DELIVERABLE D2.2

REPORT ON DVB-NGH RECEIVER ISSUES

13TH NOVEMBER 2012

Editor: Tero Jokela

University of Turku, Finland

EXECUTIVE SUMMARY

In ENGINES work package two (WP2), individual system architecture components were studied, and the results from these studies have been forwarded to standardization work (DVB-T2 Lite, DVB-NGH). The outcome of WP2 work is summarized in five deliverables. Deliverable 2.1 focuses on system architectural work performed by ENGINES partners. Deliverable 2.2 deals with DVB-NGH receiver implementation related issues. Devised advanced component techniques for DVB-NGH are presented in deliverable 2.3. Additionally there is work on overall architectures, especially issues not covered by direct standardization that are novel access technologies (deliverable 2.4) and end-to-end system integration (deliverable 2.5).

This deliverable focuses on algorithms applied at the receiver. These algorithms cover channel estimation, time/frequency synchronization, MIMO detection. Further, receiver complexity regarding selected algorithms is estimated and algorithms for reducing receiver complexity are presented. The performance of the algorithms is presented via simulations. The topics considered are:

- Generic channel equalization techniques for OFDM based systems in time-variant channels
- A Shuffled Iterative Receiver for the DVB-T2 Bit-Interleaved Coded Modulation: Architecture Design, Implementation and FPGA Prototyping
- Expected DVB-T2 Performance Over Time Varying Environments
- Fast GPU and CPU implementations of an LDPC decoder
- MIMO detection

The receiver issues related to novel access technologies are presented in deliverable 2.4.

TABLE OF CONTENTS

LIST OF CONTRIBUTORS

- BBC
- INSA-IETR
- Parrot/Dibcom
- Telecom Bretagne
- Universidad Politécnica de Valencia/ iTEAM
- University of the Basque Country
- Åbo Akademi University
- ???

1 INTRODUCTION

This deliverable is dedicated to issues related to DVB-NGH receiver algorithms and implementation issues. As DVB-NGH is destined for mobile users, complexity of the receiver plays an important role in the design. The rest of the deliverable is structured as follows.

In Chapter 2, a generic channel equalization technique for OFDM based systems in time variant channels is presented. It is proven that the most known equalization algorithms for OFDM signals in time variant channels with mobile reception scenarios are part of this generic theoretical model. This model is developed mathematically, and based on it, a general classification for channels in terms of their time variability is presented. Besides, the equalization methodology reliability and the channel classification validity have been proved in both the TU-6 and MR channels. This generic methodology could be considered for the equalization stages in the DVB-T2/NGH receivers working in mobile scenarios.

Chapter 3 introduces an efficient shuffled iterative receiver for the second generation of the terrestrial digital video broadcasting standard DVB-T2. A simplified detection algorithm is presented, which has the merit of being suitable for hardware implementation of a Space-Time Code (STC). Architecture complexity and measured performance validate the high potential of iterative receiver as both a practical and competitive solution for the DVB-T2 standard.

Chapter 4 focuses on the performance of DVB-T2 in time varying environments. In order to model the channel impulse response, a TU6 channel is considered. The latter constitutes the most common channel model of DTT standards for mobile environments. The performance of the standard is simulated for both single and diversity 2 reception. Since DVB-T2 contains a huge number of possible configurations, focus is mainly given to two configurations: UK mode, and Germany-like candidate mode.

Chapter 5 presents two implementations of LDPC decoders optimized for decoding the long codewords specified by the second generation of digital television broadcasting standards: i.e. DVB-T2, DVB-S2, and DVB-C2. These implementations are highly parallel and especially optimized for modern GPUs (graphics processing units) and general purpose CPUs (central processing units). High-end GPUs and CPUs are quite affordable compared to capable FPGAs, and this hardware can be found in the majority of recent personal home computers.

Finally, Chapter 6 studies MIMO detection in the receiver. Considered issues are the complexity needed to perform maximum likelihood (ML) decoding for MIMO systems and iterative MIMO receiver processing. The DVB-NGH standard is the first to include a full rate MIMO scheme. Even though the number of antennas is relatively small, the complexity to implement an ML decoder can be prohibitive. This chapter proposes and studies ways to reduce complexity of the DVB-NGH MIMO reception.

2 On Approaching to Generic Channel Equalization Techniques for OFDM Based Systems in Time-Variant Channels

2.1 Introduction

Orthogonal frequency division multiplexing (OFDM) is widely considered as an attractive technique for high-speed data transmission in mobile communications and broadcast systems due to its high spectral efficiency and robustness against multipath interference [1]. It is known as an effective technique for digital video broadcasting (DVB) since it can prevent inter-symbol interference (ISI) by inserting a guard interval and can mitigate frequency selectivity by estimating the channel using the previously inserted pilot tones[1][2].

Nevertheless, OFDM is relatively sensitive to time-domain selectivity, which is caused by temporal variations of a mobile channel. In the case of mobile reception scenarios dynamic channel estimation is needed. When the channels do not change within one symbol, the conventional methods consisting in estimating channel at pilot frequencies, and afterwards, interpolating the frequency channel response for each symbol could be implemented [2][3]. The estimation of pilot carrier can be based on Least Square (LS) or Linear Minimum mean-Square-Error (LMMSE). In [3], it is proved that despite its computational complexity LMMSE shows a better performance. And in [2], low pass interpolation has been proved to have the best performance within all the interpolation techniques.

Their performance is worse for time-varying channels, which are not constant within the symbol. In such cases, the time-variations lead to inter-sub-carrier-interference (ICI), which breaks down the orthogonality between carriers so that the performance may be considerably degraded. There are several equalization methods depending on the variability. First, for slow variation assumptions, Jeon and Chang used a linearbased model for the channel response [4], whereas Wang and Liu used a polynomial basis adaptative model [5]. One of the best performances is shown by Mostofi's ICI mitigation model [6]. Second, for fast time-varying systems, Hijazi and Ros implemented a Kalman Filter with very attractive results [7].

This work presents an approach to generic channel equalization techniques for OFDM based systems in time variant channels and is organized as follows. Section II describes the mathematical behavior of the channel and Section III introduces a general equalization method based on it. Next, Section IV proposes a general classification for channels in terms of their time variability. Furthermore, in Section V several simulations are carried out to prove that the general equalization methodology works fine and that the channel classification is right. Three general equalization methods are defined based on the theoretical model and are applied to previously defined channel models.

2.2 System Model

The discrete baseband equivalent system model under consideration is described in Figure 1. In the receiver, perfect synchronization time is assumed. First, the transmitter applies an N-point IFFT to a QAM-symbols $\mathbf{[s]}_k$ data block, where *k* represents the subchannel where the symbols have been modulated.

$$[\mathbf{x}]_k = \frac{1}{N}\sum_{k=0}^{N-1} s[k]e^{j2\pi nk/N}, \qquad 0 \leq N-1 \qquad (1)$$

For a theoretical mathematical development the worst case is assumed: the channel varies within one symbol. Hence, the output can be described as follows:

$$[\mathbf{y}]_n = x[n] * h[n,l] + w[n] = \sum_{l=0}^{L-1} h[n,l]x[n-l] + w[n] \quad (2)$$

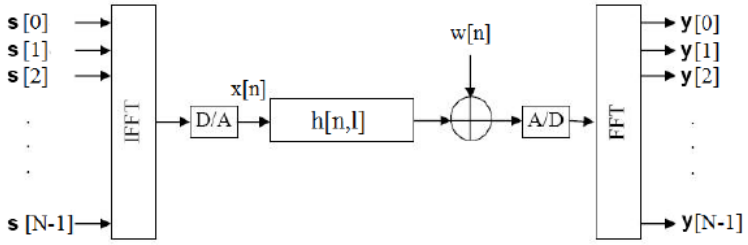

Figure 1: Equivalent baseband system model for OFDM.

The $\mathbf{[w]}_n$ represents the additive white Gaussian noise (AWGN). At the receiver, an N-point FFT is applied to demodulate the OFDM signal. The m^{th} subcarrier output can be represented by:

$$[\mathbf{y}]_m = \sum_{n=0}^{N-1} \left[\sum_{l=0}^{L-1} h[n,l]x[n-l] + w[n] \right] e^{-j2\pi nm/N} \quad (3)$$

After some operations, the expression in (3) can be simplified as a function of $\mathbf{[H]}_{m;k}$, which is the double Fourier transform of the channel impulse response [8], by terms of a convolution:

$$[\mathbf{y}]_m = \frac{1}{N} \sum_{k=0}^{N-1} X[k]H[m-k,k] + W[m] \quad (4)$$

Subsequently, let $\mathbf{[Z]}_{m;k}$ denote the matrix defining the circular-shifted convolution matrix of the expression in (4):

$$[\mathbf{Z}]_{m,k} = H[\langle m-k \rangle_N, k], \qquad 0 \leq m,k \leq N-1 \quad (5)$$

Providing this expression is analysed in depth, the channel matrix $\mathbf{[Z]}_{m;k}$ might be expressed as a sum of two terms. On the one hand, $\mathbf{[Z]}^{ici}$, the **[Z]** matrix diagonal, which is related to the channel attenuation due to the multipath fading. And, on the other hand, $\mathbf{[Z]}^{d}$ which is set as the **[Z]** matrix sub-diagonals, and it is connected to the ICI due to the Doppler effect.

$$[\mathbf{y}]_m = (Z^d[m,k] + Z^{ici}[m,k])X[m] + W[m] \quad (7)$$

It can be shown that each value of $\mathbf{[Z]}^{d}$ in (7) corresponds to the mean of the tap variability for the corresponding channel impulse response path [6].

$$[\mathbf{Z}]^d_{m,k} = diag\{[\mathbf{Z}]_{m,k}]\} = [\mathbf{H}]_{0,m}, \quad 0 \le m \le N-1 \quad (8)$$

$$[\mathbf{Z}]^d_{m,k} = \frac{1}{L}\left[\sum_{l=0}^{L-1} S[0,l]e^{-j2\pi kn/L}\right] \quad (9)$$

where,

$$\frac{1}{L}[\mathbf{S}]_{0,l} = \frac{1}{L}\sum_{n=0}^{N-1} h[n,l] = h^{ave}[l]\ , \qquad 0 \le l \le L-1 \quad (10)$$

Therefore, **[Z]**^{d} can be expressed as the Fourier Transform of the channel tap average:

$$[\mathbf{Z}]^d_{m,k} = diag\{FFT\{[\mathbf{h}]^{ave}_l\}\} = [\mathbf{H}]^{ave}_m \quad (11)$$

$$[\mathbf{y}]_m = H^{ave}[m]X[m] + Z^{ici}[m,k]X[m] + W[m] \quad (12)$$

2.3 General Channel Equalization Methodology

In this section, it is proposed general theoretical methodology for equalization based on the aforementioned mathematical model for both variant and invariant channels (see Figure 3). As it has been proved in (5) when we are dealing with LTV channels the received symbol is affected by a two dimensional channel impulse response instead of the characteristic one dimensional for LTI scenarios. That is to say, in the receiver, a two dimensional equalization method is needed.

Therefore, the CIR (Channel Impulse Response) cannot be directly estimated from the received symbol as the received signal must be pre-processed. Due to this the received symbol ICI term (12), **[Z]**^{ici}, should be completely removed. Then, the symbol impulse response, **[h]**^{sym}, must be estimated minimizing as much as possible the influence of the AWGN. It should be noted that in time-variant scenarios this estimation and the channel response are different since the transmitted signal is affected by a two dimensional CIR. Anyway, **[h]**^{sym} can be calculated as a conventional CIR using the pilot-tones (called comb-type pilot) inserted into each OFDM symbol at the transmitter side. The conventional channel estimation methods consist in estimating the channel at pilot frequencies and next interpolating the channel frequency response. The different methods and their results have already been studied in depth [2][3][9].

Subsequently, we get a N samples length symbol impulse response which has the information of the N^2 samples that complete the actual **[H]** matrix. Hence, at this point those N^2 samples should be estimated from **[h]**^{sym}. As previously mentioned (10), this function is connected to the bidimensional channel impulse response mean by the inverse Fourier Transform. Providing that these mean values match up with the $(N/2)^{th}$ value of the channel impulse response matrix, the estimated impulse response of Q symbols can be grouped, and then interpolated in order to get the signal variation within each symbol (See Figure 2). The interpolation method should be chosen according to the type of time-variability. For example, a linear interpolation should work when the time variability of each path within a symbol is nearly linear.

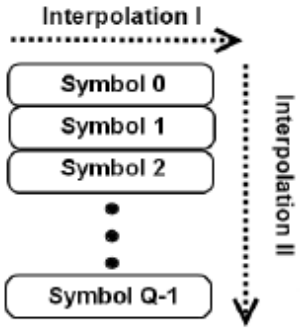

Figure 2: General equalization interpolation dimensions.

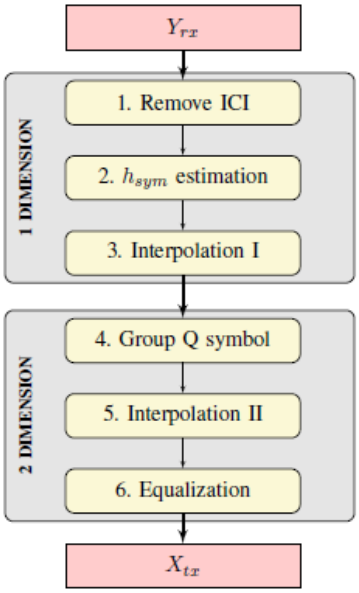

Figure 3: Equivalent General equalization block diagram.

In this way, the two dimensional channel impulse response for each symbol is obtained. Then, before the last bidimensional equalization is performed, each symbol **[Z]** matrix should be calculated using the double Fourier Transform and a circular shift (5). Eventually, the transmitted symbol is obtained equalizing each symbol using this matrix.

2.4 Channel Classification

In the general equalization method explained in the previous sections it has been proved that the channel time variability affects the result accuracy depending on two terms. First, the importance of the noisy term ICI added to the symbol impulse response, and then, the assumption that the received response matches up with the mean of the whole **[H]**. The analysis of these two terms will permit classifying channels into LTI and LTV. Likewise, LTV systems should be considered either slow-varying or rapid-varying. As mentioned before, the channel time variability is related to the relative Doppler frequency change, which indicates the degree of time variation of the CIR within a symbol. This change can be calculated by the ratio of the symbol period Tu to the inverse of the Doppler frequency [6].

First, the inter-carrier interference term, mse_{ici}, is calculated. Its value indicates the weight of the ICI term in the symbol impulse response. Hence, when it is very low it can be assumed that the distortion due to mobility is negligible and the channel should be considered slow-variant.

$$\mathbf{mse}_l^{ici} = E\{(h^{sym}[l] - h^{ave}[l])^*(h^{sym}[l] - h^{ave}[l])\} \quad (13)$$

Before the second error term is calculated, it is assumed that in a previous step the noisy influence due to the AWGN noise and the ICI component has been removed. Afterwards, we calculate, mse_{lin}, which gives the difference between the estimated symbol response (channel response mean value) and the theoretical matrix $(N/2)^{th}$ channel response.

$$\mathbf{mse}_l^{lin} = E\{(h^{ave}[l] - h[N/2, l])^*((h^{ave}[l] - h[N/2, l])\} \quad (14)$$

Therefore, when the $\mathbf{mse}_{lin}$ is low the $\mathbf{[h]}^{ave}$ matches up with the $(N/2)^{th}$ value of the bidimensional impulse response matrix. Then, these channels are considered just as LTV channels with linear time variability and the 5^{th} step interpolation could be done by a linear one. However, when this term is too high the equalization is going to deal with rapid-variant channels. In this type of channel the problem is that another interpolation method is needed and a priori the channel variation within a symbol is unknown.

2.5 Results

To demonstrate the reliability of the proposed general equalization method approach for both LTI and LTV multipath channels, the following simulations were performed. Firstly, a 4QAM-OFDM system with N = 1024 subcarriers is considered, where roughly Lu = 896 of the subcarriers are used for transmitting data symbols. The system also occupies a bandwidth of 10MHz operating in the 890MHz frequency band. The sample period is T_{sample} = 0.1us. Besides, the OFDM symbol has a guard interval with OFDM _G = 1/4 sample periods and there are N_p = N/8 (i.e., Lf=8) equally spaced pilot carriers. In the following simulations, the system will be restricted to a moving terminal with many uniformly distributed scatterers in the close vicinity of the terminal, leading to the typical classical Doppler spectrum [10]. The analyzed channel models are the TU-6 and MR models as recommended by COST 207 [11] and the WING-TV project [12], with parameters shown in the Table 1 and Table 2. Two types of simulations have been carried out. On the one hand, the equalization method weaknesses are analyzed in terms of their steps' mse, and on the other hand, the BER performance of the general method in terms of f_dT_u.

Table 1: TU-6 channel definition

TABLE I
TU-6 CHANNEL DEFINITION.

Tap Number	Delay(μs)	Amplitude (dB)	Doppler Spectrum
1	0	-3	Classical
2	0.2	0	Classical
3	0.5	-2	Classical
4	1.6	-6	Classical
5	2.3	-8	Classical
6	5.0	-10	Classical

Table 2: MR channel definition

TABLE II
MR CHANNEL DEFINITION.

Tap Number	Delay(μs)	Amplitude (dB)	Doppler Spectrum
1	0	0	Gauss1
2	0.5	-1.3	Classical
3	1.0	-3.4	Classical
4	1.8	-6.8	Classical
5	2.5	-10.2	Classical
6	3.1	-12.9	Classical
7	3.9	-16.3	Classical
8	4.8	-19.5	Classical
9	5.5	-21.7	Classical
10	6.4	-23.3	Classical
11	7.0	-24.2	Classical
12	9.0	-25.8	Classical

2.5.1 A. MSE Results

Figure 4 and Figure 5 show the mse^{ici} and mse^{lin} in terms of f_dT_u for TU-6 and MR channels, respectively. It is observed that for both channels the mse evolution is almost the same and that the ICI term can be considered negligible for low fdTu values. That is to say, the channels should be considered slowvariant and this is why the one dimensional equalization works for this type of channels. It is noticed that when the channel variability increases $\mathbf{mse}_{lin}$ can be as important as $\mathbf{mse}_{ici}$. Therefore, as this term represents the linearity of the variation within a symbol, the intersection of the two curves points the place where the channel variation within a symbol is not linear any more, and hence, the channel should be considered rapidvariant.

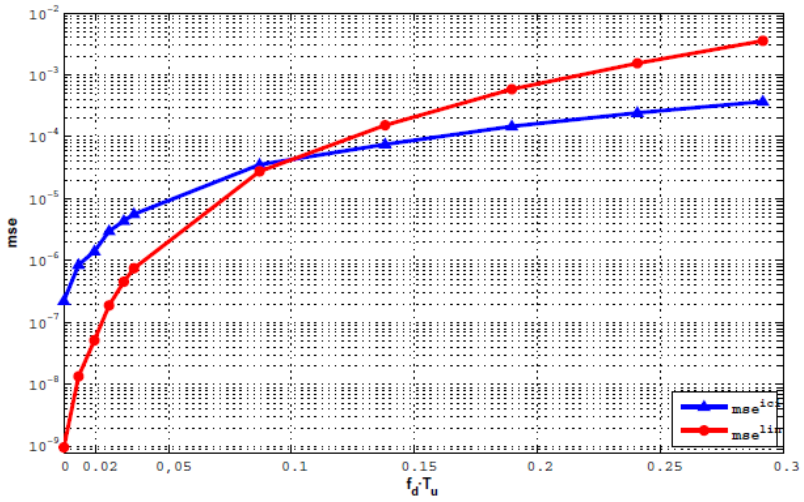

Figure 4: TU-6 Channel mse analysis.

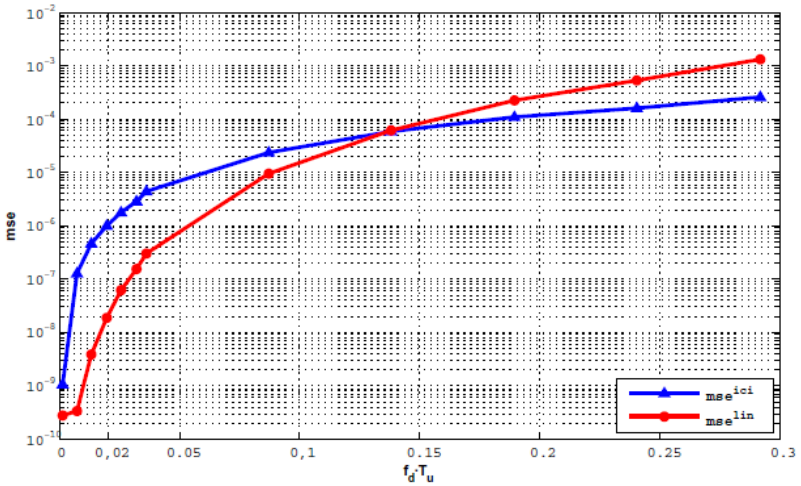

Figure 5: MR Channel mse analysis.

2.5.2 B. BER Results

Figure 6 and Figure 7 show the performance of the equalization method proposal in terms of f_dT_u for TU-6 and MR channels, respectively. Indeed, three cases of the general equalization method are considered based on the theoretical **[Z]** matrix described in (6). The first one, *1D method*, assumes that the time variability is not so important and **[Z]** is assumed to be a diagonal matrix representing the distortion due to multipath. In the second one, *lin method*, it is assumed a lineal variation within a symbol, and therefore, it is enough to know two values of each channel tap, whereas the other ones are interpolated to obtain the whole matrix. Nevertheless, in the third, *2D method*, all the **[Z]** matrix values are used.

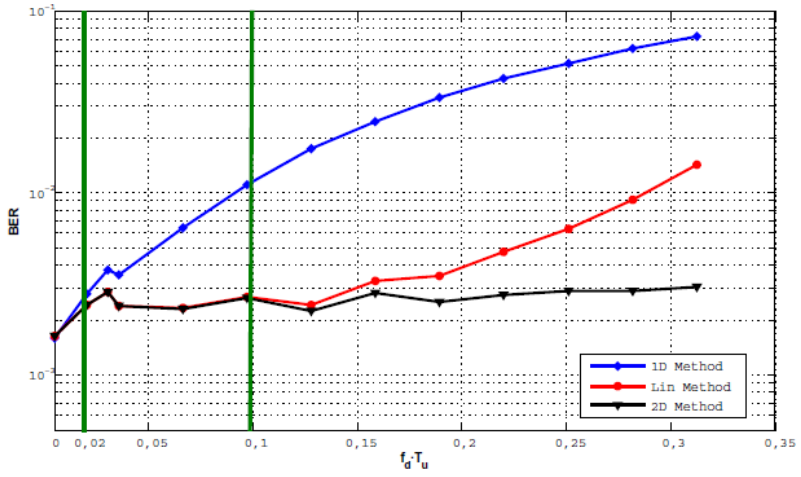

Figure 6: General method equalization algorithm for fdTu in TU-6 channels.

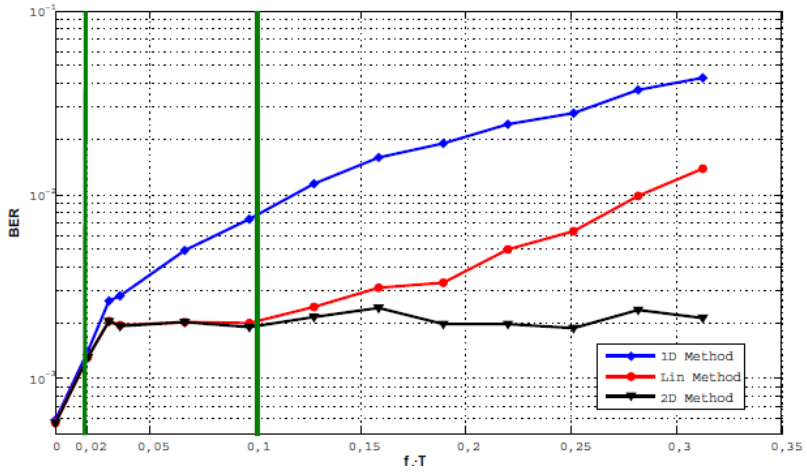

Figure 7: General method equalization algorithm for fdTu in MR channels.

As it was expected when the channel are slow-variant, up to $f_dT_u = 0.02$, the three cases show practically the same results, and therefore, in terms of simplicity the one dimension equalization is enough. But, when the time variability within a symbol starts to be important, $f_dT_u > 0.02$ the one dimension equalization performance is very poor. Hence, is clearly shown that from $f_dT_u = 0.02$ until $f_dT_u = 0.1$, the *lin* and *2D* equalizations should be used. Eventually, when the channel variability within a symbol arises to a non-linear form ($f_dT_u > 0.1$) the *2D method* is the only one which remains constant, while the linear method results worsen. What is more, these channel classifications are reinforced with the Section V **mse** results. These statements are valid for both MR and TU6 channel, and the linearity variation within variant channels boundary, coincides with the limit defined for other equalization methods [6][13].

Figure 8 and Figure 9 give the BER performance of the general equalization, 2D method, compared to conventional one, *1D method*, for both the TU-6 and MR channels. They are tested for $f_dT_u = 0.01$ and for $f_dT_u = 0.1$ when the **[Z]** has been perfectly recovered. It is shown that for slow-variant channels both methods work fine. Anyway, when the system is dealing with variant channels, the one dimensional equalization method performance is very poor, while the two dimensional method is nearly the same as for slow-variant channel. As expected, both improve with the SNR.

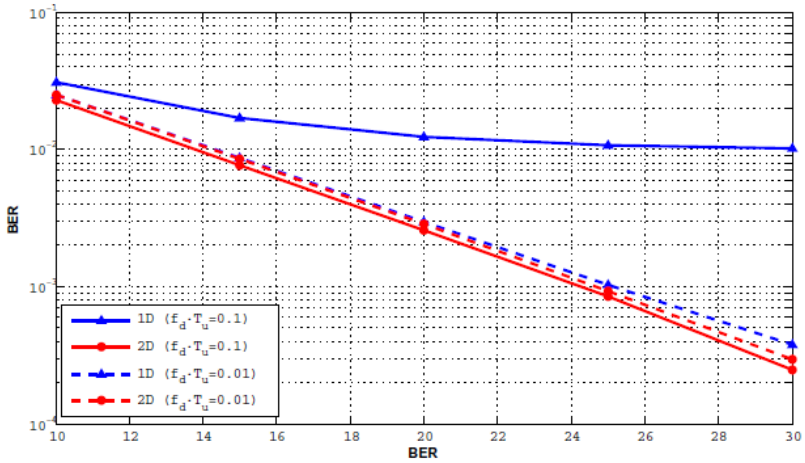

Figure 8: Comparison of TU-6 BER for fdTu=0.01 and fdTu=0.1.

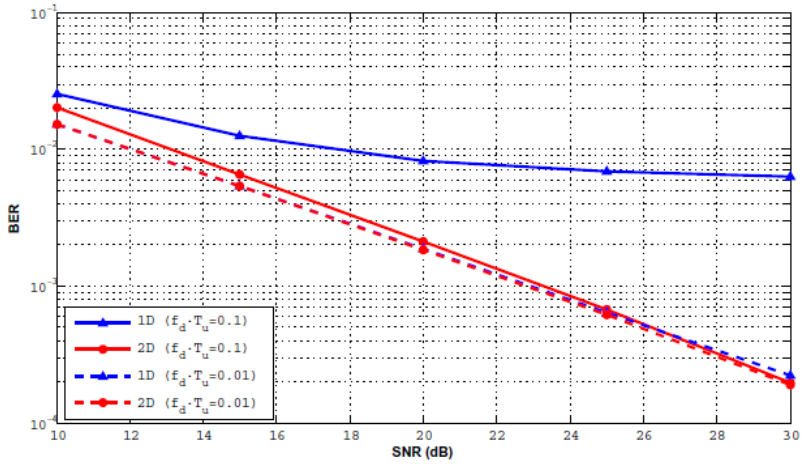

Figure 9: Comparison of MR BER for fdTu=0.01 and fdTu=0.1.

2.6 Conclusions

In this work, we have presented a general equalization method for both LTI and LTV channels. We have proved its reliability based on a theoretical analysis and some simulation results. Besides, using this mathematical analysis a general channel classification in terms of the time variability is presented. Up to f_dT_u = 0.02 the channel variation could be considered negligible, and therefore, these channels are conceived as slow variant channels. Afterwards, from this point to f_dT_u = 0.1 the channels are considered time variant, as the variation within a symbol is linear. Finally, when the variation is higher than f_dT_u > 0.1 the channel is rapid variant.

3 A SHUFFLED ITERATIVE RECEIVER FOR THE DVB-T2 BIT-INTERLEAVED CODED MODULATION: ARCHITECTURE DESIGN, IMPLEMENTATION AND FPGA PROTOTYPING

3.1 Simplified Decoding of High Diversity Multi-Block Space-Time (MB-STBC) Codes

This section presents a simplified detection algorithm, suitable for hardware implementation, for a Space-Time Code (STC) proposed by Telecom Bretagne as a response to the DVB-NGH Call for Technology. The performance of this STBC code is reported in the MIMO section of Deliverable D2.3 "Final report on advanced concepts for DVB-NGH".

3.1.1 Encoding of the proposed MB-STBC

The proposed STBC calls for a 2x4 matrix of the following form:

$$\mathbf{X} = \begin{bmatrix} s_1'' & s_3'' & s_5'' & s_7'' \\ s_2'' & s_4'' & s_6'' & s_8'' \end{bmatrix} \quad (1)$$

This structure allows the transmission of 8 signals $s_1'' \cdots s_8''$ through 2 antennas over 4 time slots. The first (second) row of the matrix contains the 4 signals successively sent through the first (second) transmit antenna.

We assume that the channel coefficients are constant during the two first and the two last time slots. In other words, a quasi-orthogonal STBC structure spread over 4 slots. In a multi-carrier transmission system, this property can be obtained by transmitting the signals of columns 1 and 2 (respectively of columns 3 and 4) of **X** over adjacent subcarriers while the signals of columns 1 (respectively 2) and 3 (respectively 4) are transmitted over distant subcarriers.

Two different channel matrices have then to be considered: **H** for the transmission of signals in columns 1 and 2 and H' for the transmission of signals in columns 3 and 4:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \text{ and } \mathbf{H}' = \begin{bmatrix} h'_{11} & h'_{12} \\ h'_{21} & h'_{22} \end{bmatrix} \quad (2)$$

Let us consider 8 modulation symbols $s_1 \ldots s_8$ taken from an *M*-order 2-dimensional constellation **C**, where in-phase *I* and quadrature *Q* components are correlated. This correlation can be obtained by applying a rotation to the original constellation. The rotation angle should be chosen such that every constellation point is uniquely identifiable on each component axis separately. This is equivalent to the first step performed for SSD [14]. The representation of s_i in the complex plane is given by, $s_i = I_i + jQ_i$, $i = 1 \cdots 8$. The proposed construction of **X** involves the application of a two-step process:

Step 1: the first step consists in defining two subsets S_1' and S_2' of modified symbols s_i' obtained from *I* and *Q* components belonging to different symbols s_i. Each subset must only contain one component of each symbol s_i of **C**. For instance:

$S_1' = \{s_1', s_2', s_3', s_4'\}$ and $S_2' = \{s_5', s_6', s_7', s_8'\}$

where $\begin{matrix} s'_1 = I_1 + jQ_7 \\ s'_2 = I_2 + jQ_8 \\ s'_3 = I_3 + jQ_5 \\ s'_4 = I_4 + jQ_6 \end{matrix}$ and $\begin{matrix} s'_5 = I_5 + jQ_3 \\ s'_6 = I_6 + jQ_4 \\ s'_7 = I_7 + jQ_1 \\ s'_8 = I_8 + jQ_2 \end{matrix}$.

Symbols s'_i belong to an extended constellation **C'** of size M^2.

Step 2: the symbols $s''_1 \cdots s''_8$ transmitted by **X** are defined as

$$\begin{matrix} s''_1 = as'_1 + bs'_2 \\ s''_2 = as'_3 + bs'_4 \\ s''_3 = -cs'^*_3 - ds'^*_4 \\ s''_4 = cs'^*_1 + ds'^*_2 \end{matrix} \text{ and } \begin{matrix} s''_5 = as'_5 + bs'_6 \\ s''_6 = as'_7 + bs'_8 \\ s''_7 = -cs'^*_7 - ds'^*_8 \\ s''_8 = cs'^*_5 + ds'^*_6 \end{matrix}.$$

where *s** represents the complex conjugate of *s*.

a, *b*, *c* and *d* are complex-valued parameters of the STBC. Signals *s*'' belong to the STBC constellation signal set **C''** different from **C'**.

3.1.2 Simplified decoding of the MB-STBC code

The proposed MB-STBC code enjoys a structure that enables a simplified detection. Indeed, inspired by the decoding process in [15], the decoding complexity can be greatly simplified without the need for a sphere decoder [16]. If we denote by r_k^j the signal received by the j^{th} reception antenna, j = 1, 2, during time slot k, where $k = 1 \ldots 4$.

The four signals successively received by antenna 1 can be written as:

$$\begin{aligned} r_1^1 &= h_{11}\left[a(I_1 + jQ_7) + b(I_2 + jQ_8)\right] \\ &\quad + h_{12}\left[a(I_3 + jQ_5) + b(I_4 + jQ_6)\right] + n_1^1 \end{aligned} \tag{3}$$

$$\begin{aligned} r_2^1 &= h_{11}\left[-c(I_3 - jQ_5) - d(I_4 - jQ_6)\right] \\ &\quad + h_{12}\left[c(I_1 - jQ_7) + d(I_2 - jQ_8)\right] + n_2^1 \end{aligned} \tag{4}$$

$$\begin{aligned} r_3^1 &= h'_{11}\left[a(I_5 + jQ_3) + b(I_6 + jQ_4)\right] \\ &\quad + h'_{12}\left[a(I_7 + jQ_1) + b(I_8 + jQ_2)\right] + n_3^1 \end{aligned} \tag{5}$$

$$\begin{aligned} r_4^1 &= h'_{11}\left[-c(I_7 - jQ_1) - d(I_8 - jQ_2)\right] \\ &\quad + h'_{12}\left[c(I_5 - jQ_3) + d(I_6 - jQ_4)\right] + n_4^1 \end{aligned} \tag{6}$$

Simplified decoding is possible under the condition that the *I* and *Q* components of any s_i constellation symbol are mapped to two different *s*' symbols who are multiplied by the same STBC parameter *a*, *b*, *c* or *d*. This constraint is respected in the structure of the STBC matrix **X.** Therefore, by re-arranging equations (3) to (6) we obtain the following terms y_k^j:

$$\begin{aligned} y_1^1 &= r_1^1 - b\left[h_{11}(I_2 + jQ_8) + h_{12}(I_4 + jQ_6)\right] \\ &= a\left[h_{11}(I_1 + jQ_7) + h_{12}(I_3 + jQ_5)\right] + n_1^1 \end{aligned} \tag{7}$$

$$\begin{aligned} y_2^1 &= r_2^1 - d\left[h_{12}(I_2 - jQ_8) - h_{11}(I_4 - jQ_6)\right] \\ &= c\left[h_{12}(I_1 - jQ_7) - h_{11}(I_3 - jQ_5)\right] + n_2^1 \end{aligned} \tag{8}$$

$$\begin{aligned} y_3^1 &= r_3^1 - b\left[h'_{11}(I_6 + jQ_4) + h'_{12}(I_8 + jQ_2)\right] \\ &= a\left[h'_{11}(I_5 + jQ_3) + h'_{12}(I_7 + jQ_1)\right] + n_3^1 \end{aligned} \tag{9}$$

$$\begin{aligned} y_4^1 &= r_4^1 - d\left[h'_{12}(I_6 - jQ_4) - h'_{11}(I_8 - jQ_2)\right] \\ &= c\left[h'_{12}(I_5 - jQ_3) - h'_{11}(I_7 - jQ_1)\right] + n_4^1 \end{aligned} \tag{10}$$

In equations (7) to (10), the first line terms only depend on the *I* and *Q* components of even symbols *s*. Vice-versa, second line terms depend solely on odd symbols. Therefore, applying a detection conditioned by the knowledge of even terms is possible. In other words, for a loop on all possible values for $S_2 = I_2 + jQ_2$, $S_4 = I_4 + jQ_4$, $S_6 = I_6 + jQ_6$ and $S_8 = I_8 + jQ_8$ (for a total of M^4 terms where *M* represents the order of the constellation *s*) intermediate Z_k terms can be computed as follows:

$$Z_1 = \frac{h_{11}^* y_1^1 + h_{21}^* y_1^2}{a} + \frac{h_{12} y_2^{1*} + h_{22} y_2^{2*}}{c^*} \tag{11}$$

$$Z_2 = \frac{h_{12}^* y_1^1 + h_{22}^* y_1^2}{a} - \frac{h_{11} y_2^{1*} + h_{21} y_2^{2*}}{c^*} \tag{12}$$

$$Z_3 = \frac{h_{11}'^* y_3^1 + h_{21}'^* y_3^2}{a} + \frac{h'_{12} y_4^{1*} + h'_{22} y_4^{2*}}{c^*} \tag{13}$$

$$Z_4 = \frac{h_{12}'^* y_3^1 + h_{22}'^* y_3^2}{a} - \frac{h'_{11} y_4^{1*} + h'_{21} y_4^{2*}}{c^*} \tag{14}$$

By properly combining Z_k terms, we obtain:

$$\begin{aligned} &\mathrm{Re}\{Z_1\} + j\,\mathrm{Im}\{Z_4\} = \left(|h_{11}|^2 + |h_{12}|^2 + |h_{21}|^2 + |h_{22}|^2\right) I_1 \\ &+ j\left(|h'_{11}|^2 + |h'_{12}|^2 + |h'_{21}|^2 + |h'_{22}|^2\right) Q_1 + \mathrm{Re}\{N_1\} + j\,\mathrm{Im}\{N_4\} \end{aligned} \tag{15}$$

$$\begin{aligned} &\mathrm{Re}\{Z_2\} + j\,\mathrm{Im}\{Z_3\} = \left(|h_{11}|^2 + |h_{12}|^2 + |h_{21}|^2 + |h_{22}|^2\right) I_3 \\ &+ j\left(|h'_{11}|^2 + |h'_{12}|^2 + |h'_{21}|^2 + |h'_{22}|^2\right) Q_3 + \mathrm{Re}\{N_2\} + j\,\mathrm{Im}\{N_3\} \end{aligned} \tag{16}$$

$$\begin{aligned} &\mathrm{Re}\{Z_3\} + j\,\mathrm{Im}\{Z_2\} = \left(|h'_{11}|^2 + |h'_{12}|^2 + |h'_{21}|^2 + |h'_{22}|^2\right) I_5 \\ &+ j\left(|h_{11}|^2 + |h_{12}|^2 + |h_{21}|^2 + |h_{22}|^2\right) Q_5 + \mathrm{Re}\{N_3\} + j\,\mathrm{Im}\{N_2\} \end{aligned} \tag{17}$$

$$\begin{aligned} &\mathrm{Re}\{Z_4\} + j\,\mathrm{Im}\{Z_1\} = \left(|h'_{11}|^2 + |h'_{12}|^2 + |h'_{21}|^2 + |h'_{22}|^2\right) I_7 \\ &+ j\left(|h_{11}|^2 + |h_{12}|^2 + |h_{21}|^2 + |h_{22}|^2\right) Q_7 + \mathrm{Re}\{N_4\} + j\,\mathrm{Im}\{N_1\} \end{aligned} \tag{18}$$

With the noise terms N_k being:

$$N_1 = \frac{h_{11}^* n_1^1 + h_{21}^* n_1^2}{a} + \frac{h_{12} n_2^{1*} + h_{22} n_2^{2*}}{c^*} \quad N_2 = \frac{h_{12}^* n_1^1 + h_{22}^* n_1^2}{a} - \frac{h_{11} n_2^{1*} + h_{21} n_2^{2*}}{c^*} \quad N_3 = \frac{h_{11}'^* n_3^1 + h_{21}'^* n_3^2}{a} + \frac{h_{12}' n_4^{1*} + h_{22}' n_4^{2*}}{c^*}$$

$$N_4 = \frac{h_{12}'^* n_3^1 + h_{22}'^* n_3^2}{a} - \frac{h_{11}' n_4^{1*} + h_{21}' n_4^{2*}}{c^*}$$

Equations (15) to (18) show that the combinations of Z_k dependent terms are each a function of only one $s_i = I_i + jQ_i$ symbol. Therefore a simple linear detection can be performed separately on all symbols in the same loop since every I_i and Q_i couple is unique. In addition, the diversity of 8 is clearly observed since the I and Q components of every symbol depend on 4 different channel coefficients. Therefore, since SSD is applied, every complex s_i signal enjoys an overall diversity of 8.

The detection of odd symbols on the second antenna is similar to the first antenna. For the joint detection of even symbols, the following distance should be minimized:

$$\begin{aligned} D(s_2, s_4, s_6, s_8) = & \left| y_1^1 - a\left[h_{11}(I_1 + jQ_7) + h_{12}(I_3 + jQ_5)\right]\right|^2 \\ & + \left| y_2^1 - c\left[h_{12}(I_1 - jQ_7) - h_{11}(I_3 - jQ_5)\right]\right|^2 \\ & + \left| y_3^1 - a\left[h_{11}'(I_5 + jQ_3) + h_{12}'(I_7 + jQ_1)\right]\right|^2 \\ & + \left| y_4^1 - c\left[h_{12}'(I_5 - jQ_3) - h_{11}'(I_7 - jQ_1)\right]\right|^2 \\ & + \left| y_1^2 - a\left[h_{21}(I_1 + jQ_7) + h_{22}(I_3 + jQ_5)\right]\right|^2 \\ & + \left| y_2^2 - c\left[h_{22}(I_1 - jQ_7) - h_{21}(I_3 - jQ_5)\right]\right|^2 \\ & + \left| y_3^2 - a\left[h_{21}'(I_5 + jQ_3) + h_{22}'(I_7 + jQ_1)\right]\right|^2 \\ & + \left| y_4^2 - c\left[h_{22}'(I_5 - jQ_3) - h_{21}'(I_7 - jQ_1)\right]\right|^2 \end{aligned} \quad (19)$$

The distance $D(s_2, s_4, s_6, s_8)$ of equation (19) can be directly computed from terms y_k^j (which depend on s_2, s_4, s_6 and s_8) of equations (7) to (10) and by replacing the I and Q components of odd constellation symbol terms by their detected values from equations (15) to (18). Since $D(s_2, s_4, s_6, s_8)$ should be computed for all possible combinations of even constellation symbols, **the total number of computed terms is in the order of M^4**.

Note that the simplified detection does not depend on the choice of the STBC parameters *a*, *b*, *c* and *d*. These should be chosen depending on the rank, determinant, and shaping considerations.

3.2 A shuffled iterative receiver architecture for Bit-Interleaved Coded Modulation systems

This section presents the design and implementation by Telecom Bretagne of an efficient shuffled iterative receiver for the second generation of the terrestrial digital video broadcasting standard DVB-T2. The scheduling of an efficient message passing algorithm with low latency between the demapper and the LDPC decoder represents the main contribution of this study. The design and the FPGA prototyping of the resulting shuffled iterative BICM receiver are then described. Architecture complexity and measured performance validate the potential of iterative receiver as a practical and competitive solution for the DVB-T2 standard.

3.2.1 Introduction

The second generation of terrestrial video broadcasting standard (DVB-T2) was defined in 2008. The key motivation behind developing a second generation is to offer high definition television services. One of the key technologies in DVB-T2 is a new diversity technique called rotated constellations [17]. This concept can significantly improve the system performance in frequency selective terrestrial channels thanks to Signal Space Diversity (SSD) [18]. Indeed, SSD doubles the diversity order of the conventional BICM schemes and improves the performance in fading channels especially for high coding rates [14]. When using conventional QAM constellations, each signal component, in-phase (I) or quadrature (Q), carries half of the binary information held in the signal. Thus, when a constellation signal is subject to a fading event, I and Q components fade identically. In the case of severe fading, the information transmitted on I and Q components suffers an irreversible loss. The very simple underlying idea in SSD involves transmitting the whole binary content of each constellation signal twice and separately yet without loss of spectral efficiency. Actually, the two projections of the signal are sent separately in two different time periods, two different OFDM subcarriers or two different antennas, in order to benefit from time or frequency or antenna diversity respectively. When concatenated with Forward Error Correcting (FEC) codes, simulations [14] show that rotated constellation provides up to 0.75 dB gain over conventional QAM on wireless channels. In order to achieve additional improvement in performance, iterations between the decoder and the demapper (BICM-ID) can be introduced. BICM-ID with an outer LDPC code was investigated for different DVB-T2 transmission scenarios [14]. It is shown that an iterative processing associated with SSD can provide additional error correction capability reaching more than 1.0 dB over some types of channels. Thanks to these advantages, BICM-ID has been recommended in the DVB-T2 implementation guidelines [19] as a candidate solution to improve the performance at the receiver.

However, designing a low complexity high throughput iterative receiver remains a challenging task. One major problem is the computation complexity at both the rotated QAM demapper and at the LDPC decoder. In [20], a flexible demapper architecture for DVB-T2 is presented. Lowering complexity is achieved by decomposing the rotated constellation into two-dimensional sub-regions in signal space. In [21], a novel complexity-reduced LDPC decoder architecture based on the vertical layered schedule [22] and the normalized Min-Sum (MS) algorithm is detailed. It closely approaches the full complexity message passing decoding performance provided in the implementation guidelines of the DVB-T2 standard. Another critical problem is the additional latency introduced by the iterative process at the receiver side. Iterative Demapping (ID), especially due to interleaver and de-interleaver, imposes a latency that can have an important impact on the whole receiver. Therefore, a more efficient information exchange method between the demapper and the decoder has to be applied. We propose to extend the recent shuffled decoding technique introduced in the turbo-decoding field [23] to avoid long latency. The basic idea of shuffled decoding technique is to execute all component decoders in parallel and to exchange extrinsic information as soon as it is available. It forces however a vertical layered schedule for the LDPC decoder as explained in [22]. In this context, processing one frame can be decomposed into multiple parallel smaller sub-frame processing each having a length equal to the parallelism level. While having a comparable computational complexity as the standard iterative schedule, the receiver with a shuffled iterative schedule enjoys a lower latency. However, such a parallel processing requires good matching between the demapping and the decoding processors in order to guarantee a high throughout pipeline architecture. This calls for an efficient message passing between these two types of processors.

Two main contributions are presented in this work. The first is the investigation of different schedules for the message passing algorithm between the decoder and the demapper. The second represents the design and FPGA prototyping of a shuffled iterative bit-interleaved coded modulation receiver. Section 3.2.2 summarizes the basic principles of the BICM-ID with SSD adopted in DVB-T2. Then, a shuffled iterative

receiver for BICM-ID systems is detailed in Section 3.2.3. In Section 3.2.4 the characteristics of efficient iterative receiver architecture are presented. Finally, an implementation of the iterative BICM receiver and its experimental setup onto FPGA device are given in Section 3.2.5.

3.2.2 BICM-ID system description

The BICM system is described in Figure 10. At the transmitter side, the messages **u** are encoded as the codeword **c**. Afterwards, this codeword **c** is interleaved by π and becomes the input sequence **v** of the mapper. At each symbol time *t*, *m* consecutive bits of the interleaved sequence **v** are mapped into complex symbol x_t. At the receiver side, the demapper calculates a two-dimensional squared Euclidean distance to obtain the bit LLR $\hat{v}_t^i$ of the i^{th} bit of symbol v_t. These demapped LLRs are then de-interleaved and used as inputs of the decoder. The extrinsic information is finally generated by the decoder and fed back to the demapper for iterative demapping. The SSD introduces two modifications to the classical BICM system shown in Figure 10. The classical QAM constellation is rotated by a fixed angle α. Its Q component is delayed for *d* symbol periods. Therefore, the in-phase and quadrature components of the classical QAM constellation are sent at two different time periods, doubling the constellation diversity of the BICM scheme. When a severe fading occurs, one of the components is erased and the corresponding LLRs could be computed from the remaining component. The channel model used to simulate and emulate the effect of erasure events is a modified version of the classical Rayleigh fading channel. More information about this model is given in [20].

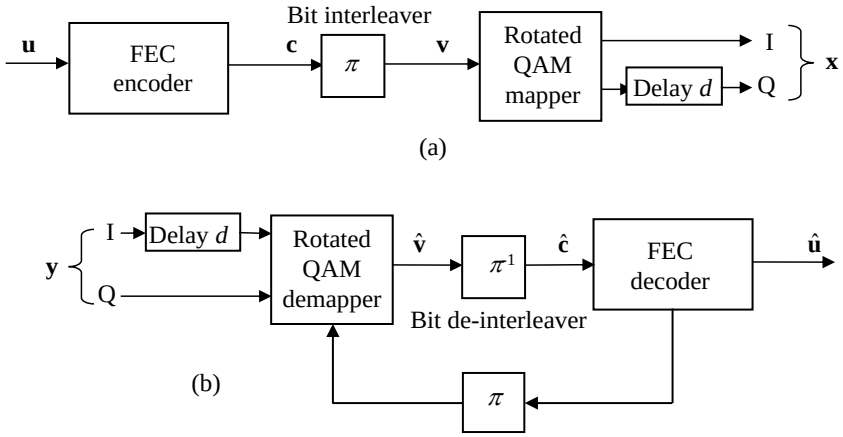

Figure 10: (a) The BICM with SSD transmitter (b) Conventional BICM-ID receiver.

A large set of transmitter configurations based on BICM system has been adopted into the DVB-T2 standard. This wide choice is motivated by the sheer nature of a broadcast network. It should be able to adapt to different geographical locations characterized by different terrain topologies. In the context of DVB-T2, the DVB-S2 LDPC code (an Irregular Repeat Accumulate -IRA- code) was adopted as FEC code. An IRA code is characterized by a parity check matrix composed of two submatrices: a sparse sub-matrix and a staircase lower triangular sub-matrix. Moreover, periodicity has been introduced in the matrix design in order to reduce storage requirements. Two different frame lengths (16200 bits and 64800 bits) and a set of different code rates (1/2, 3/5, 2/3, 3/4, 4/5 and 5/6) are supported. A blockwise bit interleaver and a bit to constellation symbol multiplexer is applied before mapping except for QPSK. Eight different Gray mapped constellations with and without rotation are also supported by the standard, ranging from QPSK to 256-QAM.

3.2.3 A shuffled iterative receiver for DVB-T2

As previously explained, a major challenge in designing iterative receiver is to reduce the computation complexity of the different parts of the receiver. In order to do this, the demapping and decoding algorithms have to be derived to take hardware limitations into account.

3.2.3.1 The rotated demapping algorithm

For Gray-mapped QAM constellations, the demapper calculates two-dimensional Euclidean distance for the computation of the LLR $\hat{v}_t^i$ related to the i^{th} bit of v_t. The resulting $\hat{v}_t^i$ becomes:

$$\hat{v}_t^i = \sum_{x_t \in \chi_0^i} \oplus \left\{ -\frac{D_{euc}(x_t)}{\sigma_w^2} + \sum_{j=0, j\neq i, b_j=0}^{m-1} ext_j^{(t)} \right\} - \sum_{x_t \in \chi_1^i} \oplus \left\{ -\frac{D_{euc}(x_t)}{\sigma_w^2} + \sum_{j=0, j\neq i, b_j=0}^{m-1} ext_j^{(t)} \right\} \tag{20}$$

where $D_{euc}(x_t)$ is the square of the Euclidean distance between the constellation point and the equalized observation, *i.e*,

$$D_{euc}(x_t) = \left[\rho_{t-d} (y_{eq,t-d}^I - x_{t-d}^I) \right]^2 + \left[\rho_t (y_{eq,t}^Q - x_t^Q) \right]^2 \tag{21}$$

the operator $\oplus$ denotes the Jacobian logarithm, *i.e.*,

$$x \oplus y = \begin{cases} \max(x,y) + \log\left(1 + \exp\left(-|x-y|\right)\right), \text{ if } |x-y| \leq 5 \\ \max(x,y) + \log\left(1 + \exp(-5)\right), \text{ else} \end{cases} \tag{22}$$

$ext_j^{(t)}$ is the *a priori* information of the i^{th} mapping bit b^i of the symbol x_t provided by the decoder after the first iteration. $y_{eq,t-d}^I$ and $y_{eq,t}^Q$ respectively represent the in-phase and quadrature components of the equalized complex symbol $y_{eq,t}$. ρ_t is a scalar representing the channel attenuation at time *t*. χ_b^i represents the subset of constellation symbols with i^{th} bit $b^i = b$, $b \in \{0,1\}$. σ_w^2 is the Additive White Gaussian Noise (AWGN) variance.

To reduce the computation complexity of (20), a sub-region selection algorithm [20] is proposed to avoid a complete search of signals in the constellation plane. However, when iterative processing is considered, this algorithm becomes greatly sub-optimal since the selected region may not contain the minimum Euclidean distance for the extrinsic information. Therefore, in this work the *Max-log* approximation represents the only applied demapping simplification.

3.2.3.2 A vertical layered decoding scheme using a normalized

Min-Sum (MS) algorithm LDPC codes can be efficiently decoded using the Belief Propagation (BP) algorithm. This algorithm operates on the bipartite graph representation of the code by iteratively exchanging messages between the variable and check nodes along the edges of the graph. The schedule defines the order of passing messages between all the nodes of the bipartite graph.

Since a bipartite graph contains some cycles, the schedule directly affects convergence rate of the algorithm and hence its computational complexity. Efficient layered schedules have been proposed in literature [22]. Indeed, the parity check matrix can be viewed as a horizontal or a vertical layered graph decoded sequentially. Decoding iteration is then split into sub-layer iterations. In [21], we have detailed a normalized MS decoder architecture based on a Vertical Shuffled Schedule (VSS). The proposed VSS Min-Sum (VSS MS) introduces only a small penalty with respect to a VSS using a BP algorithm while greatly reducing

decoding computational complexity. However, in the context of a BICM-ID receiver, the VSS MS algorithm introduces an additional penalty and therefore reduces the expected performance gain. The main simplification in the MS algorithm is that the check node update is replaced by a selection of the minimum input value. In order to increase the accuracy of the check node processing, it is also possible to select more than two minimum input values. In our case, we have considered three minimum input values for the check node processing. It is denoted by VSS MS3 algorithm in the rest of this paper. According to our investigations, the VSS MS3 algorithm offers the best compromise between Bit Error Rate (BER) performance and decoding computational complexity for a BICM-ID receiver.

3.2.3.3 A joint algorithm for a shuffled iterative process

Iterative receiver hardware latency is often seen as a brake for their use in practical systems. The fact that data are treated several times by rotated demapping and FEC decoding imposes a long delay before delivering decoded bits. Consequently, the global scheduling of an iteration has to be optimized to limit latency of the receiving process. In order to address this issue, we propose a vertical shuffle scheduling for the joint QAM demapping and LDPC decoding. The shuffled demapping and decoding algorithm is summarized in *Algorithm1*. It is applied onto groups of Q symbols. First, a demapping process is applied to estimate Q LLR values. Then, the decoding process is split into four tasks: check node processing, variable node processing, variable node update and check node update. Both steps are repeated until the maximum iteration number is achieved or a codeword has been found. The main advantage of such a scheduling is the decrease of BICM-ID scheme latency. It also leads to a decrease in the number of required iterations for similar BER performance.

***Algorithm 1*: Shuffled Parallel Demapping and Decoding Algorithm**

Initialization

$Q \in [1, 45, 90, 120, 180, 360], n \in [1, Q]$ and $i = \pi(n)$

repeat

$t = t + 1$

Demapping part

for all *i* do

$$\hat{v}_t^i \approx \max_{x_t \in \chi_0^i} \left\{ -\frac{1}{\sigma_w^2} D_{euc}(x_t) + \sum_{j=0, j\neq i, b_j=0}^{m-1} ext_j^{(t)} \right\} - \max_{x_t \in \chi_1^i} \left\{ -\frac{1}{\sigma_w^2} D_{euc}(x_t) + \sum_{j=0, j\neq i, b_j=0}^{m-1} ext_j^{(t)} \right\}$$

end for

Decoding part

for all *n* do

Check node processing

$$E_{mn}^{(t)} = \begin{cases} = 0 \text{ for } t = 1 \\ = \begin{cases} \alpha_m . \eta . sgn\left(T_{mn}^{(t-1)}\right) . M_n^1 \; if \; n = P_m^0 \\ \alpha_m . \eta . sgn\left(T_{mn}^{(t-1)}\right) . M_n^0 \text{ else} \end{cases} \text{for } t > 1 \end{cases}$$

Variable node processing

$$LLR_n = \hat{v}_t^i, \quad \text{where } n = \pi^{-1}(i)$$

$$T_n^{(t)} = \begin{cases} LLR_n, & t = 1 \\ LLR_n + \sum_{m \in M(n)} E_{mn}{}^{(t)}, & \text{else} \end{cases}$$

$$T_{mn}^{(t)} = T_n^{(t)} - E_{mn}^{(t)}$$

Variable node updating

$$ext_n^{(t)} = T_n^{(t)} - LLR_n$$

Check node updating

$$\alpha_m = \alpha_m \cdot \operatorname{sgn}\left(T_{mn}^{(t-1)}\right) \cdot \operatorname{sgn}\left(T_{mn}^{(t)}\right), \quad m \in M(n)$$

$$\begin{cases} M_m^0 = \min_{1\text{st}}\left(\left|T_{mn}^{(t)}\right|, \left|T_{mk'}^{(t-1)}\right|\right), & P_m^0 = \operatorname{index}\left(M_m^0\right) \\ M_m^1 = \min_{2\text{nd}}\left(\left|T_{mn}^{(t)}\right|, \left|T_{mk'}^{(t-1)}\right|\right), & P_m^1 = \operatorname{index}\left(M_m^1\right) \\ M_m^2 = \min_{3\text{rd}}\left(\left|T_{mn}^{(t)}\right|, \left|T_{mk'}^{(t-1)}\right|\right), & P_m^2 = \operatorname{index}\left(M_m^2\right) \end{cases}$$

where $k' \in N(m) \setminus n$

end for

until $t \leq t_{max}$ or convergence to a codeword is achieved.

The decoded bits are estimated through $sign(T_n^{(i)})$

Several possible message passing schedules between the decoder and the demapper can be proposed. They correspond to the different parallelism combinations between the partial update strategies at the demapper and the decoder process. Schedules under consideration in our study, called *A* and *B*, are based on a VSS decoding process with parallelism of 90. In other words, 90 variable nodes get updated and generate 90 extrinsics that are fed back to up to 90 demappers. If all bits originate from different symbols, then the processing requires 90 demappers working in parallel. This clearly represents a worst case processing scenario. The difference between schedule *A* and schedule *B* is in the number of the LLRs that is equal to $90.\log_2(M)$ and 90, respectively. Simulations have been carried out for both schedules. A comparison of simulated BER performance for rotated 256-QAM over a fading channel with 15 % of erasures (DVB-T2 64K LDPC, rate R=4/5) is given in Figure 11. There is around 1.2 dB performance improvement @ 10^{-4} of BER for the iterative floating point VSS BP receiver when compared to the non-iterative receiver. In a BICM-ID context, the proposed VSS MS3 receiver entailed a small penalty of 0.3 dB with respect to VSS BP. In both cases, schedules *A* and *B* have similar performance. Note that we have chosen schedule *B* for the design of our iterative receiver architecture.

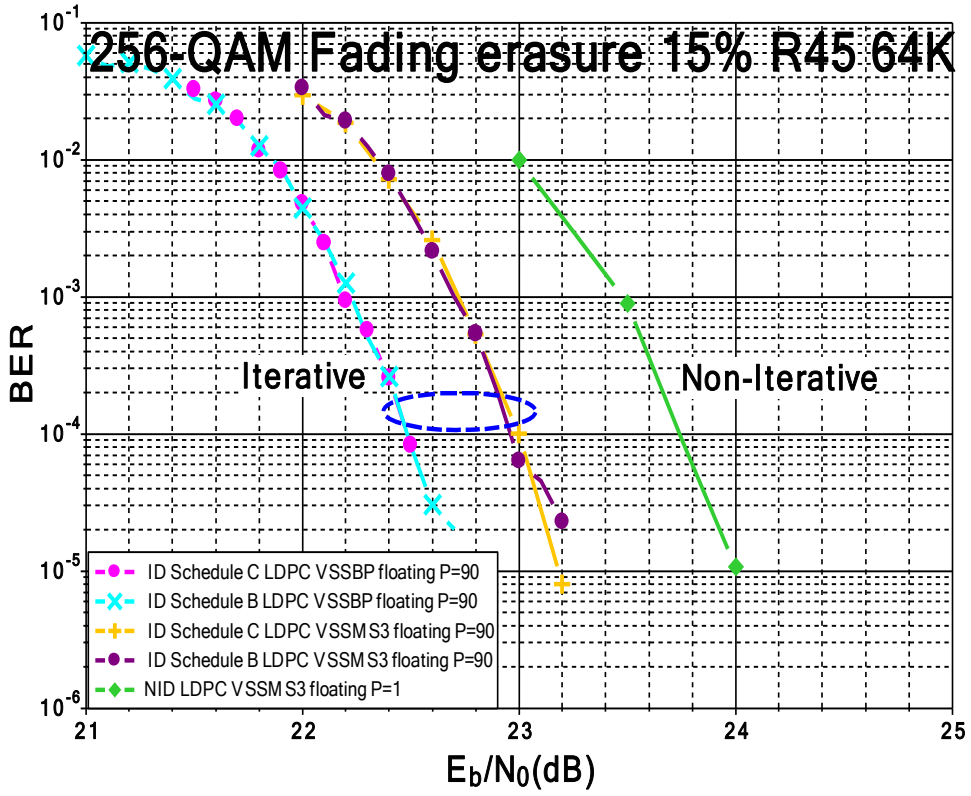

Figure 11: Performance comparison for rotated 256-QAM over a fading channel with 15 % of erasures. DVB-T2 64K LDPC, rate R=4/5

3.2.4 Design of an efficient iterative receiver architecture

The proposed architecture for the BICM-ID receiver is illustrated in Figure 12. One main demapper progressively computes the Euclidean Distances (ECD) and corresponding LLR values. All this information has to be memorized in LLR and ECD RAMs. Two types of those RAMs are allocated: one in charge of reception and one in charge of decoding. The decoding part is composed of 90 check node processors and 90 variable node processors. In charge of updating LLRs, 90 simplified demappers process extrinsic feedback generated by the decoder and the LLR RAM. Euclidean distances between the received observation and constellation symbols are memorized instead of I and Q components and the according CSI information in order to minimize the delay of the feedback-demapper. The updated LLRs are available only after two cycles of introducing updated extrinsic information. In this way, the decoding part processes the latest updated LLRs, even for the bits with a check node degree equal to 3.

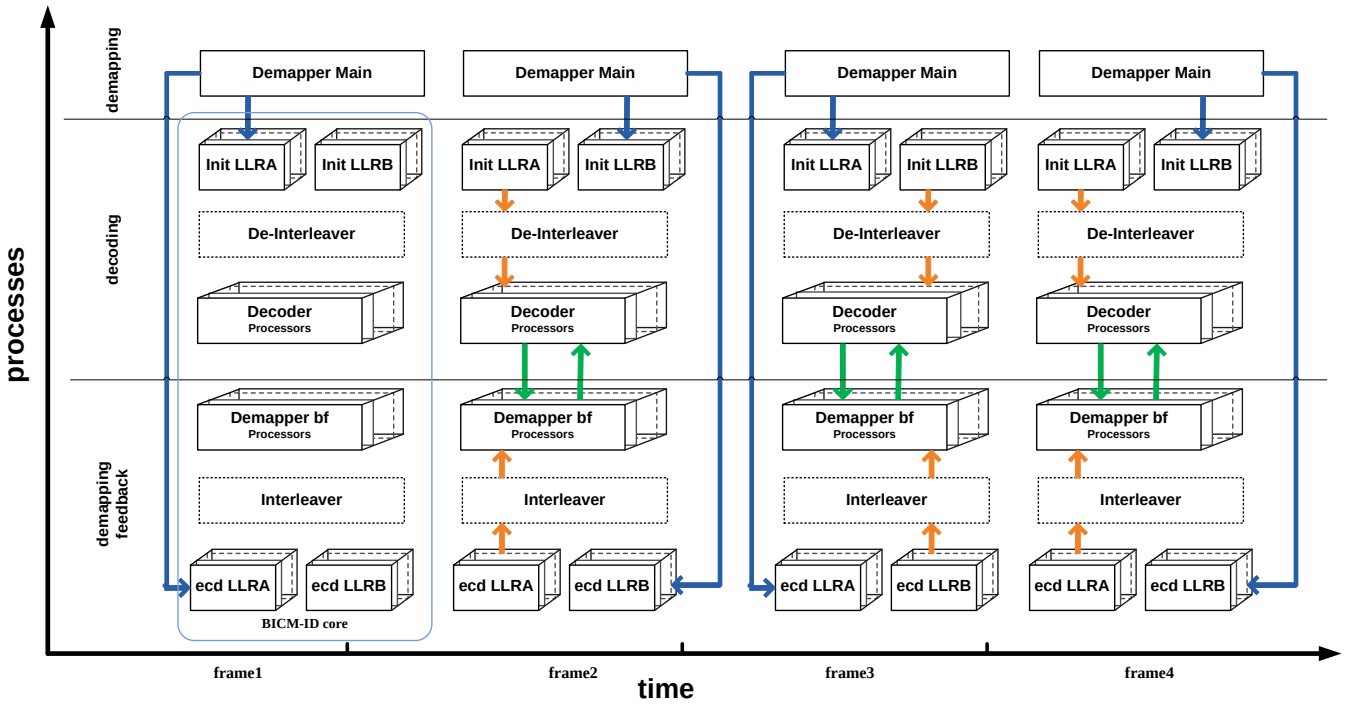

Figure 12: The proposed architecture of the vertical iterative receiver

Classically, the deinterleaving process is done by first writing the interleaved LLRs produced by the main demapper into a memory and then by reading them in the deinterleaving order by the decoding part. For interleaving, the decoded LLRs are first written into a memory and then are read in the interleaving order by the demapper. The DVB-T2 bit interleavers have been designed according to this principle. Encoded bits are written into a block memory column by column, they are read row by row, and then are permuted by a demultiplexer. Note that it is possible to replace the memory blocs by tables that directly address the connections between the demapper and the decoder. In this case, the table specification has to take into account the parallelism degree in the receiver architecture. Note that the DVB-T2 bit interleavers have been designed for a parallelism degree of 360 in the decoding part as explained in [24]. Another critical difficulty is the memory access conflicts for layered decoder architecture. These are due to the DVB-T2 parity check matrix structure and can cause significant performance loss. To deal with this constraint, we extended the reordering mechanism of the DVB-T2 parity check matrix detailed in [10] to a vertical layered schedule. We also solved the message updating inefficiency caused by the double diagonal sub-matrices during the decoder design as explained in [21]. The joint algorithm for a shuffled iterative receiver clearly brings benefits compared to the non-iterative and iterative frame-by-frame conventional receiver. The proposed schedule directly targets updating variable node. It facilitates the extrinsic information exchange between demapping and decoding processors. Indeed, both the demapped and decoded extrinsic information can be exchanged before the end of one frame processing. Let's take for example a 256-QAM constellation and a 64K-LDPC with a code rate of 4/5 having 630 non-zero elements in its 360 by 360 parity-check matrix. A parallelism degree of 90 is considered for the receiver. In order to perform one iteration for one coded frame, a classical frame-by-frame horizontal schedule has a latency l_{HSS} that can be expressed as:

$$l_{HSS} = \left[\left(630 * \frac{360}{90}\right) + 64800 * 2\right] \text{ cycles} \quad (23)$$

In comparison, the proposed shuffled iterative receiver architecture has a latency l_{VSS}:

$$l_{HSS} = \left[\left(630 * \frac{360}{90}\right) + \Delta\right] \text{ cycles} \quad (24)$$

where Δ is the delay of the interleaver table accesses. The iterative process convergence is then achieved with a lower latency.

3.2.5 FPGA implementation and prototyping

Figure 13 shows the different components of the experimental setup implemented onto one Xilinx Virtex5 LX330 FPGA. A Pseudo Random Generator (PRG) sends out pseudo random data streams at each clock period. An LDPC encoder processes the data streams. The codeword bits are then reordered thanks to the DVB-T2 interleaver. The last task of the transmitter is the mapping. The channel emulator is obtained from an AWGN generator of multiples variables. The hardware emulator is achieved using the Wallace method. Moreover, erasure event modeling was added to the channel emulator. The BICM-ID receiver is made up of a main rotated demapper and a BICM-ID core. This core is composed of 90 simplified demappers and 90 LDPC decoders. The proposed BICM-ID receiver was synthesized and implemented onto the FPGA. Computational resources of the BICM-ID MS core takes up about 15 % and 51% of a Xilinx XC5VLX330 FPGA slice registers and slice LUTs, respectively. If a BICM-ID MS3 core is implemented, 17% slice registers and 44% slice LUTs are necessary.

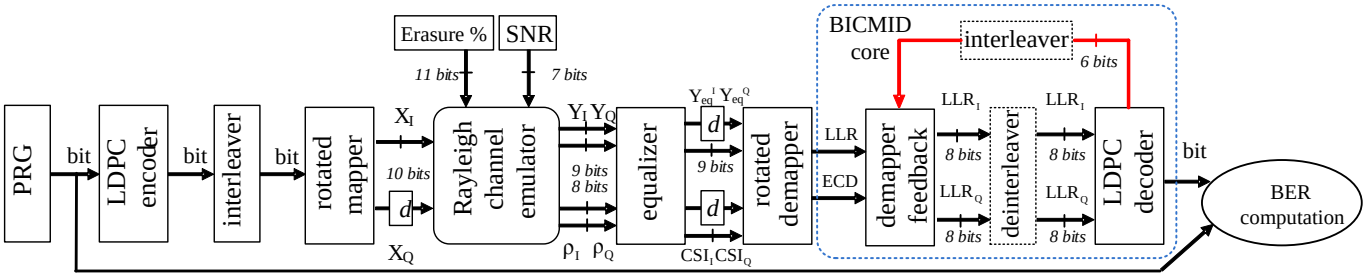

Figure 13: Experimental setup for prototyping the BICM-ID receiver

Table 3: HW resources for the two different BICM-ID cores

XC5VLX330	Flip-flops	LUTs	RAMs
BICM-ID VSS MS	17,371	93,130	179
BICM-ID VSS MS3	26,078	107,438	193

The maximum frequency estimated for the BICM-ID core after place and route is 80 MHz. It results in a throughput of 107 Mbps, for R =4/5 @ 15 layered iterations. A comparison of simulated performance and experimental setup measured performance in terms of BER of the designed BICM-ID receiver with VSS MS and VSS MS3 decoding algorithm for a QPSK constellation, a code rate R =4/5 and 64,800 bit frames, is presented in Figure 14. More than 10 dB gain is observed from the BICM-ID VSS MS receiver when compared to the non-rotated QPSK in a non-iterative receiver. Moreover, an additional gain of 0.9 dB is achieved for the iterative receiver with VSSMS3 decoding algorithm. These experimental results validate the potential of BICM-ID systems as a practical solution for the DVB-T2 standard.

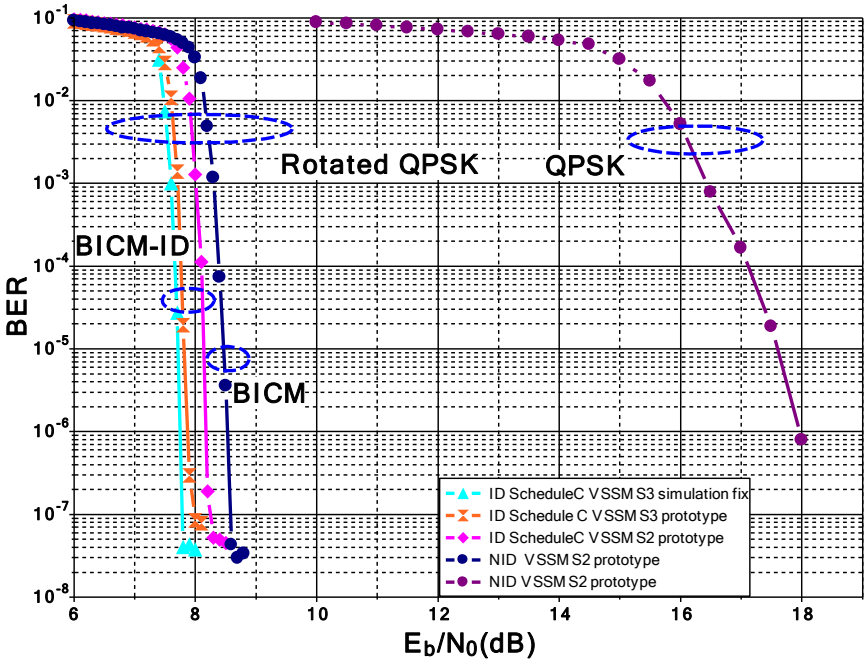

Figure 14: Performance comparison for QPSK over a fading channel with 15 % of erasures. 64K frames, DVB-T2 LDPC, rate R =4/5

3.2.6 Conclusion

BICM-ID shows best theoretical performance in the implementation guidelines of the DVB-T2 standard. In this paper, we have detailed a vertical schedule that favours an efficient data exchange between the demapper and the decoder in an ID context. The designed architecture leads to limited complexity and latency and to an acceleration of the iterative process convergence. Then, FPGA prototype characteristics and performance for BICM-ID receivers based on a vertical schedule of MS and MS3 have been discussed. The iterative receiver achieves high performance gain as expected. To the best of our knowledge, this is the first hardware implementation of a BICM-ID receiver for the DVB-T2 standard.

4 EXPECTED DVB-T2 PERFORMANCE OVER TIME VARYING ENVIRONMENTS

DVB-T2 constitutes the first second generation Digital Terrestrial Television (DTT) standard. Our focus will be mainly to assess its mobile performance for modes that have been deployed (like the UK mode) or that will soon be deployed (like the German candidate mode).

4.1 Mobile Channel Model

When receiving a DTT signal in a moving car, the mobile transfer channel can be modelled as a wideband "Frequency-Selective" channel. Indeed, in this case, the transmitted bandwidth W (usually taking values from 6 to 8 MHz) is much larger than Bc, the channel's coherence bandwidth (of 100kHz approximately). Bc is related to the maximum delay spread τmax (of about 10μs) by Bc=1/τmax. This type of fading can be modelled as a linear filter whose coefficients Ci are Gaussian complex (Rayleigh envelops), independent of each other and filtered in order to have the desired Doppler spectrum. This is actually shown on the following figure which illustrates the Typical Urban model with 6 paths (TU6 defined by COST 207).

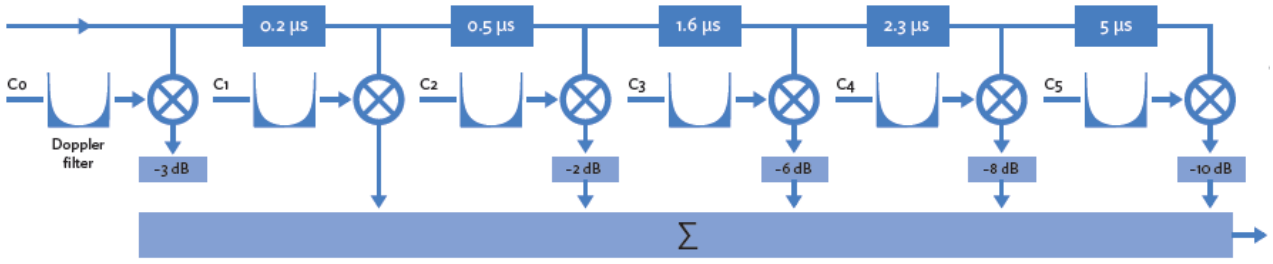

Figure 15: Typical Urban model with 6 paths (TU6).

In the following sections we intend to analyse the statistic of the received signal in Single and Diversity modes for both Narrowband (Rayleigh) and Wideband (TU6) channels. In addition, we define a performance criteria measurement for mobile environments. The main objective is to theoretically determine how much more signal power is needed for mobile reception versus fixed reception.

4.1.1 First order statistics: Signal distribution (rate independent)

The cumulative distribution function (cdf) is a first-order statistic (that is independent of the rate), which gives the probability to obtain a C/N ratio below a certain threshold. For narrowband channels with a Rayleigh distribution, the cdf formula is given in Table 1. It is also illustrated in Figures 2&3 for Single and Diversity MRCmodes (M=1 to 4 branches) versus Γ, the mean C/N and γ the C/N threshold crossed by the signal.

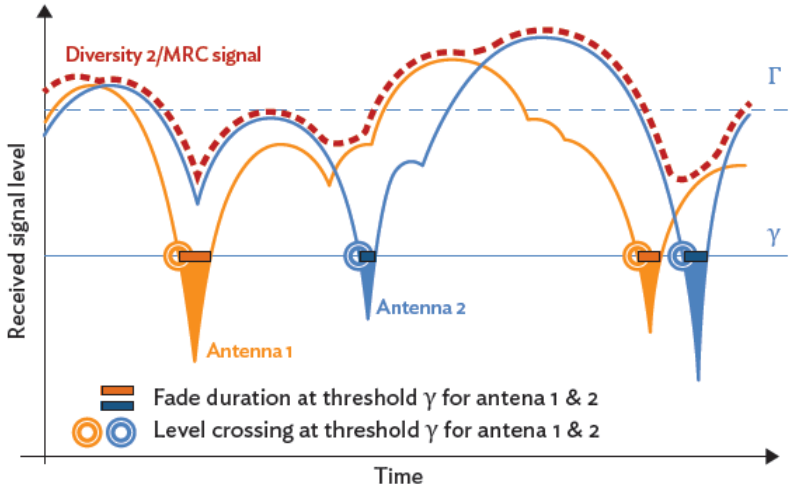

Figure 16: Single & Diversity 2 fade duration and Level Crossing.

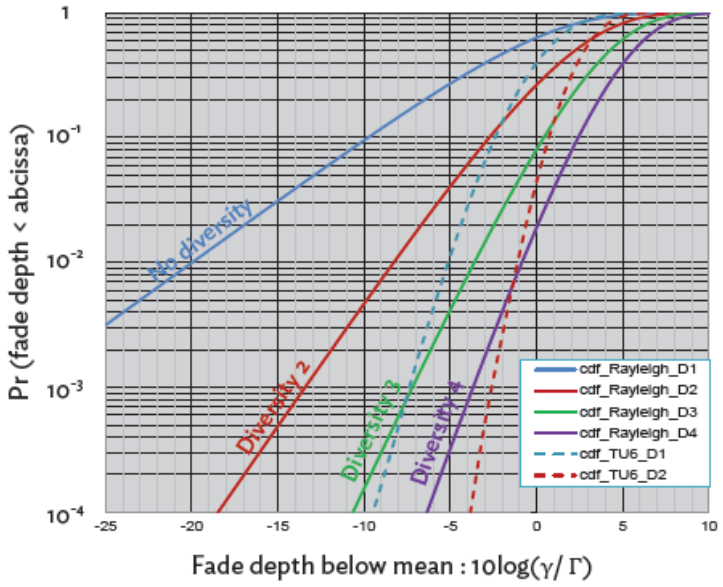

Figure 17: Rayleigh and TU6 cumulative distribution functions.

Please note that for the TU6 channel model, the cdf does not follow a Rayleigh law, but rather a "Non-central Chi-square law with 12 degrees of freedom". This is simulated and in Figure 17 it corresponds to the curves with dashed lines for the Single and Diversity 2 cases.

4.1.2 Second order statistics: Signal rate distribution (LCR, AFD, Doppler)

Second-order statistics are concerned with the distribution of the signal's rate channel change, rather than the signal itself. In a fixed or a slowly time varying environment, the Doppler effect is negligible. As soon as the receiver moves the channel varies through time and the carriers are no longer pure sine waves.

The Doppler shift Fd is given by:

$$Fd = Fc\ (v/c)\ \cos\ \alpha,$$

where Fc is the carrier frequency, v is the speed of the vehicle, c the speed of light and α the angle between the direction of motion and the arrival direction of the signal (see Figure 18). The maximum Doppler frequency is fDm = Fc (v/c) for α = 0.

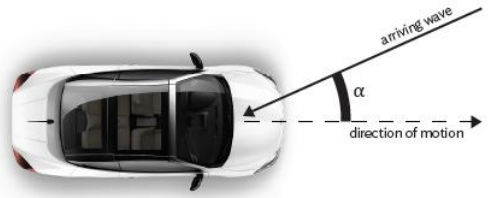

Figure 18: The Doppler effect.

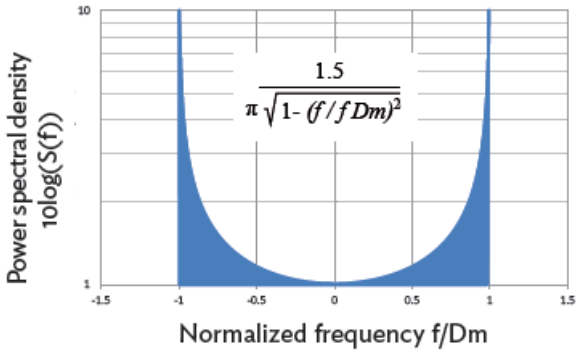

Figure 19: The classical Doppler spectrum.

Assuming a uniform distribution of the angle α, from -π to +π, the power spectrum of the received signal is called "Classical Doppler Spectrum" (or "Jake's spectrum") and is illustrated in Figure 19.

Finally, the Doppler frequency spectrum, the **Level Crossing Rate** (LCR) and the **Average Fade Duration** (AFD) characterise the dynamic representation of the mobile channel. As shown in Figure 16:

- The LCR is defined as the number of time per unit duration that the fading envelope crosses a given value in the negative, or positive, direction. Practically, the LCR gives the number of fades per second under a given threshold level and it is equal to the Erroneous Second Rate (ESR) criterion:

  LCR = ESRx, for x≤10%

- The AFD is the average time duration for which the fading envelope remains below a specified level.

Both LCR and AFD provide important information about the statistics of burst errors. The latter facilitates the design and selection of error-correction techniques. It should be pointed out that adding/increasing time interleaving will decrease both LCR and AFD, even for the case of single reception.

The following table gives the theoretical formulas for cdf, LCR, and AFD with respect to the diversity order M and γ/Γ.

Table 4: Theoretical cdf, LCR and AFD for a Rayleigh's distribution fadings combined in MRC Diversity (Γ = average C/N).

Number of receiver's antennas	cdf : Outage probability (1) P(γ<γth)	Normalized Level Crossing Rate (2) LCR / fDm	Normalized Average Fade Duration (3) AFD x fDm =(1) / (2)
M	$1- e^{-\frac{\gamma}{\Gamma}} \sum_{i=0}^{M-1} \frac{\left(\frac{\gamma}{\Gamma}\right)^i}{i!}$	$\frac{\sqrt{2\pi}}{(M-1)!} \left(\frac{\gamma}{\Gamma}\right)^{\left(M-\frac{1}{2}\right)} e^{-\left(\frac{\gamma}{\Gamma}\right)}$	$\frac{\mathrm{cd}f}{\left(\mathrm{LCR}/fDm\right)}$

For MRC Diversity (up to order 4), Figure 20 illustrates the Level Crossing Rate and Figure 21 the Average Fade Duration normalized with respect to fDm, the maximum Doppler frequency. For the Rayleigh distribution the formulas of Table 4 are used, while the TU6 statistics have been simulated.

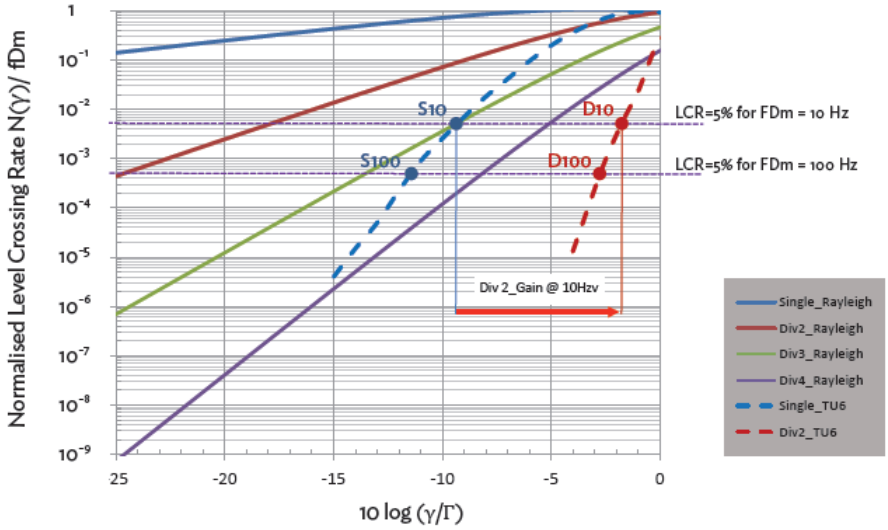

Figure 20: Normalized Level Crossing Rate with respect to (γ/Γ).

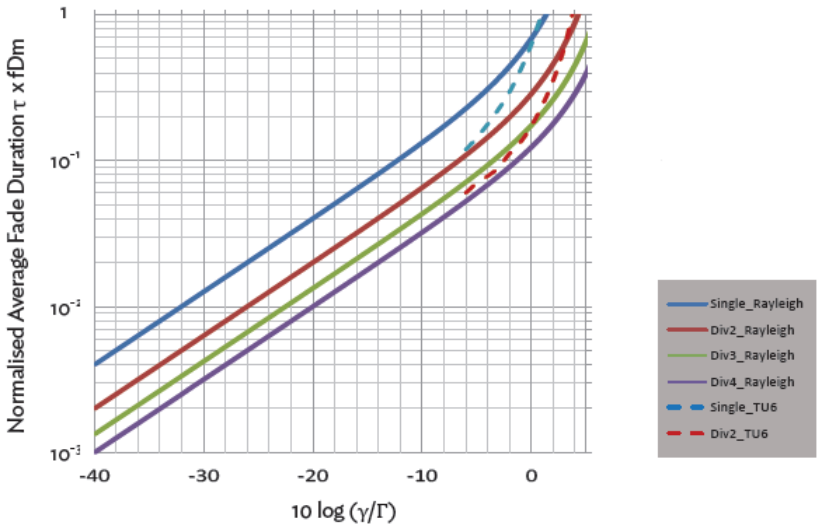

Figure 21: Normalized Average Fade Duration with respect to (γ/Γ).

When there is no time interleaving (like in DVB-T) Figure 20 is very useful for determining the simulated (γ/Γ) threshold leading to a given ESR. For example, to be consistent with the ESR5 criteria (5% of erroneous second) with a Doppler frequency of 10 Hz, Figure 20 shows that in TU6 channels Γ (i.e. the mean C/N) must be greater by at least 9.5 dB over γ (the Gaussian threshold of reception) in Single and greater by only 1.7 dB in Diversity 2, which gives a simulated Diversity gain of approximately 8dB. Considering a Doppler frequency of 100 Hz the diversity gain is slightly higher. It should be pointed out that these results are theoretical and represent the maximal performance for any standard lacking time interleaving.

Figure 21 shows that the average fading duration is divided by 2 when moving from Single to Diversity 2 and by 4 from Single to Diversity 4, independent of the (γ/Γ) value. Therefore, by adding time-interleaving, it is possible to improve mobile performance in Single vs Gaussian. However in this case, the diversity gain is reduced, since the two gains are not added.

4.1.3 Additional Doppler effects

In addition to increasing the number of fades per second, the Doppler effect spreads the OFDM sub-carriers ("FFT Leakage"), which destroys their orthogonality and therefore it creates Inter Carrier Interference (ICI).

In order to compare the theoretical mobile performance of the receiver to the one tested in the laboratory with a TU6 channel simulator, it is necessary to plot (Γ/γ) versus the Doppler frequency for the ESR5 criterion. As illustrated in Figure 22, the quasi-horizontal parts of the curves are derived directly from Figure 20. Nevertheless, as expected, after a given Doppler limit, the receiver is not able to demodulate the signal. Then, when the Doppler (i.e. the speed of the mobile) further increases, the recovery performance degrades drastically until a point where no demodulation is possible, even with a very high C/N, **which explains the quasi vertical lines measured in laboratory testing**.

In order to have good Mobile performance even with single receivers, DiBcom/Parrot´s chips integrate sophisticated signal processing algorithms such as "Dynamic FFT positioning", "Fast channel estimation", "FFT leakage removal", etc….

However as the Doppler shift increases, it becomes necessary to use Diversity which dramatically decreases both the rate and the duration of fades. This results to a gain of the required average C/N by a value between 4 to 8 dB, which depends on the standards' physical layer (i.e. existence of time interleaving, …).

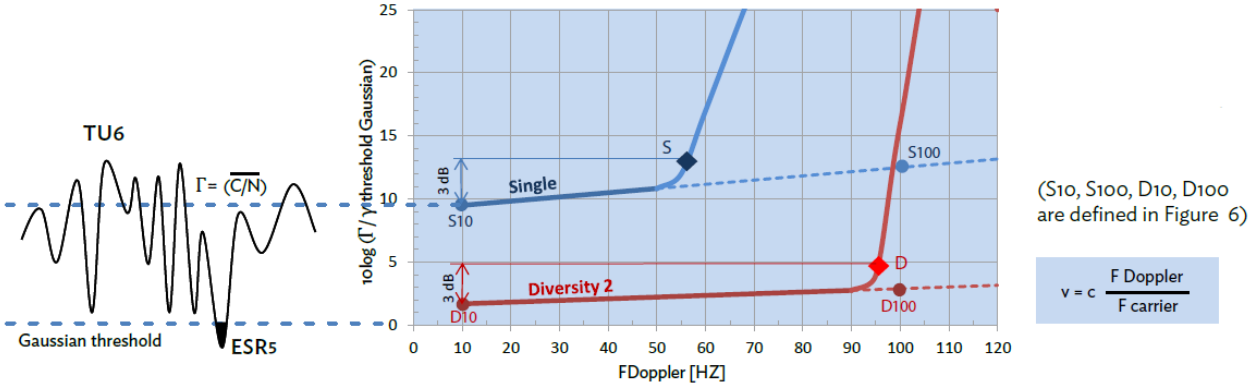

Figure 22: (γ/Γ) with respect to FDoppler.

In order to characterize the receiver speed limit it was agreed (in the Motivate, WING TV projects and MBRAI specification) to consider the maximum achievable Doppler frequency as the value for: (C/N) min @10 Hz +3dB. The asymptotic Doppler Frequency @ (C/N) max has no actual meaning, since in practise the C/N seen in the field is never higher than ~30dB. Concerning high Doppler frequency shift, the field test results obtained in a car equipped with a Parrot diversity receiver showed a strong correlation with the data obtained in a laboratory environment with a TU6 channel simulator.

In the Section 4.3 the S and D measurement points of Figure 22 are reported on a graph for most of the DTT standards in order to compare their mobile performance.

4.2 DVB-T2 Simulation Results

In order to simulate the performance of DVB-T2 we decided to consider the UK profile and a German-candidate profile. Even those two T2 modes by far are not the most optimized T2 versions for mobility, they constitute profiles that either have been already deployed (UK case) or will be shortly deployed (German case). The next Table shows the parameters that were used for the two DVB-T2 modes that are tested in this section.

Table 5: Tested DVB-T2 modes.

UK mode	(potential) German mode
FFT = 32K GI = 1/128 LDPC: 64800 CR = 2/3 256-QAM PP7 Ext. BW: On Single PLP (max time interleaving ~71ms)	FFT = 16K GI = 19/128 LDPC: 16200 CR = 2/3 16-QAM PP2 Ext. BW: Off Single PLP (max time interleaving ~83ms)

Due to time and resources limitations, for each C/N simulated point the maximum number of LDPC codewords was up to 93800 for the UK case and 179400 for the German case. These numbers correspond to simulating roughly ~100 seconds of real time signal. Once we decoded the entire 100 seconds duration of the real time signal without any errors, this point was considered to satisfy the ESR5 criterion.

In Figure 23 and Figure 24 we present the BER and BLER (block error rate, where 1 block is 1 LDPC codeword) for the UK and German modes. A TU6 channel has been simulated with a Doppler equal to 10Hz. Both single and Diversity 2 reception are depicted. For the latter case, we have generated two uncorrelated TU6 channels, and at the two signals were summed according to the well-known maximum ratio combining (MRC) method. For both modes the QEF (i.e. ESR5 criterion) diversity 2 reception offers a gain of the order of 5dBs. As expected, the German-candidate mode performs at a much lower C/N. This is mainly due to the lower constellation (more robust) and the lower FFT size (introducing a lower amount of ICI for the same Doppler frequency). The maximum achievable Doppler frequency (as previously defined, i.e. the value for: (C/N)min @10 Hz +3 dB) for the UK mode is around 16 and 28Hz for single and diversity 2. This rather poor performance was expected (largest FFT, largest constellation,…). On the other hand the simulated German-candidate mode attains a maximum Doppler of 100Hz for single and 122Hz in diversity 2 reception.

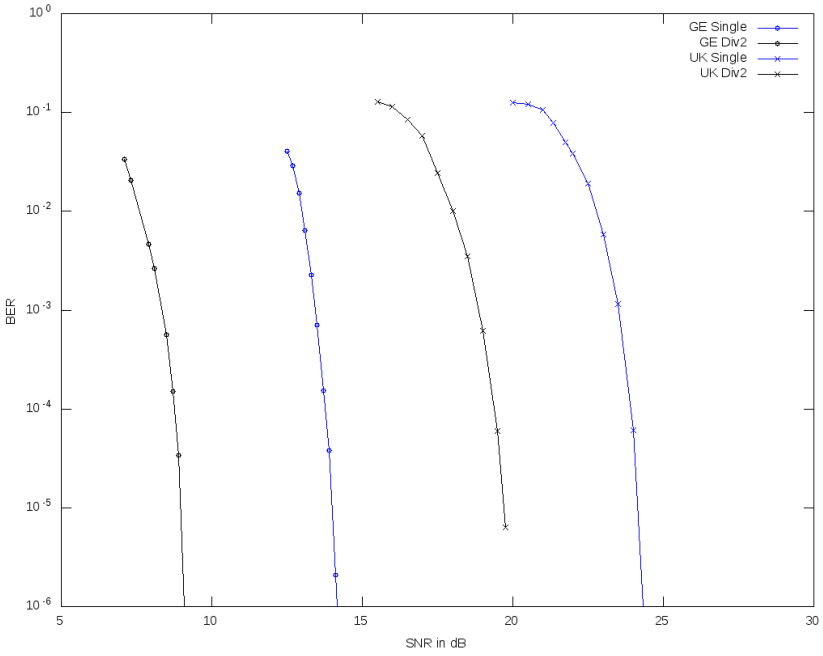

Figure 23: BER in Single and Diversity 2 for the German and UK modes.

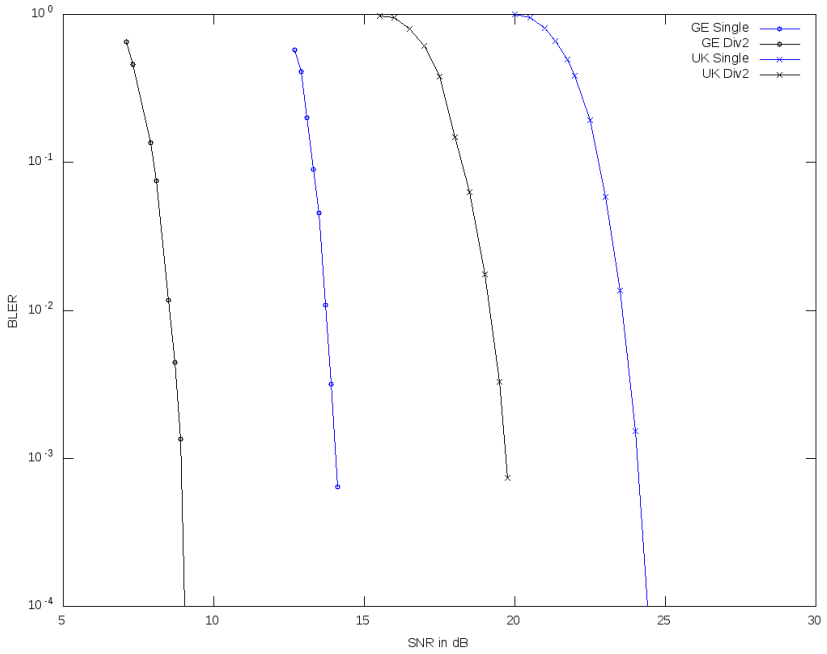

Figure 24: BLER in Single and Diversity 2 for the German and UK modes.

4.3 Mobile Performance of Worldwide DTT standards

Thanks to the multi-standard capacity of Octopus, it is possible to compare the actual mobile performances of most DTT configurations used in the World with the simulations described in the previous chapters. Table 6 offers the possibility to compare fixed (Gaussian) with mobile (TU6) performance and shows the maximum speeds attainable in Single and Diversity 2. Please note that the performance of all the standards is measured in Laboratory testing. The only exception is DVB-T2 whose performance has been simulated as it has been shown in the previous section.

Table 6: Mobile performance of worldwide DTT standards.

				FEC		Modul.	BW [MHz]	Du [Mb/s]		Spectral efficiency (bit/s/Hz)	Country	$(C/N)_{TU6@10Hz}$ - $(C/N)_{Gaussian\_Single}$		Speed [km/h]	
Standards	Use case	FFT	GI	inner	outer							Single	Div2	Single	Div2
DVB-T	Mobile	8k	1/4	2/3	188/204	16-QAM	8		13.27	1.66	Germany	11.2	3.2	90	200
	Fixed	8k	1/8	3/4	188/204	64-QAM	8		24.88	3.11	France	12.6	3.6	40	140
ISDB-T	Layer A 1 SEG	8k	1/8	2/3	188/204	QPSK	6	0.416	17.27	2.88	Japon	5.5	2.5	170	220
	Layer B 12 SEG		1/8	3/4	188/204	64-QAM	6	16.85				4.6	- 1.4	150	160
CMMB	Mobile	4k		1/2	176/240	QPSK	8		4	0.5	Shanghaï	6.1	1.3	700	860
		4k		3/4	176/240	QPSK	8		6	0.75		7.4	1.4	550	700
		4k		1/2	192/240	16-QAM	8		8.7	1.09		7	-	580	-
		4k		3/4	192/240	16-QAM	8		13.1	1.64		9	4	260	510
CTTB	Fixed / PN420	4k	1/9	0.8		16-QAM	8		21.66	2.71	Beijing	9.2	2.7	110	250
	Fixed / PN595	SC	17/108	0.8		16-QAM	8		20.79	2.6	Shanghaï	7.9	2.9	90	140
	Fixed / PN945	4k	1/4	0.6		16-QAM	8		14.44	1.8	Shanghaï	-	-	-	-
DVB-T2	Fixed	32k ext	1/128	2/3	BCH	256-QAM	8		40.21	5.03	UK	7.2	2.2	25	45
	Mobile	16k	19/128	2/3	BCH	16-QAM	8		15.27	1.9	Germany	4.7	- 0.3	155	190
DVB-SH	Terr./Sat.	2k	1/4	1/2		QPSK	5		3.36	0.67		3.5	- 0.5	670	780
DAB+	Radio	2k		1/2	RS	D-QPSK	1.71		1.12	0.65	Germany	5.6	1.6	80	120

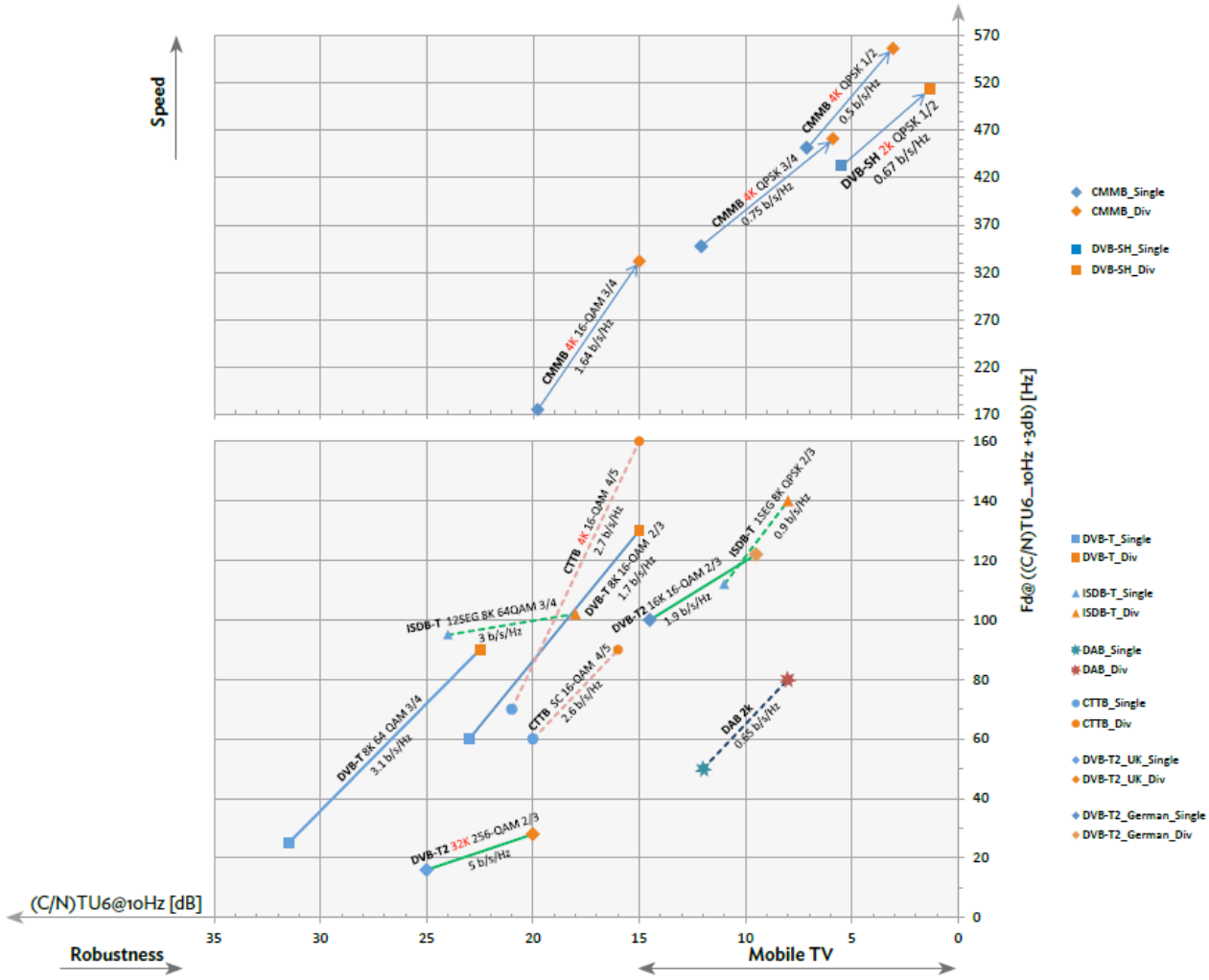

Figure 25: FdMax in Single and Diversity reception with respect to C/N for TU6@10Hz.

4.4 Conclusion

The pressure on broadcasters to gain spectrum is creating a need to use high spectrum efficiency standards such as DVB-T2. As seen on Figure 25, the UK-mode of DVB-T2 maximizes the useful bit-rate focusing only on fixed applications. Therefore, it presents a very poor performance in mobile conditions. Even if the maximum speed is ~55km/h, when using a Diversity 2 receiver, the required C/N remains quite high, at the order of ~23dB.

On the other hand, the candidate German-mode, which is very likely to adopt a 16K FFT (with either 16-QAM or 64-QAM), constitutes a good compromise between data rate and mobile performance. Its data rate is increased with respect to DVB-T and in addition it presents a good mobile performance.

5 FAST GPU AND CPU IMPLEMENTATIONS OF AN LDPC DECODER

This work has been published in [31].The DVB-T2 standard makes use of two FEC codes, featuring LDPC (low-density parity-check) codes [32] with exceptionally long codeword lengths of 16200 or 64800 bits as the inner code. As outer code, a BCH (Bose-Chaudhuri-Hocquenghem) code is employed to reduce the error floor caused by LDPC decoding. The second generation digital TV standards for satellite and cable transmissions, DVB-S2 and DVB-C2, respectively, also employ very similar LDPC codes to DVB-T2. Because of the long LDPC codewords, the decoding of these codes is one of the most computationally complex operations in a DVB-T2 receiver [33].

In this work, a method for highly parallel decoding of the long LDPC codes using GPUs (graphics processing units) and general purpose CPUs (central processing units) is proposed. While a GPU or CPU implementation is likely less energy efficient than implementations based on for example ASICs (application-specific integrated circuits) and FPGAs (field programmable gate arrays), GPUs and CPUs have other advantages. Even high-end GPUs and CPUs are often quite affordable compared to capable FPGAs, and this hardware can be found in most personal home computers. Although originally developed for graphics processing, modern GPUs are also highly reconfigurable similarly to general purpose CPUs. These advantages make a GPU or CPU implementation interesting for software defined radio (SDR) systems built using commodity hardware, as well as for testing and simulation purposes.

Algorithms and data structures that allow for reaching the LDPC decoding throughput bitrates required by DVB-T2, DVB-S2, and DVB-C2 when implemented on a modern GPU, are described in this report. While the design decisions are generally applicable to GPU architectures overall, this particular implementation is built on the NVIDIA CUDA (Compute Unified Device Architecture)[34], and tested on an NVIDIA GPU. The performance of the GPU implementation is also compared to a highly efficient multithreaded CPU implementation written for a consumer-grade Intel CPU. Furthermore, the impact of limited numerical precision as well as applied algorithmic simplifications on the error correction performance of the decoder is examined. This is accomplished through comparing the error correction performance of the proposed optimized implementations to more accurate CPU-based LDPC decoders, by simulating transmissions within a DVB-T2 physical layer simulator.

5.1 LDPC Codes

A binary LDPC code [32] with code rate $r=k/n$ is defined by a sparse binary $(n-k)\times n$ parity-check matrix, $\mathbf{H}$. A valid codeword $\mathbf{x}$ of length n bits of an LDPC code satisfies the constraint $\mathbf{Hx}^T=\mathbf{0}$. As such, the parity-check matrix $\mathbf{H}$ describes the dependencies between the k information bits and the $n-k$ parity bits. The code can also be described using bipartite graphs, i.e., with n variable nodes and $n-k$ check nodes. If $\mathbf{H}_{i,j}=1$, then there is an edge between variable node j and check node i.

LDPC codes are typically decoded using iterative belief propagation (BP) decoders. The procedure for BP decoding is the following. Each variable node v sends a message $L_{v\rightarrow c}$ of its belief on the bit value to each of

its neighboring check nodes *c*, i.e. those connected to the variable node with edges. The initial belief corresponds to the received Log-Likelihood Ratios (LLR), which are produced by the QAM (Quadrature Amplitude Modulation) constellation demapper in a DVB-T2 receiver. Then each check node *c* sends a unique LLR $L_{c\to v}$ to each of its neighboring variable nodes *v*, such that the LLR sent to *v'* satisfies the parity-check constraint of *c* when disregarding the message $L_{v'\to c}$ that was received from the variable node *v'*. After receiving the messages from the check nodes, the variable nodes again send messages to the check nodes, where each message is the sum of the received LLR and all incoming messages $L_{c\to v}$ except for the message $L_{c'\to v}$ that came from the check node *c'* to where this message is being sent. In this step, a hard decision is also made. Each variable node translates the sum of the received LLR and all incoming messages to the most probable bit value and an estimate on the decoded codeword $\hat{\mathbf{x}}$ obtained. If $\mathbf{H}\hat{\mathbf{x}}^{\mathrm{T}}=\mathbf{0}$, a valid codeword has been found and a decoding success is declared. Otherwise, the iterations continue until either a maximum number of iterations has been performed or a valid codeword has been found.

The LDPC decoder is one of the most computationally complex blocks in a DVB-T2 receiver, especially given the long codeword lengths (*n* is 16200 or 64800, while *k* varies with the code rate used) specified in the standard. The best iterative BP decoder algorithm is the *sum-product* decoder [35], which is also, however, quite complex in that it uses costly operations such as hyperbolic tangent functions. The *min-sum* [36][37] decoder trades some error correction performance for speed by approximating the complex computations of outgoing messages from the check nodes. The resulting computations that are performed in the decoder are the following. Let *C(v)* denote the set of check nodes which are connected to variable node *v*. Similarly let *V(c)* denote the set of variable nodes which are connected to check node *c*. Furthermore, let *C(v)\c* represent the exclusion of *c* from *C(v)*, and *V(c)\v* represent the exclusion of *v* from *V(c)*. With this notation, the computations performed in the min-sum decoder are the following:

1. Initialization: Each variable node *v* sends the message $L_{v\to c}(x_v) = LLR(v)$.

2. Check node update: Each check node *c* sends the message

$$L_{c\to v}(x_v) = \left(\prod_{v'\in V(c)\setminus v} \operatorname{sign}(L_{v'\to c}(x_{v'})\right) \times \min_{v'\in V(c)\setminus v} |L_{v'\to c}(x_{v'})|$$

where sign(x) = 1 if x ≥ 0 and -1 otherwise.

3. Variable node update: Each variable node *v* sends the message

$$L_{v\to c}(x_v) = LLR(v) + \sum_{c\prime\in C(v)\setminus c} L_{c\prime\to v}(x_v)$$

and computes

$$L_v(x_v) = LLR(v) + \sum_{c\in C(v)} L_{c\to v}(x_v)$$

4. Decision: Quantize $\hat{x}_v$ such that $\hat{x}_v = 1$ if $L_v(x_v) < 0$, and $\hat{x}_v = 0$ if $L_v(x_v) \geq 0$. If $\mathbf{H}\hat{\mathbf{x}}^{\mathrm{T}}=\mathbf{0}$, $\hat{\mathbf{x}}$ is a valid codeword and the decoder outputs $\hat{\mathbf{x}}$. Otherwise, go to step 2.

5.2 Hardware Architectures

In this section follows a description of the NVIDIA CUDA, and the specific GPU for which the GPU-based implementation was developed. Other relevant components of the system used for benchmarking the decoder implementations are also described, including the Intel CPU which was also the target for the CPU-optimized LDPC decoder.

5.2.1 CUDA

The NVIDIA CUDA[34] is used on modern NVIDIA GPUs. The architecture is well suited for data-parallel problems, i.e problems where the same operation can be executed on many data elements at once. At the time of writing this report, the latest variation of the CUDA used in GPUs was the Fermi architecture [38], which offers some improvements over earlier CUDA hardware architectures, such as an L1 cache, larger on-chip shared memory, faster context switching etc.

In the CUDA C programming model, we define kernels, which are functions that are run on the GPU by many threads in parallel. The threads executing one kernel are split up into thread blocks, where each thread block may execute independently, making it possible to execute different thread blocks on different processors on a GPU. The GPU used for running the LDPC decoder implementation described in this paper was an NVIDIA GeForce GTX 570, featuring 15 streaming multiprocessors (SMs) containing 32 cores each.

The scheduler schedules threads in groups of 32 threads, called thread *warps*. The Fermi hardware architecture features two warp schedulers per SM, meaning the cores of a group of 16 cores on one SM execute the same instruction from the same warp.

Each SM features 64 kB of fast on-chip memory that can be divided into 16 kB of L1 cache and 48 kB of shared memory ("scratchpad" memory) to be shared among all the threads of a thread block, or as 48 kB of L1 cache and 16 kB of shared memory. There is also a per-SM register file containing 32,768 32-bit registers. All SMs of the GPU share a common large amount of global RAM memory (1280 MB for the GTX 570), to which access is typically quite costly in terms of latency, as opposed to the on-chip shared memories.

The long latencies involved when accessing global GPU memory can limit performance in memory intensive applications. Memory accesses can be optimized by allowing the GPU to *coalesce* the accesses. When the 32 threads of one warp access a continuous portion of memory (with certain alignment limitations), only one memory fetch/store request might be needed in the best case, instead of 32 separate requests if the memory locations accessed by the threads are scattered [34]. In fact, if the L1 cache is activated (can be disabled at compile time by the programmer), all global memory accesses fetch a minimum of 128 bytes (aligned to 128 bytes in global memory) in order to fill an L1 cache line. Memory access latencies can also be effectively hidden if some warps on an SM can run arithmetic operations while other warps are blocked by memory accesses. As the registers as well as shared memories are split between all warps that are scheduled to run on an SM, the number of active warps can be maximized by minimizing the register and shared memory requirements of each thread.

5.2.2 Measurement setup and CPU

The desktop computer system, of which the GeForce GPU was one component, also contained an Intel Core i7-950 main CPU running at a 3.06 GHz clock frequency. This CPU has 4 physical cores, utilizing Intel Hyper-Threading technology to present 8 logical cores to the system [39]. 6 GB of DDR3 RAM (Double Data Rate 3 random access memory) with a clock frequency of 1666 MHz was also present in the system. The operating system was the Ubuntu Linux distribution for 64-bit architectures.

The CPU supports the SSE (Streaming SIMD Extensions) SIMD (single instruction, multiple data) instruction sets[39] up to version 4.2. These vector instructions, operating on 128-bit registers, allow a single instruction to perform an operation on up to 16 packed 8-bit integer values (or 8 16-bit values, or 4 32-bit values) at once. There are also instructions operating on up to 4 32-bit floating point values. The optimized CPU-based LDPC decoder described in this report exploits these SIMD instructions in combination with multithreading to achieve high decoding speeds. For multithreading, the POSIX (Portable Operating System Interface) thread libraries are utilized.

Another possible approach to building a CPU decoder is to compile the CUDA code directly for the Intel CPU architecture using an appropriate compiler [39]. It is also possible to write the GPU kernels within the OpenCL (Open Computing Language) framework [41] instead of CUDA, as OpenCL compilers are available for both the GPU and CPU. Both of these approaches would still most likely require tuning the implementation separately for the two target architectures in order to achieve high performance, however. As the focus here lies on performance rather than portability, the CPU decoder was implemented using more well established CPU programming methods.

5.3 Decoder Implementation

The GPU-based LDPC decoder implementation presented here consists mainly of two different CUDA kernels, where one kernel performs the variable node update, and the other performs the check node update. These two kernels are run in an alternating fashion for a specified maximum number of iterations. There is also a kernel for initialization of the decoder, and one special variable node update kernel, which is run last, and which includes the hard decision (quantization) step mentioned in section 5.1.

The architecture of the optimized CPU implementation is very similar to the GPU version. On the CPU, the kernels described above are implemented as C functions which are designed to run as threads on the CPU. Each single thread on the CPU, however, does significantly more work than a single thread running on a CUDA core.

5.3.1 General decoder architecture

For storage of messages passed between check nodes and variable nodes, 8-bit precision is used. As the initial LLR values were stored in floating point format on the host, they were converted to 8-bit signed integers by multiplying the floating point value by 2, and keeping the integer part (clamped to the range

$[-127, +127]$). This resulted in a fixed point representation with 6 bits for the integer part and 1 bits for the decimal part. The best representation in terms of bit allocation is likely dependent on how the LLR values have been calculated and the range of those values. The mentioned bit allocation was found to give good results in simulations, however this report does not focus on finding an optimal bit allocation for the integer and decimal parts. After this initial conversion (which is performed on the CPU), the LDPC decoder algorithms use exclusively integer arithmetic.

GPU memory accesses can be fully coalesced if 32 consecutive threads access 32 consecutive 32-bit words in global memory, thus filling one cache line of 128 bytes. In order to gain good parallelism with regard to memory access patterns, the decoder was designed to decode 128 LDPC codewords in parallel. When reading messages from global memory, each of the 32 threads in a warp reads four consecutive messages packed into one 32-bit word. The messages are stored in such a way that the 32 32-bit words read by the threads of a warp are arranged consecutively in memory, and correspond to 128 8-bit messages belonging to 128 different codewords. This arrangement leads to coalescing of memory accesses. Computed messages are written back to global memory in the same fashion, also achieving full coalescence. While the Core i7 CPU only has 64 byte cache lines, the CPU decoder was also designed to decode 128 codewords at once, in order to keep the data structures of the GPU and CPU implementations equal (this decision should not decrease performance).

Two compact representations, $\mathbf{H}_{VN}$ and $\mathbf{H}_{CN}$, of the parity check matrix **H** are used. The data structures were inspired by those described in [42]. To illustrate these structures, the following simple example **H** matrix is used:

$$\boldsymbol{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$\mathbf{H}_{CN}$ would then be an array of entries consisting of a cyclic index to the entry corresponding to the next one in the same row of the **H** matrix, while entries in $\mathbf{H}_{VN}$ would contain an index to the entry corresponding to the next one in the same column. Each entry in $\mathbf{H}_{CN}$ and $\mathbf{H}_{VN}$ thus represent an edge between a variable node and a check node in the bipartite graph corresponding to **H**. The $\mathbf{H}_{CN}$ and $\mathbf{H}_{VN}$ structures corresponding to the example **H** matrix are illustrated in Figure 26.

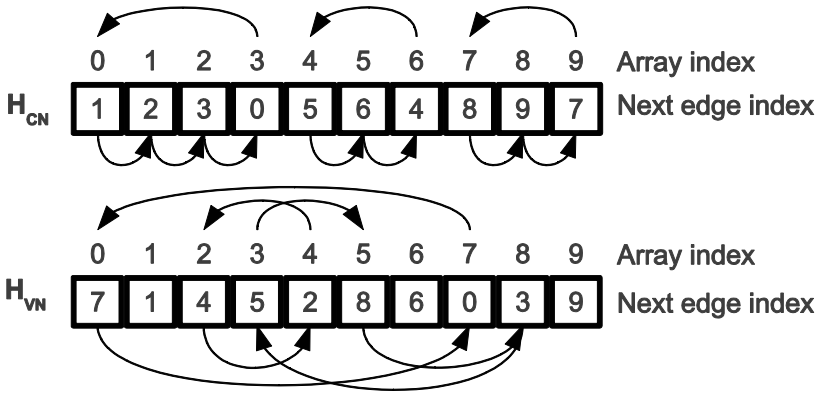

Figure 26: The arrays H_{CN} and H_{VN} corresponding to example H matrix.

A separate array structure, **M,** is used to store the actual messages passed between the variable and check node update phases. The **M** structure contains 128 messages for each one (edge) in **H**, corresponding to the 128 codewords being processed in parallel. Each entry in **M** is one byte in size. The structure is stored in memory so that messages corresponding to the same edge (belonging to different codewords) are arranged consecutively. The entry **M***(i×128+w)* thus contains the message corresponding to edge *i* for the *w*:th codeword.

Furthermore, two structures (arrays) $\mathbf{R}_f$ and $\mathbf{C}_f$ are used to point to the first element of rows and columns, respectively, of the **H** matrix. For the example **H** matrix, we have $\boldsymbol{R_f} = (0 \quad 4 \quad 7)$, and $\boldsymbol{C_f} = (0 \quad 1 \quad 2 \quad 3 \quad 9 \quad 6)$. The structure **LLR** contains the received initial beliefs for all codewords, and will have $n \times 128$ elements for an LDPC code of length n. $\boldsymbol{LLR}(x \times 128 + w)$ contains the initial belief for bit x of codeword w.

5.3.2 GPU Algorithms

In this subsection follows a more detailed description of the functionality in the GPU kernels. For the variable node update, each thread processes four consecutive codewords for one column of **H**, and similarly each thread of the check node update kernel will process one row of **H**. Thus, 32 consecutive threads will process one column or row for all 128 codewords.

The procedure for the variable node update is roughly as follows, given an LDPC code defined by an $(n-k) \times n$ parity check matrix. We launch $n \times 32$ threads in total.

1. Given global thread id t, we process column $c = \left\lfloor \frac{t}{32} \right\rfloor$ of **H**, and codewords $w = (t \mod 32) \times 4$ to $(t \mod 32) \times 4 + 3$.
2. Read four consecutive LLR values starting from $\boldsymbol{LLR}(c \times 128 + w)$ into 4-element vector $\boldsymbol{m}$. We expand these values to 16-bit precision to avoid wrap around problems in later additions.
3. Let $i = \boldsymbol{C_f}(c)$
4. For all edges in column c:
   4.1. Copy the four consecutive messages (8-bit) starting from $\boldsymbol{M}(i \times 128 + w)$ into 4-element vector $\boldsymbol{msg}$. This is achieved by reading one 32-bit word from memory.

4.2. Add, element wise, the elements of $\boldsymbol{msg}$ to the elements of $\boldsymbol{m}$, and store the results in $\boldsymbol{m}$.
4.3. Let $i = \boldsymbol{H_{VN}}(i)$. If $i = \boldsymbol{C_f}(c)$, we have processed all edges.

5. For all edges in column c:
   5.1. Again, copy four messages (8-bit) starting from $\boldsymbol{M}(i \times 128 + w)$ into 4-element vector $\boldsymbol{msg}$.
   5.2. Perform $\boldsymbol{m} - \boldsymbol{msg}$ (element-wise subtraction of four elements), clamp the resulting values to the range $[-127, +127]$ (since $\boldsymbol{m}$ contains 16-bit integers, and $\boldsymbol{msg}$ contains 8-bit integers) and store the result in $\boldsymbol{msg}$.
   5.3. Copy $\boldsymbol{msg}$ back to the memory positions of $\boldsymbol{M}(i \times 128 + w)$ to $\boldsymbol{M}(i \times 128 + w\ + 3)$.
   5.4. Let $i = \boldsymbol{H_{VN}}(i)$. If $i = \boldsymbol{C_f}(c)$, we have processed all edges.
6. Variable node update completed.

The check node update launches $(n - k) \times 32$ threads, and the procedure is the following:

1. Given global thread id t, we process row $r = \left\lfloor \frac{t}{32} \right\rfloor$ of $\boldsymbol{H}$, and codewords $w = (t \mod 32) \times 4$ to $(t \mod 32) \times 4 + 3$.
2. Define four 4-element vectors $\boldsymbol{sign}, \boldsymbol{min}, \boldsymbol{nmin}$ and $\boldsymbol{mi}$. Initialize elements of $\boldsymbol{sign}$ to 1, and elements of $\boldsymbol{min}$ and $\boldsymbol{nmin}$ to 127.
3. Let $i = \boldsymbol{R_f}(r)$.
4. Let $j = 0$ (iteration counter).
5. For all edges in row r:
   5.1. Copy four consecutive messages starting from $\boldsymbol{M}(i \times 128 + w)$ into 4-element vector $\boldsymbol{msg}$.
   5.2. For all element indices $x \in [0,3]$, if $|\boldsymbol{msg}(x)| < \boldsymbol{min}(x)$, let $\boldsymbol{min}(x) = |\boldsymbol{msg}(x)|$ and set $\boldsymbol{mi}(x) = j$. Otherwise, if $|\boldsymbol{msg}(x)| < \boldsymbol{nmin}(x)$, let $\boldsymbol{nmin}(x) = |\boldsymbol{msg}(x)|$.
   5.3. Also, for all $x \in [0,3]$, let $\boldsymbol{sign}(x)$ be negative if $\boldsymbol{msg}(x) \times \boldsymbol{sign}(x)$ is negative, and positive otherwise.
   5.4. Set j equal to $j + 1$.
   5.5. Let $i = \boldsymbol{H_{CN}}(i)$. If $i = \boldsymbol{R_f}(r)$, we have processed all edges.
6. Let $j = 0$.
7. For all edges in row r:
   7.1. Copy four consecutive messages starting from $\boldsymbol{M}(i \times 128 + w)$ into 4-element vector $\boldsymbol{msg}$.
   7.2. For all $x \in [0,3]$, if $\boldsymbol{mi}(x) \neq j$, let $\boldsymbol{msg}(x) = \text{sign}\big(\boldsymbol{sign}(x) \times \boldsymbol{msg}(x)\big) \times \boldsymbol{min}(x)$. Otherwise, if $\boldsymbol{mi}(x) = j$, let $\boldsymbol{msg}(x) = \text{sign}(\boldsymbol{sign}(x) \times \boldsymbol{msg}(x)) \times \boldsymbol{nmin}(x)$.
   7.3. Copy $\boldsymbol{msg}$ back to memory positions of $\boldsymbol{M}(i \times 128 + w)$ to $\boldsymbol{M}(i \times 128 + w + 3)$.
   7.4. Set j equal to $j + 1$.
   7.5. Let $i = \boldsymbol{H_{CN}}(i)$. If $i = \boldsymbol{R_f}(r)$, we have processed all edges.
8. Check node update completed.

The special variable node update kernel that includes hard decision, adds an additional step to the end of the variable node update kernel. Depending on if $\boldsymbol{m}(x)$, for $x \in [0,3]$, is positive or negative, a zero or one is written to index $c \times 128 + w + x$ of an array structure $\boldsymbol{B}$ as specified in the last step of the min-sum decoder procedure described in section 5.1. The $\boldsymbol{B}$ structure is copied back from the GPU to the host upon completed decoding.

5.3.3 CPU Algorithms

As mentioned, each single thread in the CPU version performs a larger amount of the total work than in the GPU case. As integer SSE instructions operating on 128-bit (16-byte) registers are used, 16 8-bit messages belonging to 16 different codewords are generally operated on in each SSE instruction. In the variable node update, each thread computes a fraction (depending on the preferred number of CPU threads) of the columns of $\boldsymbol{H}$ for all 128 codewords. Likewise, a check node update thread computes a fraction of the rows for all codewords. As in the GPU implementation, the lifetime of one CPU thread is one iteration of either a variable node update or a check node update.

The procedure for the variable node update is as follows, given an LDPC code defined by an $(n-k)\times n$ parity check matrix. We launch T_V threads, where the optimal T_V depends on factors such as CPU core count. Let $t \in [0, T_V - 1]$ denote the current thread. Hexadecimal values are written using the $0x$ prefix.

1. Given thread id t, we process columns $c \in \left[\frac{t\times n}{T_V}, \frac{(t+1)\times n}{T_V} - 1\right]$, and for each column, we process 8 groups of 16 codewords, $cg \in [0,7]$.
2. Let $w = (cg \times 16)$.
3. Read sixteen consecutive LLR values starting from $\boldsymbol{LLR}(c \times 128 + w)$ into 16-element vector $\boldsymbol{m}$.
4. Let $i = \boldsymbol{C}_f(c)$.
5. For all edges in column c:
   5.1. Copy the sixteen consecutive messages (8-bit) starting from $\boldsymbol{M}(i \times 128 + w)$ into 16-element vector $\boldsymbol{msg}$.
   5.2. Add, element-wise, the elements of $\boldsymbol{msg}$ to the elements of $\boldsymbol{m}$ and store the results in $\boldsymbol{m}$ (SSE PADDSB saturating addition instruction).
   5.3. Let $i = \boldsymbol{H}_{VN}(i)$. If $i = \boldsymbol{C}_f(c)$, we have processed all edges.
6. For all edges in column c:
   6.1. Copy the sixteen consecutive messages starting from $\boldsymbol{M}(i \times 128 + w)$ into 16-element vector $\boldsymbol{msg}$.
   6.2. Perform $\boldsymbol{m} - \boldsymbol{msg}$ and store result in $\boldsymbol{msg}$. The SSE PSUBSB saturating subtraction instruction is used for this.
   6.3. If any element in $\boldsymbol{msg}$ is equal to -128, set it to -127. Performed by comparing $\boldsymbol{msg}$ to a vector containing only -128 using the PCMPEQB instruction, followed by the PBLENDVB instruction to replace values of -128 with -127 in $\boldsymbol{msg}$.
   6.4. Copy $\boldsymbol{msg}$ back to the memory positions of $\boldsymbol{M}(i \times 128 + w)$ to $\boldsymbol{M}(i \times 128 + w + 15)$.
   6.5. Let $i = \boldsymbol{H}_{VN}(i)$. If $i = \boldsymbol{C}_f(c)$, we have processed all edges.
7. Variable node update completed.

In the CPU implementation there is also a special variable node update function including hard decision. This function calculates the hard decision using SSE instructions by right shifting the values of $\boldsymbol{m}$ by 7 bits, so that the sign bit becomes the least significant bit. All bits other than the least significant are set to zero, giving us the hard decision bit values as bytes. Elements equal to -128 are set to -127 in step 6.3 to make

the range of positive and negative values equal. Failing to do so was found to result in disastrous error correction performance.

The check node update launches T_C threads, and $t \in [0, T_C - 1]$ denotes the current thread. The procedure is the following:

1. Given thread id t, we process rows $r \in \left[\frac{t\times(n-k)}{T_C}, \frac{(t+1)\times(n-k)}{T_C} - 1\right]$, and for each column, we process 8 groups of 16 codewords, $cg \in [0,7]$.
2. Let $w = (cg \times 16)$.
3. Define 16-element vectors $\boldsymbol{sign}$, $\boldsymbol{min}$, $\boldsymbol{nmin}$, and $\boldsymbol{mi}$. Initialize elements of $\boldsymbol{sign}$ to 1, and elements of $\boldsymbol{min}$ and $\boldsymbol{nmin}$ to 127.
4. Let $i = \boldsymbol{R}_f(r)$.
5. Let $j = 0$ (iteration counter).
6. For all edges in row r:
   6.1. Copy sixteen consecutive messages starting from $\boldsymbol{M}(i \times 128 + w)$ into vector $\boldsymbol{msg}$.
   6.2. Compute $\boldsymbol{sign} \oplus \boldsymbol{msg}$, and store result in $\boldsymbol{sign}$. SSE PXOR instruction for bitwise XOR operation on two 128-bit registers is used.
   6.3. Compute element-wise absolute values of $\boldsymbol{msg}$, and store result in $\boldsymbol{msg}$, using the SSE instruction PABSB for absolute value.
   6.4. $\forall e \in [0,15]$, let the value of $\boldsymbol{mask}_1(e)$ be $0xFF$ if $\boldsymbol{msg}(e) < \boldsymbol{min}(e)$, and $0x00$ otherwise. The SSE instruction PCMPGTBr accomplishes this.
   6.5. $\forall e \in [0,15]$, let the value of $\boldsymbol{mask}_2(e)$ be $0xFF$ if $\boldsymbol{msg}(e) < \boldsymbol{nmin}(e)$, and $0x00$ otherwise (PCMPGTBr instruction).
   6.6. $\forall e \in [0,15]$, let $\boldsymbol{temp}(e) = \boldsymbol{min}(e)$ if $\boldsymbol{mask}_1(e)$ equals $0xFF$, and otherwise let $\boldsymbol{temp}(e) = \boldsymbol{msg}(e)$. The SSE instruction PBLENDVB is used.
   6.7. $\forall e \in [0,15]$, let $\boldsymbol{nmin}(e) = \boldsymbol{temp}(e)$ if $\boldsymbol{mask}_2(e)$ equals $0xFF$, and otherwise let $\boldsymbol{nmin}(e) = \boldsymbol{nmin}(e)$ (PBLENDVB).
   6.8. $\forall e \in [0,15]$, let $\boldsymbol{min}(e) = \boldsymbol{msg}(e)$ if $\boldsymbol{mask}_1(e) = 0xFF$, and otherwise let $\boldsymbol{min}(e) = \boldsymbol{min}(e)$ (PBLENDVB).
   6.9. $\forall e \in [0,15]$, let $\boldsymbol{mi}(e) = j$ if $\boldsymbol{mask}_1(e) = 0xFF$, and otherwise let $\boldsymbol{mi}(e) = \boldsymbol{mi}(e)$ (PBLENDVB).
   6.10. Set j equal to $j + 1$.
   6.11. Let $i = \boldsymbol{H}_{CN}(i)$. If $i = \boldsymbol{R}_f(r)$, we have processed all edges.
7. Let $j = 0$.
8. $\forall e \in [0,15]$, let $\boldsymbol{sign}(e)$ equal -1 $(0xFF)$ if $\boldsymbol{sign}(e) < 0$, and 0 otherwise. This is accomplished by the SSE PCMPGTBr instruction (compare to zero vector).
9. For all edges in row r:
   9.1. Copy sixteen consecutive messages starting from $\boldsymbol{M}(i \times 128 + w)$ into vector $\boldsymbol{msg}$.
   9.2. $\forall e \in [0,15]$, let the value of $\boldsymbol{mask}_1(e)$ be $0xFF$ if $\boldsymbol{mi}(e) = j$, and $0x00$ otherwise. SSE instruction PCMPEQB accomplishes this.
   9.3. $\forall e \in [0,15]$, let the value of $\boldsymbol{mask}_2(e)$ be $0xFF$ if $\boldsymbol{msg}(e) < 0$, and $0x00$ otherwise (PCMPGTBr).
   9.4. $\forall e \in [0,15]$, let $\boldsymbol{mask}_3(e) = \boldsymbol{sign}(e) \oplus \boldsymbol{mask}_2(e)$ (PXOR).
   9.5. $\forall e \in [0,15]$, let $\boldsymbol{mask}_3(e) = \boldsymbol{mask}_3(e) \vee 1$ (SSE POR instruction).

9.6. $\forall e \in [0,15]$, let $\boldsymbol{temp}_1(e)$ equal $-\boldsymbol{min}(e)$ if $\boldsymbol{mask}_3(e) < 0$, and $\boldsymbol{min}(e)$ otherwise. The SSE instruction PSIGNB is used for this.
9.7. $\forall e \in [0,15]$, let $\boldsymbol{temp}_2(e)$ equal $-\boldsymbol{nmin}(e)$ if $\boldsymbol{mask}_3(e) < 0$, and $\boldsymbol{nmin}(e)$ otherwise (PSIGNB).
9.8. $\forall e \in [0,15]$, let $\boldsymbol{msg}(e) = \boldsymbol{temp}_2(e)$ if $\boldsymbol{mask}_1(e)$ equals $0xFF$, and otherwise let $\boldsymbol{msg}(e) = \boldsymbol{temp}_1(e)$ (PBLENDVB).
9.9. Copy $\boldsymbol{msg}$ back to the memory positions of $\boldsymbol{M}(i \times 128 + w)$ to $\boldsymbol{M}(i \times 128 + w + 15)$.
9.10. Set j equal to $j + 1$.
9.11. Let $i = \boldsymbol{H}_{CN}(i)$. If $i = \boldsymbol{R}_f(r)$, we have processed all edges.
10. Check node update completed.

5.3.4 Optimization strategies - GPU

Notice that, in both main CUDA kernels, the same four elements are copied to $\boldsymbol{msg}$ from $\boldsymbol{M}$ twice (once in each loop). The second read could have been avoided by storing the elements into fast on-chip shared memory the first time. Through experiments, however, it was observed that significantly improved performance could be reached by not reserving the extra storage space in shared memory. This is mostly due to the fact that we can instead have a larger number of active threads at a time on an SM, when each thread requires fewer on-chip resources. A larger number of active threads can effectively "hide" the latency caused by global memory accesses.

Significant performance gains were also achieved by using bit twiddling operations to avoid branches and costly instructions such as multiplications in places where they were not necessary. The fact that this kind of optimizations had a significant impact on performance suggests that this implementation is instruction bound rather than memory access bound despite the many scattered memory accesses performed in the decoder. Through profiling of the two main kernels, the ratio of instructions issued per byte of memory traffic to or from global memory was found to be significantly higher than the optimum values suggested in optimization guidelines [43], further suggesting that the kernels are indeed instruction bound.

An initial approach at an LDPC decoder more closely resembled the implementation described in [42], in that one thread was used to update one message, instead of having threads update all connected variable nodes or check nodes. This leads to a larger number of quite small and simple kernels. This first implementation was however significantly slower than the currently proposed implementation. One major benefit of the proposed approach is that fewer redundant memory accesses are generated, especially for codes where the average row and/or column degree is high.

As mentioned in section 5.2.1, the Fermi architecture allows the programmer to choose between 16 kB of shared memory and 48 kB of L1 cache, or vice versa. The 48 kB L1 cache setting was chosen for the final implementation, as no shared memory was used. This clearly improved performance compared to the alternative setting.

5.3.5 Optimization strategies - CPU

On the CPU, choosing a significantly higher value for the number of threads (T_V and T_C) per variable or check node update iteration than the number of logical cores in the test setup improved performance significantly. On the test system, $T_V = T_C = 32$ was found to be a good value, although only 8 logical cores were present. It was also found important to process the 8 groups of 16 codewords for a particular row or column of $\boldsymbol{H}$ before processing another row/column, in order to improve cache utilization. Bit twiddling operations played an even more important role on the CPU than on the GPU, due to the fact that, for example, there is no 8-bit integer multiplication instruction in SSE.

It is worth noting that while the intermediate result $\boldsymbol{m}$ was expanded to a 16-bit integer in the variable node update on the GPU, precision was kept at 8-bit throughout the operation on the CPU. Expanding the intermediate values in an SSE-based implementation would have required many extra operations, sacrificing performance. This solution leads to a somewhat less precise CPU decoder. In section 5.4.3, the error correction performances of the GPU and CPU implementations are compared.

5.4 Performance

In this section, performance figures for both the CUDA-based and SSE SIMD-based LDPC decoders presented in section 5.3 are presented, both in terms of throughput and error correction performance. It is shown that the GPU implementation achieved throughputs required by the DVB-T2 standard with acceptable error correction performance.

5.4.1 Throughput Measurements

The system described in section 5.2 was used for benchmarking the two min-sum LDPC decoders. Decoder throughput was measured by timing the decoding procedure for 128 codewords processed in parallel, and dividing the codeword length used (16200 bits for short code length, and 64800 bits for long code) times 128 by the time consumed. Thus, the throughput measure does not give the actual useful bitrate, but rather the bitrate including parity data. To gain an approximate useful bitrate, the throughput figure must be multiplied by the code rate. The decoder was benchmarked for both the short and long codeword lengths supported by the DVB-T2 standard. Moreover, three different code rates were measured: 1/2, 3/4, and 5/6.

For the GPU implementation, the time measured included copying LLR values to the GPU, running a message initialization kernel, running the variable node and check node update kernels for as many iterations as desired before running the variable node update kernel including hard decision, and finally copying the hard decisions back to host memory. Timing the CPU version included the same steps, except transferring data to and from the GPU, which is not necessary in that case. In these benchmarks, checking whether we had actually arrived at a valid codeword was not included. This task was instead handled by the BCH decoder. If desired, we can check the validity of a codeword at a throughput penalty (penalty depending on how often we check for validity). This may for example be done together with hard decision in order to be able to terminate the decoder early upon successful recovery of all 128 codewords. In this case, however, we specify a set number of iterations to run before one final hard decision. Note that the $\boldsymbol{H}_{CN}$ and $\boldsymbol{H}_{VN}$ structures only need to be transferred to the GPU at decoder initialization (i.e. when LDPC code parameters change), and that this time is thus not included in the measured time.

The measured throughputs of the GPU implementation are presented in Table 7 for long code, and in Table 8 for short code configurations. The corresponding throughput figures for the CPU implementation are presented in Table 9 and Table 10. 10 batches of 128 codewords were decoded and the average time as well as the maximum time for decoding a batch was recorded. These times were used to calculate the average throughput as well as a minimum throughput (shown within parentheses in the tables) for each configuration.

Table 7: GPU decoder average throughput in Mbps (Megabits per second), long code ($n = 64800$). Minimum throughput in parentheses.

Rate	20 iterations	30 iter.	50 iter.
1/2	163.4 (160.1)	112.5 (110.9)	69.3 (68.7)
3/4	164.1 (160.6)	112.9 (111.4)	69.5 (68.9)
5/6	157.2 (153.9)	107.9 (106.3)	66.4 (65.9)

Table 8: GPU decoder average throughput in Mbps, short code ($n = 16200$). Minimum throughput in parentheses.

Rate	20 iterations	30 iter.	50 iter.
1/2	186.1 (179.4)	128.6 (125.1)	79.5 (78.2)
3/4	192.4 (185.2)	133.1 (129.6)	82.4 (81.0)
5/6	189.6 (181.8)	131.2 (127.3)	81.2 (79.7)

Table 9: CPU decoder average throughput in Mbps, long code ($n = 64800$). Minimum throughput in parentheses.

Rate	20 iterations	30 iter.	50 iter.
1/2	44.5 (43.4)	30.6 (30.0)	18.7 (18.4)
3/4	42.1 (40.8)	28.8 (28.4)	17.5 (17.4)
5/6	40.1 (38.5)	27.6 (27.4)	17.0 (16.8)

Table 10: CPU decoder average throughput in Mbps, short code ($n = 16200$). Minimum throughput in parentheses.

Rate	20 iterations	30 iter.	50 iter.
1/2	47.4 (44.3)	30.4 (28.5)	18.2 (16.5)
3/4	47.5 (45.7)	30.6 (28.5)	19.1 (17.3)
5/6	45.8 (43.4)	29.7 (27.4)	17.5 (15.5)

5.4.2 Results discussion

Annex C of the DVB-T2 standard assumes that received cells can be read from a deinterleaver buffer at 7.6×10^6 OFDM (orthogonal frequency-division multiplexing) cells per second. At the highest modulation mode supported by DVB-T2, 256-QAM, we can represent 8 bits per cell. This means that the LDPC decoder should be able to perform at a bitrate of at least 60.8 Mbps (Megabits per second). As seen from the results, the proposed GPU implementation is able to meet this realtime constraint even while performing 50 iterations.

DVB-S2 and DVB-C2 use the same codeword lengths as DVB-T2, though they specify partly different sets of code rates to suite their application domains. DVB-C2 may require processing up to 7.5×10^6 cells per second, which, coupled with a maximum modulation mode of 4096-QAM, gives us 90 Mbps maximum required throughput. DVB-S2 also may require about 90 Mbps maximum throughput [44]. By interpolation of the values in Table 7, it seems that the throughput requirements of these standards could be met at up to roughly 35 iterations.

From Table 9 and Table 10 we see the throughputs of the CPU decoder at 20, 30, and 50 iterations. We can see that the CPU implementation generally performs at slightly higher than 25% of the throughput of the GPU implementation. As the throughput increases quite linearly with a decreasing maximum number of iterations, we can derive that about 12 iterations should give us the required maximum bitrate of the DVB-T2 standard (60.8 Mbps). Indeed simulations at the slowest setting, 5/6-rate long code, revealed that at 12 iterations, 63.7 Mbps throughput was achieved with the CPU. This low amount of iterations would have a significant negative impact on error correction performance, which is demonstrated in section 5.4.3.

It should be noted that the throughput of the CPU implementation is the throughput when the CPU is completely dedicated to the task of decoding LDPC codewords. In a single processor system running a software defined receiver, this would not be the case. The CPU capacity would in that case need to be shared among all the signal processing blocks in the receiver chain (in addition to tasks such as video and audio decoding). In this respect, the GPU implementation yields an advantage in addition to higher throughput. If the GPU is assigned the task of LDPC decoding, the CPU is free to perform other tasks.

Figure 27 shows throughput of the CPU implementation (1/2-rate long code, 20 iterations) as a function of varying the amount of threads (T_V and T_C) when different numbers of cores are available to the decoder. It should be noted that a core in Figure 27 refers to a physical core, which consists of two logical cores, due to the presence of Intel Hyper-Threading technology. The Intel Turbo Boost feature, which allows a core to run at a higher than default clock frequency when other cores are idle, was disabled during this measurement. The speedup factors when utilizing two, three, and four physical cores with the optimal amount of threads are 1.9, 2.6, and 3.1, respectively. Varying the amount of cores used on the GPU is, to the authors' knowledge, not possible, and a similar scalability study was thus not performed on the GPU.

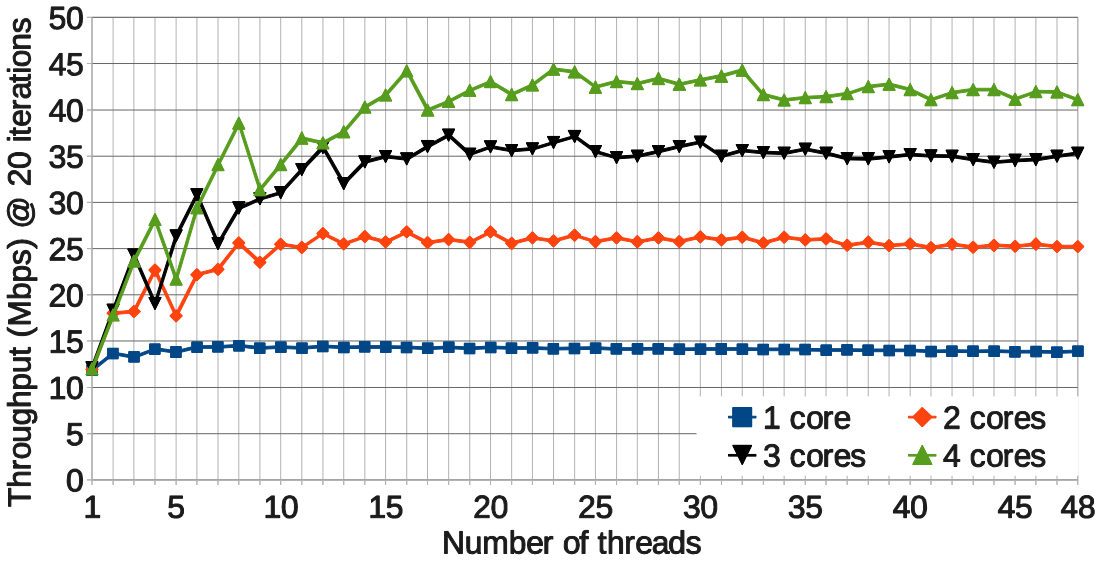

Figure 27: CPU decoder throughput with 1/2-rate long code at 20 iterations as a function of the number of threads (T_V and T_C). Different curves for 1 to 4 cores available to the decoder.

5.4.3 Error Correction Performance

Many dedicated hardware LDPC decoders use a precision of 8 bits or less for messages, and should thus have similar or worse error correction performance compared to the proposed implementations. Within the simulation framework used for testing the decoder, however, high-precision implementations of LDPC decoders using both the sum-product algorithm (SPA), as well as the min-sum algorithm, were available. These implementations were written for a standard x86-based CPU, and used 32-bit floating point message representation.

Simulations of DVB-T2 transmissions using both high-precision CPU-based implementations as well as the proposed GPU-based and CPU-based implementations, were performed in order to determine the cost of the lower precision of message representations as well as the use of min-sum over SPA in terms of decoder error correction capability.

Figure 28 and Figure 29 shows simulation results for a 16-QAM configuration at the code rates 1/2 and 5/6, respectively, of the long code. The simulations were performed on signal-to-noise ratio (SNR) levels 0.1 dB apart.

When simulating using the high-precision CPU implementations, 2000 codewords were simulated for each SNR level. As the proposed implementations were orders of magnitude faster, 16000 codewords were simulated per SNR level for these implementations, in order to be able to detect possible low error floors. The average bit error rate (BER) was calculated by comparing the sent and decoded data. A channel model simulating an AWGN (additive white Gaussian noise) channel was used. The maximum number of LDPC decoder iterations allowed was set to 50.

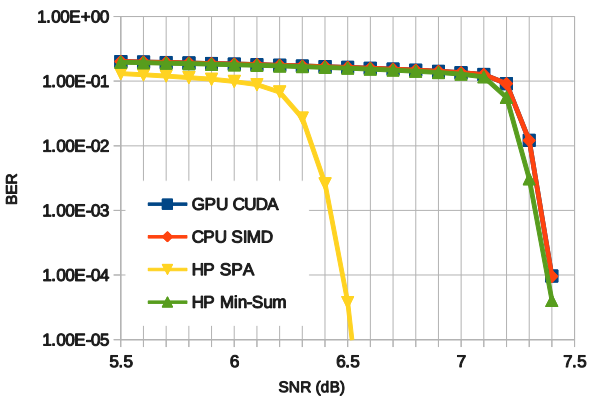

Figure 28: Simulation results for 16-QAM long code 1/2-rate configuration when using the proposed CUDA GPU and SSE SIMD CPU implementations, as well as high precision (HP) CPU implementations of SPA and min-sum algorithms.

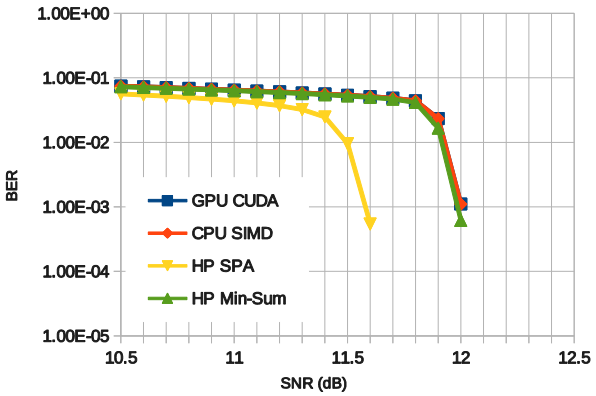

Figure 29: Simulation results for 16-QAM long code 5/6-rate.

As can be seen in Figure 28 and Figure 29, the proposed lower precision GPU and CPU implementations perform very close (within 0.1 dB) to the high-precision min-sum CPU implementation on the AWGN channel. The simulations clearly indicate that the impact of using the simplified min-sum algorithm as opposed to the superior SPA algorithm is much greater than the choice of message precision. The error correction performance advantage of the SPA algorithm also remains relatively small (please note the fine scale of the x-axes in the figures), however, with slightly less than a 1 dB advantage for 1/2-rate and roughly 0.5 dB for 5/6-rate at a BER level of 10^{-4}.

As mentioned in section 5.4.2, the CPU implementation could perform only 12 iterations in order to reach the maximum required throughput of DVB-T2, while the GPU implementation manages to perform in excess of 50 iterations under the same constraints. In Figure 30, it is demonstrated how varying the amount of maximum iterations performed by the proposed CPU min-sum decoder implementation impacts error correction performance. The figure shows simulation results for a 16-QAM configuration, with 1/2-rate long code over an AWGN channel. All SNR levels were simulated over 2048 codewords. Figure 30 reveals that 12 iterations of the min-sum decoder does not yield very good error correction performance. The difference between 12 and 50 iterations is roughly 0.7 dB at a BER level of 10^{-4}, which is perhaps not a great amount. At 12 iterations, however, the steepness of the "waterfall" region of the SNR-BER curve is notably worse than at 50 iterations, which is undesirable. Figure 30 also shows that 30 iterations does not give significantly worse results than 50 iterations.

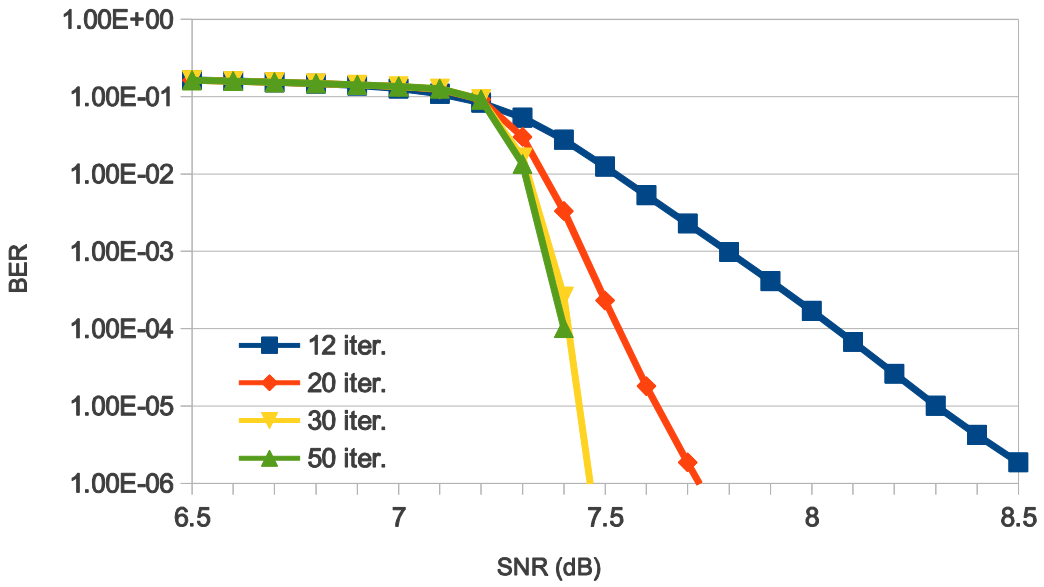

Figure 30: Simulation results for 16-QAM 1/2-rate long code configuration when varying the maximum number of LDPC decoder iterations. Simulations were performed using the proposed SIMD CPU implementation.

5.5 Conclusion

In this report, two implementations of LDPC decoders optimized for decoding the long codewords specified by the next generation digital television broadcasting standards DVB-T2, DVB-S2, and DVB-C2 have been presented. The GPU implementation is a highly parallel decoder optimized for a modern GPU architecture. The throughputs required by these standards at high numbers of iterations were reached, giving good error correction performance. It was also shown that a modern multi-core SIMD-enabled CPU is capable of quite high throughputs, though perhaps not quite enough for the most demanding configurations of the DVB standards.

In [33], it was shown that besides the LDPC decoder, the QAM constellation demapper − converting received constellation points in the complex plane to LLR values − is one of the most computationally complex blocks in a DVB-T2 receiver chain. As the demapper produces the input to the LDPC decoder (a bit deinterleaver does however separate the two signal processing blocks), a good next step would be to perform both the demapping and LDPC decoding on the GPU, further reducing the main CPU load.

6 MIMO DETECTION

6.1 Receiver structure

At the receiver side, the signal received after propagation through the MIMO equivalent channel **H** expresses simply as:

$$\mathbf{y} = \mathbf{Hx} + \mathbf{n},$$

where **y** is the matrix of the received symbols of size $N_r \times T$. The corresponding generic receiver is depicted in Figure 31. The multi-antenna equalizer takes T symbols per receive antenna and their corresponding channel estimates, *i.e.* $N_t \times N_r$ sub-channel estimates, in order to produce the estimate $\hat{S}_q$ of the transmitted symbol S_q.

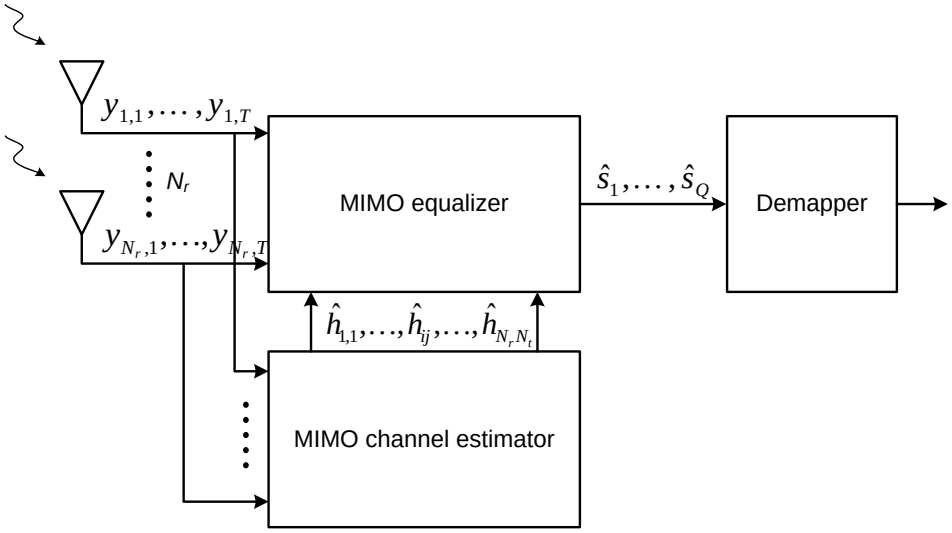

Figure 31 Generic multi-antenna receiver structure.

As detailed later on, different decoding strategies can be driven by the equalizer, depending on the type of the ST coding scheme and on complexity considerations. For example, orthogonal STBC (OSTBC) schemes yield simple maximum likelihood (ML) receiver structures, while non-orthogonal STBC need more complex decoding algorithms, either derivated from the ML approach or based on iterative interference cancellation structures. In any case, note that in SISO mode, the multi-antenna equalizer block acts exactly as a channel equalizer.

6.2 Complexity Analysis on Maximum-Likelihood MIMO Decoding

Although optimal bit-error performance is obtained with a maximum-likelihood decoder (MLD) it has the disadvantage that the complexity grows exponentially with the number of transmit antennas. Although the number of transmit antennas specified for MIMO in DVB-NGH standard is relatively small, complexity might still be an issue for low-cost portable devices, and it is worth investing in new techniques to reduce its complexity.

The first step in reducing the complexity of the decoder is to simplify the log-likelihood ratio (LLR) calculation for soft-decision by using the max-log approximation on the LLR. It was shown that the performance penalty is very small, less than 0.05 dB, for a 2-by-2 16QAM system in the context of the DVB NGH channel model. This is significantly lower than a typical hardware implementation margin.

Secondly, there has been significant research in reducing the complexity of MLD by first decomposing the MIMO channel matrix using the QR decomposition **H** = **QR**, which results in **Q**, an orthonormal matrix, and **R**, an upper triangular matrix with real diagonal values. The QR-decomposition can form the basis of reduced-complexity decoding as follows:

It is well known that finding the ML solution is equivalent to solving:

$$\hat{\mathbf{s}}_{ML} = \arg\min_{\mathbf{s}\in D} \left\| \mathbf{x} - \mathbf{Hs} \right\|^2, \quad (1)$$

where D is the search-space, **x** the received vector, **H** is the channel matrix and **s** is the transmitted vector. The QR-based decoder will first decompose **H** into **Q** and **R**, hence the ML solution will now be solving:

$$\hat{\mathbf{s}}_{ML} = \arg\min_{\mathbf{s}\in D} \left\| \mathbf{y} - \mathbf{Rs} \right\|^2, \quad (2)$$

where $\mathbf{y} = \mathbf{Q}^{\mathbf{H}}\mathbf{x}$ and the squared norm remains unaltered.

An indication of the complexity can be done by calculating the number of multiplication and addition operations required by the decoders for the max-log LLR calculation:

- MLD (2-by-2 MIMO)
  - (13 multipliers & 15 adders) x 2^{b1}
  - (13 multipliers & 15 adders) x 2^{b2}
- QR-based MLD (2-by-2 MIMO) – excluding QR decomposition
  - (5 multipliers & 6 adders) x 2^{b1}
  - (11 multipliers & 11 adders) x 2^{b2}
  - 16 multipliers & 12 adders

The $b1$ and $b2$ corresponds to the number of bits in the QAM constellation for the first and second symbol respectively. The number of multipliers/adders required by the QR decomposition depends on implementation and known to be small for a 2-by-2 matrix. Table 11 illustrates three possible QAM combinations for the 2-by-2 MIMO system.

Table 11: Resource usage for different QAM combination. Calculations do not include resources needed for QR decomposition for the QR-based MLD.

Resource	16QAM / 16QAM	16QAM / 64QAM	64QAM / 64QAM

Decoder	Multipliers	Adders	Multipliers	Adders	Multipliers	Adders
MLD	416	480	1040	1200	1664	1920
QR-based MLD	272	284	512	572	1040	1100
Savings	144	196	528	628	624	820

This shows that the QR-based MLD saves around 35%-51% multipliers and 41%-52% adders on first inspection before taking into account the resources required for QR decomposition (for the QR-based MLD). It is worth noting that the QR decomposition is only done once for every received vector.

The sphere decoding technique is another way to reduce the complexity of the decoder. The sphere decoders can be classified as a QR-based decoder and it has been known by different names throughout the research community because of its slight variant. The MLD decoding structure can be illustrated in a tree diagram as shown in Figure 32 and the search space is represented by the points at the lowest level (Layer 1).

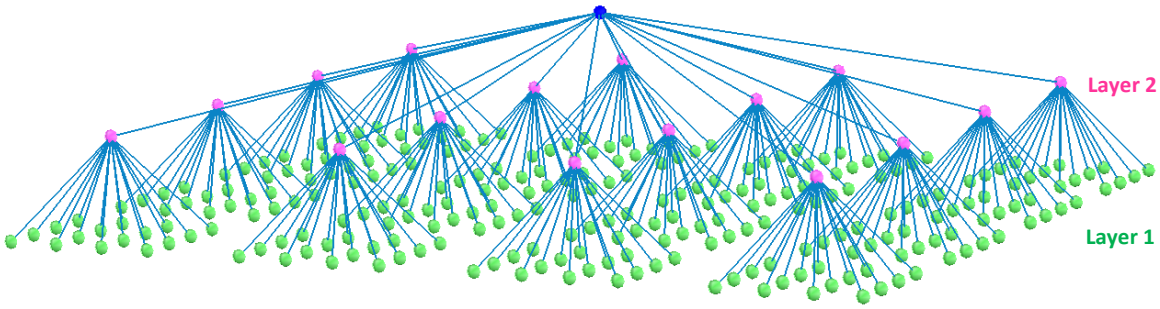

Figure 32: MIMO 2x2 tree diagram (16QAM, 16QAM).

The hard-decision MLD searches over the entire search space for the most probably transmitted symbol based on the received vector while the sphere decoder searches over a fraction of the search space by using an iterative process and boundary conditions. The sphere decoding concept can also be extended to soft-decision and it has become a good choice for MIMO decoding. However, there are still challenges in implementing sphere decoder in VLSI because of practical tradeoffs and general assumptions.

As the Space-Time codewords can be seen as a subset points of certain "lattice", the ML decoding can be recognized as searching the nearest lattice point to a given (received) point. A visual illustration of the ML decoding is given in Figure 33. As shown in the figure, the number of lattice points which are found inside a sphere is significantly smaller than the number of all possible candidates. Meanwhile, the nearest lattice point within the sphere centered by the given point is also the nearest point among all candidates. To avoid the exhaustive search of all combinations of the Space-Time codewords, the sphere decoding searches only among the points of the lattice which are located inside the sphere. This ensures only a few lattice points with more "potential" are involved in the searching processing. Therefore, by carefully selecting the radius of the sphere, the ML solution can be found by sphere decoding with much less searching complexity. Extensive description of the sphere decoder is suggested to refer to [45].

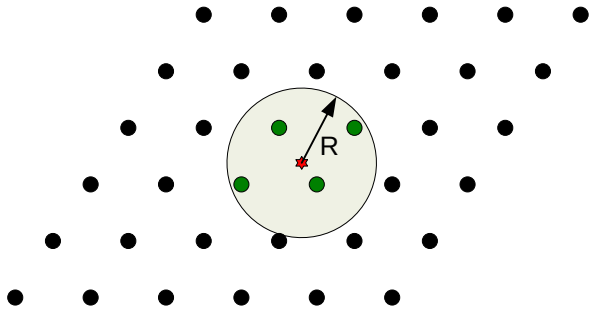

Figure 33 Principle of the sphere decoder.

It is worth mentioning that basic linear decoders such as zero-forcing and MMSE equalisers are simple and easy to implement in hardware but produce sub-optimal bit-error performances. The complexity of such decoders is not determined by the size of the QAM modulation like in MLD and the resource usage is just a very small fraction compared to MLD.

6.3 Iterative Space-Time decoding

In the case of OSTBC (Orthogonal Space-Time Block Code), the data stream is divided into several orthogonal subchannels. Hence the optimal receiver for OSTBC is made of a concatenation of ST decoder and channel decoder modules. In NO-STBC schemes, there is an inter-antenna interference (IAI) at the receiving side. The optimal receiver in this case is based on joint ST and channel decoding operations. However such receiver is extremely complex to implement and requires large memory to store the different points of the trellis. Thus the sub-optimal solution proposed here consists of an iterative receiver where the ST detector and channel decoder exchange extrinsic information in an iterative way until the algorithm converges. The iterative detection and decoding exploits the error correction capabilities of the channel code to provide improved performance. This is achieved by iteratively passing soft *a priori* information between the detector and the soft-input soft-output decoder. A more detailed description of this iterative receiver is given in Figure 34.

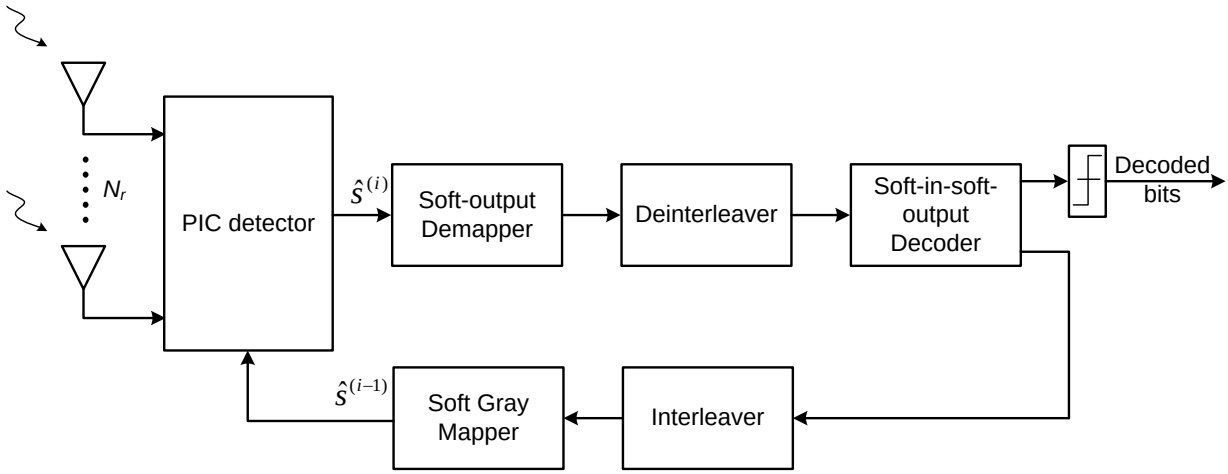

Figure 34 Iterative ST receiver structure.

6.4 Iterative MIMO decoding for DVB-NGH

DVB-NGH (Next Generation Handheld) is the next generation of mobile TV broadcasting standard developed by the DVB project. It is the mobile evolution of DVB-T2 (Terrestrial 2nd Generation) 6.4 and its deployment is motivated by the continuous grow of mobile multimedia services to handheld devices such tablets and smart-phones [47]. The main objective of DVB-NGH is to increase the coverage area and system capacity outperforming the existing mobile broadcasting standards DVB-H (Handheld) and DVB-SH (Satellite services to Handheld devices). DVB-T2 and therefore DVB-NGH, introduces the concept of Physical Layer Pipe (PLP) in order to support a per service configuration of transmission parameters, including modulation, coding and time interleaving. The utilization of multiple PLPs allows for the provision of services targeting different user cases, i.e. fixed, portable and mobile, in the same frequency channel. The main new additional characteristics of DVB-NGH compared to DVB-T2 are: use of SVC (Scalable Video Coding) for efficient support for heterogeneous receiving devices and varying network conditions, TFS (Time Frequency Slicing) for increased capacity and/or coverage area, efficient time interleaving to exploit time diversity, RoHC (Robust Header Compression) to reduce the overhead due to signaling and encapsulation, additional satellite component for increased coverage area, improved signaling robustness compared to DVB-T2, efficient implementation of local services within SFN (Single Frequency Networks) and finally, implementation of multi-antenna techniques (MIMO) for increased coverage area and/or system capacity.

The utilization of multi antenna techniques at both sides of the transmission link (MIMO) is a key technology that allows for significant increased system capacity and network coverage area. It is already included in fourth-generation (4G) cellular communication systems, e.g. *Worldwide Interoperability for Microwave Access* (WiMAX) and 3GPP´s *Long-Term Evolution* (LTE), and internet wireless networks, e.g. *Wireless Local Area Networks* (WLAN), to cope with the increasing demand of high data rate services. DVB-NGH is the first world´s broadcast system to include MIMO technology.

The gains achieved with MIMO can be further increased with the combination of iterative detection where the MIMO demapper and channel decoder exchange extrinsic information in an iterative fashion providing large gains. One big advantage of iterative demapping is that it only affects the receiver side and therefore no modification is required in standards and transmitters. However, iterative decoding significantly increases the receiver complexity, making it less suited for mobile devices. To reduce the computational complexity, numerous suboptimal MIMO receivers have been proposed, e.g. linear zero-forcing (ZF) and minimum mean square error (MMSE) receivers.

In this section we study the gains provided by MIMO in combination with iterative decoding (MIMO ID) in vehicular environments. The performance of optimal MIMO ID is compared with suboptimal MIMO ID based on MMSE filtering with a priori inputs. First the fundamentals of MIMO demodulation and complexity are described. The iterative decoding process for both, optimal decoding and suboptimal decoding based on MMSE with a priori inputs are presented. Then, the simulation setup (i.e. channel model employed and system parameters) is given and the physical layer simulation results discussed.

6.4.1 MIMO demodulation and complexity

The task of the demapper is to provide LLRs (*Log Likelihood Ratios*) to the channel decoder with reliability information of the transmitted code bits. The *optimum soft MAP (Maximum a posteriori) demapper* computes the LLR of the transmitted bit c_l with the received vector **y** and the channel estimates **H** with the

following expression

$$\Lambda_l = \log \frac{f(c_l = 1 \mid \mathbf{y}, \mathbf{H})}{f(c_l = 0 \mid \mathbf{y}, \mathbf{H})} = \ln \frac{\sum_{x \in \chi_l^1} \exp\left(- \frac{\|\mathbf{y} - \mathbf{Hx}\|^2}{\sigma_w^2} \right)}{\sum_{x \in \chi_l^0} \exp\left(- \frac{\|\mathbf{y} - \mathbf{Hx}\|^2}{\sigma_w^2} \right)}, \quad (1)$$

where σ_w^2 denotes the noise variance and χ_l^b denotes the set of transmit vectors for which c_l equals $b \in \{0, 1\}$. The computational complexity grows exponentially with the number of transmit antennas, being prohibitive even for small number of antennas. In the literature there are a vast number of algorithms and approximations to reduce the complexity. Max-log demapper applies the max-log approximation

$$\log \sum_m \exp(a_m) \approx \max_m a_m \,, \quad (2)$$

transforming (1) into the next formula

$$\tilde{\Lambda}_l = \frac{1}{\sigma_w^2} \left[\min_{x \in \chi_l^0} \|\mathbf{y} - \mathbf{Hx}\|^2 - \min_{x \in \chi_l^1} \|\mathbf{y} - \mathbf{Hx}\|^2 \right], \quad (3)$$

with a small degradation penalty [49].

Max-log approximation eases receiver implementation due to logarithm and exponential computations are changed by minimum distances calculations. Still the complexity grows exponentially with the number of transmit antennas.

Nonlinear techniques like sphere decoding further reduce the complexity finding the most likely transmitted symbol from a subset of the original ML search. Significant reduction of the receiver complexity can be obtained with linear techniques like zero forcing (ZF) and minimum mean squared error (MMSE). They apply a linear equalizer to the receive data which cancels the multi-stream interference transforming the MIMO detection problem into several independent SISO problems. Zero forcing eliminates the multi-stream interference but enhances the noise degrading the performance. MMSE equalizer trades-off interference cancellation and noise enhancement. The complexity of linear equalizer demappers scales polynomically with the number of transit antennas, significantly lower than max-log demapping.

6.4.2 Optimal and Suboptimal Iterative detection

Exploit of time, frequency and space diversity in combination with LDPC codes in BICM systems achieve spectral efficiencies very close to Shannon´s capacity limit theorem. Iterative detection reduces this gap even more. Extrinsic information is exchanged between demapper and channel decoder in an iterative manner [50]. The demapper computes extrinsic LLRs with the received vector of symbols and a priori information coming from the channel decoder. The computed extrinsic LLRs are de-interleaved to become a priori information to be fed to the channel decoder. After decoding operation the improved LLRs are used to

extract the extrinsic information, which is interleaved and fed to the demapper closing the iteration loop as it is illustrated in Figure 35. Each iteration improves the performance of the decoded stream until saturation point. After certain desired quality is achieved, the LLR decoder outputs are used for hard-decisions obtaining the final decoded bit stream.

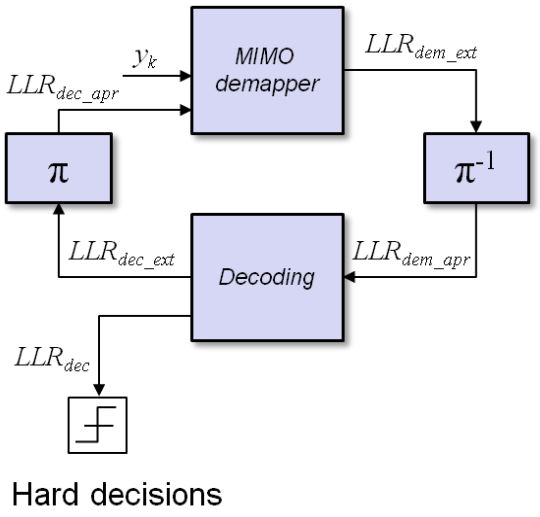

Figure 35: Iterative exchange of extrinsic information between demapper and channel decoder.

Iterative detection provides large gains at cost of higher computational complexity. The complexity increases linearly with the number of outer iterations due to the repetition of MIMO demapping and channel decoder operations, making in some cases inaccessible its real implementation. Design of number of iterations performed at the receiver (i.e. iterations of LDPC decoder and number of outer iterations) for efficient exchange of extrinsic information is out of the scope of this paper.

As explained previously, optimal MAP demapping requires high complexity due to it computes comparisons with all possible received signals. Lower complexity sub-optimal receivers based on linear equalization include ZF or MMSE. Linear equalizers reduce multi-stream interference transforming the joint MIMO demapping problem into several independent SISO problems. Therefore the receiver complexity is significantly reduced scaling polynomically with the number of transmit antennas in comparison with the exponential grow of the reference max-log MIMO demapper.

Iterative MIMO demapping can exploit the complexity reductions offered by linear equalization but exploiting the gains provided by iterative decoding. The estimates of the MMSE equalization can be improved with the information coming from the channel decoder, i.e. MMSE equalization with a priori information. This approach has been proposed for communication systems that send data over channels that suffer from ISI (Inter Symbols Interference) and require equalization [51] - [52], and in a multiuser scenario for CDMA systems [53]. MMSE linear equalizer for non-iterative schemes is illustrated in expression (4) where $\tilde{\mathbf{x}}$ is the estimated vector of transmitted symbols after linear equalization, **y** is the vector of received symbols, **H** is the MIMO channel matrix, σ_w^2 is the AWGN noise variance at the receiver and **I** is the identity matrix

$$\tilde{\mathbf{x}} = \left(\mathbf{H}^{\mathrm{H}}\mathbf{H} + \sigma_w^2\mathbf{I}\right)^{-1}\mathbf{H}^{\mathrm{H}}\mathbf{y}\,. \qquad (4)$$

Expression (4) can be generalized to take into account a priori knowledge from the channel decoder which is illustrated in expression (5)

$$\tilde{\mathbf{x}} = \bar{\mathbf{x}} + \mathrm{Cov}(\mathbf{x},\mathbf{y})\mathrm{Cov}(\mathbf{y},\mathbf{y})^{-1}(\mathbf{y}-\bar{\mathbf{y}}), \quad (5)$$

where

$$\mathrm{Cov}(\mathbf{x},\mathbf{y}) = \mathrm{Cov}(\mathbf{x},\mathbf{x})\mathbf{H}^{\mathrm{H}}, \quad (6)$$

$$\mathrm{Cov}(\mathbf{y},\mathbf{y}) = \mathbf{H}\mathrm{Cov}(\mathbf{x},\mathbf{x})\mathbf{H}^{\mathrm{H}} + \sigma_w^2\mathbf{I}, \quad (7)$$

$$\bar{\mathbf{y}} = \mathbf{H}\bar{\mathbf{x}}. \quad (8)$$

The mean and variance of the transmitted vector **x** is computed with the following expressions

$$\bar{\mathbf{x}} = \sum_{\alpha_i \in \chi} \alpha_i \cdot P(x = \alpha_i), \quad (9)$$

$$\mathrm{Cov}(x,x) = \sum_{\alpha_i \in \chi} (\alpha_i - \bar{\mathbf{x}})^2 \cdot P(x = \alpha_i), \quad (10)$$

where the extrinsic bit probabilities are calculated from the extrinsic LLRs with the following relationships

$$P(b=0) = \frac{1}{1+e^{LLR_{EXT}(b)}}, \quad (11)$$

$$P(b=1) = 1 - P(b=0). \quad (12)$$

6.4.3 Simulation setup

In this section we describe the selected system parameters and mobile channel model used in the simulations for performance evaluation of optimal and suboptimal iterative DVB-NGH MIMO receivers.

6.4.3.1 DVB-NGH channel model

The MIMO channel model used during the standardization process was developed from a sounding campaign that took place in Helsinki in June 2010 [54]. The main objective was to obtain a 2x2 MIMO channel model (Figure 36) in the UHF band representative of cross-polar MIMO propagation in order to evaluate the performance obtained by multiple antenna techniques in realistic scenarios. This measurement campaign was the first one with cross-polar antenna configuration in the UHF frequency range. In ideal conditions the MIMO channel is rich in scattering and all the spatial paths have uncorrelated fading signals leading to maximum channel capacity. However, in practice, fading between spatial paths experiments correlation due to insufficient scattering. Moreover in situations where the transmitter and the receiver have LOS (Line Of Sight) component, the fading is modeled by a Ricean distribution with a sum of a time-invariant fading component and a time-variant fading component. The power of both components is related

by the Ricean K-factor. Spatial fading correlation and LOS component diminish the MIMO capacity [48] and both effects are included in the NGH MIMO channel model.

A wide range of reception conditions are included in the set of DVB-NGH channel models. Indoor and outdoor portable scenario with typical receiver velocities of 0 km/h and 3 km/h. Vehicular scenario with receiver velocities of 60 km/h and 350 km/h. Finally, SFN (Single Frequency Network) scenarios are included with the reception from two or four transmitter sites in a SFN network.

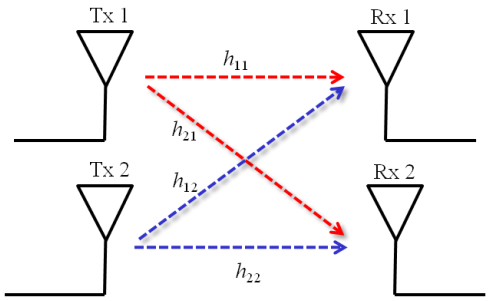

Figure 36: 2x2 MIMO system.

Vehicular scenario with receiver velocity of 60 km/h is the channel model used to evaluate the performance of the iterative MIMO receivers. Figure 37 illustrates the 8 taps PDP (Power Delay Profile) and the Doppler spectra characteristics. From both plots it can be seen the strong LOS component included in the model.

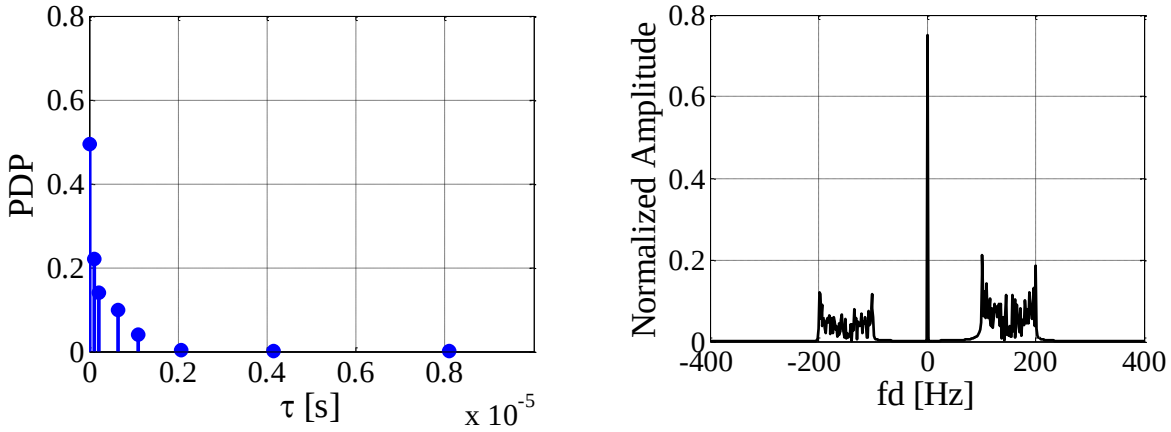

Figure 37: Power delay profile and Doppler spread spectrum for DVB-NGH portable outdoor channel model – Doppler spread of 400 Hz illustrated for visualization issue.

6.4.3.2 Simulation parameters

Table 12 summarizes the system parameters selected for the performance evaluation simulations.

Table 12: System parameters

DVB-NGH simulation platform	
FFT size	4096 carriers
Guard Interval	1/4
Memory size	260 Kcells
LDPC size	16200
Constellation order	8 bpcu (16QAM+16QAM)
Code Rates	1/3, 8/15 and 11/15
Num. iterations non iterative receiver	1x50
Num. iterations iterative receiver	25x2
QoS	Frame Error Rate after BCH 10^{-2}

The simulated system employs a FFT size of 4096 carriers and guard interval of 1/4 to trade off network cell area and resilience against Doppler spread. DVB-NGH uses half the amount of memory allowed for DVB-T2, i.e., 260 Kcells, to due to more restrictive memory requirements for handheld devices. The LDPC size is 16200 bits, to reduce power consumption and complexity in comparison with 64800 bits LDPC code word length. The constellation order selected is 8 bpcu (bits per cell unit) which implies a 16QAM constellation in each transmit antenna. We have selected the lowest, medium and highest code rate available for MIMO transmissions in DVB-NGH. The QoS (Quality of Service) selected is 1% of FER (Frame Error Rate) after BCH code.

The selection on the number of iterations performed by the receiver has a crucial impact in the performance and complexity.

Non-iterative receiver – 1x50: In this case no iterative decoding is implemented, i.e. there are zero outer iterations; the LDPC decoder performs 50 inner iterations.

Iterative receiver – 25x2: In this case, the number of outer iterations is limited to 25. In each outer iteration, the LDPC decoder performs 2 inner iterations. We note that the LDPC decoder complexity is the same in both cases since 50 inner iterations are performed in total.

6.4.4 Results

In the next section, simulation results are provided to analyze the performance of optimal and suboptimal iterative DVB-NGH MIMO receivers. We provide a performance comparison between MMSE demapper with a priori inputs and max-log demapper for both single shot and iterative receivers (MMSE non-ID, MMSE ID, max-log non-ID, max-log ID).

Figure 38, illustrates performance simulation results for code rate 1/3. For single shot receivers MMSE

demapper outperforms the max-log demapper by 0.15 dB. For the iterative receiver, max-log demapper outperforms MMSE by 0.2 dB. In both cases the performance of MMSE demapper is very similar to max-log, however complexity is significantly reduced. The iterative gain of MMSE ID demapper compared to max-log non-ID demapper is 0.8 dB.

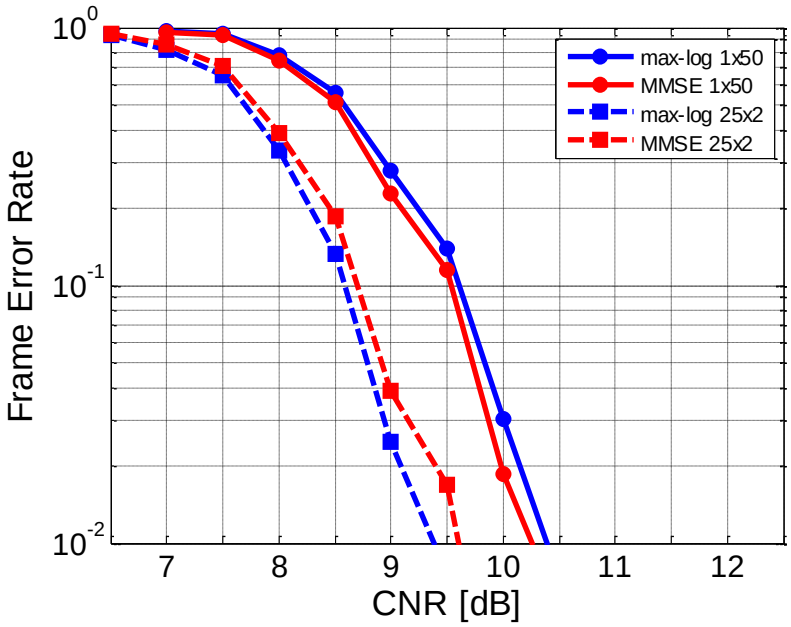

Figure 38: MMSE and max-log demapper performance comparison for single shot and iterative receivers using 8 bpcu and code rate 1/3 in vehicular DVB-NGH channel model with 60 km/h

Figure 39, shows results for code rate 8/15. In this case, MMSE demapper losses performance against max-log demapper for both cases, single shot and iterative receivers. For the former, loss is approximately by 0.4 dB and for the latter the performance loss is 0.5 dB. Still, the MMSE ID demapper outperforms max-log non-ID by 0.6 dB.

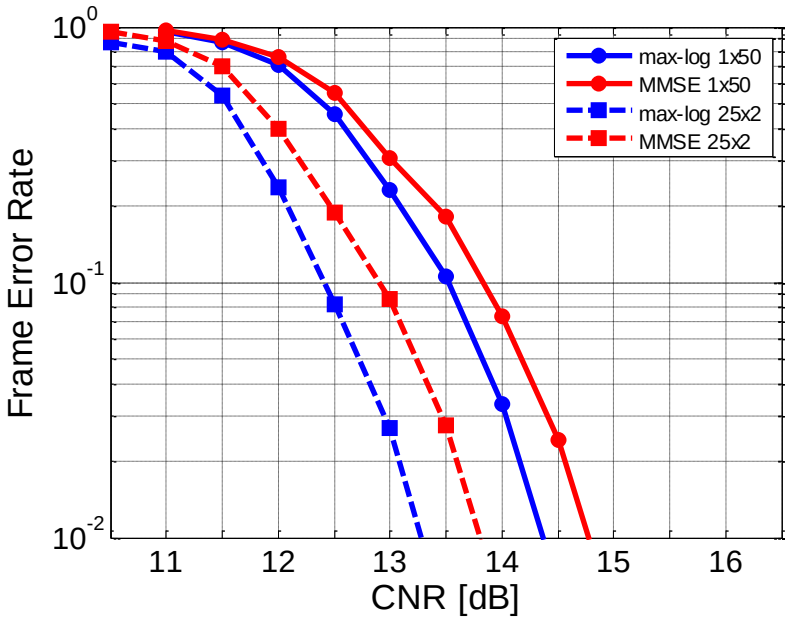

Figure 39: MMSE and max-log demapper performance comparison for single shot and iterative receivers using 8 bpcu and code rate 8/15 in vehicular DVB-NGH channel model with 60 km/h

Concluding the performance comparison between demapper options, Figure 40 shows results for code rate 11/15. In this case the difference between MMSE demapper and max-log increases. For the non-iterative case, MMSE non-ID demapper losses 1.2 dB against max-log non-ID and for the iterative case the loss of MMSE ID demapper compared to max-log ID is 1.9 dB but having similar performance to max-log non-ID.

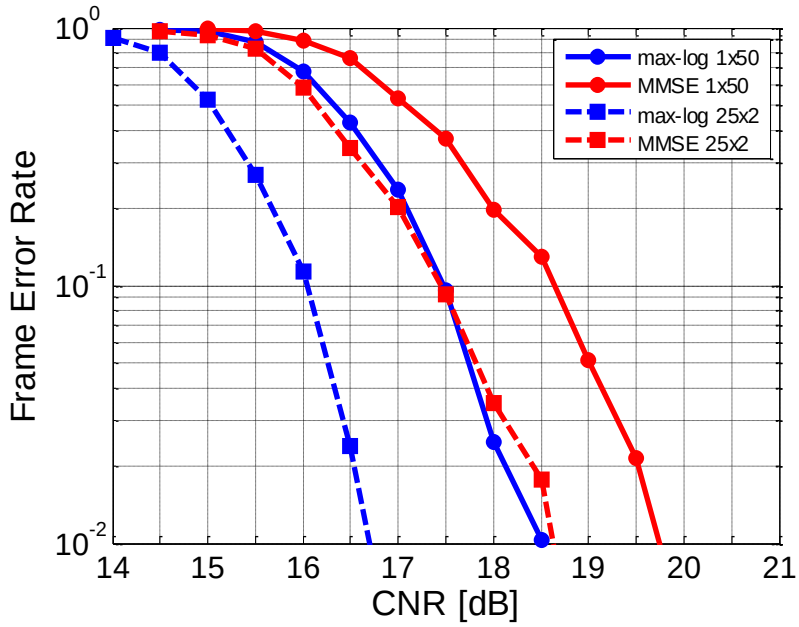

Figure 40: MMSE and max-log demapper performance comparison for single shot and iterative receivers using 8 bpcu and code rate 11/15 in vehicular DVB-NGH channel model with 60 km/h

The MMSE demapper is able to exploit the benefits of iterative detection but reducing the receiver complexity significantly. For both, non-ID and ID receivers, soft MMSE demapper has similar performance to max-log at low code rates, whereas at high rates MMSE demapper reduces its performance in comparison to max-log. These results are consistent with [55]. It is worth mentioning that the MMSE ID demapper outperforms or gives same performance than max-log non-ID demapper.

Next, we analyze the evolution of the FER with the number of outer iterations (feedback from LDPC decoder to MIMO demapper) for the two demappers under study. Figure 41 shows this evolution for code rate 1/3. The convergence of the error rate depends on the CNR available at the decoder input. For low CNR, increasing the number of iterations does not provide significant gain, e.g. 7 dB of Figure 41. On the other hand for medium or high CNR values (e.g. 8.5 dB and 9.5 dB of Figure 41), every outer iteration reduces the FER until saturation point, where feeding more information back to the demapper does not significantly improve the performance. This situation holds for both demappers and also for code rate 8/15 (Figure 42). The number of outer iterations performed at the receiver is a flexible parameter which provides a trade-off between performance and complexity.

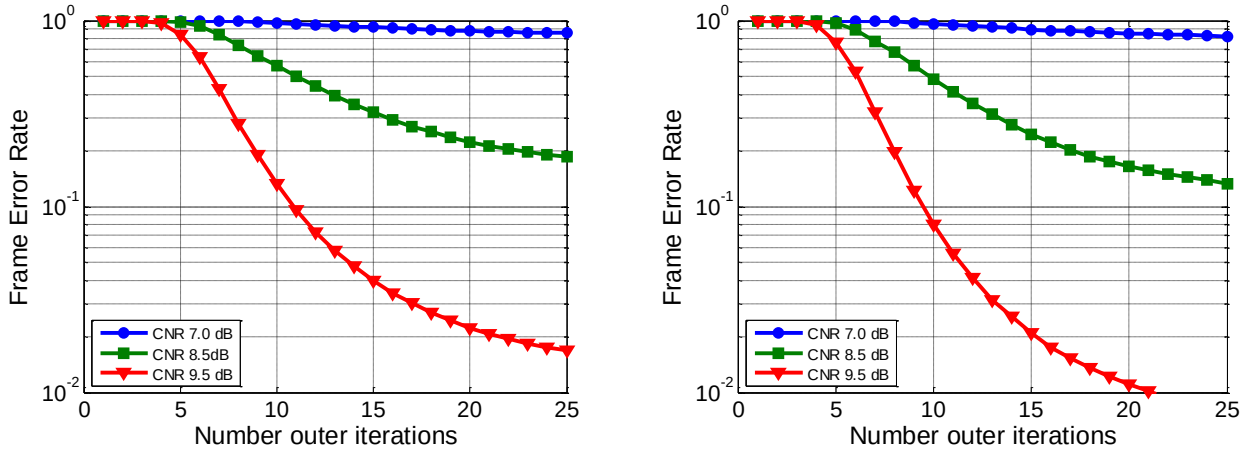

Figure 41: FER evolution with the number of outer iterations with MMSE (left) and max-log (right) demappers for 8 bpcu and code rate 1/3.

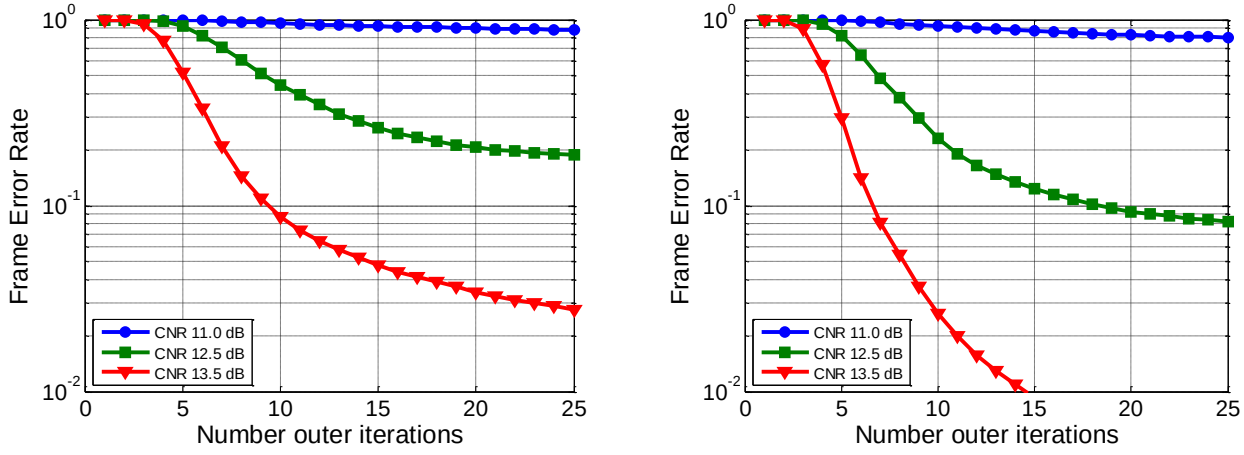

Figure 42: FER evolution with the number of outer iterations with MMSE (left) and max-log (right) demappers for 8 bpcu and code rate 8/15.

6.4.5 Conclusions

Iterative demapping provides significant gains for DVB-NGH MIMO receivers with max-log demapping. Simulation results under vehicular NGH channel model with 60 km/h show gains up to 2 dB. However, the implementation of iterative MIMO demapping requires a high computational complexity which scales exponentially with the number of transmit antennas and linearly with the number of outer iterations.

Sub-optimal soft MMSE demapper with a priori inputs is able to exploit the benefits of iterative demapping providing gains up to 1.2 dB under simulated vehicular scenario. Moreover, it significantly reduces the receiver complexity scaling polynomically with the number of transmit antennas and linearly with the number of outer iterations. Simulation results show for low code rates similar performance between soft MMSE demapper and max-log demapper for both, non-iterative and iterative receivers. At medium and high code rates MMSE demapper losses performance in comparison to max-log demapper. However iterative soft MMSE demapper provides same or improved signal quality as compared to non-iterative max-log demapper for all simulated code rates.

7 SUMMARY

The material presented here was dedicated to issues related to DVB-NGH receiver algorithms and implementation issues. A generic channel equalization technique for OFDM based systems in time variant channels was presented. A general classification for channels in terms of their time variability was presented. Besides, the equalization methodology reliability and the channel classification validity were proved in both the TU-6 and MR channels.

An efficient shuffled iterative receiver for the second generation of the terrestrial digital video broadcasting standard DVB-T2 was introduced. A simplified detection algorithm was presented, which has the merit of being suitable for hardware implementation of a Space-Time Code (STC). Architecture complexity and measured performance validate the high potential of iterative receiver as both a practical and competitive solution for the DVB-T2 standard.

Further, DVB-T2 performance in time varying environments was presented. The performance of the standard is simulated for both single and diversity 2 reception. Since DVB-T2 contains a huge number of possible configurations, focus is mainly given to two configurations: UK mode, and Germany-like candidate mode.

Highly parallel implementations of LDPC decoders optimized for decoding the long codewords specified by the second generation of digital television broadcasting standards: i.e. DVB-T2, DVB-S2, and DVB-C2 were presented. These implementations are optimized for modern GPUs (graphics processing units) and general purpose CPUs (central processing units). High-end GPUs and CPUs are quite affordable compared to capable FPGAs, and this hardware can be found in the majority of recent personal home computers.

Finally, studies on MIMO detection in the receiver were presented. Both complexity of the MIMO detection and performance of iterative MIMO detection were studied.

8 REFERENCES

[1] J. Cimini, L., "Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing," Communications, IEEE Transactions on, vol. 33, no. 7, pp. 665 – 675, Jul. 1985.

[2] S. Coleri, M. Ergen, A. Puri, and A. Bahai, "Channel estimation techniques based on pilot arrangement in OFDM systems," Broadcasting, IEEE Transactions on, vol. 48, no. 3, pp. 223 – 229, Sep. 2002.

[3] M.-H. Hsieh and C.-H. Wei, "Channel estimation for OFDM systems based on comb-type pilot arrangement in frequency selective fading channels," Consumer Electronics, IEEE Transactions on, vol. 44, no. 1, pp. 217 –225, Feb. 1998.

[4] W. G. Jeon, K. H. Chang, and Y. S. Cho, "An equalization technique for orthogonal frequency-division multiplexing systems in time-variant multipath channels," Communications, IEEE Transactions on, vol. 47, no. 1, pp. 27 –32, Jan. 1999.

[5] X. Wang and K. J. R. Liu, "An adaptive channel estimation algorithm using time-frequency polynomial model for OFDM with fading multipath channels," EURASIP J. Appl. Signal Process., vol. 2002, pp. 818–830, January 2002. [Online]. Available: http://portal.acm.org/citation.cfm?id=1283100.1283185

[6] Y. Mostofi and D. Cox, "ICI mitigation for pilot-aided OFDM mobile systems," Wireless Communications, IEEE Transactions on, vol. 4, no. 2, pp. 765 – 774, 2005.

[7] H. Hijazi and L. Ros, "OFDM high speed channel complex gains estimation using kalman filter and qr-detector," in Wireless Communication Systems. 2008. ISWCS '08. IEEE International Symposium on, 2008, pp. 26 –30.

[8] P. Bello, "Characterization of randomly time-variant linear channels," Communications Systems, IEEE Transactions on, vol. 11, no. 4, pp. 360–393, 1963.

[9] O. Edfors, M. Sandell, J. Van De Beek, S. Wilson, and P. Borjesson, "Analysis of DFT-based channel estimators for OFDM," Wireless Personal Communications, vol. 12, no. 1, pp. 55–70, 2000.

[10] W. Jakes, "Microwave Mobile Channels," New York: Wiley, vol. 2, pp. 159–176, 1974.

[11] M. Failli, "Digital land mobile radio communications COST 207," European Commission, EUR, vol. 12160.

[12] T. Celtic Wing, "project report (2006-12). Services to Wireless, Integrated, Nomadic, GPRS-UMTS & TV handheld terminals. Hierarchical Modulation Issues. D4-Laboratory test results. Celtic Wing TV, 2006."

[13] H. Hijazi and L. Ros, "Bayesian cramer-rao bound for OFDM rapidly time-varying channel complex gains estimation," in Global Telecommunications

[14] C. Abdel Nour and C. Douillard, "Improving BICM Performance of QAM constellations for broadcasting applications," *Int. Symp. on Turbo Codes and Iterative Techniques*, Lausanne, Switzerland, Sept. 2008, pp. 50 – 60.

[15] S. Sezginer and H. Sari, "Full-rate full-diversity 2×2 space-time codes

[16] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Transactions on Communications*, vol. 51, pp. 389 – 399, Mar. 2003.

[17] L. Vangelista, N. Benvenuto, S. Tomasin, C. Nokes, J. Stott, A. Filippi, M. Vlot, V. Mignone, and A. Morello, "Key technologies for next generation terrestrial digital television standard DVB-T2," *IEEE Communications Magazine*, vol. 47, no. 10, pp. 146 –153, October 2009.

[18] J. Boutros and E. Viterbo, "Signal space diversity: a power- and bandwidth-efficient diversity technique for the Rayleigh fading channel," *IEEE Transactions on Information Theory*, vol. 44, no. 4, pp. 1453 – 1467, July 1998.

[19] DVB-T2, "Implementation guidelines for a second generation digital terrestrial television broadcasting system (DVB-T2)," ETSI TR 102 831, v1.1.1, Oct. 2010.

[20] M. Li, C. Nour, C. Jego, and C. Douillard, "Design of rotated QAM mapper/demapper for the DVB-T2 standard," *IEEE Workshop on Signal Processing Systems, SiPS 2009*, October 2009, pp. 18 – 23.

[21] M. Li, C. Nour, C. Jego, and C. Douillard, "Design and FPGA prototyping of a bit-interleaved coded modulation receiver for the DVB-T2 standard," *IEEE Workshop on Signal Processing Systems, SiPS 2010*, Oct. 2010, pp. 162 – 167.

[22] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," *IEEE Workshop on Signal Processing Systems, SiPS 2004*, Oct. 2004, pp. 107 – 112.

[23] J. Zhang and M. Fossorier, "Shuffled iterative decoding," *IEEE Transactions on Communications*, vol. 53, no. 2, pp. 209 – 213, Feb. 2005.

[24] T. Yokokawa, M. Kan, S. Okada, and L. Sakai, "Parity and column twist bit interleaver for DVB-T2 LDPC codes," *5th International Symposium on Turbo Codes and Related Topics*, Sept. 2008, pp. 123 – 127.

[25] C. Marchand, J.-B. Dore, L. Conde-Canencia, and E. Boutillon, "Conflict resolution by matrix reordering for DVB-T2 LDPC decoders," *IEEE Global Communications Conference, GLOBECOM 2009*, Dec. 2009, pp. 1 – 6.

[26] Frame structure channel coding and modulation for a second generation digital terrestrial television broadcasting system (DVB-T2), ETSI EN 302 755 v.1.3.1, Apr. 2012.

[27] Microwave Mobile Communications, William C.Jakes, Wiley-Interscience.

[28] Antennas and Propagation for Wireless Communication Systems, S.R. Saunders John Wiley & Sons Ltd.

[29] Digital Communication over fading Channels, Simon & Alouini, John Wiley & Sons, Ltd.

[30] "Analytical LCR & AFD for Diversity Techniques in Nakagami Fading Channels," Iskander and Mathiopoulos, IEEE Trans. on Com., vol.50, no 8, Aug. 2002.

[31] S. Grönroos, K. Nybom and J. Björkqvist, "Efficient GPU and CPU-based LDPC decoders for long codewords", Analog Integrated Circuits and Signal Processing, Springer, 2012. DOI: 10.1007/s10470-012-9895-7

[32] R. Gallager, "Low-Density Parity-Check Codes", Ph.D. Thesis, M.I.T., 1963.

[33] S. Grönroos, K. Nybom and J. Björkqvist, "Complexity Analysis of Software Defined DVB-T2 Physical Layer", in Proceedings of the SDR '10 Technical Conference and Product Exposition, Washington, D.C., 2010.

[34] NVIDIA, "CUDA C Programming Guide v.4.0", http://www.nvidia.com, 2011.

[35] D. MacKay, "Good error-correcting codes based on very sparse matrices", in IEEE Transactions on Information Theory 45(2), 1999.

[36] N. Wiberg, "Codes and Decoding on General Graphs", in Ph.D. Thesis, Linköping University (1996).

[37] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, X. Hu, "Reduced-Complexity Decoding of LDPC Codes", in IEEE Transactions on Communications 53(8), 2005.

[38] NVIDIA, "NVIDIA's Next Generation CUDA Compute Architecture: Fermi", Whitepaper, http://www.nvidia.com, 2009.

[39] Intel Corporation, "Intel 64 and IA-32 Architectures Software Developer's Manual", Manual, http://www.intel.com, 2011.

[40] The Portland Group, "PGI CUDA-x86", http://www.pgroup.com/resources/cuda-x86.htm (accessed May 2012.)

[41] Khronos Group, "OpenCL - The open standard for parallel programming of heterogeneous systems", http://www.khronos.org/opencl (accessed May 2012.)

[42] G. Falcão, L. Sousa and V. Silva, "Massive parallel LDPC decoding on GPU", in Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming, 2008.

[43] P. Micikevicius, “Analysis-Driven Optimization”, Presented at the GPU Technology Conference 2010, San Jose, California, USA, 2010.

[44] G. Falcão, J. Andrade, V. Silva and L. Sousa, “GPU-based DVB-S2 LDPC decoder with high throughput and fast error floor detection”, in Electronic Letters 47(9), 2011.

[45] F. Oggier, and E. Viterbo, “Algebraic number theory and code design for Rayleigh fading channels”, Foundations and Trends in Communications and Information Theory, 1 (3). pp. 333-415, 2004.

[46] Frame structure channel coding and modulation for a second generation digital terrestrial television broadcasting system (DVB-T2), ETSI Std. EN 302 755, Rev. 1.2.1, 2011.

[47] Cisco visual networking index: global mobile data traffic forecast update – 2010-2015, While paper, February 2011

[48] A. Paulraj, R. U. Nabar, and D. Gore, Introduction to Space-Time Wireless Communications. Cambridge (UK): Cambridge Univ. Press, 2003.

[49] S. H. Müller-Weinfurtner, “Coding approaches for multiple antenna transmission in fast fading and OFDM,“ IEEE Trans. Signal Processing, vol. 50, no. 10, pp. 2442–2450, Oct. 2002.

[50] B. M. Hochwald and S. ten Brink, “Achieving near-capacity on a multiple-antenna channel,” IEEE Trans. Inf. Theory, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[51] C. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, “Iterative correction of intersymbol interference: Turbo equalization,” European Trans. Telecomm., vol. 6, pp. 507–511, Sept.–Oct 1995.

[52] R. Koetter, A. C. Singer, M. Tüchler, “Turbo equalization,” IEEE Signal Processing magazine, vol. 21, no. 1, pp. 67-80, January 2004.

[53] X. Wang and H. Poor, “Iterative (turbo) soft interference cancellation and decoding for coded CDMA”,IEEE Trans. Commun., vol. 47, no. 7, pp. 1046–1061, 1999.

[54] P. Moss, T. Y. Poon, and J. Boye, “A simple model of the UHF cross-polar terrestrial channel for DVB-NGH,” White Paper, BBC, 2011.

[55] P. Fertl, J. Jaldén, and G. Matz, “Capacity-based performance comparison of MIMO-BICM demodulators,” In Proc. IEEE SPAWC-2008, Recife, Brazil, July 2008, pp. 166–170.