs in the demodulation demapper by reporting an example for a 4-ASK constellation. A similar derivation can be carried out for higher order constellations.

In the Tx configurations used to verify the MRR, SM 16-QAM and SM 64-QAM (as shown in Figure 2.3 and 2.4) are used. As we have seen in Chapter 3, the M^2-QAM mapping can be decomposed into two independent M-ASK signals. This enables the separate soft-demapping of the real and imaginary part of the received signal. The SM 16-QAM signal can be decomposed into two 4-ASK signals with a bit mapping of $(b_0, b_1) = \{11, 01, 10, 00\}$ corresponding to the constellation points {-3, -1, 1, 3} $|h|^2\sqrt{E_l}$, where |h| is the gain due to the channel and $|h|^2$ is the gain after equalizing the received symbol by multiplying with h^*. The E_l is the energy level used to express the energy of different symbols in the constellations.
The Soft Demapper provides reliability values $L(b_i)$ for the demodulated bits b_i to the decoder. Applying Formula 3.3 the LLRs $L(b_i)$ for the bit b_0 and b_1 can be computed as:

$$L(b_0) = \ln \frac{\exp(-\frac{(R+|h|^2\sqrt{E_t})^2}{2|h|^2\sigma^2}) + \exp(-\frac{(R-3|h|^2\sqrt{E_t})^2}{2|h|^2\sigma^2})}{\exp(-\frac{(R+3|h|^2\sqrt{E_t})^2}{2|h|^2\sigma^2}) + \exp(-\frac{(R-|h|^2\sqrt{E_t})^2}{2|h|^2\sigma^2})} \tag{5.28}$$

$$L(b_1) = \ln \frac{\exp(-\frac{(R-|h|^2\sqrt{E_t})^2}{2|h|^2\sigma^2}) + \exp(-\frac{(R-3|h|^2\sqrt{E_t})^2}{2|h|^2\sigma^2})}{\exp(-\frac{(R+3|h|^2\sqrt{E_t})^2}{2|h|^2\sigma^2}) + \exp(-\frac{(R+|h|^2\sqrt{E_t})^2}{2|h|^2\sigma^2})} \tag{5.29}$$

where R is the received symbol from the channel after equalization, taking the real or the imaginary part ($R = \text{real}(rh^*)$ for the in-phase signal, $R = \text{imag}(rh^*)$ for the quadrature signal). The multiplication with h^* compensates the phase rotation applied by the channel, and as a result we observe symbols that are scaled with $hh^* = |h|^2$. At this point we assume that b_1 is known, ending up in a 2-ASK constellation, with the two possible following cases:

1. If $b_1 = 0$ the calculation of $L(b_0)$ becomes:

$$L(b_0) = \ln \frac{\exp(-\frac{(R-3|h|^2\sqrt{E_t})^2}{2|h|^2\sigma^2})}{\exp(-\frac{(R-|h|^2\sqrt{E_t})^2}{2|h|^2\sigma^2})} \tag{5.30}$$

and, after some simplifications, we get the first demapping result L_a shown in the following expression:

$$L_a = L(b_0) = -\frac{2}{\sigma^2}(R - 2|h|^2) \tag{5.31}$$

2. If $b_1 = 1$ the computation of $L(b_0)$ becomes:

$$L(b_0) = \ln \frac{\exp(-\frac{(R+|h|^2\sqrt{E_t})^2}{2|h|^2\sigma^2}))}{\exp(-\frac{(R+3|h|^2\sqrt{E_t})^2}{2|h|^2\sigma^2})} \tag{5.32}$$

ending up in the second demapping result L_b, obtained in a similar way of L_a, shown in the following expression:

$$L_b = L(b_0) = -\frac{2}{\sigma^2}(-R + 2|h|^2) \tag{5.33}$$

In this thesis it is assumed that no prior information about the source bits is known by the receiver. Hence, in decoding the first level no prior information from previous levels is present. After decoding the first level, the log-likelihood ratios $L(c_1)$, as estimation of a LLR of b_1, is received where:

$$L(c_1) = \ln \frac{P(c_1 = 0)}{P(c_1 = 1)} \tag{5.34}$$

where c_1 is the estimation of b_1 according to the decoder of the previous level. Reverting Equation 5.34 we can get:

$$P(b_1 = 1) = P(c_1 = 1) = \frac{1}{\exp(L(c_1)) + 1} \tag{5.35}$$

where $P(c_1 = 0) = 1 - P(c_1 = 1)$. In the same way we can get $P(b_1 = 0)$ as:

$$P(b_1 = 0) = P(c_1 = 0) = \frac{\exp(L(c_1))}{\exp(L(c_1)) + 1} \tag{5.36}$$

where $P(c_1 = 1) = 1 - P(c_1 = 0)$.

In the expressions 5.31 and 5.33 we can see that the behavior of $L(b_0)$ as function of R is piecewise linear. This scenario happens when we are totally sure that $b_1 = 0$ or $b_1 = 1$. In case of hard-decision decoding, by absence of a better option, it is usually assumed that the decoder output is correct, i.e. the assumption is that the decoder output has infinite reliability. However, if the decoded bit b_1 is wrong, the wrong soft-demapper expression is used and the decoding of that level is deteriorated.

In order to include reliability information from the previous decoding level into the soft-demapping result, we weighted the value of $L(b_0)$ in the following way:

$$L(b_0) = P(b_1 = 0) * L_a + P(b_1 = 1) * L_b \tag{5.37}$$

Replacing $P(b_1 = 0)$ with Equation 5.36 and $P(b_1 = 1)$ with Equation 5.35, we finally get :

$$L(b_0) = \frac{\exp(L(c_1))}{\exp(L(c_1)) + 1} * L_a + \frac{1}{\exp(L(c_1)) + 1} * L_b \tag{5.38}$$

The weighted value $L(b_0)$ derived from Formula 5.38 can limit the error propagation between the stages, especially if we are working at low SNR, where it's more likely to have errors. In the same way we can derive expressions for $L(b_1)$ and more generally for the soft-demapping of higher order signal constellations.

Chapter 6

Simulation results

In this Chapter the BER performances of the soft-output multistage decoder scheme are provided. In particular Section 6.1 shows the BER performance differences with respect to the hard-decision multistage decoder. Section 6.2 explains the reasons of using a scaling factor to scale the log-likelihood ratios $L(c_i)$ that are produced by the soft-output convolutional decoders. Finally Section 6.3 gives a brief insight in the complexity performance trade-off of the system, by providing simulation results with different number of stages. All the simulations are performed in the Matlab NXP chain environment.

6.1 Results

The simulation results provided in this section are obtained by applying the soft-output multistage decoder, shown in Chapter 5, decoding the data with the Max-Log-MAP algorithm. The intent is to see whether this new approach gives some improvements with respect to the scheme with hard-output decoder. Both systems are simulated for different SNR (Signal-to-Noise-Ratio) and compared at the point where the simulations achieve a BER equal to 10^{-4}. Five stages are simulated for all the channel profiles except for channel profile 5, where three stages are simulated. The order of decoding is not provided for commercial and privacy reasons.
Table 6.1 shows the SNR improvements in dB of the adopted approach for each channel profile.

Ch. Profile	Δ [dB]
1	0.2
2	0.6
3	1.4
4	0.7
5	0.5
6	1.0

Table 6.1: SNR improvements with soft-output multistage decoder.

The Δ in Table 6.1 shows the performance improvements in terms of SNR gains with respect to the hard-output multistage decoder approach. As it can be seen in Table 6.1, we get performance improvements for each channel profile. For the channel profile 1 (AWGN channel model) the two approaches are more or less similar in terms of SNR, getting a little improvement of 0.2 dB. More prominent gains are obtained instead for the other channels. In particular channel profile 3 and channel profile 6 outperform the hard-output multistage decoder approach with a gain > 1 dB. The soft-output approach supports the soft-demapper with reliability information and limits error propagation. This probably helps the multistage decoder more in the case of channels that have more time-selectiveness (more Doppler), as the results show for channel profile 3 and channel profile 6 (see channel parameters in Table 2.1).

6.2 Scaling of LLR's

This section summarizes the investigation of the impact of using an additional scaling factor μ in the multistage decoder architecture. In Figure 6.1 and 6.2 two different decoder schemes are shown: the optimal decoder and the decoder configuration used in this thesis. The theoretical decoder (Figure 6.1) employs the BCJR decoding algorithm, initially used in the multistage decoder simulations but later on discarded for complexity reasons. The BCJR algorithm is the optimal MAP decoding algorithm, as it uses exact expressions to determine LLRs (see Section 5.2). In the used scheme the Max-Log-Map algorithm is employed, that uses approximations to calculate LLRs (see Section 5.3).
As shown in Figure 6.1, at the input of the decoder next to the received signal R, there is also an estimate of the channel $|h|^2$ and an estimate of the noise variance σ^2 needed to properly carry out the soft-demapping function. The soft-demapper also assumes that the soft-values that are obtained from the previous stages are true LLRs. By using max-Log-MAP decoders, we know that the latter assumption is violated. On top of that the estimation of the noise variance might be inaccurate.

Moreover, from literature it is known that scaling of LLRs in iterative decoding schemes might compensate for approximations and, as a result, can improve the decoder performance [10]. Therefore it was investigated whether the performance of the multistage decoder could be improved by using a scaling factor μ in the parsing of LLRs from one level to another level (See Figure 6.2). With a lot of simulations a scaling factor μ was searched for, that could be used for all channel profiles. It turned out that a scaling factor of $\mu = 0.35$ is a good choice for all channel profiles.
Figure 6.2 shows the decoder scheme used and how the $L(c_i)$ are properly scaled.

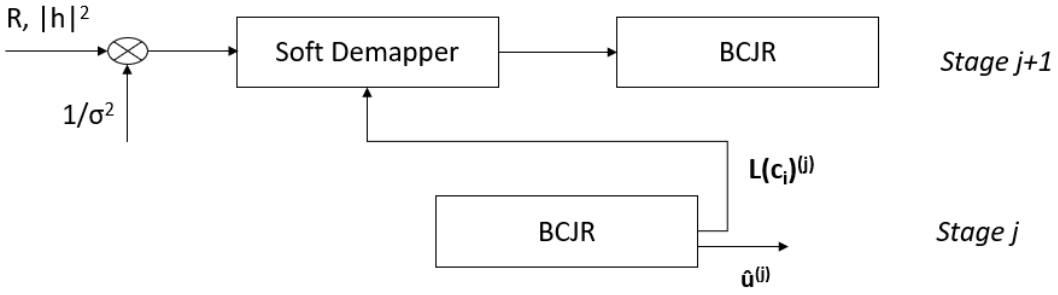

Figure 6.1: Theoretical decoder scheme.

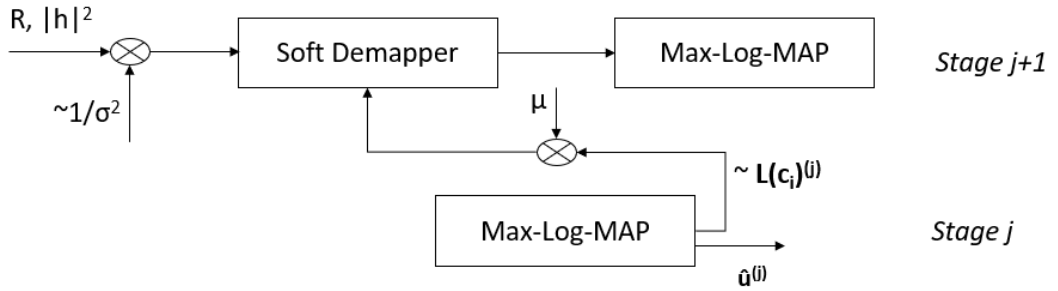

Figure 6.2: Used decoder scheme.

6.3 Complexity Performance Trade-off

In this section the trade-off between performance and decoder complexity is analysed. As we saw in Section 3.2, the minimum number of stages to be decoded is N, where N is equal to the number of levels in which the signal is encoded at the transmitter side (see Section 3.1). To try to achieve better results, more stages can be performed, helping the decoding of the stages with more and more reliable information. The simulation results illustrated in Section 6.1 are performed with the same number of stages for both the hard-output and the soft-output variant. For Channel Profile 5, 3 stages of decoding are employed, which corresponds with $N + 1 = 3$ (16-QAM modulation, 2 levels), while for the other channel profiles $N + 2 = 5$ stages of decoding are employed (64-QAM, 3-level encoding). This section is aimed to show the following two goals:

- How much the performance degrade when less stages of decoding are employed, to verify whether the performance are still better than the hard-output scheme.
- How much the performance can be improved when we apply more stages of decoding, to check whether we are close to a saturated performance level, or if there are still opportunities for improvements.

Table 6.2 shows the BER performance gains of the simulated approach with respect to the old approach, simulating N+1, N+2 and N+3 stages, where N is equal to 3 for all the channel profiles, except for channel profile 5 where N=2.

Channel Profile	Δ [dB] N+1 Stages	Δ [dB] N+2 Stages	Δ [dB] N+3 Stages
1	0.2	0.2	0.2
2	0.6	0.6	0.7
3	1.1	1.4	1.5
4	0.5	0.7	0.7
5	0.5	0.5	0.5
6	0.6	1.0	1.1

Table 6.2: SNR improvements in relation to the number of stages.

Table 6.2 shows the performances with one less and one more simulated stage with respect to the current implementation ($N + 2$ stages). The improvements are always compared to the hard-output multi stage decoder implementation. As it can be seen in Table 6.2, with N+1 stages simulated we get a gain bigger than 0.5

dB for almost all the channel profiles.
By increasing the number of stages to N+3, we get almost the same performances as the current soft-output implementation. Hence, we are confronted with the law of diminishing returns and therefore we believe that with $N + 2$ stages we are at a favourable trade-off between complexity and performance.

Chapter 7

Conclusion and Recommendations for Future Work

7.1 Conclusion

At the core of the work, the intention of the thesis was to investigate whether employing soft-output decoding in a DRM multistage decoder improves the MRR performance and if so, to quantify the amount of improvement. The subsequent question was if this improvement was sufficiently large, to investigate the extra complexity needed to realise this improvement.
To do that, the hard-output Viterbi decoder was replaced by a soft-output decoder. The Max-Log-MAP algorithm was used to produce soft-output information in the form of log-likelihood ratios (LLRs) which are used to improve the Soft Demapper in producing soft-decision information for other levels in the multistage decoder. The Soft Demapper, which produces soft-decisions on demodulated symbols, was modified to include soft-outputs obtained from the soft-output decoders.
The simulations of the modified decoder scheme showed improvements for all the DRM channel profiles, as we saw in Section 6.1. Hence, a first conclusion can be derived. The exchange of LLRs among the levels in a multistage decoder helps the decoding of the level, leading to as SNR performance improvements for a DRM receiver.
Next, we saw that these SNR improvements vary depending on the simulated channel profile. For an AWGN channel a small improvement of 0.2 dB was obtained. While for channels that have more time-selectiveness, gains of about 1 dB were obtained, with a maximum gain of 1.4 dB for channel profile 3. A

complexity versus performance trade-off investigation was conducted in Section 6.2. Increasing the number of stages to be decoded, which increases the decoder complexity, doesn't lead to significant SNR improvements. This made us believe that the results obtained with decoding $N + 2$ stages are close to a saturated performance level and therefore there is no need of investigating extra complexity to further improve the SNR performance. Simulations with one stage less than the current implementation were also performed. The results were still better than the hard-output scheme, confirming the usefulness of using soft-decision decoding in a multistage decoder.
The SNR improvements obtained with the soft-output multistage decoder have to be placed in context with the required complexity increase to realize this improvement. The used Max-Log-MAP algorithm is approximately 2.5 times more complex than the Viterbi algorithm. A system architect should consider the investigated decoder improvement with other possible improvements and their associated complexity cost.

7.2 Recommendations for Future Work

In Chapter 6 an analysis of the complexity versus performance trade-off was carried out. This analysis related the number decoded stages to the SNR performance gains. For future work we recommend a more detailed complexity analysis, in which HW/SW trade-offs in relation to platform resources are made. in the presented form the soft-output decoder has more latency than the current one. First of all the Max-Log-MAP decoder is more complex than the Viterbi decoder. Next we should consider the complexity increase in the Max-Log-MAP algorithm due to the computation of the soft-output values. As we have seen in Section 5.1.2 the decoder produces log-likelihood ratios $L(c_i)$ on the code bits. This operation brings extra complexity with respect to the common Max-Log-MAP algorithm. Another factor to take into account is the elimination of the encoder operation in the traditional multistage decoder shown in Figure 3.4. The interleaver in the re-encoding branch carries soft decision information instead of hard-decision information. Overall each block of the multistage decoder has to be taken into account and we have to consider the influence of each block on the whole system to provide an exact complexity analysis.

In this work the impact of the decoding approach was isolated from other contributors to the system performance by using an ideal channel estimation algorithm and perfect synchronisation. A next step would be performing the simulations in case of a true channel estimation and synchronization algorithm.

Another topic for future work is a more elaborate study on the scaling factor μ presented in Section 6.2. The decoder represented in Figure 6.2, implemented by using the Max-Log-MAP algorithm, can generate mismatches in the outputs due to approximations, with respect to the optimum decoder scheme in Figure 6.1, implemented by using the BCJR algorithm. Therefore various simulations with different scaling factors have been performed to look for the scaling factor that provided the best performance for all the channel profiles. To fully explore this topic is time consuming because of the experimental status of the corresponding analysis and the currently found value $\mu = 0.35$ may not work for a real case. Hence further analysis of the scaling factor can be carried out and also a more analytical way to determine the performance after infinite iterations.
The decoding algorithms explained in this thesis have been implemented in a Matlab environment considering a terminated trellis. It means that they are applied on a limited number of transmitted symbols. The current algorithm can start producing soft-outputs until the last symbols of a frame are received. However, because of latency and memory limitations on an embedded platform, one wants to start earlier with the production of soft-outputs. To this case a sliding-window BCJR or Max-Log-MAP can be implemented, presented in the next section.
Lastly we are going to present an iterative multistage decoder. It exploits jointly parallel and multistage decoding, and it might be an idea to further improve the BER performance for a DRM application.

7.2.1 Sliding-Window BCJR

As we said previously, the decoding algorithms explained in this thesis have been implemented for convolutional codes with trellis termination. The input sequence has to be received before the backward recursion can start from the final trellis state. However, in pratical decoders, this implementation cannot be used for continuous decoding of convolutional codes because we can't wait to receive the whole transmission for memory and latency issues. Hence, to overcome the latency and memory requirements, an idea is to modify the BCJR algorithm in such a way that it operates on fixed memory span and outputs the log-likelihood ratios after a given delay D (window size). This modified algorithm is called Sliding-Window BCJR algorithm. In [11] two different versions of the sliding-window algorithm are proposed. Here we will show just the first version called Sliding-Window SISO Algorithm for the BCJR algorithm as a possible implementation in an embedded system. The assumption is that the time index set K is semi-infinite with K=1, ..., inf
The steps of the Sliding-Window BCJR algorithm are the following:

- Initialization of the forward recursion $\alpha_0(s)$ according to 5.23.

- Computation of the forward recursion $\alpha_{k+1}(s)$ and the branch metric $\gamma_k(s', s)$ according respectively to 5.11 and 5.15
- Initialization of the backward recursion $\beta_k(s)$ for the time $k > D$ in the following way:
$$\beta_k(s) = \alpha_k(s) \tag{7.1}$$
where D is the window size.
- Computation of the backward recursion $\beta_k(s)$ according to 5.13, from time k-1 to time k-D
- Computation of the log-likelihood ratios $L(u_l)$ and $L(c_i)$ according respectively to 5.10 and 5.16.

The parameters of the sliding window algorithm (window size and the backward recursion run-in length) must have a certain dimension, such that an appropriate trade-off between latency, memory versus extra computations can be made. The same procedure can be applied on the case of implementation of the Sliding-Window Max-Log-MAP with the appropriate changes.

7.2.2 Iterative Soft-Outputs Multistage Decoding

Iterative multistage decoding is a decoding technique that exploits a parallel decoding of the levels jointly with a decoding by using a-priori information for iterations $i > 1$. Before going into the description of this technique, we will first have a look on parallel decoding (PLD).
The multilevel coding - parallel decoding approach (MLC/PLD) was proposed by P. Schramm in [12] and consists of a parallel decoding of each level. Each level is decoded independently, without feedback of a-priori information. Therefore a decoder D_i makes no use of decisions of others levels. The advantage of this approach is that we avoid error propagation between the levels since the levels decoding is done independently. Another advantage regards the complexity: the decoding of the level is done in parallel, so all the levels are decoded at the same time. In the multistage decoding scheme one level per time is decoded. Besides, each level has to wait for the incoming information from the lower layers before starting to decode. Figure 7.1 shows a scheme of a parallel multilevel decoding.

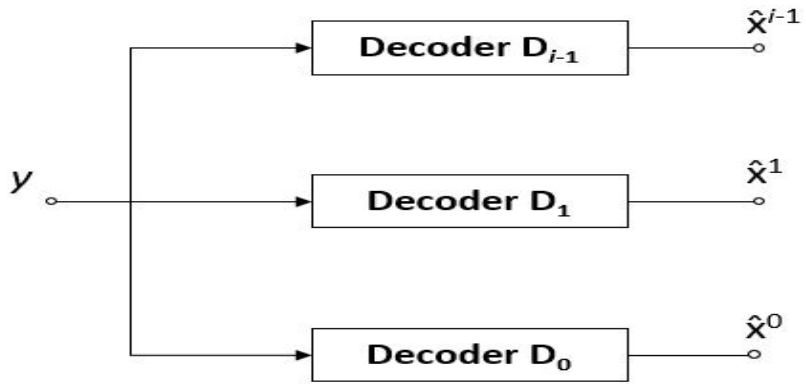

Figure 7.1: Parallel decoding of levels [6].

Since no prior information is used, while decoding, the selection of the signal point is based on the entire signal constellation. In section 3.1 we have seen that the equivalent modulator i is time variant for $i>0$. That is because it depends on the selection of the binary digits x^j of the lower levels j. In the PDL all the equivalent modulators are time-invariant, and each modulator comprise the entire signal set [6]. The lack of feedback information among the levels can obviously avoid the error propagation seen in the MSD, but from the other side, that feedback decisions can provide useful information to improve the decoding of the next level.
In [13] the MLC/MSD and the MLC/PLD are compared. The authors performed simulations over AWGN and Rayleigh fading channels. It turned out that the MLC/MSD approach provides better or at most similar performances with respect to the MLC/PLD approach.
Now that we have summarized the MLC/PLD approach, and we have seen the improvements brought by the exchange of soft decisions among the levels in a multistage decoder, it might be interesting to simulate a combination of both approaches for a DRM application. Figure 7.2 illustrates an iterative multistage decoder with exchange of soft decisions among the levels.

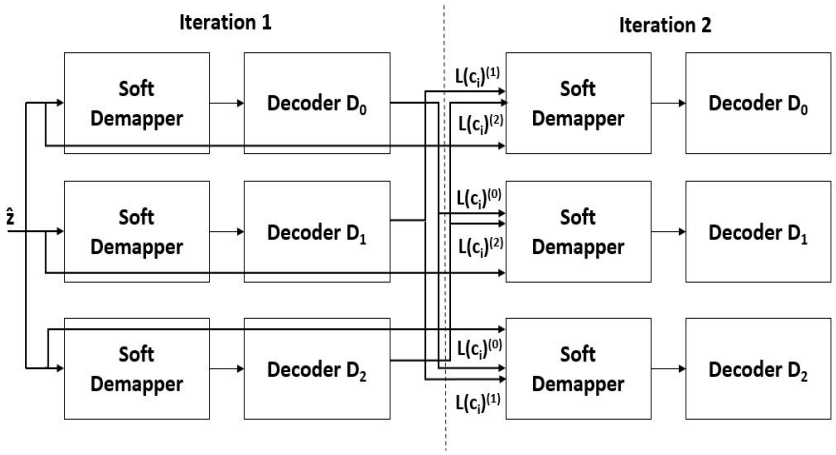

Figure 7.2: Iterative Multistage Decoder with exchange of soft decisions among the levels.

In the first iteration all the levels are decoded in parallel without a-priori information. In the second iteration the soft demappers receive soft decision information $L(c_i)$ from the other levels of the previous iteration and the levels are decoded in parallel taking decisions based also on the soft information $L(c_i)$ as we have seen in the multistage decoder. More than two iterations can be performed to see whether the performance improves, taking always in mind a complexity-performance trade-off.

References

[1] L. Bahl ; J. Cocke ; F. Jelinek ; J. Raviv. «Optimal decoding of linear codes for minimizing symbol error rate». In: *IEEE TRANSACTIONS ON INFORMATION THEORY* 20 (2 March 1974), pp. 284–287 (cit. on pp. x, 18).

[2] DRM Consortium. «The DRM Digital Broadcasting System. Introduction and Implementation Guide; » in: (Revision 4, February 2019) (cit. on p. 2).

[3] «Digital Radio Mondiale (DRM); System Specification». In: *ETSI ES 201 980* 4.1.2 (2017/04) (cit. on pp. 3–9).

[4] «Digital Radio Mondiale (DRM); Minimum Receiver Requirements for DRM receivers operating below 30 MHz (DRM30)». In: (Version 2.0, 16th February 2015) (cit. on p. 9).

[5] H. Imai and S. Hirakawa. «A new multilevel coding method using error correcting codes». In: *IEEE Trans. Inform. Theory* IT-23.10 (May 1977), pp. 71–377 (cit. on p. 10).

[6] Rober F. H. Fischer Udo Wachsmann and Johannes B. Huber. «Multilevel Codes: Theoretical Concepts and Practical Design Rules». In: *IEEE TRANSACTIONS ON INFORMATION THEORY* 45.5 (July 1999), pp. 71–377 (cit. on pp. 11, 12, 40).

[7] F. Tosato and P. Bisaglia. «Simplified soft-output demapper for binary interleaved COFDM with application to HIPERLAN/2». In: *IEEE ICC* 2 (2002), pp. 664–668 (cit. on p. 15).

[8] Shu Lin; Daniel J. Costello Jr. *Error Control Coding*. Pearson. ISBN: 978-81-317-3440-7 (cit. on pp. 19–22, 24, 26).

[9] Forney G. D. Jr. «The Viterbi Algorithm». In: *IEEE* 61.3 (1973), pp. 268–278 (cit. on pp. 25, 26).

[10] Marten van Dijk ; Augustus J. E. M. Janssen ; Arie G. C. Koppelaar. «Correcting systematic mismatches in computed log-likelihood ratios». In: *European Transactions on Telecommunications* 14.3 (28 July 2003) (cit. on p. 33).

[11] S. Benedetto; D.Divsalar; G. Montorsi; and F. Pollara. «A Soft-Input Soft-Output Maximum A Posteriori (MAP) Module to Decode Parallel and Serial Concatenated Codes». In: *TDA Progress Report* (November 1996), pp. 42–127 (cit. on p. 38).

[12] U. Wachsmann ; J. Huber ; P. Schramm. «Comparison of coded modulation schemes for the AWGN and the Rayleigh fading channel». In: *ISIT 98,Boston,USA* (September 1998) (cit. on p. 39).

[13] Dong-Feng Yuan ; Cheng-Xiang Wang ; Qi Yao ; Zhi-Gang Cao. «Comparison of Multilevel Coded Modulations with Different Decoding Methods for AWGN and Rayleigh Fading Channels». In: *IEEE* (2000) (cit. on p. 40).