Design, implementation and prototyping of an iterative receiver for bit-interleaved coded modulation system dedicated to DVB-T2

Meng Li

N° d'ordre : 2012telb0225

Thèse

présentée à

TELECOM BRETAGNE

en habilitation conjointe avec l'Université de Bretagne Sud

pour obtenir le grade de

DOCTEUR de Telecom Bretagne

Mention : *Sciences et technologies de l'information et de la communication*

par

Meng LI

Design, implementation and prototyping of an iterative receiver for bit-interleaved coded modulation system dedicated to DVB-T2

Soutenance prévue le 11 janvier 2012 :

Composition du Jury :

Directeurs	: Catherine Douillard, Professeure, Télécom Bretagne
	: Christophe Jégo, Professeur des Universités, IPB Bordeaux
Encadrant	: Charbel Abdel Nour, Maître de Conférences, Télécom Bretagne
Rapporteurs	: Fabienne Nouvel, Maître de Conférences HDR, INSA de Rennes
	: Jean-Marie Gorce, Professeur des Universités, INSA de Lyon
Examinateurs	: Gérard Faria, Directeur Général, Teamcast Rennes
	: Jean-Philippe Diguet, Directeur de Recherche CNRS, UBS

Contents

Introduction

Context:

The emergence of new market driven services such as *High Definition* (HD) television and 3D-TV have offered unprecedented user experience creating a real need for improving nowadays transmission systems. A better use of the scarce spectrum resources became a must leading to the development of next generation broadcasting systems.

Single Frequency Network (SFN) is a way to increase spectral efficiency. It consists of a broadcast network where several transmitters simultaneously send the same signal over the same frequency channel. While spectrally efficient, such a topology can lead to a severe form of multipath propagation. Indeed, the receiver sees several echoes of the same signal, the destructive interference among these echoes known as self-interference may result in additional fade events. This is problematic especially in wideband communication and high-data rate digital communications, since the frequency-selective fading and the *Inter-symbol Interference* (ISI) caused by the time spreading of the echoes greatly deteriorate the system performance in terms of *Bit Error Rate* (BER).

Spectral efficiency should not come at the price of reduced robustness. Therefore, numerous technical aspects are to be improved from first generation systems including source coding, channel coding, interleaving, modulation, diversity etc.

In 2008, the European *Digital Video Broadcasting* (DVB) standardization committee launched the *second generation of Digital Video Broadcasting-Terrestrial* (DVB-T2) standard [1]. As the successor of DVB-T, it introduces several enhancements to the transmission system including the 4th generation of the *Moving Picture Experts Group* (MPEG4) source coding, multiple physical layer pipes, a state-of-the-art forward error correcting codes: *Low Density Parity Check* (LDPC) [2] + *Bose Ray-Chaudhuri Hocquenghem* (BCH) [3], increased diversity thanks to a longer channel interleaver and the introduction of a diversity technique at the signal space level, a *Multiple Input Single Output* (MISO) Alamouti [4] based-scheme, etc.

Since the invention of turbo codes in 1993 [5], iterative processing has found its way

into numerous domains. The *Low Density Parity Check* (LDPC) codes are another branch of powerful iterative codes, which was re-found [6] after the invention of Turbo codes. In digital communications, the iterative process called turbo principle was extended to additional blocks than the traditional FEC. Indeed, an iterative process between an FEC decoder and a soft Multiple Input Multiple Output (MIMO) detector [7] or a demapper or an interference canceller has proven to improve performance. The iterative process between a demapper and a LDPC decoder was recommended in the implementation guideline of DVB-T2 standard in order to improve the performance over fading channel without and with erasures. The fading channel with erasures represents the case of a severe fading in SFN network.

Objectives:

In this document, we restrict ourselves to techniques that intended to improve throughput and reliability in the context of channel coding, diversity and modulation. The main objective of our study is to design a DVB-T2 receiver that can achieve high throughput for an acceptable hardware complexity. Moreover, the proposed receiver has to support both non-iterative process and iterative process. However, practical applications are reluctant to mandate solutions based on iterative processes due to some challenges and constraints in terms of **increased hardware complexity, memory access conflicts and additional latency**.

Signal Space Diversity (SSD) [8] can improve the robustness of the DVB-T2 system and mitigate the effects of self-interference due to SFN. While improving performance, SSD introduces additional complexity especially for spectrally efficient constellation sizes. DVB-T2 is the first standard to adopt signal space diversity with high order constellation such as 256-QAM. In this case, the classical one dimensional Max-Log demapping algorithm applied on $log(M)$ PAM based on de-coupling the I and Q components is not applicable. The quest for a hardware efficient SSD demapper is raised and not addressed yet.

The *Low Density Parity Check* (LDPC) codes are defined by their parity check matrices. The double diagonal sub-matrices in the parity check matrix of the LDPC codes induce message update conflicts problem in the shuffled LDPC decoding algorithm. In the meanwhile, the memory access problem caused by scheduling induces inefficient message passing between the check nodes and bit nodes. These are two crucial problems that have to be addressed for designing an LDPC decoder dedicated to the DVB-T2 standard.

A classical iterative receiver is frame-based, which induces large latency. The latency is introduced by the block interleaving/de-interleaving, which is based on memory writing and reading. The latency is also due to the state-of-art LDPC decoding algorithm (horizontal layer decoding algorithm). Indeed this algorithm provides the extrinsic information only after one complete iteration. Therefore, one iteration of a classical receiver consists of one complete

iteration of LDPC decoding, block de-interleaving memory writing and reading, demapping and block interleaving memory writing and reading. The resulting large latency prohibits efficient message exchange between the demapper and decoder hence reduces the throughput.

In this study, architectural solutions have to be provided to such problems for a *Bit-Interleaved Coded Modulation*(BICM) system with SSD applying an iterative processing between the demapper and the LDPC decoder.

Contributions:

Towards these objectives, some contributions are given in two domains : algorithmic domain and architecture design domain.

Contributions in algorithmic domain:

1.) Proposal of a two-dimensional Max-Log demapping algorithm based on sub-region detection to reduce the computational complexity of two-dimensional demapping algorithm and the corresponding architecture. The proposal of a linear approximation for the computation of Euclidean distance further reduces the requirement of multiplication operations, especially for high order constellations.

2.) Proposal of a Min-Sum vertical shuffled LDPC decoding algorithm. The message update conflicts problem due to the double diagonal sub-matrices and the message access conflicts due to pipeline in the case of vertical shuffled schedule are well solved.

3.) Proposal of a joint vertical shuffled iterative demapping and decoding algorithm for an iterative BICM receiver, which greatly reduce the latency of message exchange between demapper and decoder. An efficient message passing schedule between the demapper and decoder is also proposed which is suitable for a paralleled hardware implementation.

Contributions and results in hardware domain:

1.) Design and FPGA prototyping of a flexible demapper with low latency and low complexity, which supports 8 different kinds of QAM constellations.

2.) Design and FPGA prototyping of a vertical shuffled Min-Sum LDPC decoder.

3.) Prototyping of two transmission systems without OFDM modulation for the DVB-T2 standard onto a Xilinx Virtex5 LX330 device. One includes non-iterative receiver and the other one includes the iterative receiver.

4.) Integrating the proposed demapper and the LDPC decoder into a real DVB-T2 demodulator, which is provided by Teamcast company and supports various modulation schemes.

The measured performance of the three prototypes achieves expected performance gain. The estimated maximum working frequency of the iterative receiver after place and route is

80 Mhz. The corresponding throughput is equal to 107 Mbps for a 64K LDPC code with a code rate of R=4/5. To the best of our knowledge, the prototype of an iterative receiver is the first published hardware implementation for the DVB-T2 standard.

Organization:

This manuscript is organized as follow:

In Chapter 1, we first give a brief introduction about a digital communication system and error control codes. Then, a description of a wireless channel and its corresponding mathematical model is provided. The state-of-the-art of the coded modulations and the details of the coded modulation adopted in the DVB-T2 standard are presented afterwards.

In Chapter 2, we first recall the classical demapping algorithm for non-rotated QAM. Then, a two-dimensional demapping algorithm suitable for rotated QAM constellations is detailed. This section is followed by a proposal of a computational complexity reduced and hardware friendly demapping algorithm for rotated and Q-delayed QAM constellations. It applies Max-Log demapping and sub-region detection. The corresponding architecture is provided afterwards. Finally, a prototype of a complete uncoded transmission chain is introduced and the performance measurements are listed.

In Chapter 3, we first give an overview of the classical LDPC decoding algorithms and the simplification methods for the check node processing. Horizontal shuffled and vertical shuffled message passing schedules, which accelerate the decoding convergence speed, are also presented. Inspired by previous work, we propose a vertical shuffled Min-Sum LDPC decoding algorithm and its corresponding architecture design. The proposal includes methods to avoid message update conflicts due to double diagonal sub-matrices and memory access conflicts due to pipeline. A prototype of a simplified DVB-T2 transmission system is implemented to test the efficiency of the decoder. The designed demapper and LDPC decoder were also integrated in a real DVB-T2 demodulator.

In Chapter 4, we detail a novel vertical shuffled iterative processing algorithm dedicated to an iterative receiver. It applies a hardware oriented message exchange schedule between the demapper and decoder. The corresponding architecture is detailed and tested in a simplified DVB-T2 transmission system. The measured performance validates the efficiency of the proposed algorithm and the design.

résumé

L’émergence récente de nouveaux services de diffusion numérique tels que la télévision haute définition (HD) ou la télévision 3D a engendré la nécessité de définir des systèmes de diffusion numériques plus performants, capables de supporter la diffusion généralisée de tels services. En 2008, le consortium européen DVB (*Digital Video Broadcasting*) a défini le standard de télévision numérique terrestre de deuxième génération DVB-T2 qui permet à la fois une meilleure occupation des ressources spectrales et une meilleure robustesse de réception pour les récepteurs fixes, portables et même mobiles que son prédécesseur DVB-T.

Le mode de transmission préférentiel de DVB-T2 utilise des réseaux de diffusion isofréquences ou SFN (*single frequency networks*) dont tous les émetteurs envient le même signal au même instant et à la même fréquence. Les réseaux SFN permettent une utilisation optimisée du spectre radio-fréquence, permettant la diffusion d’un nombre plus important de programmes TV comparativement aux traditionnels réseaux multi-fréquences. Cependant dans les zones couvertes par deux ou plusieurs émetteurs, le récepteur doit faire face à l’arrivée de trajets multiples d’amplitudes équivalentes et présentant différents angles d’arrivée et retards, qui peuvent interférer de manière destructive et produire des phénomènes d’évanouissements ou *fadings*. Dans certains cas, ces interférences peuvent provoquer un effacement du signal. Ce type de canal à effacement est un modèle de canal de transmission typique défini dans les directives d’implémentation (*implementation guidelines*) du standard DVB-T2. Dans notre étude, nous avons principalement considéré ce modèle de canal à effacement, ainsi que le modèle plus classique de canal à *fading* sans mémoire de type Rayleigh, représentatif de la réception fixe d’un seul émetteur.

DVB-T2 a adopté plusieurs techniques innovantes de communications numériques offrant une robustesse de réception supérieure à DVB-T. Une avancée importante est l’adoption d’une modulation codée entrelacée par bit ou BICM (*bit-interleaved coded modulation*) faisant appel à la fois à un code correcteur d’erreur puissant et à une technique additionnelle de diversité de constellation. Le code correcteur d’erreur est constitué de la concaténation d’un code LDPC (low density parity-check) et d’un code BCH, chargé d’éliminer les erreurs résiduelles à la sortie du décodeur LDPC. La technique de diversité de constellation, qui

permet de doubler l'ordre de diversité de la transmission, est utilisée pour la première fois en pratique en association avec un code puissant tel qu'un LDPC.

Quand il ne met pas en œuvre de technique de diversité de constellation, l'émetteur BICM inclut habituellement le codeur correcteur d'erreurs, un entrelaceur au niveau bit et le convertisseur bits-symbole ou mappeur de la constellation de modulation. En présence de la technique de diversité de constellation, encore appelée « constellations tournées », la conversion bits-symbole est réalisée en deux étapes :

1) Les points de la constellation subissent tout d'abord une rotation d'un angle donné, qui entraîne la corrélation des axes en phase (I) et en quadrature (Q) de la constellation. Les deux composantes I et Q contiennent la totalité de l'information portée par chaque point de la constellation.
2) La composante Q est ensuite retardée par rapport à la composante I avant d'être envoyée sur le canal de transmission.

Les deux composantes I et Q de la constellation originale n'étant pas transmises simultanément, elles subissent des atténuations indépendantes sur le canal. En réception, le processus inverse est appliqué. Lorsqu'une des composantes du symbole de constellation original a été fortement atténuée ou même effacée, le contenu de celui-ci peut être récupéré à grâce à l'autre composante.

Depuis l'invention des turbocodes en 1993, le principe de décodage itératif, encore appelé principe turbo, est utilisé dans de nombreux domaines. Dans la chaîne de communication numérique, le principe turbo a été appliqué à d'autres blocs que les traditionnels décodeurs correcteurs d'erreurs ou égaliseurs. En particulier, l'application d'un processus itératif entre le démappeur et le décodeur LDPC est suggérée dans les directives d'implémentation du standard DVB-T2, afin d'améliorer les performances du système sur les canaux à, notamment lorsque ceux-ci présentent des phénomènes d'effacement.

Notre étude avait pour objectif de concevoir un décodeur BICM pour le standard DVB-T2 mettant en œuvre un processus itératif entre le démappeur et le décodeur et prenant en compte des contraintes de latence et de complexité matérielle. L'étude architecturale a été réalisée en trois phases. Dans un premier temps, nous avons conçu un démappeur de complexité matérielle réduite qui supporte les constellations tournées pour des modulations d'amplitude

en quadrature (QAM) carrées allant jusqu'à l'ordre 256. La seconde étape a consisté à concevoir une architecture de décodeur LDPC adaptée à la mise en œuvre d'un échange d'information itératif avec le démappeur. Enfin, dans la dernière phase, nous avons étudié l'optimisation du séquencement des processus de décodage et de démapping ainsi que la réalisation du récepteur itératif.

Conception d'un démappeur de complexité réduite

Pour une modulation *M*-QAM non tournée, les informations binaires portées par les composantes I et Q sont indépendantes car la modulation QAM peut être vue comme deux modulations $\sqrt{M}$-PAM (*pulse amplitude modulation*) séparées. En réception, le démappeur estime le LLR (*Log-Likelihood Ratio*) des bits portés par chaque composante en calculant $\sqrt{M}$ distances euclidiennes sur chacun des axes de la constellation.

Dans le cas d'une constellation *M*-QAM tournée, le démappeur doit calculer *M* distances euclidiennes bi-dimensionnelles pour chaque LLR ν :

$$\hat{v}_t^i = \log\left(\frac{\sum_{x_t \in \chi_1^i} \frac{P(x_t)}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{D_{euc}(x_t)}{2\sigma^2}\right)}{\sum_{x_t \in \chi_0^i} \frac{P(x_t)}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{D_{euc}(x_t)}{2\sigma^2}\right)}\right) \quad (1)$$

$$\text{où } D_{euc}(x_t) = \left(\left[\underbrace{\rho_{t-d}(y_{eq,t-d}^I - x_{t-d}^I)}_{a}\right]^2 + \left[\underbrace{\rho_t(y_{eq,t}^Q - x_t^Q)}_{b}\right]^2\right) \quad (2)$$

L'approximation linéaire communément appliquée pour simplifier l'expression des LLRs lorsque la constellation n'est pas tournée ne peut pas s'appliquer dans ce cas car la rotation introduit une inter-dépendance entre les composantes I et Q. Le calcul des LLRs est plus complexe que dans le cas classique car les deux composantes sont utilisées simultanément. Néanmoins la complexité du démappeur peut être réduite lorsque l'on applique l'approximation dite Max-Log. L'expression des LLRs devient alors :

$$\hat{v}_t^i \approx \frac{1}{2\sigma^2}\left[\min_{x_t \in \chi_0^i}\left(D_{euc}(x_t)\right) - \min_{x_t \in \chi_1^i}\left(D_{euc}(x_t)\right)\right] \quad (3)$$

Malgré ces approximations, dans le cas d'une constellation 256-QAM, 256 distances euclidiennes doivent être calculées, ce qui requiert 512 multiplications. La complexité matérielle correspondante peut dans certains cas être rédhibitoire. Afin de réduire le nombre de distances euclidiennes à calculer, nous proposons un algorithme de *demapping* basé sur la

division de la constellation en quatre sous-régions. Le choix et le dimensionnement des sous-régions suivent les règles suivantes :

1. Pour un signal reçu donné, un quart de la constellation (quadrant) est choisi en fonction du signe des composantes I et Q reçue,
2. La sous-région correspondante est dimensionnée de telle sorte que, pour tout point du quadrant sélectionné, celle-ci contienne l'ensemble des points ne différant que d'un bit du point considéré.

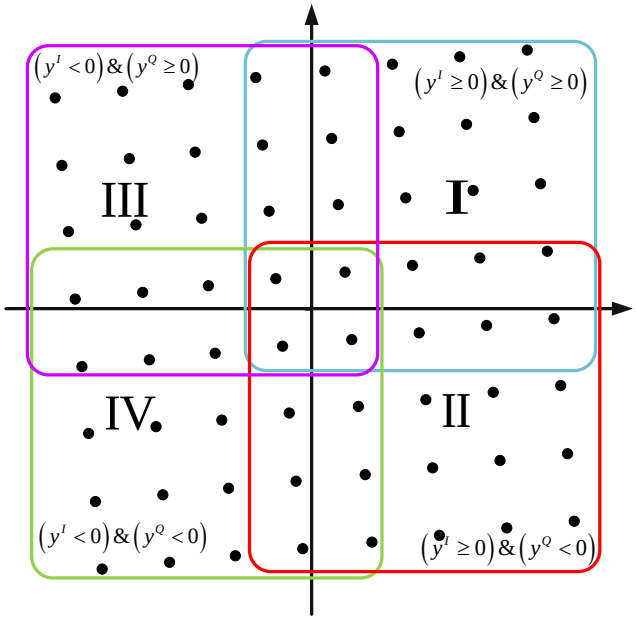


Fig. 1 Les quatre sous-régions utilisées pour la constellation 64-QAM tournée.

La Fig.1 montre la constellation 64-QAM tournée adoptée dans DVB-T2. Chaque point est porteur de six bits. Lorsque les composantes I et Q du signal reçu sont positives, la sous-région sélectionnée est la région bleue. Les trois autres sous-régions correspondent aux trois autres combinaisons de signes possibles. Pour une 64-QAM, le nombre de distances euclidiennes à calculer a ainsi été réduit de 64 à 25. Pour une 256-QAM, il est ramené de 256 à 81.

Outre la diminution du nombre de distances euclidiennes à calculer, nous avons réduit la complexité du calcul de ces distances proprement dites. Le calcul complet d'un terme de distance requiert normalement au moins deux multiplications. Afin de réduire le nombre total de multiplications, nous avons proposé l'application de l'approximation suivante dans l'équation (4) pour le calcul des distances euclidiennes :

$\mathrm{F}(a,b) = \sqrt{a^2 + b^2}$ peut être approximé par :

- $\mathrm{F}(a,b) = \max(a,b)$ si $\min(a,b) \leq \max(a,b)/4$, sinon
- $\mathrm{F}(a,b) = \max(a,b) + (\min(a,b) - \max(a,b)/4)/2$

L'application simultanée de ces simplifications ont permis la conception d'une architecture flexible de démappeur pour DVB-T2, supportant les constellations QAM tournées et non tournées d'ordre 4, 16, 64 et 256 pour des transmissions sur canaux gaussiens et canaux à *fading* avec et sans effacements.

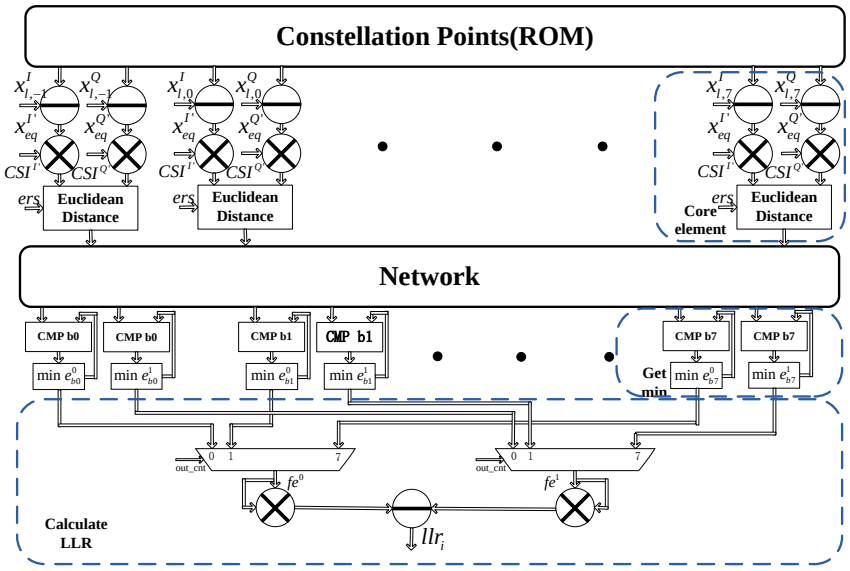


Fig. 2 Architecture d'un démappeur DVB-T2 flexible.

Cette architecture est décrite en Fig. 2. Les points de la constellation de deux sous-régions sont stockés dans une mémoire ROM. Les points des deux autres sous-régions sont déduits par symétrie. Chaque bloc élémentaire est chargé du calcul d'une distance euclidienne. Pour une constellation 256-QAM, 9 blocs élémentaires travaillent en parallèle pour calculer les 81 distances euclidiennes. Après 9 cycles de calcul, les 81 distances sont disponibles. Le réseau d'interconnexions est en charge de la sélection des distances nécessaires au calcul de chaque LLR, les deux derniers cycles étant consacrés au calcul des valeurs minimales des distances utilisées dans l'expression (6) puis au résultat de cette même expression.

Ce démappeur a été implémenté sur un FPGA Xilinx Virtex II Pro (XC2VP30). Afin de valider les performances du prototype, un premier démonstrateur matériel a été réalisé : il est constitué en émission d'un générateur pseudo-aléatoire de données sources et d'un mappeur,

d’un émulateur de canal et en réception d’un égaliseur, d’un démappeur et d’un calculateur de taux d’erreurs. Cette plate-forme fonctionne à une fréquence d’horloge de 62 MHz et la latence du démappeur est égale à 14 périodes d’horloge. Le démappeur implémenté ne requiert que 20 multiplieurs.

Toutes les constellations du standard DVB-T2 ont été vérifiées pour trois modèles de canaux de transmission: canal gaussien, canal de Rayleigh et canal de Rayleigh avec effacements. La Fig.3 compare les courbes de taux d’erreurs binaires résultant d’une part de la simulation en virgule flottante et d’autre part de mesures sur le démonstrateur, pour la transmission d’une constellation 64-QAM sur un canal de Rayleigh avec 15% d’effacements. On observe que les performances du prototype sont quasiment identiques à celles d’un modèle idéal de démappeur bi-dimensionnel.

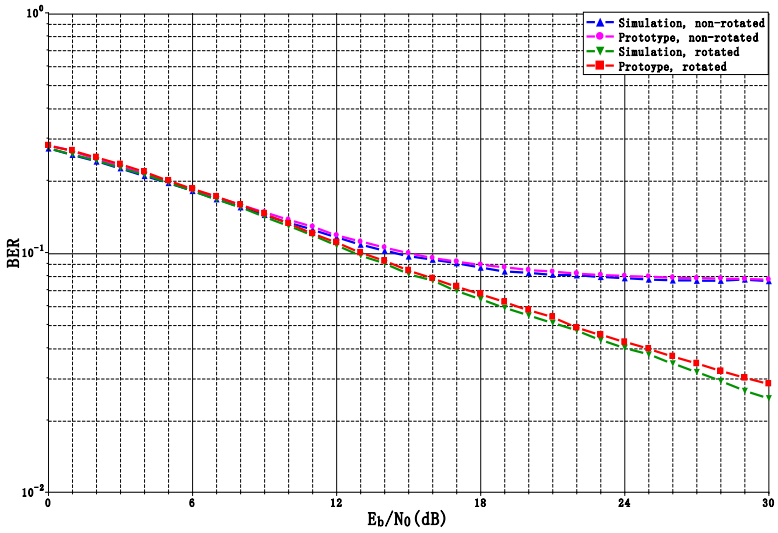


Fig. 3 Comparaison des taux d’erreurs binaires simulés et mesurés en sortie d’un démappeur 64-QAM pour une transmission sur canal de Rayleigh avec 15% d’effacements.

Afin de faciliter le passage de message entre le décodeur LDPC et le démappeur dans le cadre de la mise en place d’un récepteur itératif, nous avons proposé l’adoption d’un séquencement du décodage LDPC dit VSS (*Vertical Shuffled Scheduling*). D’autre part, nous avons implémenté la version simplifiée dite *min-somme* du décodage LDPC VSS. L’architecture correspondante est présentée en Fig. 4.

Au démarrage de l’algorithme, les informations relatives à chaque nœud de parité du graphe de décodage du code LDPC sont initialisées : signe du syndrome, valeur minimale et seconde

valeur minimale des informations provenant des nœuds de variables, ainsi que les indices correspondants. L'ensemble de ces informations sont stockées dans le banc mémoire associé aux nœuds de parités. Puis chaque itération de décodage VSS est constituée de N_{ldpc} sous-itérations exécutées de manière séquentielle. A chaque sous-itération, le processeur de nœud de parité (SISO-B) calcule une information dite *extrinsèque* qui est envoyée au processeur de nœud de variable (SISO-A). Le processeur de nœud de variable calcule l'information *a posteriori* en additionnant les LLRs provenant du canal à l'ensemble des informations extrinsèque et en extrait une information *a priori* qui est renvoyée au processeur de nœud de parité. Enfin le processeur de nœud de parité met à jour les valeurs minimales des informations provenant des nœuds de variable en utilisant les anciennes valeurs et l'information *a priori*.

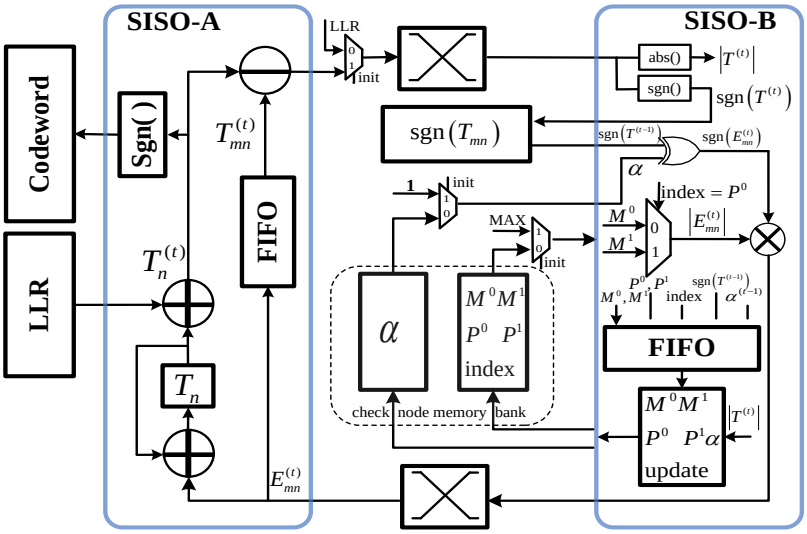


Fig.4 Architecture du décodeur LDPC VSS utilisant l'algorithme min-somme.

Lors de la conception du décodeur LDPC pour le standard DVB-T2 nous avons été confrontés à deux problèmes majeurs : des conflits lors de la mise à jour des messages dus à la présence de sous-matrices double-diagonales (DDSM) et des conflits d'accès mémoire dus à l'introduction de niveaux de pipeline. Ces problèmes ont déjà fait l'objet d'études antérieures dans de cas de la conception d'un décodeur à séquencement horizontal, mais pas, à notre connaissance, dans le cas d'un séquencement vertical.

Les matrices de parités des codes LDPC de DVB-T2 présentent un nombre important de DDSMs, dans lesquelles deux ou trois variables interviennent dans chaque relation de parité.

Lors de l’étape de mise à jour du nœud de parité, chaque nœud de variable fournit simultanément une nouvelle information extrinsèques à plusieurs nœuds de parités, ce qui cause un conflit d’accès mémoire. Le partitionnement de la matrice s’avère être une technique efficace pour réduire le nombre de DDSMs. Il ne peut néanmoins assurer la suppression de la totalité des DDSMs dans la matrice de parité. Pour résoudre le problème de conflit dans les DDSMs résiduelles, nous avons proposé de réutiliser l’information mise à jour de la première diagonale et de de renvoyer le résultat intermédiaire en tant qu’entrée pour le processus de mise à jour de la seconde diagonale. La Fig. 5 décrit le schéma logique correspondant à la mise à jour du signe d’une DDSM.

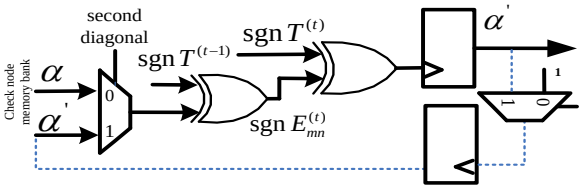


Fig. 5 Schéma logique pour la mise à jour du signe d’une DDSM.

Afin d’augmenter la fréquence maximale de fonctionnement du circuit et le débit des données en sortie du décodeur, des techniques classiques de pipelinage sont appliquées. Le séquencement du pipeline. Il se peut alors qu’une sous-itération démarre avant la fin de la précédente et qu’une information relative au nœud de parité soit lue avant d’être mise à jour. Des conflits peuvent en découler qui dégradent les performances de décodage. Un séquencement élaboré doit par conséquent être mis en place pour éviter ce type de conflits.

Nous avons proposé une méthode de modification du séquencement pour le décodage VSS. Il s’agit tout d’abord de détecter les cas de conflit et de modifier l’ordre de traitement des nœuds de parité dans une sous-itération. Puis la sous-matrice est partitionnée, la plupart des conflits étant supprimés lors de cette étape. Pour les cas problématiques résiduels, l’ordre de traitement des nœuds de variable est modifié avant de pouvoir décider de l’information décodée. Cette technique n’engendre quasiment aucun cycle d’attente entre les sous-itérations, ce qui se traduit par une augmentation conséquente du débit de décodage.

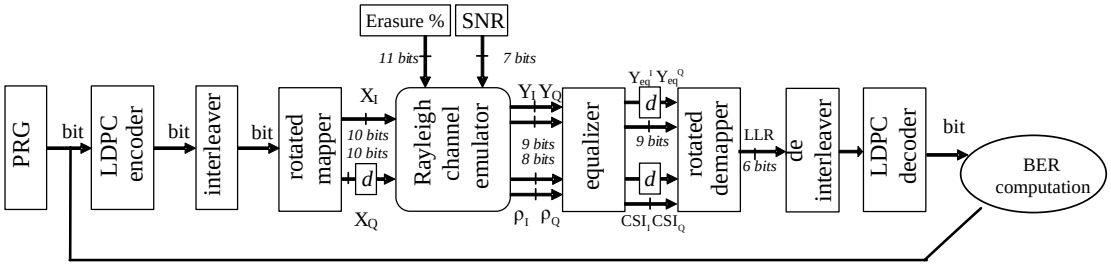


Fig. 6 Schéma-bloc de la chaîne de transmission DVB-T2 simplifiée.

Une chaîne de transmission DVB-T2 simplifiée, constituée d'une source numérique, un modulateur BICM, un émulateur de canal et un démodulateur BICM a été implémentée sur un FPGA Xilinx Virtex 7 sur la plateforme montrée en Fig. 7.

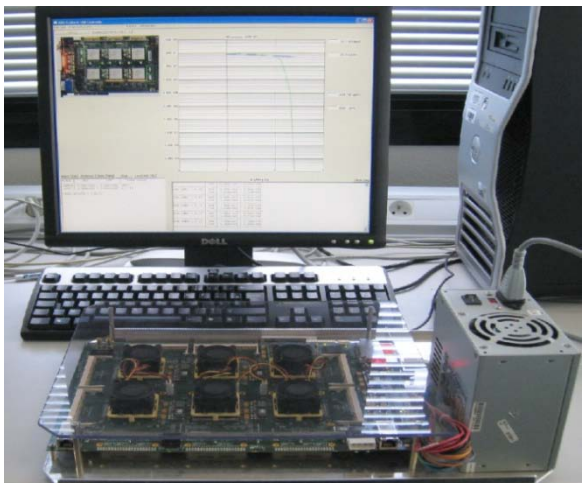


Fig.7 Plate-forme matérielle implémentant la chaîne de transmission DVB-T2 simplifiée.

L'utilisation des ressources du FPGA pour le décodeur LDPC sont listées dans la Table 1. Le décodeur fonctionne à une fréquence de 113MHz, correspondant à un débit utile en sortie du décodeur égal à 151 Mbit/s dans le cas d'un code de longueur 64k, de rendement 4/5 et pour 15 itérations de décodage VSS.

XC5VLX330	Flip-Flops	LUTs	RAMs
Décodeur LDPC	18,029 (8%)	41,032 (19%)	84 (29%)

Table 1 Utilisation des ressources matérielles du FPGA pour la réalisation du décodeur LDPC.

Les performances du prototype ont été vérifiées sur canal gaussien et de Rayleigh avec et sans effacements. Les Fig. 8 et 9 montrent le résultat des comparaisons de performance entre les mesures effectuées sur le prototype et les simulations en virgule fixe réalisées à partir d'une description en langage C du décodeur, pour le code 64k de rendement 4/5, sur canal de Rayleigh avec 15% d'effacements. Les courbes de la Fig. 8 ont été obtenues avec des constellations QAM conventionnelles (non-tournées) tandis que la Fig. 9 montre les performances avec constellations tournées.

Le démappeur et le décodeur ont également été intégrés sur le démonstrateur de démodulation DVB-T2 montré en Fig. 10, fourni par Teamcast dans le cadre du projet Eurêka/Eurostars SME42 (*SMEs for T2*). Les performances du démodulateur intégrant notre démappeur et décodeur ont été mesurées pour différents rendements de codage et constellations et ont permis de valider les algorithmes et architectures proposées.

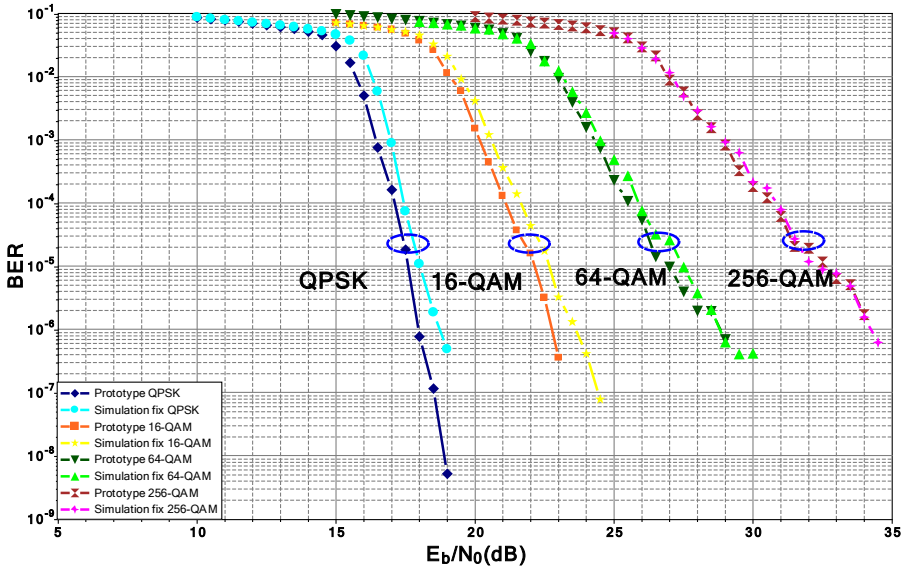


Fig.8 Performances comparées du prototype et du modèle C du décodeur BICM avec QAM non tournées pour une transmission sur canal à *fading* avec 15% d'effacements.

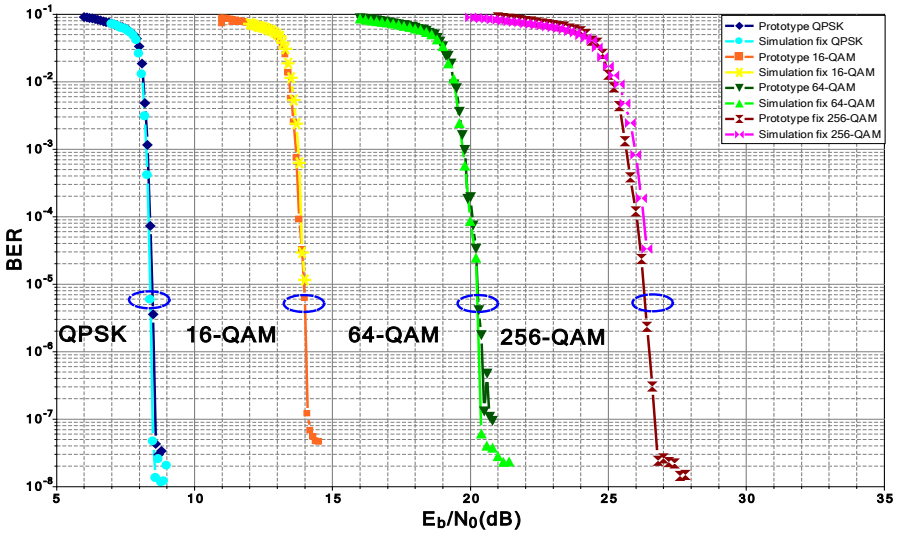


Fig.9 Performances comparées du prototype et du modèle C du décodeur BICM avec QAM tournées pour une transmission sur canal à *fading* avec 15% d'effacements.

L'introduction d'un processus itératif entre le décodeur LDPC et le démappeur permet d'améliorer les performances du décodeur BICM et/ou de diminuer le nombre d'itérations nécessaires à sa convergence. Néanmoins, la conception d'un récepteur itératif de faible latence et de complexité raisonnable présente des difficultés. En particulier, la latence constitue la contrainte principale. Elle est liée à deux causes dans le cas d'une architecture BICM conventionnelle : la présence de l'entrelacement et le désentrelacement binaire d'une part et le séquencement horizontal classiquement utilisé pour le décodage LDPC.

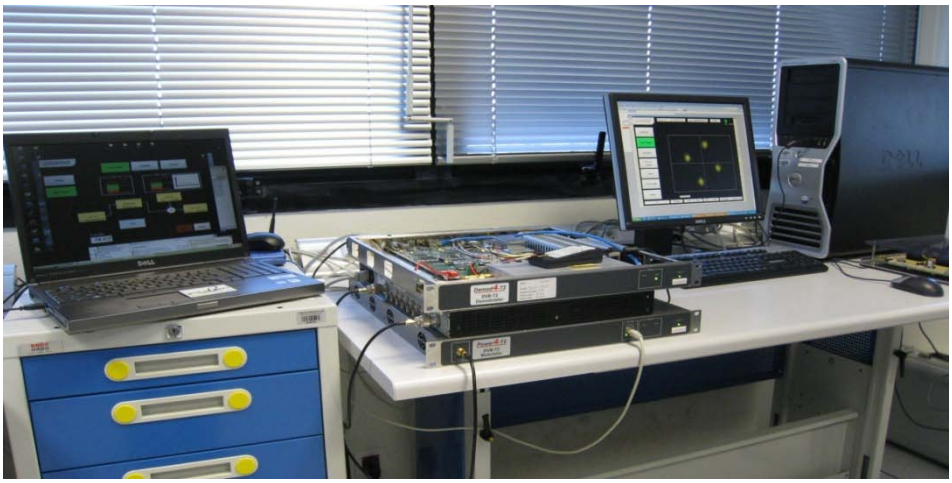


Fig.10 Plate-forme de modulation/démodulation DVB-T2 de Teamcast utilisée dans le cadre du projet SME42.

Afin de limiter la latence, nous avons divisé le bloc LDPC en plusieurs sous-blocs et appliqué le processus itératif au niveau de chaque sous-bloc. D'autre part, nous avons remplacé la RAM dédiée à l'entrelacement et au désentrelacement par des *look-up tables* pour permettre un routage rapide de l'information, puis nous avons adopté le séquencement de décodage LDPC vertical VSS précédemment étudié fin de garantir une génération rapide de l'information extrinsèque.

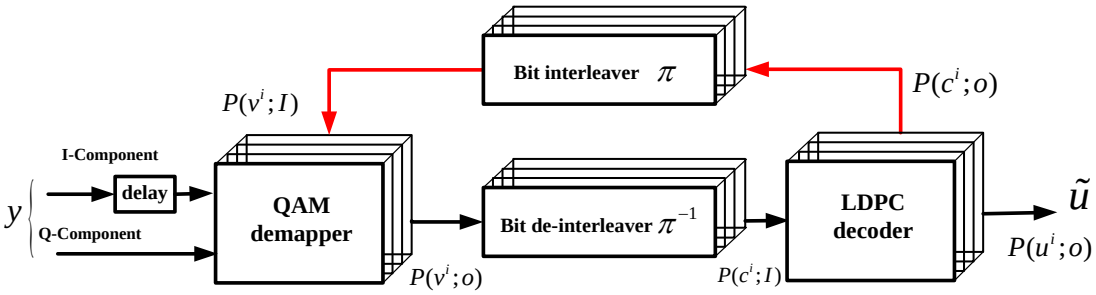


Fig. 11 Structure d'un récepteur itératif

Le passage de message lors du décodage VSS est réalisé colonne par colonne ; l'information extrinsèque et les LLRs peuvent ainsi être échangés entre le démappeur et le décodeur en un nombre de cycles limité. Plusieurs séquencements d'échange de messages peuvent être considérés, basés sur différentes stratégies de combinaison du parallélisme et de mise à jour des LLRs. Trois séquencements de référence ont été étudiés et sont listés dans la Table 2.

Séquencement par rapport au :	Séq. A démappeur	Séq. B décodeur	Séq. C décodeur
Nombre de symboles mis à jour au démappeur	1	≤ 90	≤ 90
Nombre de LLRs mis à jour au démappeur	$\log_2(M)-1$	$\leq 90\cdot(\log_2(M)-1)$	90
Niveau de parallélisme de décodage	1	90	90

Table 2 Les trois séquencements étudiés pour les échanges de messages entre le décodeur LDPC et le démappeur.

Nous avons dans un premier temps implémenté le récepteur itératif pour la constellation QPSK. Deux prototypes ont été réalisés, basés sur le séquencement C. Le premier prototype utilise l'algorithme de décodage VSS min-somme (MS) tandis que le second met en œuvre l'algorithme VSS *min-somme*-3 (MS3), dont les performances sont plus proches de celles de l'algorithme de référence *somme-produit*. Les ressources matérielles utilisées pour les deux décodeurs BICM itératifs (BICM-ID) sont recensées dans la Table 3. La fréquence maximale de fonctionnement du décodeur BICM-ID MS est égale à 80 MHz après placement-routage, ce qui correspond à un débit de 107 Mbit/s en sortie du décodeur LDPC pour un rendement de codage de 4/5 et 15 itérations de décodage VSS.

XC5VLX330	Flip-Flops	LUTs	RAMs
BICM-ID MS	23,118 (11%)	9,3130 (44%)	179 (62%)
BICM-ID MS3	26,088 (14%)	10,8126 (51%)	193 (67%)
Coût additionnel (MS→ MS3)	2970 (3%)	14996 (7%)	14 (5%)

Table 3 Utilisation des ressources matérielles FPGA la conception d'un décodeur BICM itératif.

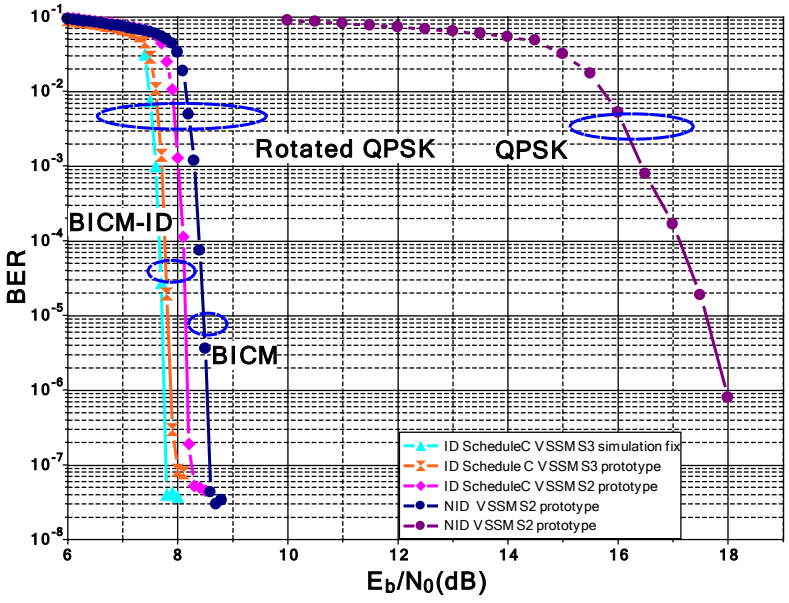


Fig. 12 Courbes de performance d'un décodeur BICM-ID QPSK sur canal à fading avec 15% d'effacements. Code LDPC 64k de rendement 4/5.

Les performances du prototype de décodeur BICM-ID QPSK ont été mesurées dans le cas du code 64K et rendement de codage 4/5 pour une transmission sur un canal à *fading* avec 15% d'effacement. Les résultats sont présentés en Fig. 12. Le gain liée à la diversité de constellation est de l'ordre de 10 dB tandis le gain additionnel lié au processus itératif est égal à 0,5 dB pour l'algorithme MS et 0,8 dB pour l'algorithme MS3. Les performances mesurées sont quasiment identiques aux courbes de référence simulées en virgule fixe. A notre connaissance, il s'agit du premier prototype de décodeur BICM-ID DVB-T2 référencé dans la littérature.

La suite de ces travaux va essentiellement consister à étendre cette dernière étude aux cas des constellations d'ordres supérieurs : 16-QAM, 64-QAM et 256-QAM.

CHAPTER 1

Background

The second generation of terrestrial video broadcasting standard (DVB-T2) was defined in 2008. The key motivation for the second generation is to provide high capacity and robust transmission to fixed, portable and mobile terminals. One of the important key technologies in DVB-T2 is the advanced *Bit-Interleaved Coded Modulation* (BICM) with *Signal Space Diversity* (SSD). The possibility of iteration between the decoder and demapper further increases the performance gain especially over a deep faded channel.

In this chapter we start with a brief introduction of the digital communication system, then we offer a review of different *Forward Error Correction* (FEC) codes. The fading channel model used in the test of our study is represented next. It is followed by a review of the existing different schemes of the coded modulation. Afterwards, we give a brief introduction of the DVB-T2 system and a detailed description of *BICM with Signal Space Diversity* (BICM-SSD) and *BICM with Signal Space Diversity and iterative process* (BICM-ID-SSD). In the remaining section, we give a detailed introduction of the LDPC codes adopted in DVB-T2 system, including the encoding method and the property of the codes.

1.1 A digital communication system

The functional diagram and basic elements of a digital communication system is illustrated in Fig. 1.1. In a digital communication system, the source may be either analog or digital signal. The messages produced by the source are converted into digital sequence. To have an efficient communication, we seek efficient representation of the source information that results in little or no redundancy. The process of efficiency converting the source into a sequence of binary digits is called source encoding or data compression.

To have a reliable communication system, the channel encoder induces some redundancy in a controlled manner. The redundancy can be used at the receiver to overcome the effects of noise and interference encountered in the transmission of the signal through the channel. The encoding involves taking k information bits at a time and mapping each k-bit sequence into a unique n-bit sequence, which is called a codeword. The ratio k/n is called code rate.

The modulator serves as the interface for the channel encoder to the communication channel. It maps the binary information sequence into a continuous-time electrical signals (waveforms). Let us suppose that the modulator may transmit b coded information bits at the same time t by using one waveform of the set of $M = 2^b$ distinct waveforms, $s_i(t), i = 0, 1, \cdots, M-1$. We call this M-ary modulation (M > 2,M=2 binary modulation).

The communication channel is the physical medium that is used to send the signal from the transmitter to the receiver. The channel may be the atmosphere, wire lines, optical fiber cables, ect. Whatever the physical medium used for the transmission of the information, the essential feature is that the transmitted signal may get corrupted and induced errors.

At the receiving end of a digital communication system, the demodulator processes the channel-corrupted transmitted waveform and reduces the waveforms to a sequence that represents estimates of the transmitted data symbols. This sequence is passed to the channel decoder, which attempts to reconstruct the original information sequence. The average probability of a bit-error at the output of the decoder is a measurement of the performance of the combination of the demodulator and the decoder, which is a function of the code characteristic, the type of the waveform of the modulator, the transmitter power, the characteristic of the channel and the demodulation and decoding algorithms. Finally, the source decoder attempts to reconstruct the original signal by processing the output of the channel decoder based on the knowledge of the source encoding method.

1.2 Error control codes

Error control codes also called *Forward Error Correction* (FEC) enable the detection and correction of the errors introduced by transmission of a modulated signal through a channel.

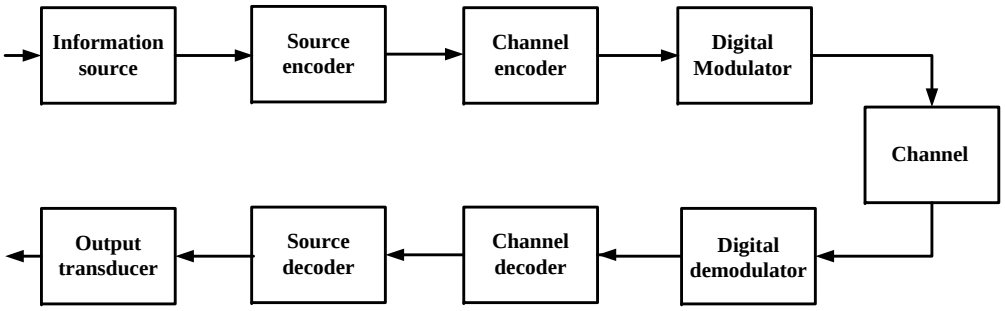


Figure 1.1 — Basic elements of digital communication system

Today's error correction codes fall into two categories: block codes and convolutional codes. However, Turbo codes and *Low density parity check* (LDPC) codes could be classified as a new branch of error control codes: the iteratively decoded codes.

1.2.1 Linear block codes

A binary block code generates a block of n coded bits from k information bits, we call this as an (n, k) binary block code, with $(n-k)$ parity bits. Hamming (7,4) code is a famous binary block code that encodes 4 bits of data into 7 bits by adding 3 parity bits. The linear block codes are encoded by $\mathbf{C} = \mathbf{U} \times \mathbf{G}$. For an (n, k) code with k information bits, denoted as: $\mathbf{U} = [u_1, u_2, \cdots, u_k]$, are encoded into the codeword, denoted as $\mathbf{C} = [c_1, c_2, \cdots, c_n]$. $\mathbf{G}_{k\times n}$ is the generator matrix and for a systematic linear block code, the generator matrix is described as $\mathbf{G}_{k\times n} = [\mathbf{I}_{k\times k}|\mathbf{P}_{k\times(n-k)}]$, where $\mathbf{I}_{k\times k}$ is the $k \times k$ identity matrix and $\mathbf{P}_{k\times(n-k)}$ matrix determines the parity bits.

$$\mathbf{G}_{k\times n} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix} \tag{1.1}$$

$$\mathbf{G}_{k\times n} = [\mathbf{I}_{k\times k}|\mathbf{P}_{k\times(n-k)}] = \left[\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & g_{11} & g_{12} & \cdots & g_{1(n-k)} \\ 0 & 1 & \cdots & 0 & g_{21} & g_{22} & \cdots & g_{2(n-k)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & g_{k1} & g_{k2} & \cdots & g_{k(n-k)} \end{array}\right] \tag{1.2}$$

The parity check matrix is used to decode linear block codes. The parity check matrix corresponding to the generator matrix $\mathbf{G}_{k\times n} = [\mathbf{I}_{k\times k}|\mathbf{P}_{k\times(n-k)}]$ is defined as: $\mathbf{H}_{(n-k)\times n} = [\mathbf{I}_{(n-k)\times k}|\mathbf{P}_{(n-k)\times(n-k)}]$. It is easy to verify that $\mathbf{G} \times \mathbf{H}^\top = \mathbf{0}_{k\times(n-k)}$. Recall that $\mathbf{C} = \mathbf{U} \times \mathbf{G}$,

we can get $\mathbf{C} \times \mathbf{H}^{\top} = \mathbf{0}_{k\times(n-k)}$. Thus, multiplication of any valid codeword with the parity check matrix results in all-zero vector, this is called syndrome testing and is used to determine the valid codeword.

One powerful class of block codes is the *Bose-Chadhui-Hocquenghem* (BCH) codes, which were invented in 1959 by Hocquenghem, and independently in 1960 by Bose and Ray-Chaudhuri [3]. BCH codes are polynomial codes over a finite field with a particularly chosen generator polynomial, so it provides a large selection of block length. The BCH codes are cyclic codes, in which the high rates BCH codes typically outperform all other block codes with the same n and k at moderate to high SNRs.

Reed-Solomon (RS) codes, which are non-binary BCH codes, with symbols as coefficients of a polynomial p(x) over a finite field $GF(q)$, $(q > 2)$, invented by Irving S. Reed [9] and Gustave Solomon. Reed-Solomon codes achieve a minimum distance of $d_{min} = N - K + 1$, which is the largest possible minimum distance between codewords for any linear code (n, k). The RS(204,188) based on $GF(2^8)$ shortened from RS(255, 239), is the RS codes adopted in the DVB-T standard with the generator polynomial $p(x) = 1 + x^2 + x^3 + x^4 + x^8$. Berlekamp-Massey decoding algorithm is the most popular hard decision decoding algorithm for BCH and RS codes, which was discovered by Elwyn Berlekamp [10] and James Massey [11]. While Chase-Pyndiah algorithm [12] is a soft input soft output decoding algorithm well used in the turbo decoding of product codes composed of BCH or Reed-Solomon component codes.

1.2.2 Convolutional codes

Convolutional codes differ from block codes in that the encoder contains memory so the output of the encoder at any given time is not only determined by the input but also by the previous memorized inputs. Convolutional codes are commonly specified by three parameters (n, k, m), where n is the number of output bits, k is the number of input bits and m is the number of memory registers. The octal generated polynomial is also used for defining a convolutional code. The constrain length $K = k \cdot (m - 1)$ represents the number of bits in the encoder memory that affect the generation of the n output bits, The code in Fig. 1.2 is a $(3, 1, 3)$ convolutional code, with a code rate of R=1/3 and the constrain length as 2. Viterbi [13] in 1967 proposed a *maximum likelihood* (ML) decoding algorithm that was relatively easy to implement for soft-decision decoding of the convolutional codes. In 1974, Bahl, Coke, Jelinek and Raviv (BCJR) [14] introduced a *maximum a posteriori probability* (MAP) decoding algorithm for convolutional codes with unequal *a prior* probability for the information bits. The BCJR has been widely applied to soft-decision iterative decoding scheme in which the *a prior* probability information changes from iteration to iteration.

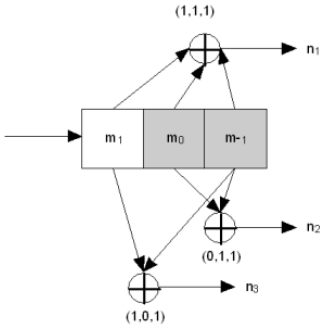


Figure 1.2 — A (3,1,3) convolutional code

1.2.3 Concatenated codes

Concatenated codes form a class of error-correcting codes that are derived by combining an inner code and an outer code. They were conceived in 1966 by Dave Forney [15] as a solution for the problem of finding a code that has an exponentially decreasing error probability with increasing block length and a polynomial-time decoding complexity. The inner code is typically designed to remove most of the errors introduced by the channel and the outer code is typically a less powerful code that further reduces error probability when the received bits have a relatively low error probability. The concatenated codes frequently have the inner and outer codes separated by an interleaver to break up bursts of errors. In the DVB-T standard, the inner code is a punctured convolutional code with five code rates $1/2, 2/3, 3/4, 5/6,$ and $7/8$. The Viterbi decoder tends to have some residual errors in bursts. The punctured Reed-Solomon (204,188) as the outer code has good burst error correcting properties. The combination of the inner code with outer code plus interleaver can achieve very low error probability.

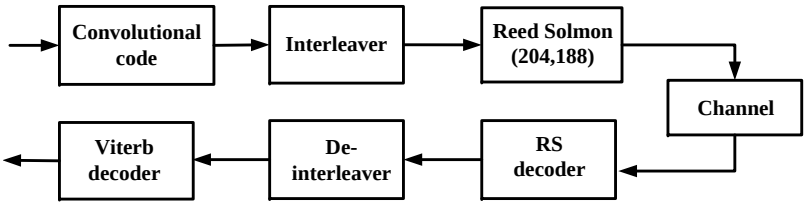


Figure 1.3 — The FEC of DVB-T standard

1.2.4 Turbo codes

Shannon set out the performance limits of channel coding and modulation schemes as early as 1948 [16] [17], however he gave no indication on how to construct good practical codes. The achievement of the Shannon capacity limit has been the goal of channel coding theorists

ever since. However the performance of the mentioned Reed-Solomon codes, convolutional codes, product codes and concatenated codes is still a long way from the Shannon limit. Turbo codes, invented by Berrou and Glavieux [5] in 1993, is the first codes that are capable of approaching Shannon's limit.

These codes involve a parallel concatenation of two *recursive systematic convolutional* (RSC) codes. A general structure of a turbo encoder is shown in Fig. 1.4. Two component codes are used for encoding the same input bits m, but an interleaver is placed between the encoders. The output of the encoder is $(m, X1, X2)$ for a code rate of R=1/3. Higher code rates are obtained by puncturing.

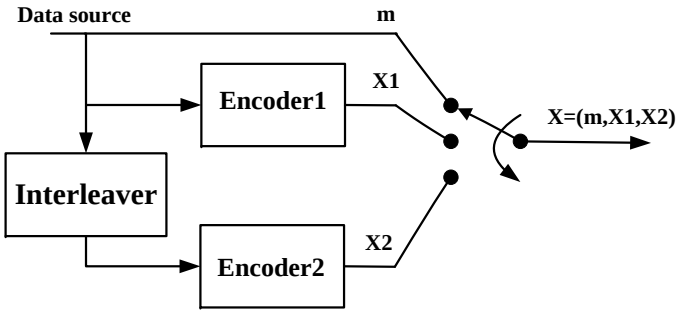


Figure 1.4 — A classical structure of Turbo encoder

Fig. 1.5 shows a general structure of Turbo decoder. Two component decoders are linked by interleavers in a structure similar to that of the encoder. The inputs of the decoder are multiplexed as $(R0, R1, R2)$, according to the systemic bit and the other two parity bits. Each decoder takes three inputs: the systematic bit, the parity bit transmitted from the corresponding component encoder and the information from the other component decoder, which is referred to as *a priori* information. Each decoder needs to provide the probability of the decoded bit sequence, so a *Soft Input Soft Output* (SISO) decoding algorithm is required.

BCJR-based decoding and Max-Log MAP decoding represent two classical decoding algorithms for Turbo codes. A soft output version of the Viterbi decoding (SOVA) is also a wide spread decoding algorithm for these concatenated codes. During the decoding process, each decoder alternately builds upon the results of the other to gradually enhance the reliability of the decisions via the exchange of *extrinsic* information on the systematic bits.

1.2.5 Low density parity check codes

Low density parity check (LDPC) codes were originally invented by Gallager [2] in 1963. However, these codes were ignored until the introduction of turbo codes or more precisely iterative decoding. LDPC codes were re-born by Mackay and Neal [6] in 1997.

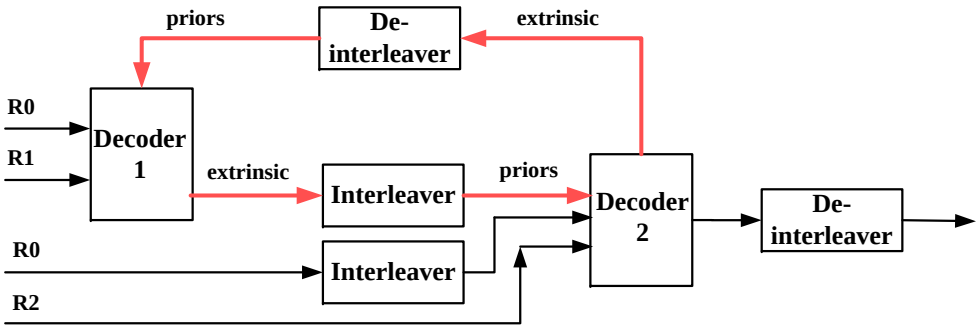


Figure 1.5 — A structure of Turbo decoder

These codes are linear block codes based on simple parity check equations and specified by a sparse parity-check matrix containing mostly zeros and a few ones (hence low density). They are often represented as bipartite graphs (Tanner Graphs) [18] which contain loops or cycles. An LDPC code is said to be regular if the check node and bit node degrees are constant and irregular if they are not. The degree correspond to the number of ones in the rows (for check nodes) or columns (for bit nodes) of the parity check matrix.

The widely used decoding algorithm is *Belief Propagation* (BP) also named as *Message Passing* (MP), since the messages are passed between the check nodes and bit nodes through the connection defined by the parity check matrix. A well-known instance of BP is the sum-product algorithm first proposed by Gallager in [2], which may also be realized in the log domain. The method to simplify the check node process were well studied by Chen and Fossorier [19]. Among them normalized min-sum is mostly used in current LDPC decoders. Different ways of scheduling the bit and check node update can have a significant impact on the convergence speed of the decoding process. The default approach used in classical BP is called flooding with two phase, where all of the bit nodes are updated in parallel followed by the update of all the check nodes. Faster convergence can be achieved with shuffled scheduling [20] [21]. A detailed explanation of the decoding algorithm and scheduling will be presented in Chapter 3.

1.3 The fading channel model

The communication channel represents a physical medium between the transmitter and the receiver. The channel model is a representation of the input-output relationship in mathematical or algorithmic form. Unlike wired channels whose characteristics are stationary and predictable, wireless channels are not predictable. They introduce significant levels of interference, distortion, and noise. Modelling wireless channels has been one of the most difficult parts in the wireless system design.

1.3.1 General description of fading channel

Developing mathematical models for the propagation of signals over a transmission medium requires a good understanding of the underlying physical phenomena. In wireless mobile communications, the electromagnetic waves often do not directly reach the receiver because of the obstacles that block the *Line Of Sight* (LOS) path, such as buildings, mountains or foliage. A signal travels from transmitter to receiver over multiple reflective paths; this phenomenon is called *multipath propagation*. This effect can cause fluctuations in the received signal's amplitude, phase and angle of arrival, which can be constructive or destructive. A typical scenario of mobile radio communications is shown in Fig. 1.6, where the three main mechanisms that impact the signal propagation are depicted.

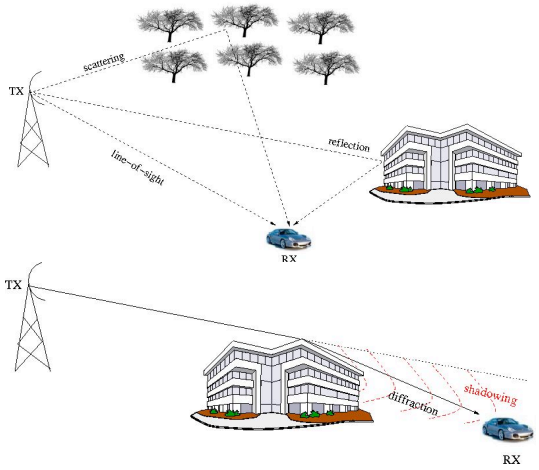


Figure 1.6 — A typical scenario of mobile radio communications

Those mechanisms are: 1.) *Reflection*. It occurs when the electromagnetic wave bumps against a smooth surface, whose dimensions are large compared to the signal wavelength. 2.) *Diffraction*. When a building whose dimensions are larger than the signal wavelength obstructs a path between transmitter and receiver, new secondary waves are generated. This phenomenon is often called shadowing, because the diffracted field can reach the receiver even when shadowed by an impenetrable obstruction. 3.) *Scattering*. It happens when a radio wave bumps against a rough surface whose dimensions are equal to or smaller than the signal wavelength. In the urban area, lampposts, street signs, and foliage are typical obstacles that cause scattering. Another negative influence on the characteristics of the radio channels is the Doppler effect, due to the motion of the mobile receiver. The Doppler effect causes a frequency shift of each portion of transmitted waves, as described in equ. (1.3).

$$f = f_{max} \cdot \cos \alpha \tag{1.3}$$

where $f_{max} = (v/c_0) \cdot f_0$ is the maximum Doppler frequency. The value of f_{max} depends

on the ration of the speed of the receiver v, the speed of the light c_0 and the carrier frequency f_0. α is the angle of the arrival of the wave with respect to the mobile receiver.

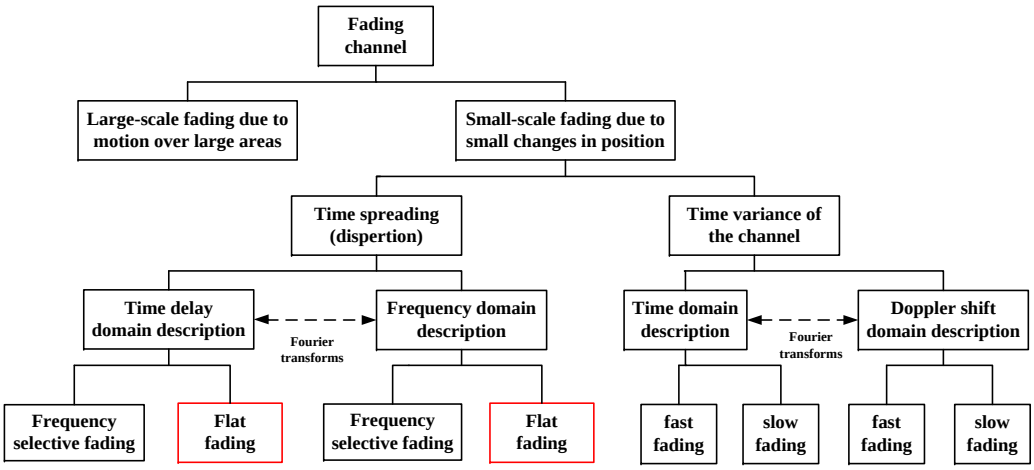


Figure 1.7 — Fading types and their corresponding manifestation

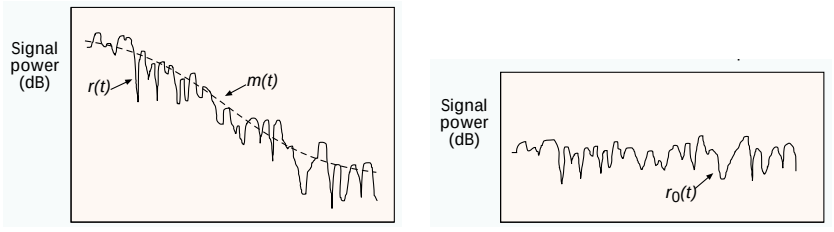
Fig. 1.7 represents an overview of the manifestation of a fading channel [22] [23]. It falls into two main categories: large-scale fading and small-scale fading. Large-scale fading represents the average signal power attenuation or path loss due to motion over large areas. This phenomenon is affected by prominent terrain contours (hills, forests, clumps of buildings, etc) between the transmitter and the receiver. The signal suffered large-scale fading is said to be shadowed by these obstacles. The amplitude change caused by shadowing is often modelled by a log-normal distribution with a standard deviation according to the log-distance path loss.

Small-scale fading refers to the dramatic changes in signal amplitude and phase that can be experienced as a result of small changes (as small as half-wavelength) between the receiver and transmitter. If the multiple reflective paths are large in number and there is no line-of-sight signal component, the envelope of the received signal is statistically described by a *Rayleigh probability distribution function* (pdf). When there is a dominant non-faded signal component, such as a line-of-sight propagation path, the small-scale fading envelope is described by a *Rician* pdf. The small-scale fading manifests itself into two distinct mechanisms, namely, time spreading of the signal and time variance of the channel. The former one is due to multipath and the later one is due to motion.

Frequency selective fading and *Flat fading* are the two kinds of fading in the signal dispersion manifestation,which could get explained both in time domain and frequency domain. From time domain point of view, frequency selective fading occurs when the multipath delay spread is greater than the duration of symbol. The frequency selective fading is also known as *Intersymbol Interference* (ISI), which leads to an irreducible BER degradation. From the

frequency point of view, frequency selective fading occurs when the coherence bandwidth of the channel is smaller than the bandwidth of the signal. The coherence bandwidth means the statistical measure of the range of frequencies over which the channel passes all spectral components with approximately equal gain and linear phase. In this case, different frequency components of the signal therefore experience decorrelated fading. While in flat fading, the coherence bandwidth of the channel is larger than the bandwidth of the signal. Therefore, all frequency components of the signal will experience the same magnitude of fading.

Fast fading and *slow fading* are the two kinds of fading in the time variance manifestation. *Fast fading* describes a condition when the time duration in which the channel behaves in a correlated manner is short compared to the time duration of a symbol. Therefore, it can be expected that the fading character of the channel will change several times while a symbol is propagating, which leads to distortion of the baseband pulse shape and yields an irreducible error. While in the *slow fading* the time duration that the channel behaves in a correlated manner is longer compared to the time duration of the transmission symbol.



(a) received signal with both large and small-scale fading

(b) received signal with large-scale fading removed

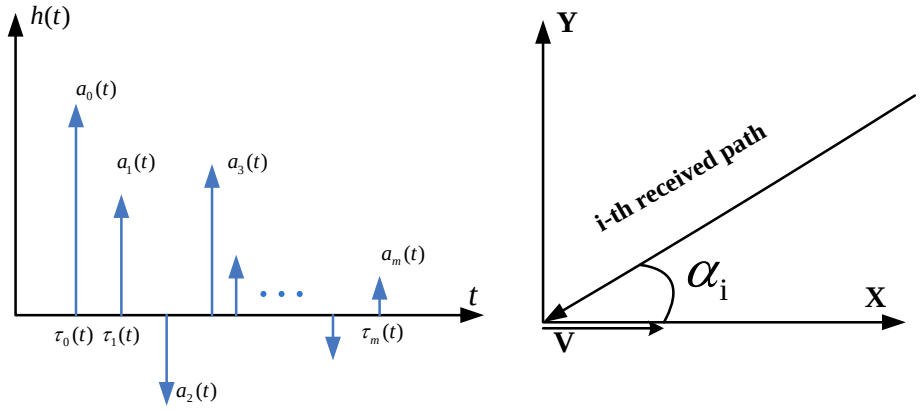
Figure 1.8 — Large-scale and small-scale fading

Any wireless signal $r(t) = m(t) \cdot r_0(t)$ transmitted over large physical distances is suffered both large-scale fading $m(t)$ as well as small-scale fading $r_0(t)$. Since large-scale fading affects only the average strength of the received signal, it will not be considered in the rest of our study. We restrict our study to small-scale fading and especially to flat fading and flat fading with erasures.

1.3.2 Rayleigh fading channel model

The mathematical model of the multipath channel can be presented by using the method of the impulse response used for linear systems. At time 0, Dirac delta function as $x(t) = \delta(t)$ is used to describe the transmitted single. At the receiver side, due to the presence of the multiple electromagnetic paths, more than one pulse will be received (we suppose here that the channel has infinite bandwidth, thus the pulse shape is not modified at all), and each one of them will arrive at different times $\tau(t)$, with different energy strengths $\beta(t)$ and different

angles $\alpha(t)$. The phase of the path is uniformly distributed between 0 and 2π. Fig. 1.9(b) describes one path of the mobile fading channel model with the receiver moving in the x-direction with Doppler shift Δf_i which is described in equ. (1.3). Fig. 1.9(a) illustrates the multipath channel impulse response, where $a(t) = \beta(t) \cdot \exp(\alpha(t))$.



(a) Channel impulse response (b) Model of one path

Figure 1.9 — Mathematical model of the multipath impulse response and the reception of the mobile receiver with one incoming path

Let $S(t)$ be the transmitted complex signal having a carrier frequency f_0 modulated by a baseband complex signal $x(t)$, it can be written as:

$$S(t) = x(t) \exp(j2\pi f_0 t) \tag{1.4}$$

The received signal suffers a multipath channel with m distinct waves. With the additive white Gaussian noise omitted, it can be expressed as equ. (1.5)

$$S^{'}(t) = \sum_{i=1}^{m} a_i(t) S\left(t - \tau_i(t)\right) = \sum_{i=1}^{m} a_i(t) \exp\left(-j2\pi f_0 \tau_i(t)\right) \cdot x\left(t - \tau_i(t)\right) \cdot \exp(j2\pi f_0 t) \tag{1.5}$$

where $a_i(t) = \beta_i(t) \cdot \exp(\alpha_i(t))$ and $\tau_i(t)$ represent the attenuation and the delay of the i-th path. The received signal $S^{'}(t)$ can be rewritten as the baseband signal $y(t)$:

$$y(t) = \sum_{i=1}^{m} c_i(t) x\left(t - \tau_i(t)\right) \tag{1.6}$$

where

$$c_i(t) = a_i(t) \exp\left(-j2\pi f_0 \tau_i(t)\right) \tag{1.7}$$

The Doppler shift affects the attenuation $a_i(t)$ periodically and is comparably small when compared with the carrier frequency f_0.

As a result, $y(t)$ can still be considered as narrowband and the central limit theorem can be applied with high value of m. Then $c_i(t)$ can be modelled as complex, mutually independent Gaussian processes as follows:

$$c_i(t) = p_i(t) + jq_i(t) \tag{1.8}$$

where $p_i(t)$ and $q_i(t)$ are independent Gaussian process having the same variance σ_i^2.

In the case of flat fading, the maximum delay $\tau_i(t)$ is much smaller than the symbol duration T, hence $x(t - \tau_i(t))$ can be approximated by $x(t)$. The received signal $y(t)$ can be rewritten as:

$$y(t) = x(t) \sum_{i=1}^{m} c_i(t) \tag{1.9}$$

In the discrete time domain, with sample time T, the received signal becomes as:

$$y(nT) = x(nT) \sum_{i=1}^{m} c_i(nT) \tag{1.10}$$

Here we introduce the complex Gaussian process $c_n = \sum_{i=1}^{m} c_i\,(nT)$, which is the sum of m independent complex Gaussian process $c_i\,(nT)$, then according to the central limit theorem c_n becomes as follows:

$$c_n = p_n + jq_n \tag{1.11}$$

where p_n and q_n represent two independent Gaussian processes having the same variance $\sigma_n^2 = \sum_{i=1}^{m} \sigma_i^2$. c_n can also be expressed in the form as:

$$c_n = \rho_n \exp(j\varphi_n) \tag{1.12}$$

where $\rho_n = \sqrt{p_n^2 + q_n^2}$ is the amplitude of c_n with $E\left(\rho_t^2\right) = 1$. φ_n is the phase of c_n uniformly distributed in $[0, 2\pi]$.

Taking equ. (1.9), equ. (1.12) and the additive complex Gaussian noise b_n into account, the discrete received signal can be expressed as:

$$y_n = \rho_n \exp(j\varphi_n) x_n + n_n \tag{1.13}$$

If a perfect phase detection is assumed, y_n becomes as:

$$y_n = \rho_n x_n + n_n \tag{1.14}$$

The received discrete-time baseband signal that suffered a flat Rayleigh fading is finally described as equ. (1.14), based on thesis [24]. In the rest of the thesis, we assume perfect phase detection and perfect *Channel State Information* (CSI).

1.3.3 Single Frequency Network

A *Single-frequency Network* (SFN) is a broadcast network where several transmitters simultaneously send the same signal over the same frequency channels. The aim of SFN is an efficient utilization of the radio spectrum, allowing a higher number of radio and TV programs in comparison to traditional *multi-frequency network* (MFN) transmission. The receiver gets several echoes of the same signal at the same time with the same frequency, the effect can be constructive or destructive. In the fringe areas covered by two or more SFN transmitters, any drift will cause reception degradation. Therefore, SFN transmission can be considered as a severe form of multipath propagation in a negative point of view.

If only two paths are considered, an undesired static echo at a specific delay will cause the magnitude of the received signal to change up or down depending on its relative phase shift. In an extreme case, the undesired echo may arrive at 180° and at an equal energy level (0-dB) relative to the desired signal. In this case, the received signal is cancelled or erased. If the interference is dynamic, i.e., with a very small Doppler shift, the received signal will suffer erasure periodically even the reception is stable.

1.3.4 Channel model for the fading channel with erasures

In a single-frequency network, the received signal may suffer erasures. This is a typical scenario in the state-of-art of the broadcasting system. In the DVB-T2 implementation guidelines [25], a 0-dB echo channel is defined as a fading channel with dynamic erasures having two paths with the same energy and one of them has 1Hz Doppler shift. Therefore, in the rest of the study, we focus on the fading channel with erasures as well. The received discrete-time baseband signal, that suffers a Rayleigh fading with erasures is described as equ. (1.15):

$$y_t = \rho_t e_t x_t + n_t \tag{1.15}$$

where e_t is a random discrete process, that takes value 0 with a probability of P_e and value 1 with a probability of $1 - P_e$. At the receiver side, the transmitted energy has to be normalized by a factor $\sqrt{1 - P_e}$ in order to compensate the loss of transmitted power. Based on the 0-dB echo channel model, the erasure ratio of 15% corresponding to $Pe = 0.15$ have been chosen in our study.

1.4 Coded Modulation

Coded Modulation is a technology of combining bandwidth efficient modulation and coding to achieve coding gain without bandwidth expansion or reducing data rate. There are several

approaches for constructing such bandwidth efficient coded modulation techniques. Two of these: *Trellis coded modulation* (TCM) and *Bit-Interleaved Coded Modulation* (BICM) are studied in the following content.

1.4.1 Trellis coded modulation

Trellis coded modulation (TCM) is a modulation scheme which enables highly efficient transmission over band-limited channels, proposed by Ungerboeck in 1982 [26]. This breakthrough is achieved by joint optimization of channel coding and modulation, which uses multi-level/phase signal modulation with set-partition mapping and simple convolutional coding.

Fig. 1.10 represents a classical encoding scheme of TCM for 8PSK modulation with a code rate of R=2/3 . A binary convolutional encoder operates on 2 information bits to produce 1 coded bit **P**. This coded bit **P** selects one of 2^2 subsets of the signal constellation, in which the minimum distance is expended to $2\sqrt{E_s}$ from $\sqrt{(2-\sqrt{2})\cdot E_s}$. Then the information bit b_1 and b_0 choose one of the point in the selected subset consecutively. Finally the selected point is passed to a PAM modulator and the transmitted symbols is achieved finally. At the receiver side, a sequence detector is employed (i.e., Viterbi decoder) to determine the maximum likelihood sequence transmitted.

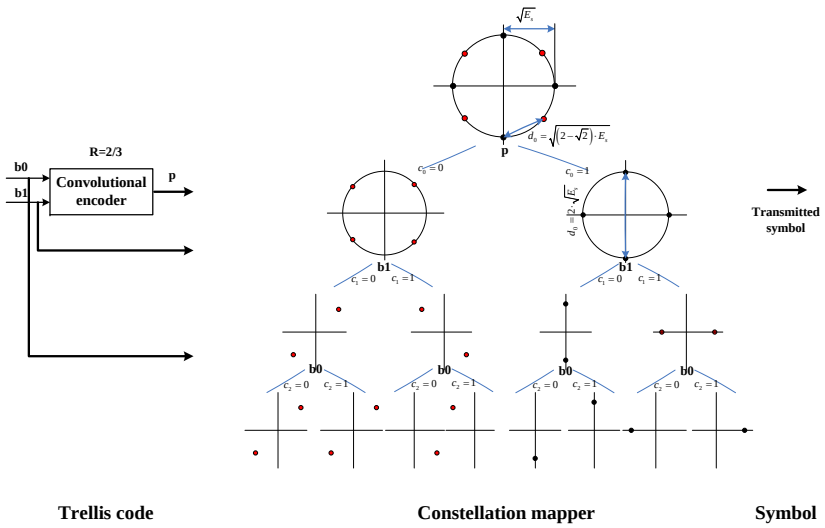


Figure 1.10 — General coding scheme of TCM

To achieve a reliable communication over a fading channel, the primary design criterion is to increase the diversity order of coded modulation. Since TCM is designed to maximize the minimum free Euclidean distance and not diversity order, the performance of TCM over

Rayleigh fading channel is degraded. Adding a symbol interleaver in conjunction with a trellis code and avoiding parallel transition is a common technique used to improve TCM performance. The diversity order for any symbol interleaved system is limited to the minimum number of distinct symbols along any error event, so the performance gain achieved by adding symbol interleaver is limited. Increasing the constraint length of the code is another way to increase the diversity order.

1.4.2 Pragmatic trellis coded modulation

TCM as created by Ungerboeck requires predefined association between the code and the modulation. Viterbi [27] used a basic convolutional code (code rate 1/2 with 64 states) to produce a wide range of rate $k/(k+1)$ TCM codes with the QPSK, 8PSK and 16QAM TCM signals, which is called Pragmatic TCM. Pragmatic means practical but not necessarily optimum, however this Pragmatic TCM is straightforward to implement. It uses a currently available industry standard Viterbi decoder and only with little modification the same decoder can be applied for a variety of modulation schemes although sacrifices very little performance loss compared to TCM. This work implied giving up the joint decoder and demodulator in favor of two separate entities.

1.4.3 Bit-interleaved coded modulation

Based on this concept, Zehavi [28] in 1992 recognized the code diversity. He showed that the diversity order can be increased, which can achieve the smallest number of distinct bits (rather than channel symbols) along any error events. This is achieved by bit-wise interleaving after encoding and by using an appropriate soft-decision bit metrics as an input to the Viterbi decoder. This technique is later known as bit-interleaved coded modulation. It further improves the performance of coded modulation over Rayleigh fading channel because the diversity order is increased to the minimum Hamming distance of the code.

Although first introduced by Zehavi [28], it was Caire [29] in 1998 who presented the underlying theory behind BICM and provided a general information-theoretical framework for this concept, which is regarded as the guidelines of BICM system design.

A conventional BICM is modelled as a serial concatenation of a conventional encoder, random bit interleaver and a memoryless modulator as shown in Fig. 1.11. At the transmitter side, the information sequence u is encoded via a binary outer encoder. Then the encoded sequence c is bitwise interleaved. The purpose of the bit interleaver π is to break the sequential fading correlation and to increase the diversity order to the minimum Hamming distance of a codeword. At the sample time t, m consecutive bits of the interleaved sequence are grouped and mapped to symbol x_t according to a 2^m-ary constellation by an m-bit signal label μ_m.

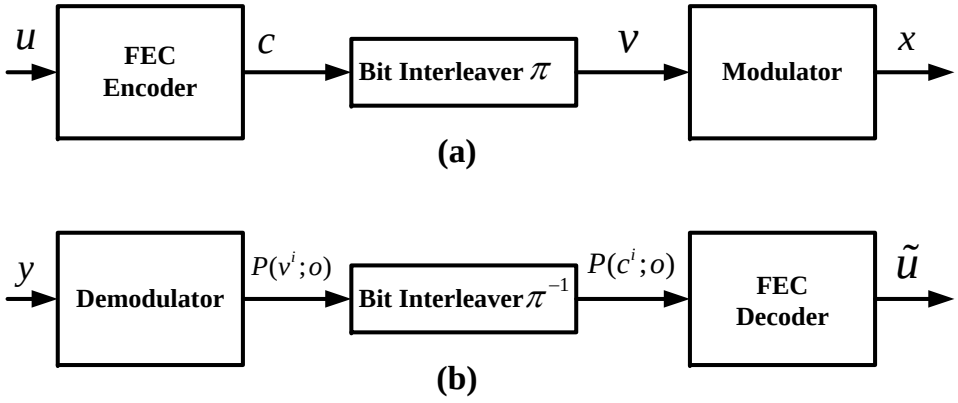


Figure 1.11 — Transmitter(a) and receiver(b) in the BICM system

$$x_t = \left\{ \left(x_t^I, x_t^Q \right) \in \chi \right\} \tag{1.16}$$

x_t^I and x_t^Q represent the in-phase(I) and quadrature(Q) components of μ_m at the sampling time t. χ represents the possible 2^m signals in the constellation set.

The bit interleaver function for every transmitted bits is defined as equ. (1.17), where $i = \{0, 1, ..., m-1\}$ and N_s is the number of symbols of the constellation in the coded frame, t and $t' \in [1, N_s]$. The interleaving function is a modulo m mathematical operation that divides the coded sequence into m subsequences. Uniform interleaver and S-random interleaver are the two possible interleaving schemes. The interleaving function is considered to break any existing correlation between the m subsequences.

$$\pi\left(mt + i\right) = \quad \bmod\left((mt' + i), N_s m\right); \tag{1.17}$$

Bits-to-symbol mapping plays a critical role in the BICM system. It greatly affects the bit error correcting performance. There are several ways to label the symbols: Gray, mixed, set-partitioning, modified set-partitioning and random. The mostly used is Gray mapping, which characterized by only one different bit between one constellation signal and its neighbors.

At the receiver side, the demodulator (demapper) provides probabilities on transmitted bit sequence v^i, $i = \{0, 1, ..., m-1\}$. At sample time t, the output of the demodulator which is the probability of error on bit b_t^i noted $P\left(v_t^i = b|y; O\right)$ is expressed as equ. (1.18):

$$P\left(v_t^i = b|y; O\right) = \sum_{x_t \in \chi_b^i} P\left(x_t|y_t\right) = \sum_{x_t \in \chi_b^i} P\left(y_t|x_t\right) \cdot P\left(x_t\right) \tag{1.18}$$

where $b = \{0, 1\}$, χ_b^i represents the set of signals in the signal space. It contains the signals whose i^{th} bit equal to value b. $P\left(x_t\right)$ designates the *a priori* probability of transmitted x_t.

With perfect CSI, the soft output of the demodulator is illustrated as equ. (1.19):

$$P\left(v_t^i = b|y;O\right) = \sum_{x_t \in \chi_b^i} \frac{P(x_t)}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{\left|y_t^I - \rho_t x_t^I\right|^2 + \left|y_t^Q - \rho_t x_t^Q\right|^2}{2\sigma^2}\right) \tag{1.19}$$

where σ^2 represents the channel variance and ρ_t represents the amplitude of the fading. $P(x_t)$ is not available at the receiver side. It is assumed to be 2^{-m} with each bit having an equal probability 2^{-1}. After the de-interleaving step, $P\left(c_t^i = b|y;O\right)$ becomes the input of the soft input decoder $P\left(c_t^i = b|y;I\right)$.

1.4.4 Improving the performance of BICM system over a fading channel

Signal space diversity improves the diversity order of BICM. It consists of rotating the constellation and interleaving one of the component axis(I or Q) with respect to the other. The coordinate interleaving was first proposed in [30], whereas the constellation rotation coupled with coordinate interleaving was first proposed in [8].

The choice of the rotated angle α is selected based on maximizing of the so-called product distance introduced in [8]. The author of [31] used the bit error rate as the criterion to determine the optimal rotation matrices. In [32] a new criterion was proposed to achieve good performance for the fading channel with and without erasures. [33] determines the optimal rotation angle by maximizing average mutual information at the output of demapper.

In [34] and [35], it has been shown that the performance of a BICM system can be improved by iterative decoding (ID) using hard-decision feedback by proper bit labeling of the constellation. So the minimum inter signal Euclidean distance of a BICM system can be increased while retaining the desirable Hamming distance. BICM with iterative decoding greatly outperforms TCM and compares favorably with bandwidth-efficient turbo TCM over AWGN, while significantly outperforms over a Rayleigh and Ricean channel. The authors in [36] firstly adopted the soft-input soft-output(SISO) module for the convolutional decoding and optimised the bit-metric calculation of the soft-output feedback from the decoder to further improved the performance of BICM-ID.

Combining iterative BICM with signal space diversity (BICM-ID-SSD) can further improve the performance over Rayleigh fading channel. The authors in [32] first applied BICM-ID-SSD with modern codes (Turbo codes, LDPC codes), which provides attractive performance gains over fading channel with and without erasures. Fig. 1.13 shows some performance comparison for 16-QAM with the 64K LDPC code in DVB-T2 (coding rate R=4/5) over fading channel with 15% erasures. The value of performance gain from signal space diversity

and iterative process chances according to different coding rate and different constellation size.

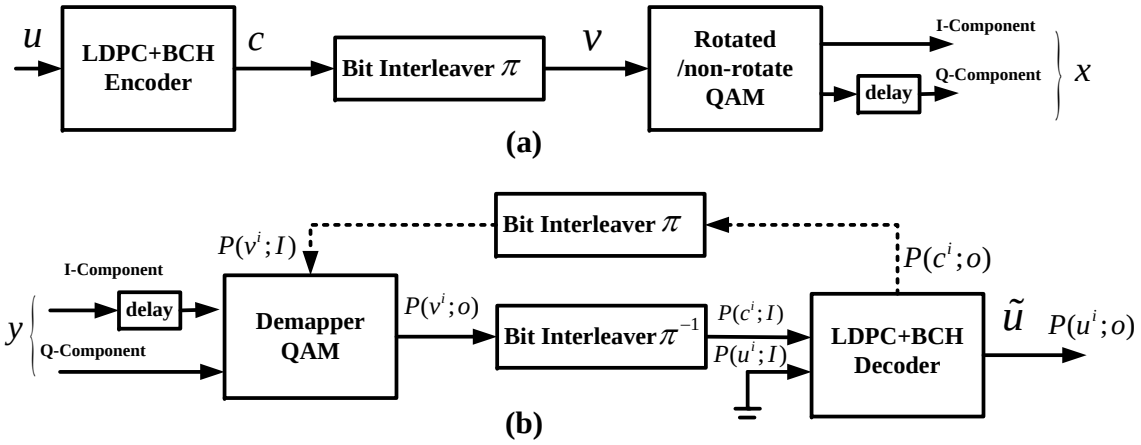


Figure 1.12 — System description of the BICM-SSD and BICM-ID-SSD mode

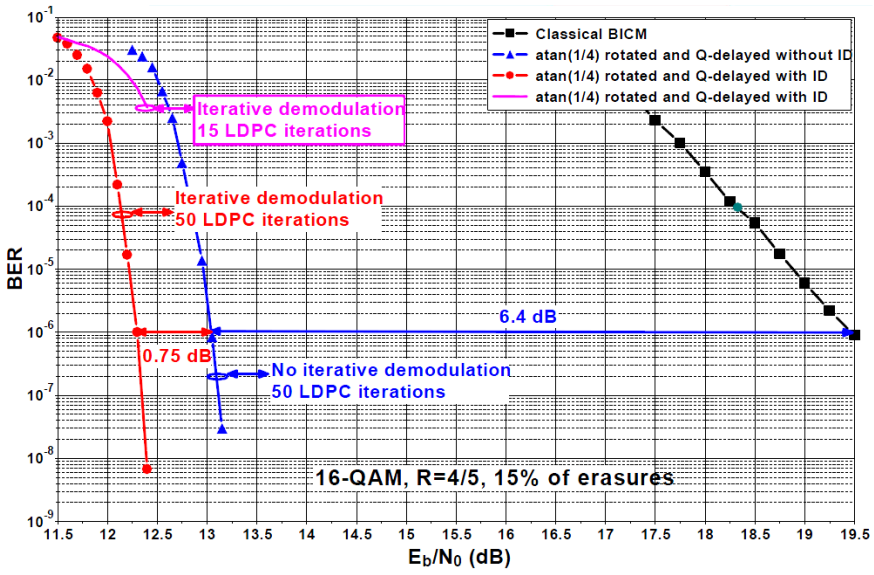


Figure 1.13 — Performance comparison for 16-QAM over fading channel with 15% erasures
64K LDPC code rate R=4/5

1.5 DVB-T2 standard introduction

Digital Video Broadcasting-Terrestrial (DVB-T) is the most widely developed digital terrestrial television system worldwide over thirty countries. In order to increase its spectral efficiency and enable new services, the DVB organization has developed a next generation of DVB-T2 standard [37]. The latest advanced coding and modulation technologies have been included in the specification of DVB-T2 to provide high capacity and improve robustness

in the terrestrial transmission environment. *Multiple-Input Single-Output* (MISO) technique, *Low-Density Parity Codes* (LDPC), rotated constellation, new pilot pattern are the most remarkable new techniques in the DVB-T2 standard.

Table 1.1 lists the system configuration for both DVB-T2 and DVB-T. The LDPC codes combined with BCH codes are used as the *Forward Error Correcting* (FEC) part in DVB-T2, which improve coding gain when compared to the concatenated codes in DVB-T. DVB-T2 introduces a higher order constellation 256-QAM and the four QAM constellations (from QPSK to 256-QAM) can be rotated to achieve high coding diversity. Four different FFT sizes are added in the DVB-T2. 16K and 32K FFT increase the date rate while 1K makes the system more robust in the mobile channel. The added three different guard intervals provide more flexible selection of the OFDM parameters and let the receiver can endure high delayed multipath. In addition, two new bandwidth are included: 10MHz for professional application, 1.712MHz for the narrow RF channel such as L-band. Eight different scattered pilots are available for DVB-T2, which is selected according to the FFT size and guard interval. Moreover, DVB-T2 can support *Multiple Input Single Output* (MISO) which can improve the overall performance in SFN using a transmitter diversity technique based on Alamouti encoding [4].

	DVB-T	**DVB-T2**
FEC	Convolutional Code + RS 1/2, 2/3, 3/4, 5/6, 7/8	LDPC + BCH 1/2, 3/5, 2/3, 3/4, 4/5, 5/6
Modulations	QPSK, 16QAM, 64QAM	QPSK, 16QAM, 64QAM 256QAM rotated or non
FFT Size	2K,8K	2K, 8K 1K, 4K, 16K, 32K
Guard Interval	1/4, 1/8, 1/16, 1/32	1/4, 1/8, 1/16, 1/32 19/256, 19/128, 1/128
Bandwidth	5, 6, 7, 8 MHz	5, 6, 7, 8 1.712, 10 MHz
Scattered Pilots	8% of total	1%, 2%, 4%, 8% of total
Continual Pilots	2.6% of total	0.35% of total

Table 1.1 — Comparison of available modes in DVB-T and DVB-T2

Fig. 1.14 illustrates the basic physical layer structure of a DVB-T2 transmitter. The system input may be one or more MPEG-2 or MPEG-4 *Transport Streams* (TS) and/or one or more Generic Stream. The input Pre-Processor is also a part of the DVB-T2 system, which has a service splitter or de-multiplexer for *Transport Streams* to separate the services into the T2 system inputs. Those are the input of the DVB-T2 transmitter. One logical data stream

is carried by one *Physical Layer Pipe* (PLP). The mode adaptation modules, which operate separately on the contents of each PLP, slice the input data stream into data fields which will form baseband frames (BBFRAMEs) after stream adaptation.

Each BBFRAME shall be processed by the FEC coding subsystem (outer coding BCH and inner coding LDPC) to generate a FECFRAME. Then the FECFRAME shall be mapped to a coded and modulated FEC block by de-multiplexing the input bits into parallel cell words first and then mapping these cell words into constellation values. These cell words are then undertaken cell interleaver and time interleaver to mitigate the effect of selective and /or time varying fading. In the rest of our study, we mainly focus on the LDPC, bit-interleaver and mapper highlighted in blue in Fig. 1.14 and the corresponding dual functions at the receiver side.

The frame builder assembles the cells produced by the time interleavers for each of the PLPs and the cells of the modulated Layer1 signalling into arrays of active OFDM cells corresponding to each OFDM symbol which makes up the overall frame structure. The Layer1 signalling provides the receiver with a means to access physical layer pipes within the DVB-T2 frames.

The function of the OFDM generation module is to take the cells produced by the frame builder to insert the relevant reference information, known as pilots, which allow the receiver to compensate for the distortions introduced by the transmission channel, and to produce from this the basis for the time domain signal for transmission. It then inserts guard intervals to produce the completed DVB-T2 signal.

1.5.1 Advanced bit-interleaved coded modulation for the DVB-T2 standard

Signal Space Diversity has been widely studied with simple codes. It had never been studied and published with advanced forward error correcting code, such as Turbo codes or LDPC codes before 2008. In [32] and [38], BICM-SSD and BICM-SSD-ID have been studied with Turbo codes and LDPC codes. The exhibited gains lead to their adoption in the DVB-T2 standard.

A description of this system is given in Fig. 1.12 and a solution to double the diversity order relies on two independent parts as shown in Fig. 1.15.

1. Correlating the in-phase I and quadrature Q components of the transmitted symbols by constellation rotation. Rotation is applied for the squared Gray labelled QAM to make the coordinates correlated, hence maximizes the diversity order. In the rotated constellation, each symbol could be uniquely identified based on the projection on axis **I** or **Q**.

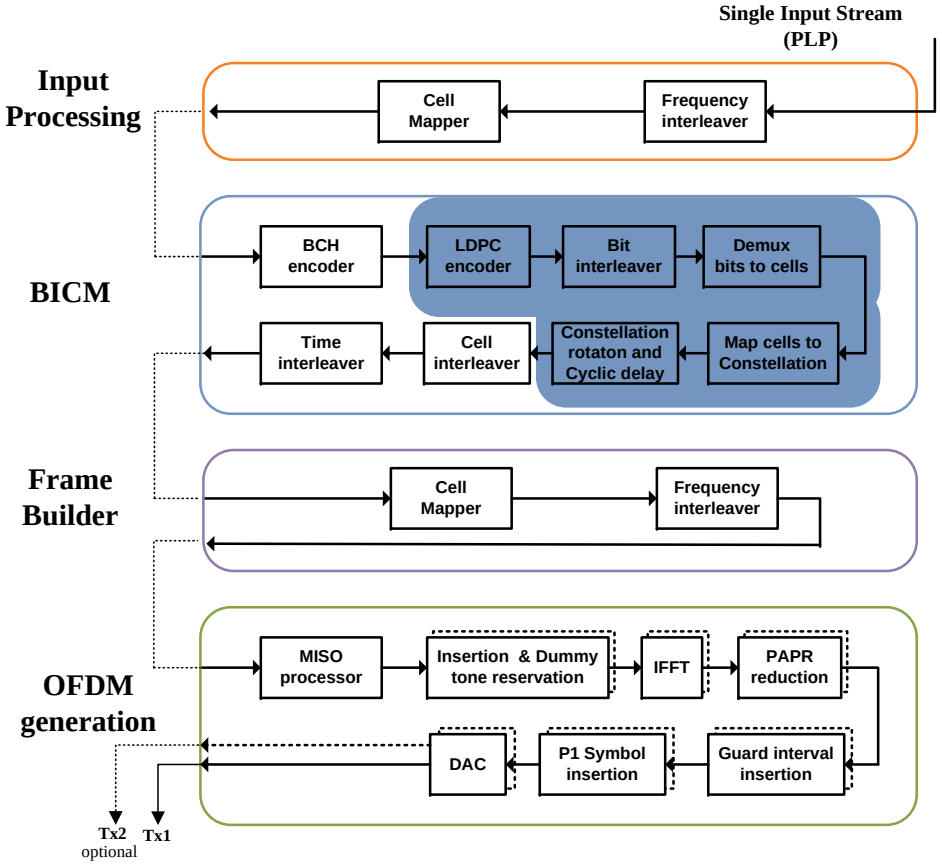


Figure 1.14 — Physical layer structure of a DVB-T2 transmitter

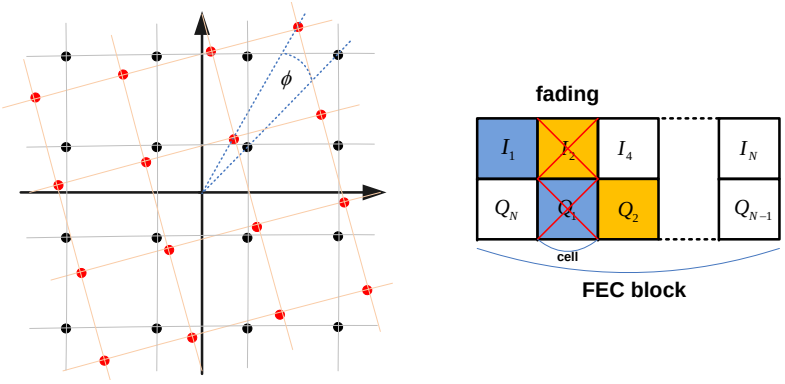


Figure 1.15 — Signal space diversity scheme

The in-phase I contains the intrinsic information of quadrature Q component and vice-versa. The optimum choice of the angle depends on constellation order is shown in Table. 1.2. The rotated constellation for 16-QAM is shown in Fig. 1.16.

2. Making these two components faded independently. The in-phase I and quadrature Q components for a same symbol can be made to be faded independently if they are not concurrently transmitted during the same fading time interval. This can be reached by adding

Modulation	QPSK	16-QAM	64-QAM	256-QAM
ϕ (degrees)	$29,0$	$16,8$	$6,8$	atan (1/16)

Table 1.2 — Rotation angle for each modulation type

independent interleaver for I and Q components of the transmitted symbol. A practical solution is to replace the coordinates interleaver with a cyclic delay of one of the component (Q is adopted in the standard) before the cell-interleaver and time-interleaver.

In consequence, the detection of the transmitted signal is still possible with a cyclic delayed and constellation rotated system in case the transmitted signal suffers a completely fading.

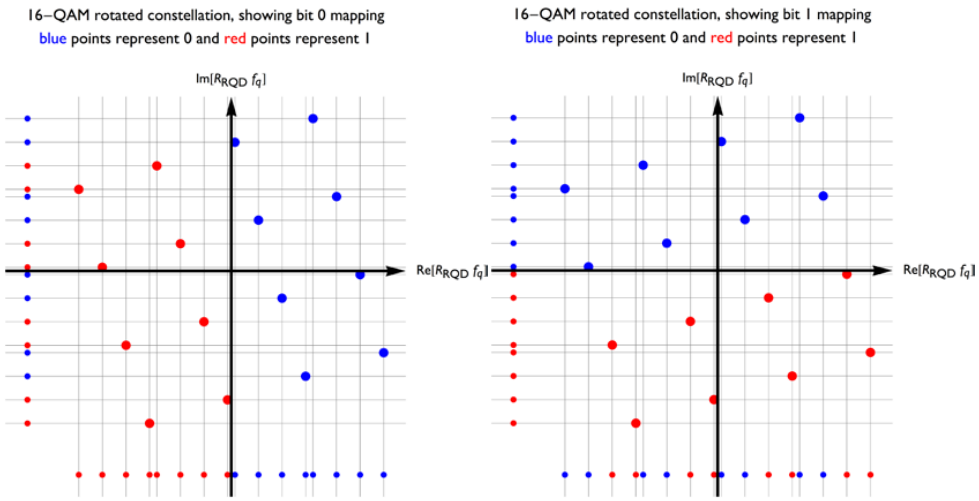


Figure 1.16 — Constellation of non-rotated and rotated 16-QAM

1.5.2 LDPC codes of DVB-T2

The LDPC codes in the DVB-T2 standard are the *irregular repeat accumulate* (IRA) LDPC codes in the same family of LDPC code in DVB-S2, which support two different frame lengths (16200 bits for short code length and 64800 for long code length) and a set of code rates (R1/4, R1/2, R3/5, R2/3, R3/4, R4/5, R5/6). It is defined by the parity check matrix $\mathbf{H}_{(n-k)\times n} = [\mathbf{I}_{(n-k)\times k} | \mathbf{P}_{(n-k)\times(n-k)}]$, as shown in Fig. 1.17.

There are two types of variable nodes, the information and parity nodes, corresponding to the systematic bits $(i_0, i_1, \cdots, i_{k-1})$ and parity bits $(p_0, p_1, \cdots, p_{n-k-1})$ respectively. The information part of the parity check matrix consists of two sets, $\mathbf{A}$ with a bit node degree larger than 3 (the value changes according to code rate), $\mathbf{B}$ with a constant bit node degree 3, as shown in Fig. 1.17. The parity part is a staircase lower triangular matrix. The code can also be represented by a bipartite graph, called Tanner graph [18], as illustrated in Fig. 1.18. The connection between the information nodes and the check nodes is semi-random

via a permutation, which is defined by the encoding rules defined in [37]. The connection between the parity nodes and the check nodes is a fixed zigzag pattern, which results in the accumulative encoding property.

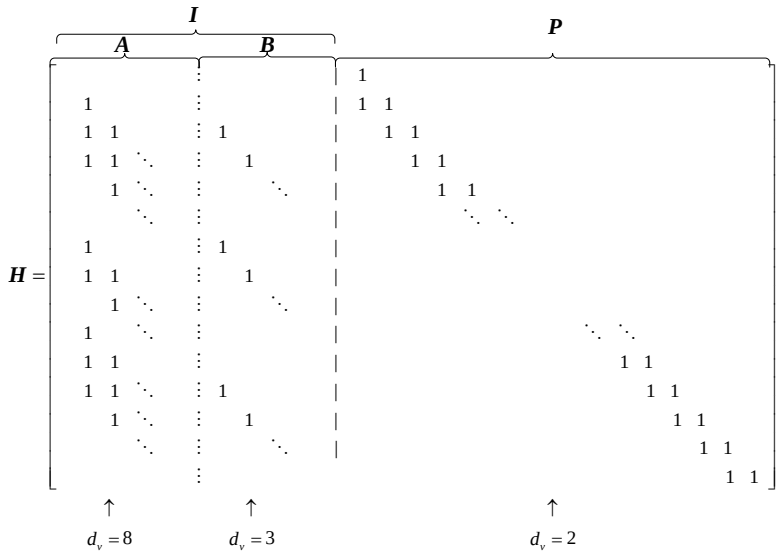


Figure 1.17 — Parity Check Matrix of the LDPC code in DVB-T2

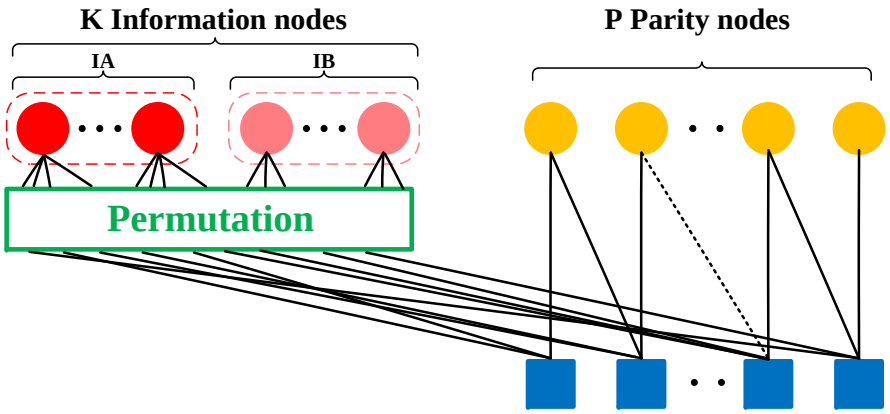


Figure 1.18 — Tanner graph of the LDPC code in DVB-T2

1.5.2.1 Encoding method of LDPC codes in DVB-T2

The set of the valid codeword C have to satisfy $H^{\intercal} \cdot C = 0$, as illustrated in equ. (1.20). The zigzag pattern of the connection between the parity bits and check nodes shows the accumulative property of the code (equ. (1.21)). Therefore, the parity bits can be derived from the a recursive addition operation on GF(2). The encoding algorithm can be explained by equ. (1.22).

Algorithm 1 Encoding algorithm of the LDPC code in DVB-T2

1: **Initialization:**

2: $p_0 = p_1 = \cdots = p_{n-k-1} = 0$

3: **Encoding:**

4: `for the ith bit, where mod(i,360)=0`

5: $p_j = p_j \oplus b_i, \quad j = 1, 2, \cdots, dv_i$

{ p_j *are the parity bit addresses specified in annex A and B in the standard*}

6: `for the ith bit, where mod(i,360)`$\neq$ `0`

7: $p_x = p_x \oplus b_i, \quad x = 1, 2, \cdots, dv_i$, where $p_x = mod\left((p_j + mod(i, 360) \cdot q), (n_{ldpc} - k_{ldpc})\right)$

{ q *is specified in the standard, see table.1.3*}

8: `for the ith parity bit, where i = 1,2,`$(n_{ldpc} - k_{ldpc})$

9: $p_0 = p_0$, $p_i = p_i \oplus p_{i-1}$

$64K$		$16K$	
rate	q	rate	q
1/2	90	1/2	54
3/5	72	3/5	80
2/3	60	2/3	108
3/4	45	3/4	120
4/5	36	4/5	126
5/6	30	5/6	135

Table 1.3 — q value for all code rates of 64K and 16K LDPC codes

$$\left[\begin{array}{cccc|ccccc} a_{00} & a_{01} & ... & a_{0(k-1)} & 1 & 0 & ... & 0 & 0 \\ a_{10} & a_{11} & ... & a_{1(k-1)} & 1 & 1 & ... & 0 & 0 \\ a_{20} & a_{21} & ... & a_{2(k-1)} & 0 & 1 & ... & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & 1 & \vdots & \vdots \\ a_{(p-2)0} & a_{(p-2)1} & ... & a_{(p-2)(k-1)} & 0 & 0 & ... & 1 & 0 \\ a_{(p-1)0} & a_{(p-1)1} & ... & a_{(p-1)(k-1)} & 0 & 0 & ... & 1 & 1 \end{array}\right]^{\mathsf{T}} \cdot \begin{bmatrix} i_0 \\ \vdots \\ i_{k-1} \\ p_0 \\ \vdots \\ p_{n-k-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{1.20}$$

$$\begin{array}{cccccc} a_{00}i_0 & \oplus a_{01}i_1 & ... & \oplus a_{0(k-1)}i_{(k-1)} & \oplus p_0 & \oplus 0 = 0; \\ a_{10}i_0 & \oplus a_{11}i_1 & ... & \oplus a_{1(k-1)}i_{(k-1)} & \oplus p_0 & \oplus p_1 = 0; \\ a_{20}i_0 & \oplus a_{21}i_1 & ... & \oplus a_{2(k-1)}i_{(k-1)} & \oplus p_1 & \oplus p_2 = 0; \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots = 0; \\ a_{(p-2)0}i_0 & \oplus a_{(p-2)1}i_1 & ... & \oplus a_{(p-2)(k-1)}i_{(k-1)} & \oplus p_{n-k-3} & \oplus p_{n-k-2} = 0; \\ a_{(p-1)0}i_0 & \oplus a_{(p-1)1}i_1 & ... & \oplus a_{(p-1)(k-1)}i_{(k-1)} & \oplus p_{n-k-2} & \oplus p_{n-k-1} = 0; \end{array} \tag{1.21}$$

$$\begin{array}{lllll} p_0 & = a_{00}i_0 & \oplus a_{01}i_1 & \cdots & \oplus a_{0(k-1)}i_{(k-1)} \quad \oplus 0; \\ p_1 & = a_{10}i_0 & \oplus a_{11}i_1 & \cdots & \oplus a_{1(k-1)}i_{(k-1)} \quad \oplus p_0; \\ p_2 & = a_{20}i_0 & \oplus a_{21}i_1 & \cdots & \oplus a_{2(k-1)}i_{(k-1)} \quad \oplus p_1; \\ \vdots & \vdots & \vdots & \ddots & \vdots \quad \vdots \\ p_{n-k-2} & = a_{(p-2)0}i_0 & \oplus a_{(p-2)1}i_1 & \cdots & \oplus a_{(p-2)(k-1)}i_{(k-1)} \quad \oplus Pt_{n-k-3}; \\ p_{n-k-1} & = a_{(p-1)0}i_0 & \oplus a_{(p-1)1}i_1 & \cdots & \oplus a_{(p-1)(k-1)}i_{(k-1)} \quad \oplus Pt_{n-k-2}; \end{array} \tag{1.22}$$

1.5.2.2 Properties of LDPC codes in DVB-T2

From the Annex A and Annex B of the DVB-T2 standards [37], we could not only know the way of encoding but also know the construction of the codes. For bit i which satisfies mod(i,360) = 0, the index of j of a_{ij}, whose value equals to one, is defined in the standard. The rest of a_{ij} equal to zero. We denote the index of j, which is defined in the standard, as $j^* = j_1^*, j_2^*, \cdots, j_{dv-1}^*$. For the other bit i (mod(i,360)$\neq$0), the index of j for a_{ij}, whose value equals to one can be calculated based on equ. (1.23).

$$mod\left((j^* + mod(i, 360) \cdot q), (n_{ldpc} - k_{ldpc})\right) \tag{1.23}$$

Table 1.4 and Table 1.5 give the codes parameters of the long and short LDPC codes.

rate	$Pldpc$	$Kldpc$	dv_a		dv_b		dc	Total edges
			value	column	value	column		
1/2	90x360	90x360	8	36x360	3	54x360	7	630x360
3/5	72x360	108x360	12	28x360	3	80x360	10	720x360
2/3	60x360	120x360	13	12x360	3	108x360	10	600x360
3/4	45x360	135x360	12	15x360	3	120x360	14	630x360
4/5	36x360	144x360	11	18x360	3	126x360	18	648x360
5/6	30x360	150x360	13	15x360	3	135x360	22	660x360

Table 1.4 — All the parameters of the long codes (N=64800)

In fact, it is possible to process one group of the connection between the check nodes B and bit nodes C simultaneously. With the re-arrangement of the check node order as equ. (1.24) and parity variable bits as shown in equ. (1.25), the parity check matrix of the LDPC codes becomes quasi-cyclic, as illustrated in Fig. 1.19.

$$B = \{l \cdot M, l \cdot M + 1, \cdots, l \cdot M + M - 1\}, l \in [0, 1, \cdots, \frac{N_{ldpc}}{M}], M = 360 \tag{1.24}$$

$$C = \{r, r \cdot q, \cdots, r \cdot (M-1) \cdot q\}, r = mod(j^*, q) \tag{1.25}$$

rate		*Pldpc*	*Kldpc*	dv_a		dv_b		dc	Total edges
identifier	effective			value	column	value	column		
1/4	1/5	36x360	9 x360	12	4x360	3	5 x360	Max 4	135x360
1/2	4/9	25x360	20x360	8	5x360	3	15x360	Max 7	135x360
3/5	3/5	18x360	27x360	12	5x360	3	22x360	9	162x360
2/3	2/3	15x360	30x360	13	3x360	3	27x360	10	150x360
3/4	11/15	12x360	33x360	12	1x360	3	32x360	Max 13	132x360
4/5	7/9	10x360	35x360	0	0x360	3	35x360	Max 13	125x360
5/6	37/45	8 x360	37x360	13	1x360	3	36x360	Max 19	137x360

Table 1.5 — All the parameters of the short codes (N=16200)

The check node indexes with the re-arrangement are listed in equ. (1.26).

$$(0, q, 2q, \cdots, 359q), (1, q+1, 2q+1, \cdots, 359q+1), \cdots, (q-1, 2q-1, 3q+1, \cdots, 360q-1) \tag{1.26}$$

Having such a structure, decoder can decode cyclic blocks in parallel. However, the parallel decoding can introduce additional constraints. In fact, memory conflicts can occur when a double-diagonal is observed in certain sub-matrix, in which the row and column degree is bigger than one. A detailed explanation and corresponding solution is detailed in Chapter 3.

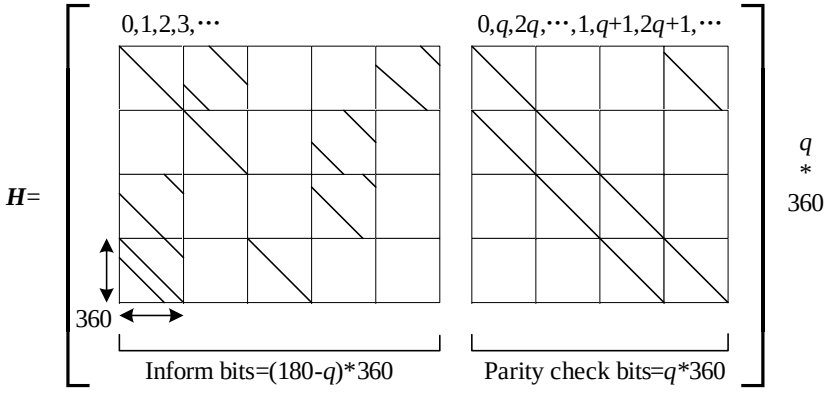


Figure 1.19 — The LDPC code in the DVB-T2 standard with quasi-cyclic property

1.6 Conclusion

In this chapter, the considered digital communication system and corresponding elements were reviewed first. A brief introduction of the error control codes including block codes, convolutional codes and iteratively decoded codes (Turbo codes and LDPC codes) was carried out. Then we made a description of the fading channel model which is used in the rest of our

study and coded modulation (TCM, BICM). As a final section, we made a description of our study environment: the DVB-T2 standard and especially the FEC part of DVB-T2.

CHAPTER 2

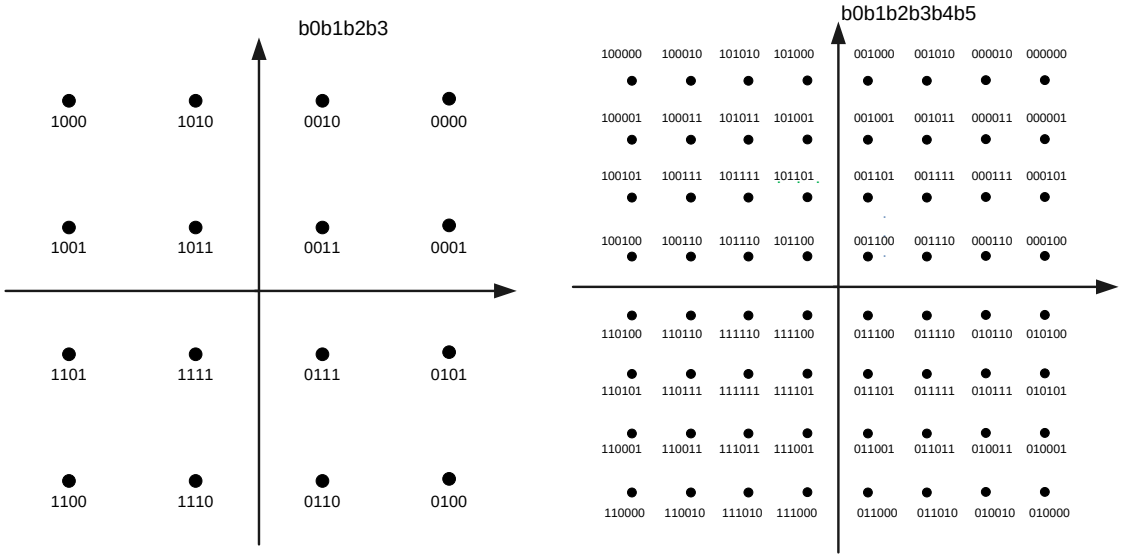
Design and implementation of a flexible demapper

In this chapter we carry out a detailed study of different demapping algorithms in order to design a prototype for non-rotated and rotated QAM. A review of the traditional piecewise linear approximation (one-dimensional) for non-rotated QAM is presented. The demapping algorithm for the rotated QAM does not follow this approach. In fact, a one-dimensional demapping algorithm for the rotated QAM cannot guarantee the performance gain of a BICM-SSD system. A two-dimensional demapping algorithm is required. Unfortunately, its complexity is a crucial problem for high order constellations, like 256-QAM. A novel demapping algorithm based on sub-region detection with Max-Log approximation is proposed for the DVB-T2 standard. It greatly reduces the hardware complexity while maintaining good performance gains.

Simulation of different demapping algorithms confirms the efficiency of the sub-region detection algorithm. A flexible Max-Log demapper based on the sub-region detection algorithm has been designed and implemented on an FPGA target. Moreover, a prototype of an uncoded transmission chain has been developed to demonstrate the efficiency of the design. To the best of our knowledge, such a sub-region based detection is quite new and could be applied both for the rotated and non-rotated QAM especially for very high order modulations, such as 1024-QAM or 4096-QAM adopted in the coming DVB-C2 standard.

2.1 Demapping algorithm for non-rotated QAM

According to the system description of sub-section 1.2.3, the demapper computes the *Log Likelihood Ratio* (LLR) of v_t^i related to the i^{th} bit based on v_t for the received symbol y_t $\left(y_t^I, y_t^Q\right)$. A positive LLR would indicate that v_t^i is more probably transmitted as 0 and a negative value would indicate v_t^i is more probably transmitted as 1. The probability computation is written in equ. (2.1).



(a) non-rotated 16-QAM (b) non-rotated 64-QAM

Figure 2.1 — Signal space panel for non-rotated 16-QAM and 64-QAM

Assuming y is the received symbol as indicated in Fig. 2.1(a), y^I and y^Q are the In-phase and Quadrature components of y. Let us take a 16-QAM constellation as an example to explain.

$$LLR(v_t^i) = log\left(\frac{Pr(\hat{v}_t^i = 0)|y_t^I, y_t^Q}{Pr(\hat{v}_t^i = 1)|y_t^I, y_t^Q}\right) \tag{2.1}$$

$$= log\left(\frac{\sum\limits_{x_t \in \chi_0^i} \frac{P(x_t)}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{\left|y_t^I - \rho_t x_t^I\right|^2 + \left|y_t^Q - \rho_t x_t^Q\right|^2}{2\sigma^2}\right)}{\sum\limits_{x_t \in \chi_1^i} \frac{P(x_t)}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{\left|y_t^I - \rho_t x_t^I\right|^2 + \left|y_t^Q - \rho_t x_t^Q\right|^2}{2\sigma^2}\right)}\right) \tag{2.2}$$

The sets of signals equal to the value zero and the value one for mapping bit v_t^i in a 16-QAM of the DVB-T2 standard are listed as:

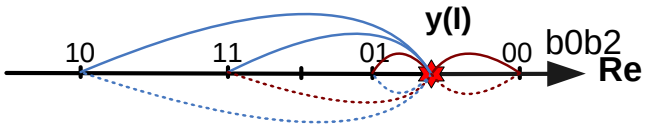


Figure 2.2 — Simplified LLR computation for v^0 and v^2 , in a PAM mode

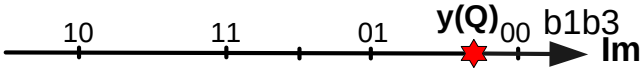


Figure 2.3 — Simplified LLR computation for v^1 and v^3 , in a PAM mode

$$\chi_0^0 = \{0,1,2,3,4,5,6,7\} \quad \chi_1^0 = \{8,9,10,11,12,13,14,15\}$$
$$\chi_0^2 = \{0,1,4,5,8,9,12,13\} \quad \chi_1^2 = \{2,3,6,7,10,11,14,15\}$$
$$\chi_0^1 = \{0,1,2,3,8,9,10,11\} \quad \chi_1^1 = \{4,5,6,7,12,13,14,15\}$$
$$\chi_0^3 = \{0,1,4,5,8,9,12,13\} \quad \chi_1^2 = \{2,3,6,7,10,11,14,15\}$$

The M-QAM could be regarded as two independent $\sqrt{m}$-PAM, because of the independence of I and Q components. LLR computation for v^0, v^2 and v^1, v^3 are depicted in Fig. 2.2 and Fig. 2.3, respectively. Hence the computation of the two-dimensional Euclidean distance can be reduced into one dimensional, thanks to the independency of I and Q components in the case of non-rotated QAM.

The simplified computation of the LLRs is illustrated as equ. (2.3), where χ_0^0 is reduced to the set $\{3,1\}$ and χ_1^0 is reduced to the set $\{9,11\}$ for v^0 and v^2.

$$LLR(\hat{v}_t^i) = log\left(\frac{\sum\limits_{x_t \in \chi_0^i} \frac{P(x_t)}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{\left|y_t^I - \rho_t x_t^I\right|^2}{2\sigma^2}\right)}{\sum\limits_{x_t \in \chi_1^i} \frac{P(x_t)}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{\left|y_t^I - \rho_t x_t^I\right|^2}{2\sigma^2}\right)}\right) \tag{2.3}$$

For the sake of simplicity, $P(x_t)$ is assumed to be equivable (2^{-m}) and the signal space is normalized. For the moment we focus on the AWGN channel, hence ρ_t is equal to one. Under this scenario, the demapping equation for v^0 can be written as:

$$LLR(\hat{v}_t^0) = log\left(\frac{\exp\left(-\frac{\left(y_t^I - \frac{1}{\sqrt{10}}\right)^2}{2\sigma^2}\right) + exp\left(-\frac{\left(y_t^I - \frac{3}{\sqrt{10}}\right)^2}{2\sigma^2}\right)}{\exp\left(-\frac{\left(y_t^I + \frac{1}{\sqrt{10}}\right)^2}{2\sigma^2}\right) + exp\left(-\frac{\left(y_t^I + \frac{3}{\sqrt{10}}\right)^2}{2\sigma^2}\right)}\right) \tag{2.4}$$

The Max-Log approximation shown in equ. (2.5) is also applied to reduce complexity.

$$\ln\left(\exp(a_1) + \ldots + exp(a_k)\right) \approx \max_{i=1,\ldots,k}(a_i) \tag{2.5}$$

Therefore, equ. (2.4) is rewritten as equ. (2.6)

$$LLR(\hat{v}_t^0) = log\left(\frac{\max\left\{\exp\left(-\frac{\left(y_t^I - \frac{1}{\sqrt{10}}\right)^2}{2\sigma^2}\right), exp\left(-\frac{\left(y_t^I - \frac{3}{\sqrt{10}}\right)^2}{2\sigma^2}\right)\right\}}{\max\left\{\exp\left(-\frac{\left(y_t^I + \frac{1}{\sqrt{10}}\right)^2}{2\sigma^2}\right), exp\left(-\frac{\left(y_t^I + \frac{3}{\sqrt{10}}\right)^2}{2\sigma^2}\right)\right\}}\right) \tag{2.6}$$

The LLR computation for v_t^0 and v_t^2 can further be reduced to equ. (2.7) and equ. (2.8). These equations are more efficient in terms of hardware implementation.

$$LLR(\hat{v}_t^0) = \begin{cases} \frac{4}{\sigma^2 \cdot \sqrt{10}} \cdot \left(y_t^I - \frac{1}{\sqrt{10}}\right) & \text{if } y_t^I > \frac{2}{\sqrt{10}} \\ \frac{2}{\sigma^2 \cdot \sqrt{10}} & \text{if } \frac{2}{\sqrt{10}} \geq y_t^I \geq -\frac{2}{\sqrt{10}} \\ \frac{4}{\sigma^2 \cdot \sqrt{10}} \cdot \left(y_t^I + \frac{1}{\sqrt{10}}\right) & \text{if } y_t^I < -\frac{2}{\sqrt{10}} \end{cases} \tag{2.7}$$

$$LLR(\hat{v}_t^2) = \begin{cases} \frac{2}{\sigma^2 \cdot \sqrt{10}} \cdot \left(y_t^I - \frac{2}{\sqrt{10}}\right) & \text{if } y_t^I > 0 \\ -\frac{2}{\sigma^2 \cdot \sqrt{10}} \cdot \left(y_t^I + \frac{2}{\sqrt{10}}\right) & \text{if } y_t^I \leq 0 \end{cases} \tag{2.8}$$

The LLR computation for v_t^1 and v_t^3 follow the one for v_t^0 and v_t^2, but is based on the quadrature component of the received channel symbol. Fig. 2.4 illustrates the relationship between the LLR output and y_t^I of the input symbol for 256-QAM.

$$LLR(\hat{v}_t^1) = \begin{cases} \frac{4}{\sigma^2 \cdot \sqrt{10}} \cdot \left(y_t^Q - \frac{1}{\sqrt{10}}\right) & \text{if } y_t^Q > \frac{2}{\sqrt{10}} \\ \frac{2}{\sigma^2 \cdot \sqrt{10}} & \text{if } \frac{2}{\sqrt{10}} \geq y_t^Q \geq -\frac{2}{\sqrt{10}} \\ \frac{4}{\sigma^2 \cdot \sqrt{10}} \cdot \left(y_t^Q + \frac{1}{\sqrt{10}}\right) & \text{if } y_t^Q < -\frac{2}{\sqrt{10}} \end{cases} \tag{2.9}$$

$$LLR(\hat{v}_t^3) = \begin{cases} \frac{2}{\sigma^2 \cdot \sqrt{10}} \cdot \left(y_t^Q - \frac{2}{\sqrt{10}}\right) & \text{if } y_t^Q > 0 \\ -\frac{2}{\sigma^2 \cdot \sqrt{10}} \cdot \left(y_t^Q + \frac{2}{\sqrt{10}}\right) & \text{if } y_t^Q \leq 0 \end{cases} \tag{2.10}$$

Equ. (2.7) to equ. (2.10) show that the LLR computation is steadily expressed by a linear equation, $a(x - b)$, in a particular interval. Therefore, the LLR computation can be performed via a piecewise linear approximation. This approximation greatly reduces hardware complexity while having a limited impact on performance. It represents the state of the art for demapping algorithms.

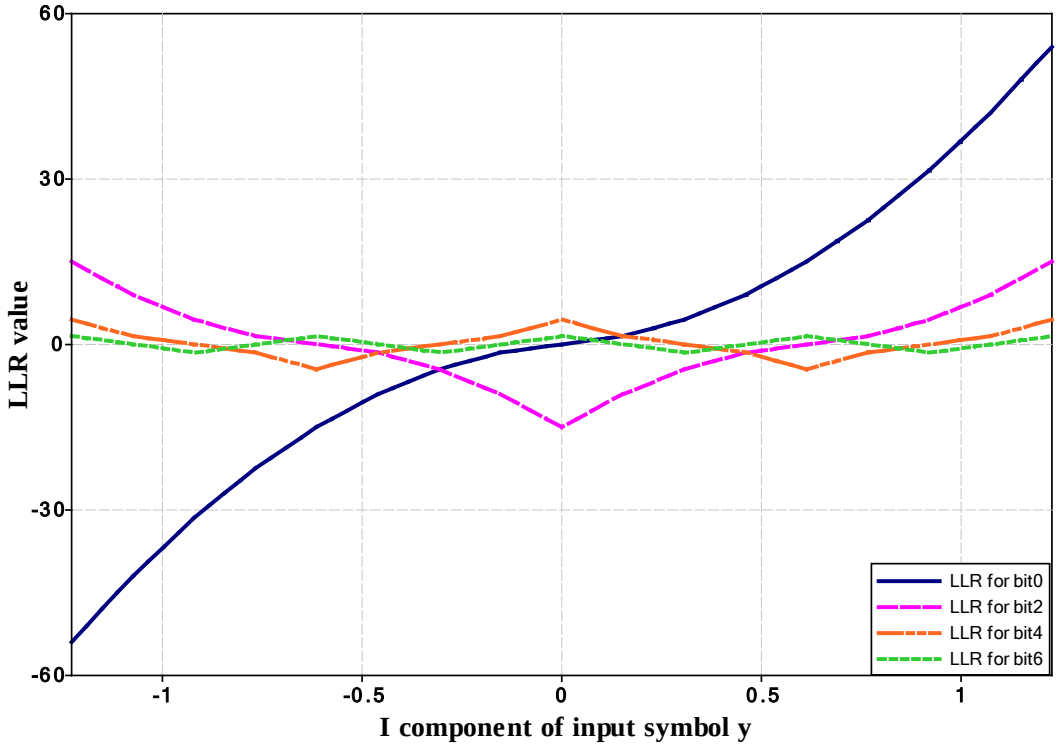


Figure 2.4 — LLR value of bits related to in-phase for 256-QAM over AWGN channel (@SNR=9dB)

2.2 Demapping algorithms for rotated QAM

In this section, two different demapping algorithms are described. The first algorithm has a low complexity since it is obtained with a method based on one-dimensional detection. However, the lower complexity comes at the price of a penalty in terms of performance. The second algorithm is a two-dimensional based algorithm. It achieves the expected gains when introducing SSD. Due to its high complexity, we have also proposed a novel simplified algorithm based on sub-region detection.

2.2.1 One-dimensional demapping algorithm

The number of signal space projections on every component axis for a non-rotated M-QAM is $\sqrt{M}$. However, the corresponding number of projections for a rotated M-QAM is M. In order to reuse the hardware demapping model of a non-rotated M-QAM, a de-rotation process can be applied prior to the piecewise linear demapping.

According to equ. (1.14), the received symbol is defined as $y_n = \rho_n \cdot x_n + n_n$. Hence the received symbol in the case of BICM with signal space diversity and Q delay becomes as:

$$y_t = y_{t-1}^I + j \cdot y_t^Q = \left(\rho_{t-1} \cdot x_{t-1}^I + n_{t-1}^I\right) + j \cdot \left(\rho_t \cdot x_{t-1}^Q + n_{t-1}^Q\right) \tag{2.11}$$

where $\rho_{t-1} = \rho^I$ and $\rho_t = \rho^Q$, if a Q-delayed BICM-SSD system is considered.

Equ. (2.1) becomes equ. (2.12) over a AWGN channel for the rotated constellation. (s_t^I, s_t^Q) denotes the transmitted symbol at sampling time t and (x^I, x^Q) is one of the signals in a non-rotated signal space.

$$LLR(v_t^i) = \ln\left(\frac{\displaystyle\sum_{x_t \in \chi_0^i} \frac{P(x_t)}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\parallel (s_{t-1}^I + n_{t-1}^I, s_t^Q + n_t^Q), (x^I, x^Q) \cdot e^{j\frac{2\pi\phi}{360}} \parallel^2}{2\sigma^2}\right)}{\displaystyle\sum_{x_t \in \chi_1^i} \frac{P(x_t)}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\parallel (s_{t-1}^I + n_{t-1}^I, s_t^Q + n_t^Q), (x^I, x^Q) \cdot e^{j\frac{2\pi\phi}{360}} \parallel^2}{2\sigma^2}\right)}\right) \tag{2.12}$$

Multiplied by a vector $\parallel e^{-j\frac{2\pi\phi}{360}} \parallel^2$, with an amplitude of one, the Euclidean distances are not changed. So the original demapping algorithm becomes:

$$LLR(v_t^i) = \ln\left(\frac{\displaystyle\sum_{x_t \in \chi_0^i} \frac{P(x_t)}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\parallel e^{-j\frac{2\pi\phi}{360}} \cdot (s_{t-1}^I + n_{t-1}^I, s_t^Q + n_t^Q), (x^I, x^Q) \parallel^2}{2\sigma^2}\right)}{\displaystyle\sum_{x_t \in \chi_1^i} \frac{P(x_t)}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\parallel e^{-j\frac{2\pi\phi}{360}} \cdot (s_{t-1}^I + n_{t-1}^I, s_t^Q + n_t^Q), (x^I, x^Q) \parallel^2}{2\sigma^2}\right)}\right) \tag{2.13}$$

This means that the LLR computation for the rotated QAM is based on the computation of non-rotated QAM with a de-rotation process of the input symbol.

Over a fading channel, the constellations are no longer square but shaped by the attenuation of in-phase and quadrature components. Any type of a rectangular constellation shaped by the attenuation of the fading channel is not suitable for the de-rotation algorithm. Fig. 2.5 represents the performance degression of a demapper with de-rotation algorithm over fading channel.

A compensation process can lead to a squared constellation, which supports the de-rotation process. Fig. 2.6 illustrates the process of a one-dimensional demapping algorithm over fading channel. The received symbol is firstly compensated to square and de-rotated to the non-rotated QAM. Then the piecewise linear demapper can be applied to the $\sqrt{M}$-PAM for the bits corresponding to each component. Equalization can also lead to a squared constellation before applying the de-rotation process. We call it CR-1D algorithm.

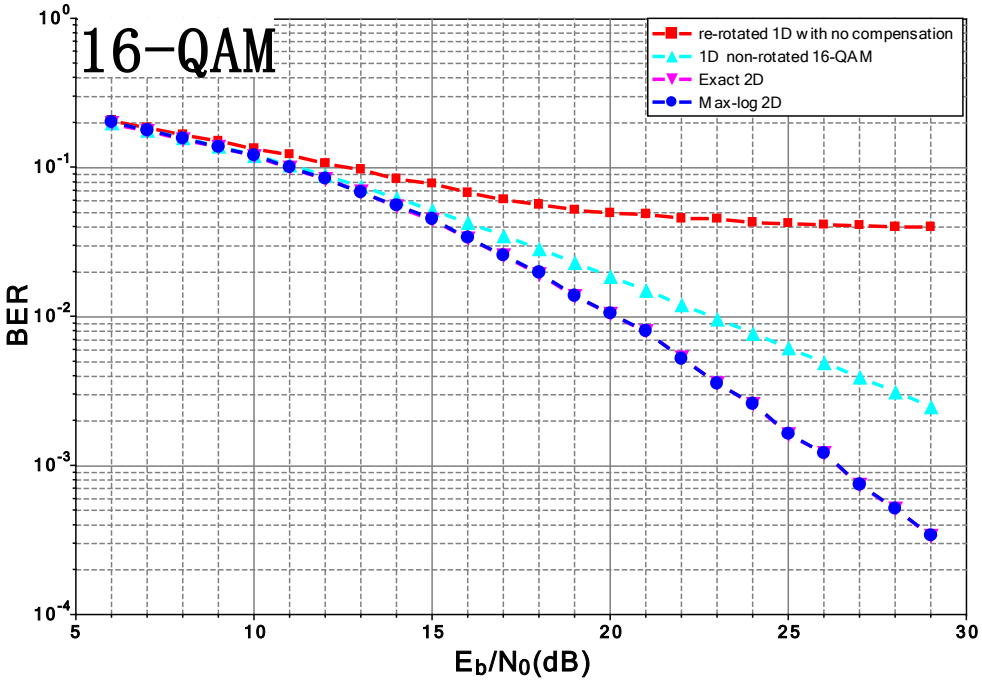


Figure 2.5 — Performance degradation for a non-compensated but re-rotated piecewise linear 1D demapping algorithm for different rotated QAM constellation over fading channel

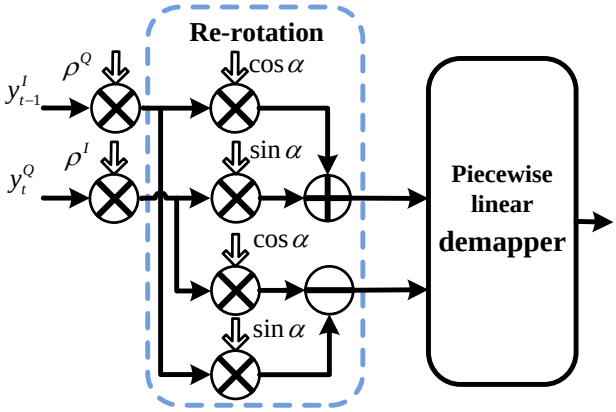


Figure 2.6 — The re-rotation and compensated one dimensional demapper for fading channel

2.2.2 Two-dimensional demapping algorithm and simplification

The rotated QAM breaks the independency between the I and Q components of the signals in the signal space panel, both I and Q components contribute to the estimation of the log likelihood of each transmitted bit. Therefore, the one-dimensional picewise linear approximation based directly on the channel observations widely used for non-rotated QAM is no longer applicable. A two-dimensional demapping algorithm shown in equ. (2.1) is required in order to get the performance gain from SSD.

By using the Max-Log approximation illustrated in equ. (2.5), the equ. (2.1) can be simplified to equ. (2.14), where $\mathbf{D}_{euc}$ denotes the squared Euclidean distance between the received symbol and the signals x in the set of χ_0^i or χ_1^i . Nevertheless, the demapping algorithm is of high computational complexity. For a 256-QAM, 256 squared two dimensional Euclidean distances have to be calculated in order to find the minimum values for each mapping bit, which requires 1024 multiplications.

$$LLR(v_t^i) = \frac{1}{2\sigma^2} \cdot \Big(min_{x_t \in \chi_0^i} \left(\mathbf{D}_{euc}(x_t) \right) - min_{x_t \in \chi_1^i} \left(\mathbf{D}_{euc}(x_t) \right) \Big) \tag{2.14}$$

In order to reduce the required computations, we propose a detection method based on the decomposition of the constellation into two-dimensional sub-regions in the signal space, called SR-2D. It leads to the computation of the squared Euclidean distances on a limited number of constellation points. This approach is particularly attractive for high order QAM constellations. It can also be applied to a fading channel with erasures.

The equ. (2.14) is a function of two different kinds of Euclidean distances: the shortest distance to the constellation signal where the value of the mapping bit is zero and the corresponding distance to the signal where the value of the mapping bit is one. The selected sub-region must contain the two shortest distances for each transmitted bit. A closer look at the Gray mapping of the constellation allows the partitioning of the constellation into sub-regions according to the sign of the received channel observation y_{t-1}^I and y_t^Q.

We take the rotated 64-QAM as an example. Table 2.1 shows the interval of x^I that corresponding to the projection of signals on the **Re** axis for the 3 different mapping bits v^i which takes the values 1 and 0, respectively. The same reasoning is applied to the Q component x^Q. From Table 2.1, we can see that bit v^0 sets the critical condition on the choice of the sub-region since $v^0 = 1$ for all $x^I < 0$ (respectively $v^0 = 0$ for all $x^I \geq 0$). In the region where $x^I < 0$, v^1 is symmetric with respect to $x = 4.01$ and v^2 is symmetric with respect to $x = 1.91$ and $x = 6.00$ according to the value 1 and 0. Consequently, $[-7.96, 1.90]$ is a good sub-region for $x^I < 0$ containing the signals which have the minimum Euclidean distance for each transmitted bit taking respectively the value 1 and 0. ($[-1.90, 7.96]$ for

$x^I \geq 0$ respectively) An illustration of the proposed sub-regions as a function of the sign of the received I and Q components is shown in Fig. 2.7.

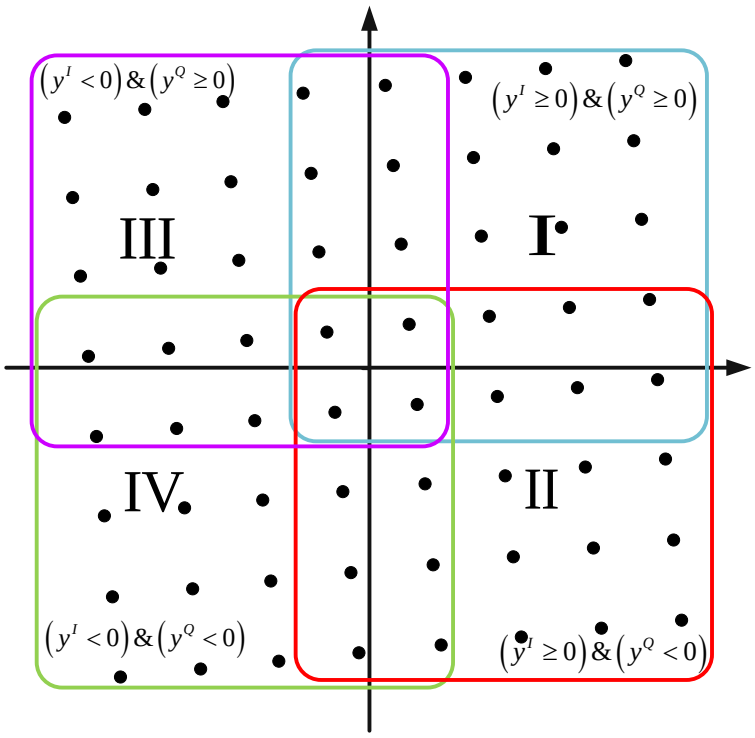


Figure 2.7 — The rotated 64-QAM constellation of DVB-T2 with the proposed sub-regions over fading channel

	Region of x^I with $v^i = 1$	Region of x^I with $v^i = 0$
v^0	$[-7.96, 0.06]$	$[-0.06, 7.96]$
v^1	$[-4.01, 4.01]$	$[-7.96, -3.89], [3.89, 7.96]$
v^2	$[-6.00, 1.91], [1.91, 6.00]$	$[-7.96, -5.89], [-2.04, 2.04], [5.89, 7.96]$

Table 2.1 — The interval of x^I for each bit v^i that takes the value 1 and 0 respectively

When an erasure occurs, the detection is limited to the projections of all the M signals on the non-erased remaining axis. Hence a one-dimensional demapping algorithm is applied in this case, which is illustrated in equ. (2.15).

$$LLR(v_t^i) = log\left(\frac{\sum\limits_{x_t \in \chi_0^i} \frac{P(x_t)}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{\left|y_t^I - \rho_t x_t^I\right|^2}{2\sigma^2}\right)}{\sum\limits_{x_t \in \chi_1^i} \frac{P(x_t)}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{\left|y_t^I - \rho_t x_t^I\right|^2}{2\sigma^2}\right)}\right) \tag{2.15}$$

where χ_0^i represents the projections of the set of signals in the constellation, whose i^{th} bit

equals to value zero. χ_1^i represents the projections of the set of signals in the constellation, whose i^{th} bit equals to value one.

The region partitioning should be based on the projections of all the constellation points on the non-erased axis. The sub-region can be divided depending on the sign of the non-erased component. Each sub-region contains $\log_2(M) \cdot (1 + \log_2(M)/2)$ points for M-QAM, as shown in Fig. 2.8. We denote the resulting algorithm as SR-1D.

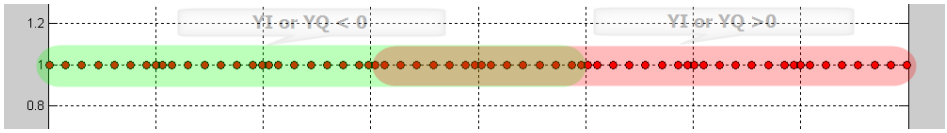


Figure 2.8 — The proposed sub-regions over fading erasure channel for rotated 64-QAM

The detection based on sub-region represents the key contribution of this work. This detection principle can be applied for a Max-Log demapping, both for two-dimensional and one-dimensional demapping. The selection of the sub-region over a fading channel is determined by the sign of the received symbol and contains $(1 + \log_2(M)/2)^2$ points. On the other hand, the sub-region selection over an erased channel is determined by the sign of the non-erased component of the received symbol and contains $\log_2(M) \cdot (1 + \log_2(M)/2)$ points. For the sake of clarity, during the comparison of the performance of different algorithms, all sub-region based detection is denoted as SR-2D.

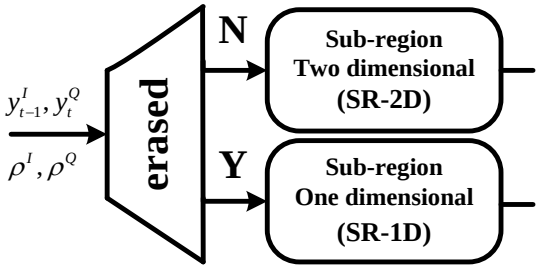


Figure 2.9 — The sub-region detection demapping algorithm for rotated QAM

2.2.3 Performance comparison

In this sub-section we provide simulation results for the un-coded QPSK, 16-QAM, 64-QAM and 256-QAM modulations both over fading and fading channel with 15% of erasures. Each figure contains at least four curves:

Exact 2D: the full complexity two-dimensional demapping algorithm detailed in equ.(2.1) for a rotated QAM.

Max-Log 2D: the Max-Log approximation of two-dimensional dempping algorithm detailed in equ.(2.14) for a rotated QAM.

Exact 1D: the full complexity one-dimensional demapping algorithm for non-rotated QAM detailed in equ.(2.3).

Max-Log SR 2D: the sub-region detection algorithm for rotated QAM as illustrated in Fig. 2.9.

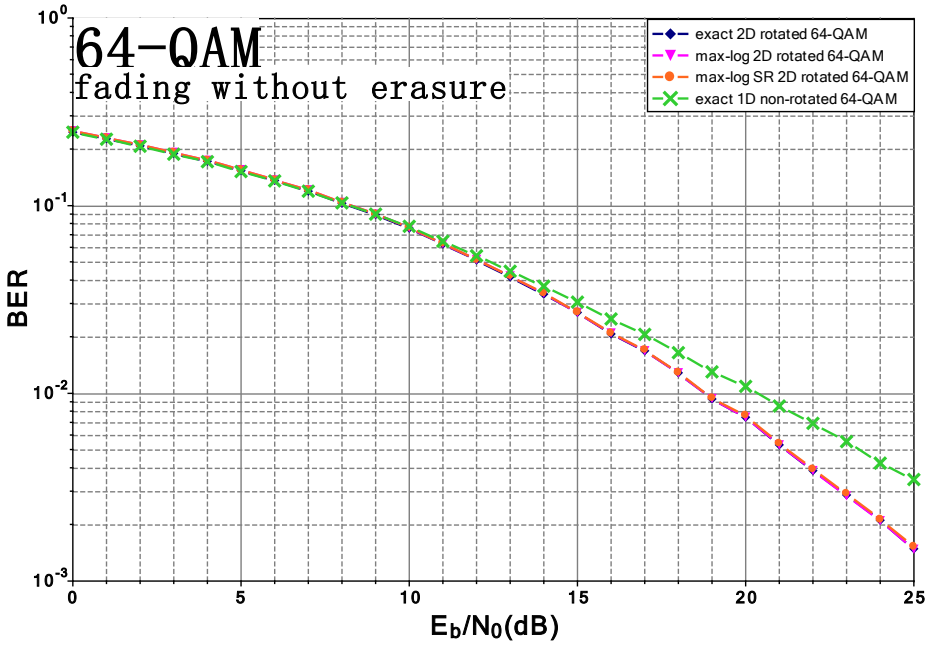


Figure 2.10 — BER comparison for 64-QAM uncoded over fading channel without erasure

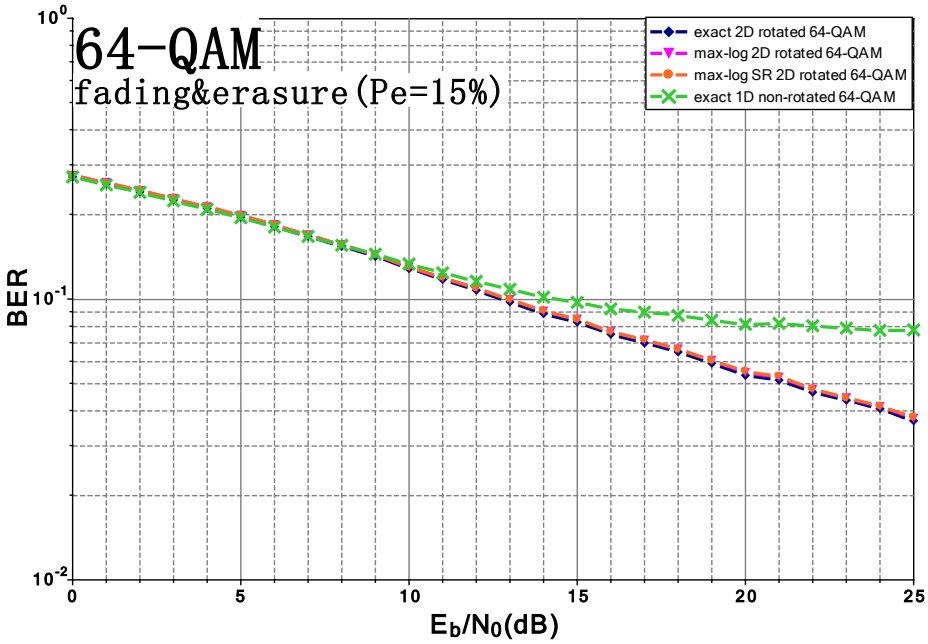


Figure 2.11 — BER comparison for 64-QAM uncoded over fading channel with 15% erasure

The simulation results show that the two-dimensional demapping algorithm with Max-Log approximation has almost no performance loss when compared to the full complexity two-dimensional demapping algorithm. Moreover, the two-dimensional demapping based on the sub-region detection with Max-Log approximation has no impact on performance compared to the whole region detection with Max-Log approximation both over a fading channel and a fading channel with erasures.

As a conclusion, the sub-region detection with a Max-Log approximation demapping algorithm is an appropriate method for hardware implementation. Indeed, this approach achieves almost the same performance as the un-simplified two-dimensional demapping algorithm for rotated QAM both over a fading and a fading erasure channel while greatly reduces computation complexity for high order modulations. This work is a notable contribution to the state-of-the-art demapping algorithm if a rotated constellation is considered.

2.3 Architecture of a flexible demapper for DVB-T2

The proposed demapping algorithm based on sub-region detection has shown good performance both over a fading and a fading channel with erasures and shows a good tradeoff between performance and hardware complexity. Therefore, we have designed and implemented the equivalent demapper.

The original demapping algorithm illustrated in equ. (2.1) can be rewritten as equ. (2.16), if the input of the demapper is replaced by the equalized symbol (s_{t-1}^I, s_t^Q).

$$LLR(\hat{v}_t^i) = log\left(\frac{\sum\limits_{x_t\in\chi_0^i}\frac{P(x_t)}{\sigma\sqrt{2\pi}}\cdot\exp\left(-\frac{\rho_{t-1}^2\cdot\left(s_{t-1}^I-x^I\right)^2+\rho_t^2\cdot\left(s_t^Q-x^Q\right)^2}{2\sigma^2}\right)}{\sum\limits_{x_t\in\chi_1^i}\frac{P(x_t)}{\sigma\sqrt{2\pi}}\cdot\exp\left(-\frac{\rho_{t-1}^2\cdot\left(s_{t-1}^I-x^I\right)^2+\rho_t^2\cdot\left(s_t^Q-x^Q\right)^2}{2\sigma^2}\right)}\right)$$

where $y_{t-1}^I = \rho_{t-1}^I s_{t-1}^I$, and $y_t^Q = \rho^Q s_t^Q$.

It can be further rewritten as:

$$LLR(\hat{v}_t^i) = log\left(\frac{\sum\limits_{x_t\in\chi_0^i}\frac{P(x_t)}{\sigma\sqrt{2\pi}}\exp\left(-\left(CSI^I\cdot(s_{t-1}^I-x^I)\right)^2-\left(CSI^Q\cdot(s_t^Q-x^Q)\right)^2\right)}{\sum\limits_{x_t\in\chi_1^i}\frac{P(x_t)}{\sigma\sqrt{2\pi}}\exp\left(-\left(CSI^I\cdot(s_{t-1}^I-x^I)\right)^2-\left(CSI^Q\cdot(s_t^Q-x^Q)\right)^2\right)}\right) \tag{2.16}$$

where CSI is denoted as

$$CSI = \frac{\rho}{\sigma} \tag{2.17}$$

σ denotes the standard variance of the channel noise.

2.3.1 Simplification of the Euclidean distance computation

Even though the sub-region detection method reduces the number of the squared Euclidean distances to be calculated, there are still $(1 + \log_2(M)/2)^2$ terms to be computed over fading channel. It means that $2 \cdot (1 + \log_2(M)/2)^2$ multiplications have to be performed. In the case of 256-QAM, 162 multiplications are needed. It definitely represents the bottleneck for the architecture design. The computation complexity can be further reduced by introducing a linear approximation for the computation of Euclidean distance.

In fact, the Euclidean distance between two points $P\left(p^I, p^Q\right)$ and $Q\left(q^I, q^Q\right)$ is defined as $\sqrt{(p^I - q^I)^2 + (p^Q - q^Q)^2}$. It can also be reformulated as $\sqrt{a^2 + b^2} = a \cdot \sqrt{1 + (b/a)^2}$, where $a = p^I - q^I$ and $b = p^Q - q^Q$. If a is chosen as the maximum of (a, b), then (b/a) is smaller than one. A linear approximation of the Euclidean distance is then performed as:

Algorithm 2 Simplification of the Euclidean distance computation

if $(\min(a,b) <= \max(a,b)/4)$ **then**

$F(a,b) = \max(a,b)$

else

$F(a,b) = \max(a,b) + (\min(a,b) - (\max(a,b)/4)/2)$

end if

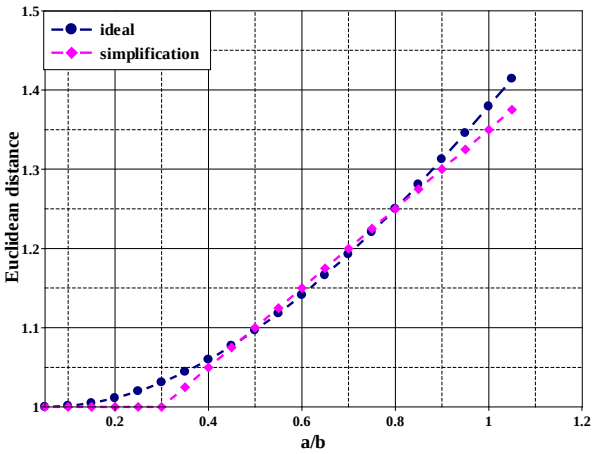


Figure 2.12 — Linear approximation of the Euclidean distance

Fig. 2.12 shows that the linear approximation has perfect estimation of the exact computation in floating point. Moreover, the difference between the approximated method and the

exact method is negligible in fixed point computation, with the quantization schedule as: 1 bit for the sign, 2 bits for the integral magnitude and 4 bits for the precision.

The simplified linear approximation of the Euclidean distances greatly reduces the computation needed in equ.(2.16). Then, the hardware oriented demapping algorithm with Max-Log approximation is then rewritten as equ. (2.18).

$$LLR(\hat{v}_t^i) = log\left(\frac{\min\limits_{x_t\in\chi_0^i}\left(\exp\left(-\left(\sqrt{\left(CSI^I\cdot(s_{t-1}^I-x^I)\right)^2+\left(CSI^Q\cdot(s_t^Q-x^Q)\right)^2}\right)^2\right)\right)}{\min\limits_{x_t\in\chi_1^i}\left(\exp\left(-\left(\sqrt{\left(CSI^I\cdot(s_{t-1}^I-x^I)\right)^2+\left(CSI^Q\cdot(s_t^Q-x^Q)\right)^2}\right)^2\right)\right)}\right)$$

$$= log\left(\frac{\min\limits_{x_t\in\chi_0^i}\left(\exp\left(-\left(Eucldiean_0\right)^2\right)\right)}{\min\limits_{x_t\in\chi_1^i}\left(\exp\left(-\left(Eucldiean_1\right)^2\right)\right)}\right) \tag{2.18}$$

(2.19)

Based on equ. (2.18), only $2\cdot(1+\log_2(M)/2)^2+2$ multiplications are necessary for the computation over a fading channel and only $2\cdot(1+\log_2(M)/2)\cdot\log_2(M)/2+2$ multiplications are needed for the computation over fading channel with erasures for M-QAM modulation. The corresponding values for considered constellation are listed in Table 2.2.

	The squared Euclidean distance		The number of the multiplication	
	Max-Log	Max-Log Sub-region	Max-Log	Max-Log Sub-region
64-QAM	64	25	256	52
256-QAM	256	81	1024	162

Table 2.2 — The gain of the simplification for high order modulations

The architecture for the Euclidean distance computation based on the proposed approximation is given in Fig. 2.13. Only two comparators, one adder, one subtracter and one multiplexer are allocated.

2.3.2 Architecture of a 2D demapper based on sub-region detection

The top level scheme of the architecture of the 2D demapper is given in Fig. 2.14. The architecture consists of couples of ROM tables, nine core modules, one network, eight comparison (get min) modules and one LLR computation module.

The ROM tables contain the values of the I and Q components for eight different QAM modulation modes. The sub-region detection is applied for 64-QAM and 256-QAM. In these

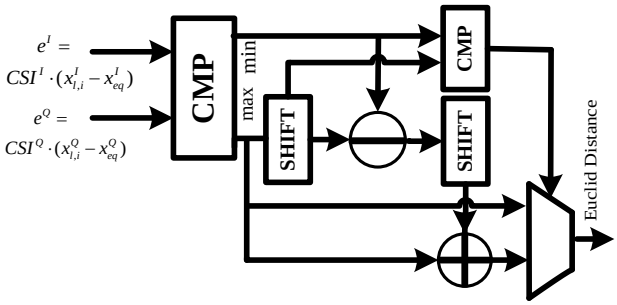


Figure 2.13 — Architecture of the Euclidean distance computation

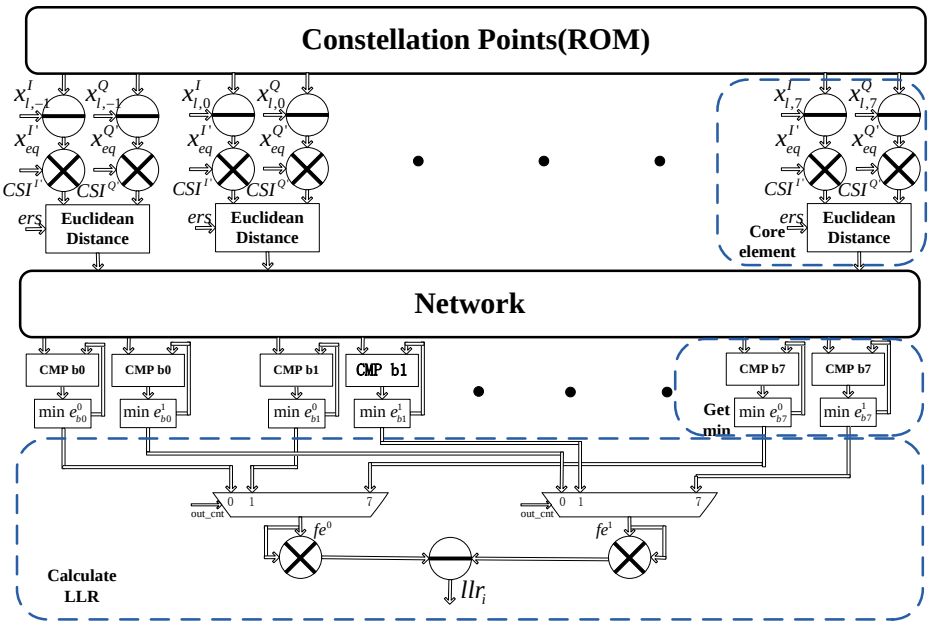


Figure 2.14 — Architecture of the flexible 2D demapper

two modes, only the values of signals in region I and II are memorized, as shown in Fig. 2.7, in the case of classical Rayleigh fading channel. The values in region III and IV can be obtained by a sign inversion of the stored values due to the symmetric property of the signals in the constellation. However, only the in-phase values of the signals in the union of regions I and II are stored in the case of a channel with erasures since the one dimensional demapping algorithm is applied on the remaining component as explained in Fig. 2.8. The in-phase values of the signals in the union of regions III and IV can also be achieved by a sign inversion of the stored values. If the non-erased component is quadrature component, then the corresponding values can be computed by a pre-processor illustrated in Fig. 2.15.

Nine core modules are assigned in the architecture. They enable a parallel computation of the Euclidean distances. In fact, the highest computation complexity configuration for the 2D-demapping is the rotated 256-QAM over a fading channel. In this case, the demapper has

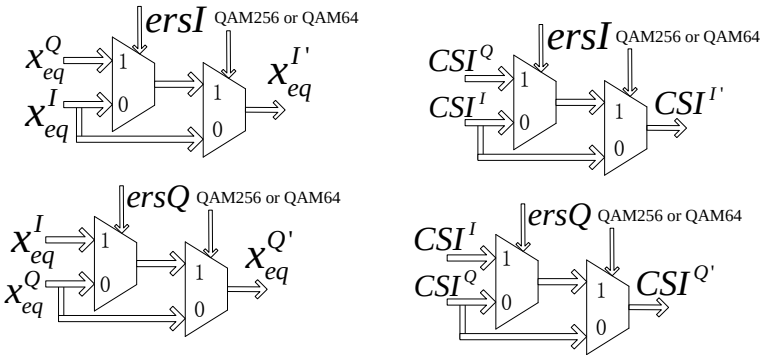


Figure 2.15 — Architecture of the pre-processor module for the demapper

to compute 81 Euclidean distances (9 by 9). The architecture computes 9 Euclidean distances from the received symbol to signal $(x^I_{l,q}, x^Q_{l,q})$ in parallel, where q denotes the column number $q \in [-1, 7]$ and l denotes the row number $l \in [0, 8]$. After 9 period times, 81 Euclidean distances are obtained. In other words, the ROM generates the values of the signals in the signal space panel row by row. In our point of view, this architectural solution offers a good tradeoff between throughput and complexity.

It is possible to re-use the 9 core elements for a fading channel with erasures. The demapper needs to compute $(log_2(M)/2) \cdot (log_2(M)/2 + 1)$ Euclidean distances if one-dimensional region algorithm is applied. However, in the case of an erased channel, every Euclidean distance depends only on one component of the received symbol (the non-erased one). In consequence, every core module can compute two one-dimensional Euclidean distances together and then selects and sends the minimum one to the network module. This process is obtained thanks to the structure of the core module that enables two computations of one-dimensional Euclidean distances in parallel. However, one pre-processor is needed to keep the input of $x^I_{l,q}$ and $x^Q_{l,q}$ equal to the non-erased value. Following this reasoning, all the Euclidean distances can be computed in 4 system clocks for 64-QAM and in 8 system clocks for 256-QAM respectively.

The network works according to the configuration of the previous modules and has two different kinds of function. One is the distribution of the Euclidean distances and the other one is the comparison of those to get the temporary minimum of each mapping bit of the received symbol at each period time.

The network is in charge of the distribution of the $(log_2(M)/2 + 1)$ Euclidean distances to M Get min modules belonging to M transmitted bits for an M-ary modulation. The generated $(log_2(M)/2 + 1)$ Euclidean distances are delivered to dedicated registers in every processing time. The delivery is determined by the value of the M bits of the $(log_2(M)/2+1)$ symbols in each row of the constellation. Let us take the rotated 64-QAM as an example

to explain this idea. If the received symbol has positive sign for both I and Q components, then region I is selected, as shown in Fig. 2.7. At the first processing time, 5 Euclidean distances $Ecd_{l,q}$ from the received symbol to 5 signals $\left(x_{l,q}^{I}, x_{l,q}^{Q}\right)$ in the first row of the signal space panel, where $q \in [-1, 3]$ and $l = 0$, are delivered to some registers. The registers are in the M get-min models according to 5 different transmitted bits. We first look at bit v^0, the transmitted value of the first bit of the 5 signals at the first processing time are $\{1, 0, 0, 0, 0\}$. Therefore, $Ecd_{0,-1}$, which is the Euclidean distance from the received symbol to symbol 10100$[bin]$ in the constellation, is delivered to register e_{b0}^{1}. e_{bi}^{j} denotes the temporary minimum distance for bit bi, whose value equals to $j \in [0, 1]$. At the same time, four distances $\{Ecd_{0,0}, Ecd_{0,1}, Ecd_{0,2}, Ecd_{0,3}\}$ which are the Euclidean distances from the received symbol to the four symbols $001000[bin], 001010[bin], 00010[bin], 00000[bin]$ are delivered to register e_{b0}^{0}. For bit v^1, the transmitted value of the second bit of the 5 signals are $\{0, 0, 0, 0, 0\}$. So none of the Euclidean distance is delivered to register e_{b1}^{1} and five Euclidean distances $\{Ecd_{0,-1}, Ecd_{0,0}, Ecd_{0,1}, Ecd_{0,2}, Ecd_{0,3}\}$ are all delivered to register e_{b1}^{0}. The same reasoning applies for the remaining 2 bits. The comparison is done for each register belonging to a Get min module. Only the minimum value in each group is registered as the temporary minimum value e_{bi}^{0} and e_{bi}^{1}. This network module is composed of 68 comparators.

Then eight Get min modules are still necessary to obtain the final minimum for the eight mapping bits with value one and value zero. $2 \cdot (log_2(M)/2+1)$ temporary values are obtained through the network module after $(log_2(M)/2 + 1)$ system clocks for a fading channel and $log_2(M)/2$ system clocks for a fading channel with erasures, respectively. Each of the Get min module consists of two comparators and two registers. The registers are in charge of the memorization the final minimum values e_{bi}^{0} and e_{bi}^{1}.

After all these steps, the LLRs are computed by two multiplexers, two multipliers and one subtractor.

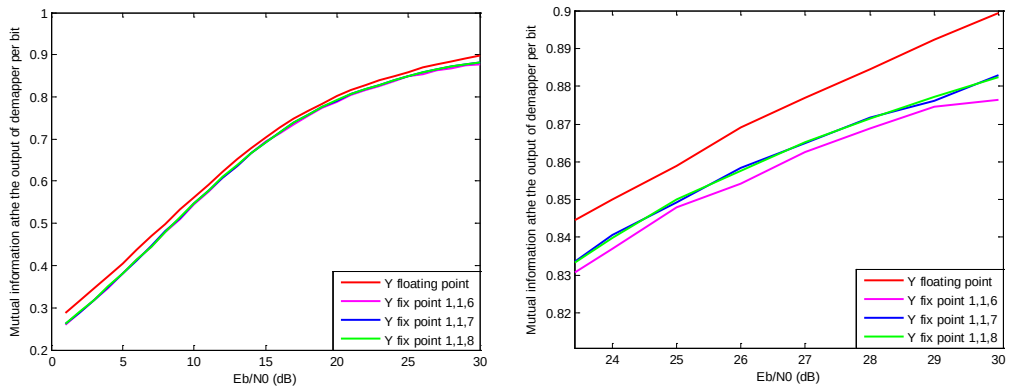
Only 20 multipliers are assigned to the proposed architecture. They are in charge of 1024 multiplications for the Max-Log demapping algorithm and 162 multiplications in the sub-region based Max-Log demapping algorithm for the most complicated 256-QAM case.

The proposed design not only enjoys low complexity but also has low latency. In fact, the delay of the output is only 14 system clocks for rotated 256-QAM, which guarantees in time feedback when an iterative modulation is applied.

2.3.3 Choice of the number of quantization bits

Fig. 2.16 and Fig. 2.17 give the theoretical guideline about how to choose the quantization bits for the input symbol (y^I, y^Q) and CSI. Mutual information per bit for different quantization schemes in the case of 256-QAM over a fading channel with 15% erasures are plotted. First, the

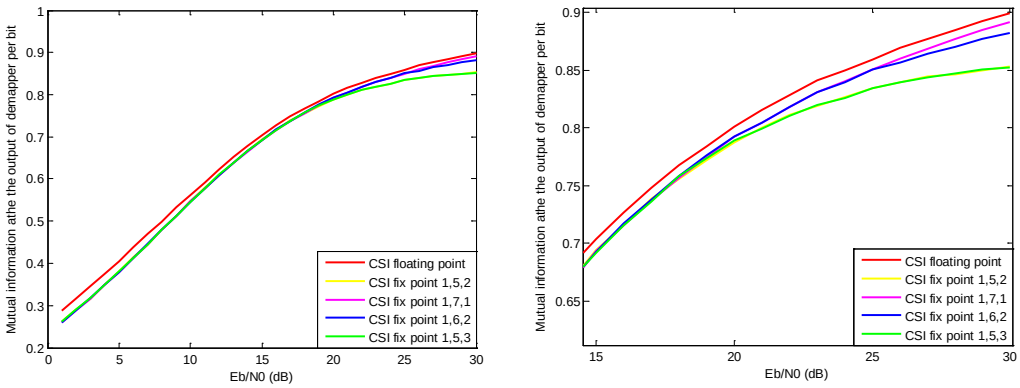
quantization bits assigned for CSI is fixed as $(1, 10, 4)$, which means 1 bit for the sign, 10 bits for the integer part and 4 bits for the fractional part. Then the mutual information per bits at the output of the demapper is calculated with different quantization schemes for (y^I, y^Q). From Fig. 2.16 we can observe that 9 bits quantization outperforms 8 bits quantization and the scheme $(1, 1, 7)$ and $(1, 1, 8)$ provide almost the same mutual information. Afterwards, the quantization bits for the input symbol (y^I, y^Q) is fixed at $(1, 1, 7)$ and the mutual information is calculated with different quantization schemes of CSI. From Fig. 2.17 we can observe that $(1, 7, 1)$ and $(1, 6, 2)$ are the two schemes show best performance.



(a) Mutual information per bit

(b) Mutual information between 0.8-0.9 per bit

Figure 2.16 — Mutual information per bit in function of the quantization of the received symbol in the fading channel with 15% erasures 256-QAM



(a) Mutual information per bit

(b) Mutual information between 0.6-0.9 per bit

Figure 2.17 — Mutual information per bit in function of the quantization of CSI in the fading channel with 15% erasures 256-QAM

Based on this guideline, quantization scheme $(1, 1, 7)$ is assigned for the input symbol

(y^I, y^Q) for all the constellation modes and $(1, 6, 2)$ is assigned for CSI in case of 256-QAM. However, the quantization scheme of CSI changes according to the constellation mode, $(1, 5, 3)$ is assigned for 64-QAM and $(1, 4, 4)$ is assigned for 16-QAM and QPSK.

2.3.4 Logic synthesis results

The flexible QAM demapper was synthesized and implemented onto a Virtex II Pro FPGA using Xilinx ISE tools. Computational resources of the demapper take up 791 slice Flip-Flops and 4,667 slice LUTs. The corresponding occupation rates are about 2% and 17% of a Xilinx XC2VP30 FPGA for slice registers and slice LUTs, respectively. In addition, multiplication resources for the demapper module take up 20 DSP blocks. It represents 14% of the total DSP blocks available in the chosen device. No block memory resource is used in our design because the ROM module was mapped in slice LUTs. The maximum clock frequency f_0 reaches 62 MHz and one input symbol lasts for 10 clocks. So an output data rate of 49.6 MLLR/s for 256-QAM is achieved at the input date rate of 6.2 Msymbol/s. Importantly, a reduction of the number of multipliers from 512 to 20 is achieved for 256-QAM.

2.3.5 BER performance

In order to validate the flexible QAM demapper, BER performance measures have been carried out. For this reason, a flexible mapper is designed, which supports eight different kinds of constellation modes in DVB-T2. In addition, a channel emulator was integrated. It supports a Rayleigh channel with and without erasures and that is based on Gaussian generators using Wallace method [39]. This channel emulator needs 4,907 slice Flip-Flops and 6321 slice LUTs. In addition, 59 DSP resources are necessary for multiplications and 13 Block RAMs are also assigned.

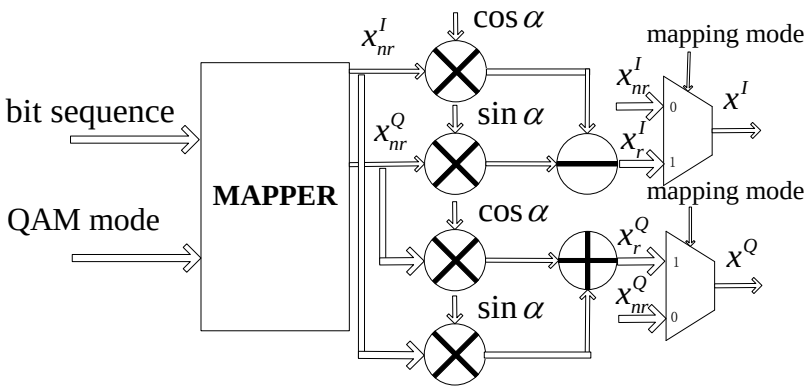


Figure 2.18 — Architecture of the flexible 2D Mapper

The experimental setup is carried out on a development board which contains a Xilinx

XC2VP30 device. Fig. 2.19 shows the different components of the experimental setup implemented onto the FPGA. A *Pseudo Random Generator* (PRG) sends out pseudo random data streams at each clock period f_0. This module is composed of flip-flops and XOR gates. The proposed flexible mapper generates the data streams. Then, the channel emulator generates Rayleigh fading samples with or without erasures and adds the noise onto the previous data streams. The SNR value and the percentage of the erasures is configurable by interface. The reception part is composed of two elements: an equalizer and the flexible demapper. The equalizer takes charge of calculation of the in-phase y^I_{equ} and quadrature y^Q_{equ} components of the received complex symbol, depends on the received information (y^I, y^Q), the Rayleigh distributed fading coefficients (ρ^I, ρ^Q) and the channel noise variance σ^2. A BER measurement is also implemented for a real time verification of the demapping performance. Computation resources of the complete system take up 12,115 slice Flip-Flops and 12,953 slice LUTs. Moreover, 83 DSP resources and 13 Block RAMs are also necessary.

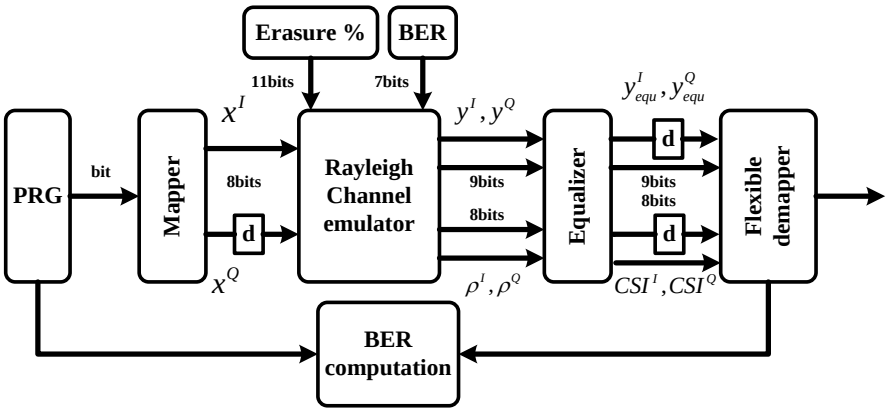


Figure 2.19 — Experimental setup for the prototype for an uncoded transmission chain

The comparisons of simulated floating point performance and measured performance in terms of BER of the flexible demapper for 64-QAM and 256-QAM are presented from Fig. 2.20 to Fig. 2.23 . The prototype shows quasi-identical performance when compared to simulation over a Rayleigh fading channel with and without 15% of erasures.

2.4 Conclusion

In this chapter, the study of demapping algorithm for the DVB-T2 standard is presented. It includes two main contributions. The first contribution is dividing the signal space into sub-region. The detection is then performed on a limited space, which enables an important decrease of complexity. The second contribution consists of a linear approximation for the computation of Euclidean distance. This simplification enables an important reduction in the

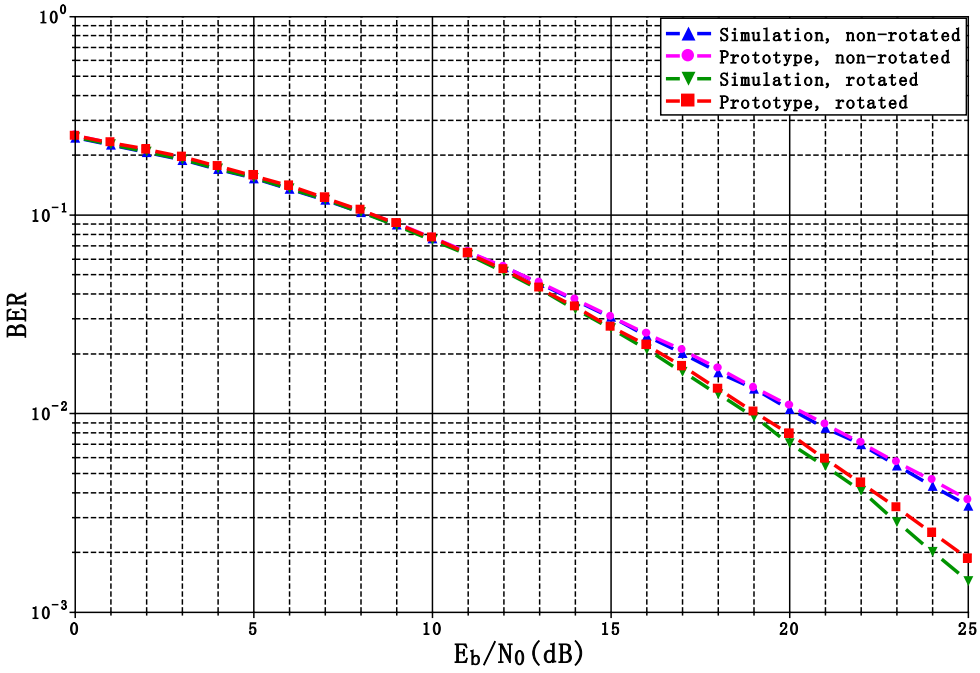


Figure 2.20 — Performance of uncoded system for 64-QAM over fading channel

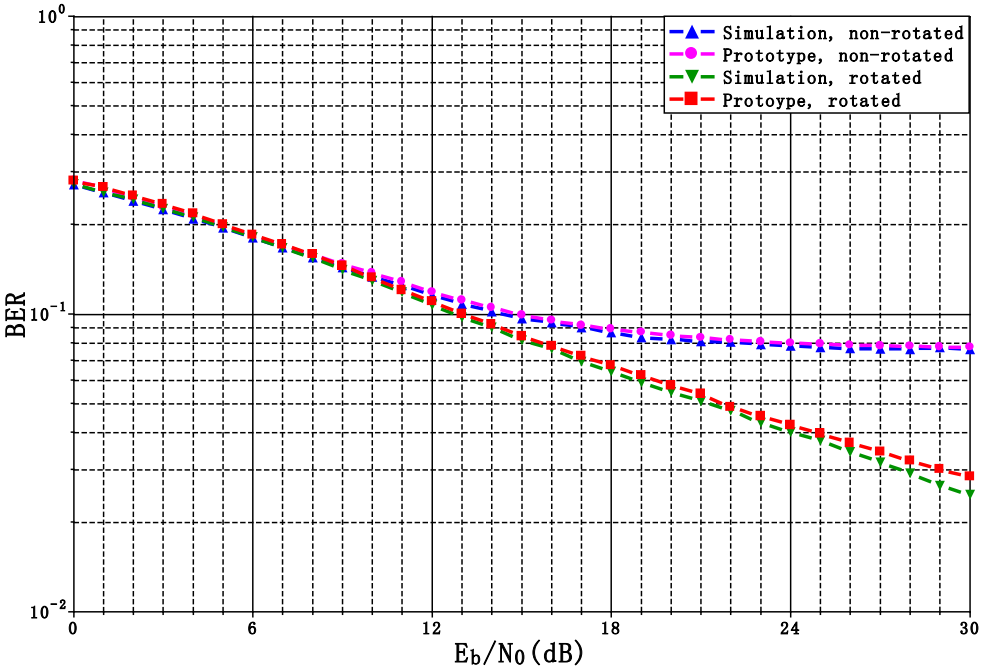


Figure 2.21 — Performance of uncoded system for 64-QAM over fading erasure (15%) channel

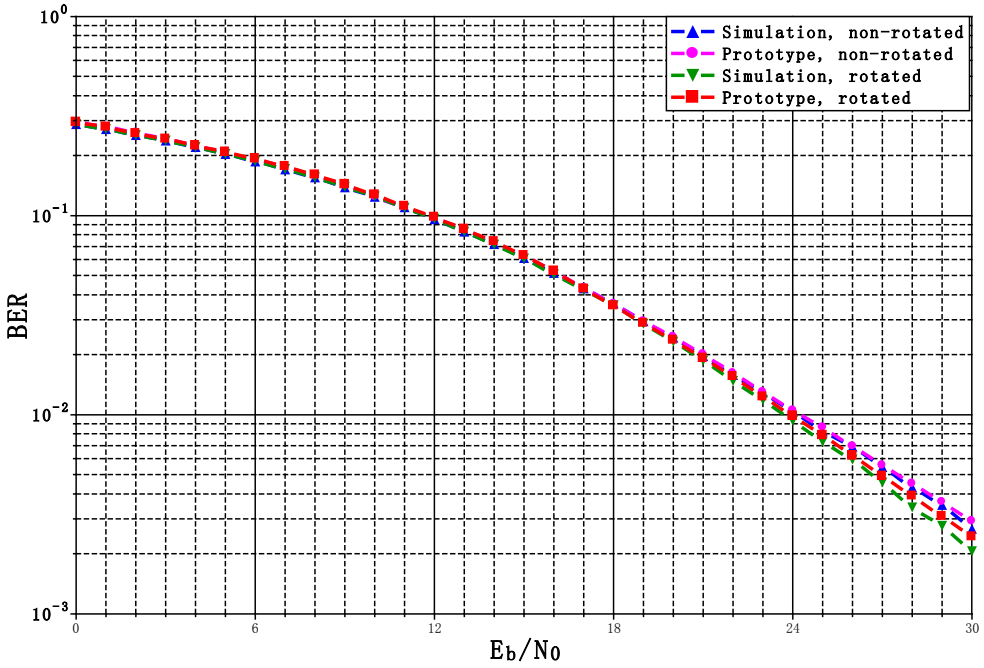


Figure 2.22 — Performance of uncoded system for 256-QAM over fading channel

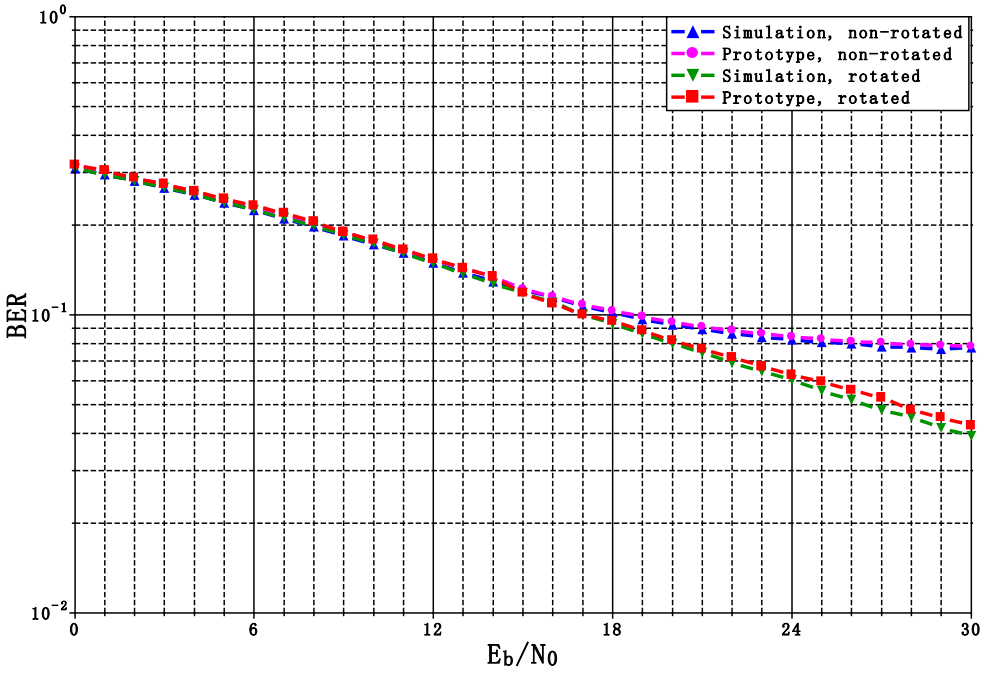


Figure 2.23 — Performance of uncoded system for 256-QAM over fading erasure (15%) channel

number of multiplication needed for demapping.

As a proof of principle, an uncoded platform has been designed and implemented on FPGA board. The measurement of the prototype over AWGN, fading channel with and without erasures demonstrates the flexibility and efficiency of the designed demapper for DVB-T2.

CHAPTER 3

Design and implementation of a vertical shuffled LDPC decoder

The aim of the thesis is to design an iterative BICM receiver for the DVB-T2 standard, with an efficient message exchange schedule between the notated demapper and the LDPC decoder. However, the state-of-art LDPC decoder using *horizontal shuffled* decoding algorithm prohibits an immediate message exchange between the demapper and decoder, since the *a posteriori* and *extrinsic* message are only accessible after a complete decoding iteration.

In this chapter, we propose a *vertical shuffled* Min-Sum LDPC decoding algorithm with the corresponding architecture. The *vertical shuffled* decoding algorithm can provide the *a posteriori* information of each bit node with a latency of only *dv* cycles. This enables an efficient exchange of *extrinsic* information between the rotated demapper and the LDPC decoder if an iterative receiver is considered.

Memory access conflicts due to double diagonal sub-matrix or pipeline is the major problem for the design of an efficient LDPC decoder for the DVB-T2 standard. Some solutions have been proposed for a *horizontal shuffled* decoder, however no previous work has addressed this kind of problem for a *vertical shuffled* decoder.

In the following chapter, we first give a brief introduction of different schedulings for

the decoding algorithms and the way to simplify the check node process. Then, we give a detailed description of the algorithm and the corresponding architecture for the proposed *vertical shuffled* Min-Sum LDPC decoder. Moreover, the way to solve the message updating conflicts by double diagonal sub-matrix and the memory access conflicts caused by pipeline architecture are also detailed. Afterwards, a prototype of a simplified transceiver is implemented onto an FPGA target. The proposed demapper and decoder are also integrated into a real DVB-T2 reference demodulator. Performance tests are carried out for these two systems to demonstrate the efficiency of the proposed LDPC decoder.

3.1 Background

The *Low Density Parity-Check* (LDPC) codes were first proposed by Gallager [2] in 1963. LDPC codes were re-discovered by MacKay [40] in 1996. Since the first LDPC code adopted by the DVB-S2 standard in 2003, LDPC codes found their way into many communication standards, such as DVB-T2, DVB-C2, DVB-NGH, WiMax, Wifi, etc.

The design of a low complexity and high throughput LDPC decoder has been widely studied since the beginning of the 21th century. The first LDPC decoder of a length 1024 with a code rate of R=1/2 LDPC code was published in 2002 [41] and the first LDPC decoder for DVB-S2 was published in 2005 [42]. However, both of them used the belief propagation or *Two Phase Message Passing* (TPMP) algorithm. This algorithm requires amount of memory and therefore leads to high hardware complexity. The messages from bit nodes to check nodes and from check nodes to bit nodes have both to be memorized. These messages are exchanged between bit nodes and check nodes along routed message wires, called network. This network quickly becomes the bottleneck of the hardware implementation and prohibits the design of a parallel decoder for long codes. The architecture-aware LDPC codes, such as *quasi-cyclic* (QC) LDPC codes, that are described by sparse parity-check matrices comprised of circulant sub-matrices, soon became the hot topic among the researches and industry field.

Hocevar [43] proposed a layered LDPC decoding algorithm in 2004. Then Mansour [21] gave a detailed explanation and proposed the corresponding architecture for architecture-aware LDPC codes. This layered decoding algorithm is also called as *Turbo Decoding Message Passing* (TDMP). The TDMP algorithm accelerates the convergence behavior of the standard TPMP by roughly a factor of two in terms of decoding iteration number and attains an order of magnitude improvement in *Bit Error Rate* (BER) at high *Signal-to-Noise Ratios* (SNR). It greatly reduces the required memory size and increases the convergence speed.

As a dual part of TDMP decoding, Fossorier [20] in 2005 proposed a vertical shuffled iterative decoding, in which the schedule of the message passing follows a column-wise processing other than row-wise. A comparison of state-of-the-art decoding schedules are carried out in [44], where the authors classified the decoding schedule into three different kinds: *folding schedule* , *horizontal shuffle schedule* (HSS) and *vertical shuffle schedule* (VSS).

3.2 Two phase message passing decoding algorithm

The most popular LDPC decoding algorithm is *Belief Propagation* (BP) algorithm which is introduced in Gallager's work [2]. This is also called *Message Passing* (MP) iterative algorithm, because during each iteration messages are passed from bit nodes to check nodes and then from check nodes back to bit nodes. The messages sent from bit nodes to check

nodes are computed based on the observed value of the bit nodes and the messages provided by the neighboring check nodes to the considered bit node.

Before describing the algorithm, we have to introduce some notations that will be used hereafter. Let $M(n)$ denote the set of check nodes connected to the bit node n, and let $N(m)$ denote the set of bit nodes that participated in the m^{th} parity-check equation. Furthermore, $N(m)\backslash n$ represents the set $N(m)$ excluding the variable n and $M(n)\backslash m$ represents the set $M(n)$ excluding the check node m, similarly.

Symbol $\oplus$ denotes modulo-2 summation and $\mathbb{L}(u)$ represents the log-likelihood ratio of a variable u in GF(2). For statistically independent random variable $u1$ and $u2$, the log-likelihood ratio of the sum of $u1$ and $u2$: $\mathbb{L}(u1 \oplus u2)$, is denoted as equ. (3.2).

$$\mathbb{L}(u) = \log\left(\frac{Pr(u=1)}{Pr(u=0)}\right) \tag{3.1}$$

$$\mathbb{L}(u1 \oplus u2) = \log\left(\frac{1 + e^{\mathbb{L}(u1)}e^{\mathbb{L}(u2)}}{e^{\mathbb{L}(u1)} + e^{\mathbb{L}(u2)}}\right) \tag{3.2}$$

Based on the derivation [45], we define the symbol $\boxplus$ as the notation of the addition defined by equ. (3.3).

$$\mathbb{L}(u_1) \boxplus \mathbb{L}(u_2) \triangleq \mathbb{L}(u_1 \oplus u_2) \tag{3.3}$$

The LLR summation $\boxplus$ follows some additional rules:

$$\mathbb{L}(u) \boxplus \infty = \mathbb{L}(u); \mathbb{L}(u) \boxplus -\infty = -\mathbb{L}(u); \mathbb{L}(u) \boxplus 0 = 0 \tag{3.4}$$

When the modulo-2 addition is extended to n variables, one can further prove that

$$\sum_{j=1}^{J} \boxplus \mathbb{L}(u_j) \triangleq \mathbb{L}\left(\sum_{j=1}^{J} \oplus u_j\right) \tag{3.5}$$

$$= log\left(\frac{\prod_{j=1}^{J}\left(e^{(\mathbb{L}(u_j))}+1\right) + \prod_{j=1}^{J}\left(e^{(\mathbb{L}(u_j))}-1\right)}{\prod_{j=1}^{J}\left(e^{(\mathbb{L}(u_j))}+1\right) - \prod_{j=1}^{J}\left(e^{(\mathbb{L}(u_j))}-1\right)}\right) \tag{3.6}$$

$\sum_{j=1}^{J} \boxplus \mathbb{L}(u_j)$ can be further rewritten as equ. (3.8) by using equ. (3.7).

$$\tanh\left(u/2\right) = \frac{e^u - 1}{e^u + 1} \tag{3.7}$$

$$\begin{aligned}\sum_{j=1}^{J} \boxplus \mathbb{L}\left(u_j\right) &= log\left(\frac{1 + \prod_{j=1}^{J} \tanh\left(\mathbb{L}(u_j)/2\right)}{1 - \prod_{j=1}^{J} \tanh\left(\mathbb{L}(u_j)/2\right)}\right) \\ &= 2tanh^{-1}\left(\prod_{j=1}^{J} \tanh\left(\mathbb{L}(u_j)/2\right)\right)\end{aligned} \tag{3.8}$$

Based on the decision rule described in equ. (3.9), where c_n denotes the transmitted bit with index n and y represents the received sequence, one of the decoding target is to compute the sign of equ. (3.10).

$$\begin{aligned}\log\left(\frac{Pr\left(c_n = 0|y\right)}{Pr\left(c_n = 1|y\right)}\right) \geqslant 0 \Longrightarrow c_n = 0; \\ \log\left(\frac{Pr\left(c_n = 0|y\right)}{Pr\left(c_n = 1|y\right)}\right) < 0 \Longrightarrow c_n = 1;\end{aligned} \tag{3.9}$$

$$\frac{Pr\left(c_n = 0|y\right)}{Pr\left(c_n = 1|y\right)} \tag{3.10}$$

The received codeword $y = (y_1, y_2, \cdots, y_n)$ can be split into two sets: y_n and $y_{n' \neq n}$. Then the log-likelihood ratio of equ. (3.10) is re-written as equ. (3.12) based on the derivation shown in equ. (3.11):

$$\begin{aligned}Pr\left(c_n|y\right) &= Pr\left(c_n|y_n, y_{n' \neq n}\right) \\ &= \frac{p\left(c_n, y_n, y_{n' \neq n}\right)}{p\left(y_n, y_{n' \neq n}\right)} \\ &= \frac{p\left(y_n|c_n, y_{n' \neq n}\right) \cdot p\left(c_n, y_{n' \neq n}\right)}{p\left(y_n|y_{n' \neq n}\right) \cdot p\left(y_{n' \neq n}\right)} \\ &= \frac{p\left(y_n|c_n\right) \cdot Pr\left(c_n|y_{n' \neq n}\right)}{p\left(y_n|y_{n' \neq n}\right)}\end{aligned} \tag{3.11}$$

$$\underbrace{log\left(\frac{Pr\left(c_n = 0|y\right)}{Pr\left(c_n = 1|y\right)}\right)}_{T_n} = \underbrace{log\left(\frac{\left(y_n|c_n = 0\right)}{\left(y_n|c_n = 1\right)}\right)}_{llr_n} + \underbrace{log\left(\frac{c_n = 0|y_{n' \neq n}}{c_n = 1|y_{n' \neq n}}\right)}_{E_n} \tag{3.12}$$

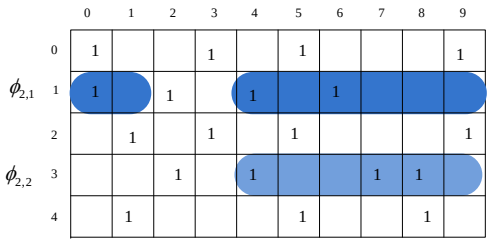


Figure 3.1 — One example of $\phi_{n,k}$ and $\phi_{n,k,l}$

Let llr_n denote the *intrinsic* channel reliability value of the bit node n, based on the channel observation of related symbol y_n. T_n denotes *a posteriori* log-likelihood ratio of variable-node n during each iteration. The sign of T_n enables to make a hard decision on c_n and the absolute value $|T_n|$ represents the reliability of the decision. E_n is the *extrinsic* information of the bit c_n. E_{mn} denotes the information sent from check node m to variable-node n. T_{mn} denotes the information sent from bit node n to check node m.

Let $\phi_{n,k}$ denote the k^{th} parity check constraint (syndrome equals to zero) of the bits in $M(n)$ without the bit c_n, $k \in \{1, 2, \cdots, |M(n)|\}$. Fig. 3.1 illustrates the bit node set of $\phi_{2,1}$ and $\phi_{2,2}$ in one parity check matrix. The probability that $c_n = 1$ is the probability that all the syndromes related to bit n are equal to zero with $c_n = 1$. It means the parity check equations are satisfied.

$$Pr\left(c_n = 1 | y_{n' \neq n}\right) = Pr\left(\phi_{n,1} = 1, \cdots, \phi_{n,|M(n)|} = 1 | y_{n' \neq n}\right) \tag{3.13}$$

In a cycle free Tanner Graph, the events that $\phi_{n,|M(n)|} = 1$ are independent in the case of $y_{n' \neq n}$. So E_n is expressed as:

$$\begin{aligned} E_n &= log\left(\frac{\prod_{k=1}^{|M(n)|} Pr\left(\phi_{n,k} = 0 | y_{n' \neq n}\right)}{\prod_{k=1}^{|M(n)|} Pr\left(\phi_{n,k} = 1 | y_{n' \neq n}\right)}\right) \\ &= \sum_{k=1}^{|M(n)|} log\left(\frac{Pr\left(\phi_{n,k} = 0 | y_{n' \neq n}\right)}{Pr\left(\phi_{n,k} = 1 | y_{n' \neq n}\right)}\right) \end{aligned} \tag{3.14}$$

Consequently, E_n is the summation of $E_{k,n}$, which is defined in equ. (3.15). $E_{k,n}$ is the information given by each of the parity check constraints belonging to the k^{th} row of the parity check matrix.

Furthermore, $c_{n,k,l}$ means the l^{th} bit in the parity check constraint of $\phi_{n,k}$, which belongs to any bit in $N(m)$ excluding n. In Fig. 3.1, $\phi_{2,1,1} = \{v_4 \oplus v_6\}$, $\phi_{2,1,2} = \{v_0 \oplus v_6\}$ and $\phi_{2,1,3} = \{v_0 \oplus v_4\}$. With this notation and equ. (3.8), $E_{k,n}$ is derived as:

$$\begin{aligned} E_{k,n} &= log\left(\frac{Pr\left(\phi_{n,k}=0|y_{n'\neq n}\right)}{Pr\left(\phi_{n,k}=1|y_{n'\neq n}\right)}\right) \\ &= 2\tanh^{-1}\prod_{l=1}^{|\phi_{n,k}|}\tanh\frac{1}{2}log\left(\frac{Pr(c_{n,k,l=0}|y_{n'\neq n})}{Pr(c_{n,k,l=1}|y_{n'\neq n})}\right) \end{aligned} \tag{3.15}$$

Let $log\left(\frac{Pr(c_{n,k,l=0}|y_{n'\neq n})}{Pr(c_{n,k,l=1}|y_{n'\neq n})}\right)$ denote $T_{k,l}$, which means the information sent by bit node l to its connected check node k. Then $E_{k,n}$ is rewritten as:

$$E_{k,n} = 2\tanh^{-1}\prod_{l\in N(k)\backslash n}\tanh\frac{T_{k,l}}{2} \tag{3.16}$$

where $T_{k,l}$ denotes the messages sent from bit nodes to check nodes.

$$T_{k,l} = T_k - E_{k,l} \tag{3.17}$$

Finally, the *a posteriori* information of bit c_n can be rewritten as:

$$T_n = I_n + \sum_{k=1}^{|M(n)|} E_{k,n} \tag{3.18}$$

$E_{k,n}$ is called the message sent from check nodes to bit nodes.

The *Two Phase Belief Propagation* decoding algorithm is described in Algorithm 3. It is the classical message passing algorithm, in which each iteration consists of two phase computations. The message update of all bit nodes are done during phase 1 and the message is then sent to neighboring check nodes. Then the update of all check nodes is done during phase 2 and then the message is sent to neighboring bit nodes. An important aspect is that the message sent from a bit node v to a check node c may not take into account the message sent during the previous iteration from c to v. The same principle is also applied for the message passed from check nodes to bit nodes. The check node processor is a *Soft Input Soft Out* (SISO) core. This latter uses the *a priori* information to generate the *extrinsic* information by using equ. (3.14). The bit node processor is also a SISO core that uses the *extrinsic* information to provide the *a posteriori* and the *a priori* information.

Algorithm 3 Two phase belief propagation decoding algorithm

Initialization:

$E_{mn}^{(0)} = 0$

Decoding:

for $t = 1, .., t_{max}$ {***iteration*** }

  for all Check nodes

    $E_{mn}^{(t)} = 2 \tanh^{-1} \prod_{k \in N(m) \backslash n} \tanh \frac{T_{mk}^{(t-1)}}{2}$

  for all Bit nodes

    $T_n^{(t)} = llr_n + \sum_{k \in M(n)} E_{kn}^{(t-1)}$

    $T_{mn}^{(t)} = T_n^{(t)} - E_{mn}^{(t-1)}$

Hard decision according to $T_n^{(t)}$

3.3 Check node process simplification

The most complex computation in the LLR based decoding algorithm is the check node process, in which the non-linear function $\tanh(x)$ is not convenient for a hardware implementation. Indeed, the quantization for the sharp non-linear function and the infinite value is particularly critical and costly in terms of hardware implementation. In this subsection, we give a detailed introduction of check node process algorithms and the corresponding simplification methods.

3.3.1 Check node process based on Gallager's approach

The original check node update computation equ. (3.19) based on the $\tanh(x)$ function [2] can be rewritten as equ. (3.20). The computation is split into the sign part and the amplitude part. The separation process of the sign and the amplitude is applied to all the following check node processing algorithms.

$$E_{mn}^{(t)} = 2 \tanh^{-1} \prod_{k \in N(m) \backslash n} \tanh \frac{T_{mk}^{(t-1)}}{2} \tag{3.19}$$

$$E_{mn}^{(t)} = \left(\prod_{k \in N(m) \backslash n} sign \left(T_{mk}^{(t-1)} \right) \right) \cdot 2 \tanh^{-1} \left(\prod_{k \in N(m) \backslash n} \tanh \frac{|T_{mk}^{(t-1)}|}{2} \right) \tag{3.20}$$

Following Gallager's proposal, the check node process can be rewritten as equ. (3.21) by

using function $\varphi(x)$ in equ. (3.23).

$$E_{k,n} = \varphi^{-1}\left(\sum_{l\in N(k)\backslash n} \varphi\left(T_{mk}\right)\right) \tag{3.21}$$

$$= \left(\prod_{k\in N(m)\backslash n} sign\left(T_{mk}^{(t-1)}\right)\right) \cdot \varphi^{-1}\left(\sum_{l\in N(k)\backslash n} \varphi\left(|T_{mk}|\right)\right) \tag{3.22}$$

$$\varphi(x) = ln\left(\frac{e^x+1}{e^x-1}\right) \tag{3.23}$$

The function $\varphi(x)$ is illustrated in Fig. 3.2 and has the properties of:

$$\varphi(\varphi(x)) = x \quad and \quad \varphi(x)^{-1} = \varphi(x) \tag{3.24}$$

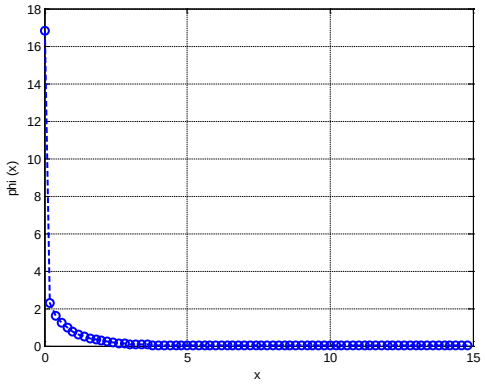


Figure 3.2 — The $\varphi(x)$ function

The $\varphi(x)$ function can be elegantly implemented by software with floating point (very high precision). However, the quantization precision becomes an important constraint for hardware implementation, which leads to a very high implementation cost.

3.3.2 Check node process based on Jacobian logarithm

The $\varphi(x)$ function can be further rewritten as equ. (3.25) by using the Jacobian logarithm [46]:

$$\begin{aligned}\mathbb{L}\left(u \oplus v\right) = {} & sign(\mathbb{L}(u))sign(\mathbb{L}(v))\min(|\mathbb{L}(u)|, |\mathbb{L}(v)|) \\ & + \log\left(1 + e^{-|\mathbb{L}(u)+\mathbb{L}(v)|}\right) - \log\left(1 + e^{-|\mathbb{L}(u)-\mathbb{L}(v)|}\right)\end{aligned} \tag{3.25}$$

with a two time usage of function $g(x)$, where $g(x) = \log\left(1 + e^{-x}\right)$ is shown in Fig. 3.3.

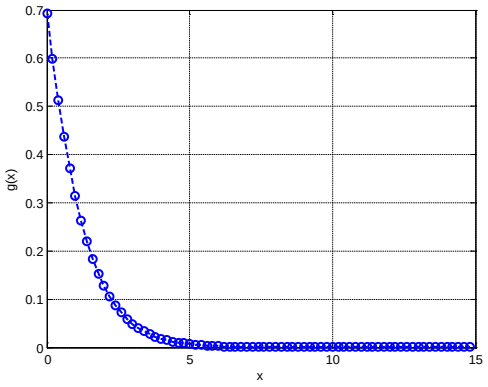


Figure 3.3 — The $g(x) = \log\left(1 + e^{-x}\right)$ function

Assuming that the check node m with dc degree has the connection to bit nodes $N(m) = (n_1, n_2, \cdots, n_{dc})$, that are denoted as $b_1, b_2, \cdots, b_{dc}$. We induce two kinds of temporary variables: forward temporary variable f_i and backward temporary variable b_i.

$$f_1 = b_1, \quad f_2 = b_2 \oplus f_1, \quad f_3 = b_3 \oplus f_2, \quad \cdots, \quad f_{dc} = b_{dc} \oplus f_{dc-1}$$

$$d_{bc} = b_{dc}, \quad d_{dc-1} = d_{bc} \oplus b_{dc-1}, \quad \cdots \quad , d_1 = d_2 \oplus b_1$$

where the operation $\oplus$ denotes the modulo-2 summation.

All the bit nodes should meet the parity check constraint:

$$b_1 \oplus b_2 \oplus \cdots \oplus d_{dc} = 0$$

Therefore, any bit node b_x can be computed by:

$$b_x = f_{x-1} \oplus d_{x+1}$$

Based on this idea, $\mathbb{L}(f_1), \mathbb{L}(f_2), \cdots, \mathbb{L}(f_{dc})$ and $\mathbb{L}(d_{dc}), \mathbb{L}(d_{dc-1}), \cdots, \mathbb{L}(d_1)$ can be calculated in a recursive manner. The computation of the messages from check node m to dc bit nodes can be achieved by applying equ. (3.26) based on equ. (3.25).

$$E_{mn} = \begin{cases} \mathbb{L}(d_2) & \text{i=1,} \\ \mathbb{L}(f_{i-1} \oplus d_{i+1}) & \text{i=2,3,...,dc-1} \\ \mathbb{L}(f_{dc-1}) & \text{i=dc.} \end{cases} \tag{3.26}$$

The approach based on Jacobian logarithm is more appealing for a hardware implementation when compared to Gallager's approach. In fact, the function $g(x) = \log\left(1 + e^{-x}\right)$ is easier to be quantized by using a lookup table or/and a piecewise linear approximation. Indeed, the slope of $g(x)$ (see Fig. 3.3) is not steep and its maximum value remains below 1. This is an

undeniable advantage when compared to the $\varphi(x)$ function (see Fig. 3.2) since $\varphi(x) \to \propto$ when $x \to 0$. In addition, the computation of *extrinsic* messages is based on the backward-forward algorithm, that is already widely used for turbo decoding. This algorithm requires $3(dc-2)$ computations of the log likelihood ratio for a modulo-2 variable, while Gallager's approach requires $2dc$ computations. For check node degree $dc \geq 3$, the backward-forward algorithm reveals to be less complex.

3.3.3 Check node process based on normalized Min-Sum

The former two check node processing algorithms are based on the full complexity belief propagation algorithm. An additional simplification is possible if the following approximation is applied: the magnitude of $\mathbb{L}(U \oplus V)$ is less than or equal to the minimum of the magnitude of $\mathbb{L}(U)$ or $\mathbb{L}(V)$.

$$\mathbb{L}(U \oplus V) \leq \min\left(\mathbb{L}(U), \mathbb{L}(V)\right) \tag{3.27}$$

Based on this lemma, equ. (3.20) is then simplified as:

$$E_{mn}^{(t)} = \left(\prod_{k \in N(m) \backslash n} sign\left(T_{mk}^{(t-1)}\right) \right) \times \left(\eta \cdot \min_{k \in N(m) \backslash n} \left(|T_{mk}^{(t-1)}| \right) \right) \tag{3.28}$$

where η is a normalization factor ($\eta \leq 1$), which tries to minimize the effect of the approximation in equ. (3.27). Note good BER performance for an LDPC decoder is achieved with an adaptive η, that varies with SNRs and iterations. In practice, this η value is set to be constant around value 0.8.

The check node processing determines the two minimum amplitude value of T_{mn} as well as the product of the sign of T_{mn}. For BP based check node processing algorithm, *dc extrinsic* messages of the signed value have to be memorized. Whereas for a normalized Min-Sum check node processing algorithm only the sign of the *dc extrinsic* messages, the minimum and the second minimum plus the index of the minimum have to be memorized [47].

3.3.4 Check node process based on offset Min-Sum

An additional simplification based on Min-Sum approximation called offset Min-Sum is presented in [19]. The corresponding check node processing is illustrated as:

$$E_{mn}^{(t)} = \left(\prod_{k \in N(m) \backslash n} sign\left(T_{mk}^{(t-1)}\right) \right) \times \max\left\{ \min_{k \in N(m) \backslash n} |T_{mk}^{(t-1)}| - \varepsilon, 0 \right\} \tag{3.29}$$

where ε is a factor ($\varepsilon \leq 1$). In practice, this value is kept constant and set to 0.15. The major difference to the normalized Min-Sum is the total removal of some *extrinsic* messages with low probability. In fact, the ones with a magnitude lower than ε are set to zero. Therefore these messages will not be taken into account for the next iteration. When the minimum value or the second minimum value are selected, the offset Min-Sum uses a comparator instead of the multiplication with a normalized factor as it is the case for the normalized Min-Sum. This operation takes up less resources and achieves higher working frequency that is necessary for a higher throughput.

Normalized and offset Min-Sum greatly reduce check node processing complexity. However, this advantage comes at the price of a performance penalty. In [19], a detailed performance comparison between the Min-Sum based algorithms and BP based algorithms is performed. It has shown that the performance penalty increases when lowing the coding rate.

3.3.5 Check node process based on lambda-Min-Sum

The λ-Min-Sum was first proposed in [48] to offer a good trade-off between hardware complexity and BER performance, especially for low code rates. The performance of λ-Min-Sum with an offset can get very close to the Jacobian-based Belief Propagation algorithm for the regular (5,10) short (N=816) LDPC code proposed by MacKay [49] and the irregular LDPC code ($N = 2000$) with a code rate of R=0.85 proposed by Urbanke [50].

In the approach of the Jacobian logarithm (equ. (3.25)) the two terms of correlations are negligible in the case of $U \gg V$.

$$\mathbb{L}\left(u \oplus v\right) = sign(\mathbb{L}(u))sign(\mathbb{L}(v))|\mathbb{L}(v)| \qquad if U \gg V \tag{3.30}$$

Therefore, the computation of the check node process can be simplified by limiting the number of magnitude computations of $\mathbb{L}\left(u \oplus v\right)$. The corresponding λ-Min-Sum algorithm is illustrated in equ. (3.31). $N_\lambda(m)$ represents the set having the minimum λ magnitudes of $|T_{mk}^{(t-1)}|$. The check node process differs in two cases. If the bit node n belongs to the set $N_\lambda(m)$, then the amplitude of $E_{mn}^{(t)}$ is processed over $\lambda - 1$ minimum values of $|T_{mk}^{(t-1)}|$. If the bit node n does not belong to the set $N_\lambda(m)$, then the amplitude of $E_{mn}^{(t)}$ is processed over λ minimum values.

$$E_{mn}^{(t)} = \left(\prod_{k \in N(m) \backslash n} sign\left(T_{mk}^{(t-1)}\right) \right) \cdot \left(\sum_{k \in N_\lambda(m) \backslash n} \oplus \mathbb{L}(|T_{mk}^{(t-1)}|) \right) \tag{3.31}$$

In practice, the check node process carries out the comparison of the amplitude of each coming $|T_{mk}^{(t-1)}|$, the minimum λ values and the corresponding indexes of the addresses are registered. Afterwards, at each time period, two of these values or the temporary results are selected to compute equ. (3.25). However, the process of the sign is based on all the coming $T_{mk}^{(t-1)}$ excluding the value connected to bit node n, similar to normalized Min-Sum. In the case of a regular LDPC code, with a check node degree equalling to 20, the required memory for *extrinsic* messages for a λ-Min-sum decoder(λ=2), is only 30% of the required memory in a corresponding belief propagation based decoder.

3.3.6 Summary

Table 3.1 summarizes the properties of the different check node process methods mentioned above. Two kinds of belief propagation methods have almost the same properties in the listed content in Table 3.1. However, the Jacobian based belief propagation is more suitable for a hardware implementation. Normalized Min-Sum and offset Min-Sum requires the least amount of memories at the price of a performance penalty especially for low code rates. λ-Min-Sum represents a good tradeoff between performance and complexity.

	BP Gallager	BP Jacobien	λ-Min-Sum	Normalized Min-Sum	Offset Min-Sum
Performance	+ + ++	+ + ++	+ + +	++	+
Hardware Complexity	+	++	+ + +	+ + ++	+ + + + +
CSI sensitive	Yes	Yes	Yes	No	No
Memory for check node process	Wdc + dc	Wdc + dc	$W(\lambda+1)$+ $\lambda log_2(dc)$ + dc	2W + dc	2W + dc

Table 3.1 — A comparison of the check node process methods

Indeed, sensitivity to the estimation of channel information is very important for any non-linear function, such as $\varphi(x)$ and $g(x)$. A poor estimation could lead to a severe penalty. However, unlike BP-based algorithm, the Min-Sum based algorithm described above does not require a perfect estimation of the channel.

For different LDPC codes and different applications, requirements and constraints differ. The designer has to choose a well adapted decoding method based on these constraints.

3.4 Horizontal shuffled decoding algorithm

If a horizontal shuffled decoding schedule is considered, the parity check matrix is viewed as horizontal layers. The horizontal shuffled decoding schedule is performed by applying the decoding algorithm to each successive layer. The decoding algorithm for one particular layer

uses the result from the previous decoded layers. The horizontal shuffled decoding schedule differs from the two phase message passing schedule, in which all check node processing of one layer are processed and the corresponding bit node processing is calculated immediately.

One iteration in horizontal shuffled decoding algorithm is composed of $P_{ldpc} = N_{ldpc} - K_{ldpc}$ sub-iterations, if a serial decoding is considered here. First, the *a posteriori* messages T_n are initialized with the intrinsic channel reliability values. The *extrinsic* messages of the check nodes processors E_{mn} are initialized to zero. Each sub-iteration involves two steps: check node process and bit node addition. The *a priori* messages are generated by subtracting the *extrinsic* messages $E_{mk}^{(t-1)}$ of the previous iteration from the *a posteriori* messages T_n. The outputs $E_{mk}^{(t)}$ are stored for the next iteration and the *a posteriori* messages T_n are updated by adding the current *extrinsic* messages $E_{mk}^{(t)}$ to the *a priori* messages $T_{mn}^{(t-1)}$. Both $E_{mk}^{(t)}$ and T_n participate in the next sub-iteration. The latest updated T_n can provide the latest information to the following sub-iterations. In contrast to the HSS schedule, the updated T_n in the TPMP algorithm is sent to check node processors only after all the bit node processing is finished. Consequently, the horizontal shuffled schedule doubles the decoding convergence speed.

Algorithm 4 Horizontal shuffled belief propagation algorithm

Initialization:

$T_n^{(0)} = llr_n$

$E_{mn}^{(0)} = 0$

Decoding:

for $t = 1, .., t_{max}$ {***iteration*** }

  for $m = 1, .., P_{ldpc}$ { ***sub-iteration***}

    Check node processing

$$E_{mn}^{(t)} = \prod_{k \in N(m) \backslash n} sign\left(T_n^{(t-1)} - E_{mk}^{(t-1)}\right) \cdot \varphi^{-1}\left\{\sum_{k \in N(m) \backslash n} \left(T_n^{(t-1)} - E_{mk}^{(t-1)}\right)\right\},\ m \in M(n)$$

    Bit node processing

$$T_n^{(t)} = T_n^{(t-1)} - E_{mn}^{(t-1)} + E_{mn}^{(t)}\ ,\ n \in N(m)$$

Hard decision according to $T_n^{(t)}$

3.4.1 Horizontal shuffled normalized Min-Sum decoding algorithm

In a horizontal shuffled normalized Min-Sum decoding algorithm, shown in Algorithm 5, the check node process first makes a comparison between dc incoming *a priori* messages $T_{mn}^{(t-1)} = T_n^{(t-1)} - E_{mn}^{(t-1)}$ for each layer. Then the value of the minimum plus the corresponding index and the second minimum value are memorized. Moreover, the magnitude of the *extrinsic* message

is achieved by applying the normalization to the selected minimum or second minimum values, as illustrated from line 10 to line 13 in Algorithm 5. The split of the magnitude and sign processes of the *extrinsic* messages greatly reduce the required memory size for *extrinsic* messages. In a classical belief propagation based decoding algorithm, all the $E_{mn}^{(t-1)}$ messages have to be memorized. The number of *extrinsic* messages are equal to the number of elements in the parity check matrix. However, in the Min-Sum based decoding, only the sign of all the $E_{mn}^{(t-1)}$ messages plus the magnitude of M_m^0, M_m^1 and P_m^0 for each check node have to be memorized. P_m^0 is the index of k^{th} bit node of M_m^0 for the check node m. A benefit for any LDPC code with check node degree larger than 3 can be observed. The memory reduction ratio increases when the check node degree increases.

The horizontal Min-Sum algorithm enjoys low complexity and fast convergence speed, therefore it is widely used.

Algorithm 5 Horizontal shuffled normalized Min-Sum algorithm

1: **Initialization:**
2: $T_n^{(0)} = llr_n$
3: $E_{mn}^{(0)} = 0$
4: **Decoding:**
5: **for** $t = 1, .., t_{max}$ {***iteration*** }
6: **for** $m = 1, .., P_{ldpc}$ { ***sub-iteration***}
7: **Check node processing**
8: $T_{mn}^{(t-1)} = T_n^{(t-1)} - E_{mn}^{(t-1)}$
9: $M_m^0 = min(|T_{mn}^{(t-1)}|), M_m^1 = secmin(|T_{mn}^{(t-1)}|)$
10: **if** $(|T_{mn}^{(t-1)}| = M_m^0)$ **then**
11: $E_{mn}^{(t)} = \prod_{k \in N(m) \backslash n} \cdot sign(T_{mk}^{(t-1)}) \eta \cdot M_m^1$
12: **else**
13: $E_{mn}^{(t)} = \prod_{k \in N(m) \backslash n} \cdot sign(T_{mk}^{(t-1)}) \eta \cdot M_m^1$
14: **Bit node processing**
15: $T_n^{(t)} = T_n^{(t-1)} - E_{mn}^{(t-1)} + E_{mn}^{(t)}$, $n \in N(m)$
16: **Hard decision according to** $T_n^{(t)}$

3.5 Vertical shuffled decoding algorithm

In order to obtain the objective of our work: design an efficient iterative receiver for DVB-T2 with low complexity and high throughput, a vertical shuffled LDPC decoding is mandatory to increase the throughput. Similarly to the horizontal shuffled schedule, the main idea of

the vertical shuffled schedule is to update the check node information as soon as possible. In difference from horizontal shuffled schedule, the vertical shuffled decoding algorithm operates on the bit nodes. Each iteration is composed of n_{ldpc} sub-iteration, if serial decoding is considered. Each sub-iteration uses the newest T_{mn}, which have been updated during the previous sub-iteration to update the check node information β_m and α_m as depicted in equ. (3.32) and equ. (3.33).

$$\beta_m = \sum_{\substack{k \in N(m)\backslash n \\ k < n}} \varphi\left(|T_{mn}^{(t-1)}|\right) + \sum_{\substack{k \in N(m)\backslash n \\ k > n}} \varphi\left(|T_{mn}^{(t-1)}|\right) \tag{3.32}$$

$$\alpha_m = \prod_{\substack{k \in N(m)\backslash n \\ k < n}} sign\left(T_{mn}^{(t-1)}\right) + \prod_{\substack{k \in N(m)\backslash n \\ k > n}} sign\left(T_{mn}^{(t-1)}\right) \tag{3.33}$$

First, the messages T_{mn} and α_m, β_m are initialized with corresponding *intrinsic* channel reliability values llr_n. Each sub-iteration involves three steps: check node processing, bit node processing and the update of α_m, β_m. The check node process is based on the property of $x = \varphi(\varphi(x))$. The *a posteriori* message T_n for the bit node n is computed by adding all the $E_{mn}^{(t)}$ with llr_n, hence $T_{mn}^{(t)}$. The sign of T_n is the decoded codeword. Afterwards, the updated $T_{mn}^{(t)}$ is sent back to check node processor to update the α_m, β_m. Thus the latest check node information is provided in the following sub-iteration. α_m and β_m values are updated dc_m times per iteration, where dc_m means the check node degree of check node m.

The faster convergence speed is due to the usage and the update of the variables β_m and α_m. Actually, they always include the latest information and send the latest information to the bit nodes.

3.5.1 Vertical shuffled normalized Min-Sum decoding algorithm

Base on the vertical shuffled schedule, a normalized Min-Sum algorithm for the vertical shuffled scheduling is proposed to reduce hardware complexity. Following the principle of Min-Sum algorithm, vertical shuffled normalized Min-Sum decoding algorithm uses the minimum value M_m^0 and the second minimum value M_m^1 of all the $E_{mn}^{(t)}$ connected to the check node m to simplify the check node processing.

The principle of the shuffled normalized Min-Sum decoder is that the M_m^0 and M_m^1 should be updated as soon as the new *extrinsic* messages $T_{mn}^{(t)}$ are available. So the incoming bit node processing for bit n in $N(m)$ can use the latest check node information from the m^{th} check node processor.

Algorithm 6 Vertical shuffled belief propagation algorithm

{**Initialization:** for all bit nodes n, where $m \in M(n)$ }

$T_{mn}^{(0)} = llr_n$

$\alpha_m^{(0)} = \prod_{n \in N(m)} sgn(llr_n)$

$\beta_m^{(0)} = \sum_{n \in N(m)} \varphi(|llr_n|)$

{**Decoding:**}

for $t = 1, .., t_{max}$ {***iteration*** }

  for $n = 1, .., N_{ldpc}$ { ***sub-iteration***}

    Check node processing

    $E_{mn}^{(t)} = \alpha_m \cdot sgn\left(T_{mn}^{(t-1)}\right) \cdot \varphi\left(\beta_m - \varphi\left(|T_{mn}^{(t-1)}|\right)\right)$, $m \in M(n)$

    Bit node processing

    $T_n^{(t)} = llr_n + \sum_{m \in M(n)} E_{mn}^{(t)}$, $n \in N(m)$

    $T_{mn}^{(t)} = T_n^{(t)} - \sum_{m \in M(n)} E_{mn}^{(t)}$, $n \in N(m)$

    check node update for next sub-iteration

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

    $\beta_m = \beta_m - \varphi\left(|T_{mn}^{(t-1)}|\right) + \varphi\left(|T_{mn}^{(t)}|\right)$, $m \in M(n)$

Hard decision according to $T_n^{(t)}$

In practice, two more variables are necessary to register the index of M_m^0 and M_m^1, denoted by P_m^0 and P_m^1. If the processing bit n neither equals to P_m^0 nor P_m^1, then the candidates of new M_m^0 are $\left\{M_m^0, |T_{mn}^{(t)}|\right\}$ and the candidates for a new M_m^1 are $\left\{M_m^1, |T_{mn}^{(t)}|\right\}$. However, if the processing bit n equals to P_m^0, then the candidates for a new M_m^0 are $\left\{M_m^1, |T_{mn}^{(t)}|\right\}$ and the candidates for a new M_m^1 are $\left\{M_m^1, |T_{mn}^{(t)}|\right\}$. In this case, $|T_{mn}^{(t-1)}|$ is the old M_m^0. Therefore, the update of check node information should exclude $|T_{mn}^{(t-1)}|$. The same principle is applied if the processing bit n is equal to P_m^1. In this case, the candidates for a new M_m^0 are $\left\{M_m^0, |T_{mn}^{(t)}|\right\}$ and the candidates for a new M_m^1 are $\left\{M_m^0, |T_{mn}^{(t)}|\right\}$. The update of P_m^0 and P_m^1 is carried out simultaneously with the update of M_m^0 and M_m^1.

3.6 Performance comparison

Fig. 3.4 shows the BER performance comparison of different decoding algorithms of DVB-T2 LDPC code with a frame size of 64800 and a code rate of R=4/5 over an AWGN channel with the maximum iteration number of 50. The legend VSSMS and HSSMS means the normalized Min-Sum for VSS and HSS, respectively. The simulation results show that the performance

Algorithm 7 Vertical shuffled Min-Sum algorithm

{**Initialization:** for all bit nodes n, where $m \in M(n)$ }
$T_{mn}^{(0)} = llr_n$
$\alpha_m^{(0)} = \prod_{n \in N(m)} sgn(llr_n)$
$M_m^0 = min\left(|llr_n|\right), P_m^0 = Index\left(M_m^0\right)$
$M_m^1 = secmin\left(|llr_n|\right), P_m^1 = Index\left(M_m^1\right)$
{**Decoding:**}
for $t = 1, .., t_{max}$ {***iteration***}
  for $n = 1, .., N_{ldpc}$ {***sub-iteration***}
    Check node processing
    if $\left(n == P_m^0\right)$
      $E_{mn}^{(t)} = \alpha_m \cdot \eta \cdot sgn\left(T_{mn}^{(t-1)}\right) \cdot M_m^1 \ , \ m \in M(n)$
    else
      $E_{mn}^{(t)} = \alpha_m \cdot \eta \cdot sgn\left(T_{mn}^{(t-1)}\right) \cdot M_m^0 \ , \ m \in M(n)$
    Bit node processing
    $T_n^{(t)} = llr_n + \sum_{m \in M(n)} E_{mn}^{(t)} \ , \ n \in N(m)$
    $T_{mn}^{(t)} = T_n^{(t)} - \sum_{m \in M(n)} E_{mn}^{(t)} \ , \ n \in N(m)$
    check node update for next sub-iteration
    $\alpha_m = \alpha_m \cdot sgn\left(T_{mn}^{(t-1)}\right) \cdot sgn\left(T_{mn}^{(t)}\right) \ , \ m \in M(n)$
    $M_m^0 = min\left(|T_{mn}^{(t)}|, |T_{mk'}^{(t-1)}|\right), P_m^0 = Index\left(M_m^0\right) \ , \ k' \in N(m) \backslash n$
    $M_m^1 = secmin\left(|T_{mn}^{(t)}|, |T_{mk'}^{(t-1)}|\right), P_m^1 = Index\left(M_m^1\right) \ , \ k' \in N(m) \backslash n$
Hard decision according to $T_n^{(t)}$

of the horizontal and vertical shuffled decoding schedule is very close both for the belief propagation algorithm and the normalized Min-Sum. These results demonstrate the efficiency of vertical shuffled decoding algorithm and the proposed vertical shuffled Min-Sum decoding algorithm.

In the following section, we will propose an architecture for the vertical shuffled Min-Sum LDPC decoder with parallelism level of 90. The curves of the floating point and fixed point VSSMS with maximum iteration number of 25 give the reference performance for the prototype of the corresponding LDPC decoder.

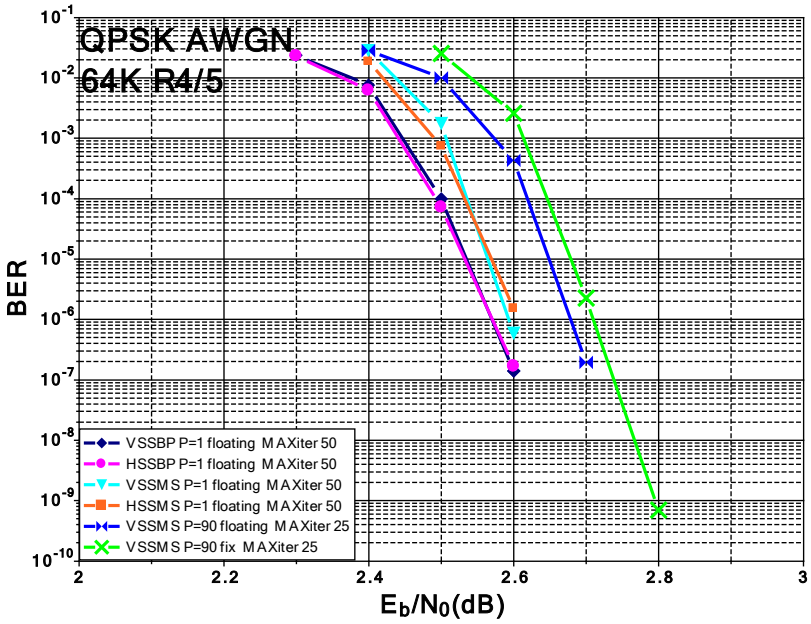


Figure 3.4 — Performance comparison for 64K LDPC code with a code rate of R=4/5 over an AWGN channel and a QPSK constellation

3.7 Design and implementation of a vertical shuffled Min-Sum LDPC decoder

In this section, the architecture design of a vertical shuffled Min-Sum LDPC decoder for the DVB-T2 standard is detailed. This is motivated by the introduction of an iterative information exchange between the decoder and the demapper.

3.7.1 The design of a vertical shuffled normalized Min-Sum LDPC decoder

3.7.1.1 The architecture of the proposed LDPC decoder

Fig. 3.5 depicts the proposed architecture based on the proposed Min-Sum algorithm detailed in sub-section. 3.5.1. The decoder is mainly composed of five major blocks: a bit node processor SISOA, a check node processor SISOB, two permutation networks, an IO processor and a control model. The *Input Output* (IO) processor includes the LLR RAM and Codeword RAM, which are connected to the bit node processor. The bit node processor and check node processor are connected by two permutation networks.

The original parallelism level of the LDPC codes in the DVB-T2 standard is 360. A paralleled LDPC decoder with the parallelism of 360 can provide high throughput, which is around 350 Mbps for the 64K LDPC with a code rate of R=4/5 with a maximum iteration number of 20 at the working frequency of 90 Mhz. However, in practice, the highest throughput of

DVB-T2 is 50.6 Mbps taking into account of pilot patterns ans guard interval. Moreover, 360 elements of check node processor and 360 elements of bit node processor require large amount of hardware resources. In order to get a good tradeoff between the throughput and the hardware complexity, the parallelism level in our design is fixed to 90. So 90 elements of bit node processors in SISOA and 90 elements of check node processors in SISOB have been designed to work in parallel. Any data path inside the decoder is a data bus composed of 90 words.

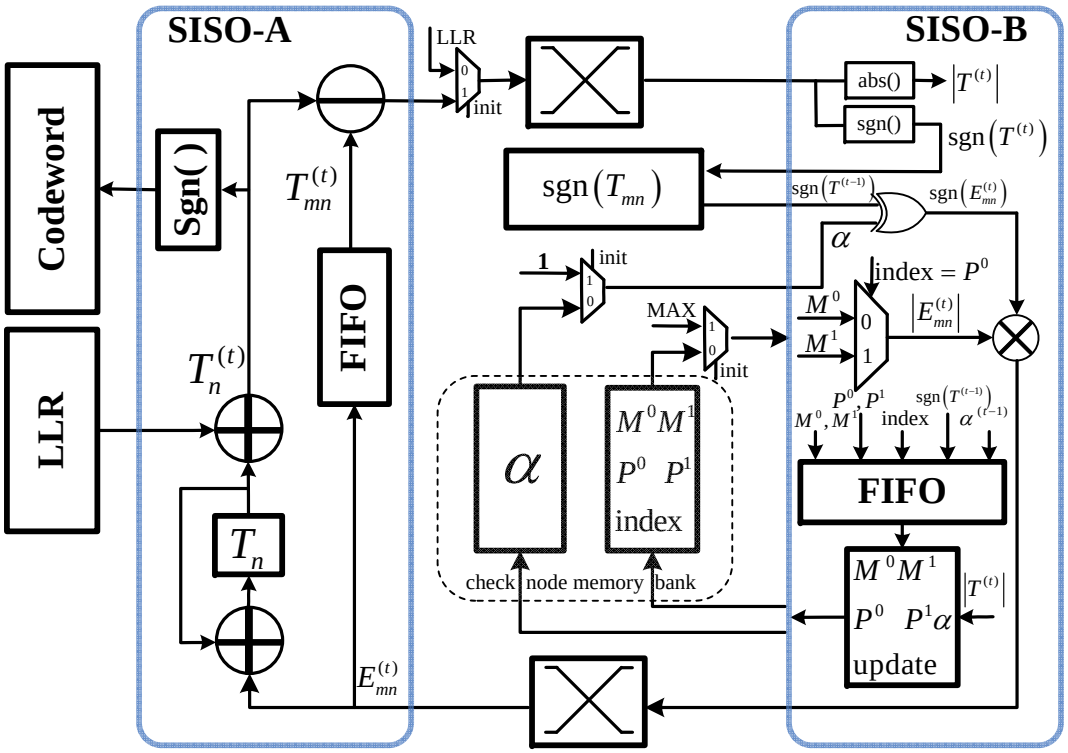


Figure 3.5 — Architecture of the proposed vertical shuffled normalized Min-Sum LDPC decoder

3.7.1.2 The timing schedule of the proposed LDPC decoder

A top-down description of decoding steps is illustrated in Fig. 3.6. The IO processing and decoding processing are described by *Finite State Machine* (FSM) form in Fig. 3.7. The IO processor is triggered by a frame start signal. As mentioned in sub-section 1.5.2.2, the parity check matrix of the LDPC code has a quasi-cyclic property only after the permutation of the check nodes and parity bit nodes. Therefore, the input processor should permute the parity bit nodes with a uniform step of Q. The input processor is also in charge of getting the LLRs data bus (90 words) from each single serial input LLR and writing them into the LLR RAM. The output processor is in charge of reading the codeword bus from the Codeword RAM and

providing them to the output serially. The decoding processor is also triggered by the frame start signal and it uses the pre-processed LLRs as input to start the decoding process.

The decoding process is composed of the MP initialization state and the iteration state. During the MP initialization state, the LLRs belonging to each column group are read from the LLR RAM, transmitted through the permutation network and then sent to check node processors. The check node processors provide: M_0,M_1,P_0,P_1,α, for each check node group. Those values are then registered into the check node memory bank. The sign of LLRs are passed through SISOB and registered into a SgnT RAM.

One decoding iteration is composed of N_{ldpc}/P sub-iterations, where P is the parallelism level of the decoder. During each sub-iteration, M_0,M_1,P_0,P_1, α are first read out from the memory bank to compute the *extrinsic* message $E_{mn}^{(t)}$. Then, the *extrinsic* messages are permuted by a barrel shifter to different elements of bit nodes processor, in which the *a posteriori* message T_n is computed based on all the *extrinsic* messages belonging to the bit node n. Afterwards, the *a priori* message $T_{mn}^{(t)}$ is sent back to the check node processor. SISOB uses the *a priori* message $T_{mn}^{(t)}$ to update M_0,M_1,P_0,P_1 and α. The iteration state stops the execution either when the iteration number reaches the maximum iteration number or when the decoder detects that the syndrome computation is satisfied.

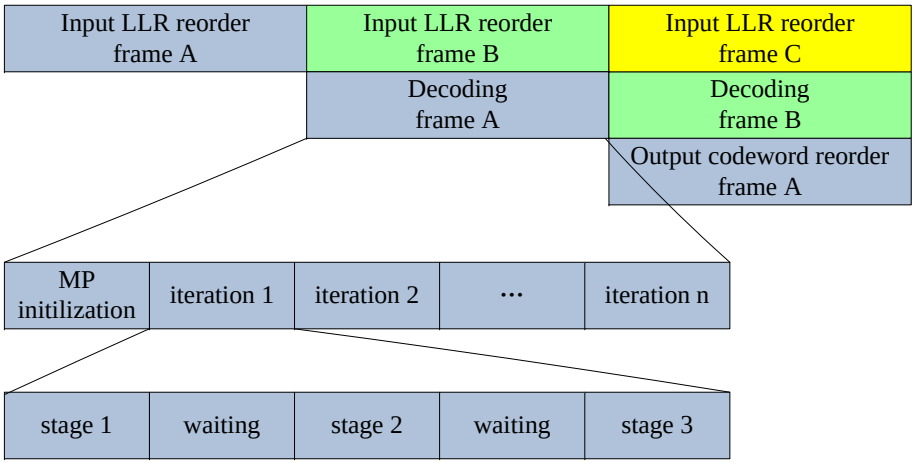


Figure 3.6 — Timing of the vertical shuffled normalized Min-Sum LDPC decoder

3.7.1.3 Memory management

LLR RAM and codeword RAM are the two block memories that are directly connected to the bit node processor. Two RAMs are assigned for each of these two kinds of RAMs, with the address size of N_{ldpc}/P, where P is the parallelism level. One of the LLR RAMs and one of the codeword RAMs are assigned for the decoding state, at the same time one of the

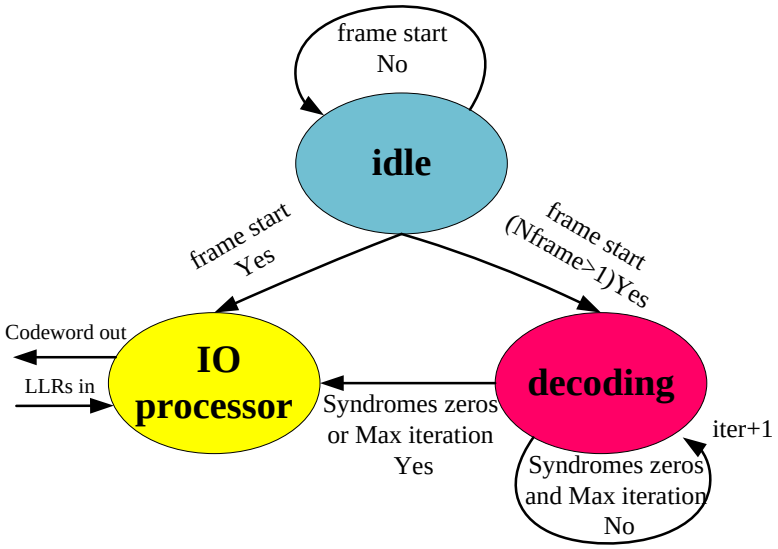


Figure 3.7 — Finite state machine of the VSS normalized Min-Sum decoder

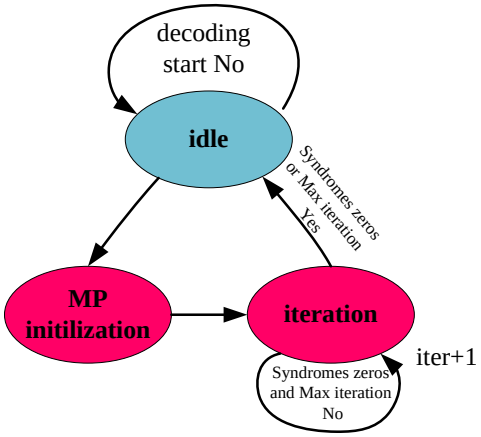


Figure 3.8 — Finite state machine of the decoding state

LLR RAMs is in charge of memorizing the input LLRs and one of the codeword RAMs is in charge of providing the hard decision. Two of the LLR RAMs switch their roles for each frame and so does the codeword RAMs. The elements of the data bus are organized in a sequential order, as illustrated in Fig. 3.9. (a), labelled as column order.

Table 3.2 depicts how the input LLRs are memorized in the LLR RAM with a parallelism level of 90, where $K_b = \frac{N_{ldpc}}{360}$. A 360x360 sub-matrix is split into four 90x90 sub-matrices. A detailed explanation will be given in sub-section 3.7.1.4.

SgnT RAM and the check node memory bank are the two memories that are directly connected to the check node processor. The elements of the data bus are organized in a row order, as illustrated in Fig. 3.9. (a). The size of SgnT RAM is Te/P, where Te is the total number of 1s in the parity-check matrix. The check node memory bank includes five small

address	element0	element1	element2	···	element89
0	0	4	8	···	356
1	1	5	9	···	357
2	2	6	10	···	358
3	3	7	11	···	359
4	$360+0$	$360+4$	$360+8$	···	$360+356$
5	$360+1$	$360+5$	$360+9$	···	$360+357$
6	$360+2$	$360+6$	$360+10$	···	$360+358$
7	$360+3$	$360+7$	$360+11$	···	$360+359$
	⋮	⋮	⋮	...	⋮
$4(K_b-1)$	$360(K_b-1)+0$	$360(K_b-1)+4$	$360(K_b-1)+8$	···	$360(K_b-1)+356$
$4(K_b-1)+1$	$360(K_b-1)+1$	$360(K_b-1)+5$	$360(K_b-1)+9$	···	$360(K_b-1)+357$
$4(K_b-1)+2$	$360(K_b-1)+2$	$360(K_b-1)+6$	$360(K_b-1)+10$	···	$360(K_b-1)+358$
$4(K_b-1)+3$	$360(K_b-1)+3$	$360(K_b-1)+7$	$360(K_b-1)+11$	···	$360(K_b-1)+359$
$4K_b$	$0+0+Kldpc$	$4Q+0+Kldpc$	$8Q+0+Kldpc$	···	$356Q+0+Kldpc$
$4K_b+1$	$1Q+0+Kldpc$	$5Q+0+Kldpc$	$9Q+0+Kldpc$	···	$357Q+0+Kldpc$
$4K_b+2$	$2Q+0+Kldpc$	$6Q+0+Kldpc$	$10Q+0+Kldpc$	···	$358Q+0+Kldpc$
$4K_b+3$	$3Q+0+Kldpc$	$7Q+0+Kldpc$	$11Q+0+Kldpc$	···	$359Q+0+Kldpc$
$4K_b+4$	$0+1+Kldpc$	$4Q+1+Kldpc$	$8Q+1+Kldpc$	···	$356Q+1+Kldpc$
$4K_b+5$	$1Q+1+Kldpc$	$5Q+1+Kldpc$	$9Q+1+Kldpc$	···	$357Q+1+Kldpc$
$4K_b+6$	$2Q+1+Kldpc$	$6Q+1+Kldpc$	$10Q+1+Kldpc$	···	$358Q+1+Kldpc$
$4K_b+7$	$3Q+1+Kldpc$	$7Q+1+Kldpc$	$11Q+1+Kldpc$	···	$359Q+1+Kldpc$
	⋮	⋮	⋮	...	⋮
$4N_b-4$	$0+(Q-1)+Kldpc$	$4Q+(Q-1)+Kldpc$	$8Q+(Q-1)+Kldpc$	···	$356Q+(Q-1)+Kldpc$
$4N_b-3$	$1Q+(Q-1)+Kldpc$	$5Q+(Q-1)+Kldpc$	$9Q+(Q-1)+Kldpc$	···	$357Q+(Q-1)+Kldpc$
$4N_b-2$	$2Q+(Q-1)+Kldpc$	$6Q+(Q-1)+Kldpc$	$10Q+(Q-1)+Kldpc$	···	$358Q+(Q-1)+Kldpc$
$4N_b-1$	$3Q+(Q-1)+Kldpc$	$7Q+(Q-1)+Kldpc$	$11Q+(Q-1)+Kldpc$	···	$359Q+(Q-1)+Kldpc$

Table 3.2 — The address of initial LLR RAM

memories for M_0,M_1,P_0,P_1 and α. The read and write addresses and the timing of those memories are identical. Therefore, we call them the check node memory bank. The size of check node memory bank is P_{ldpc}/P, where P_{ldpc} is the number of parity check nodes.

3.7.1.4 Sub-matrix split

The throughput of DVB-T2 LDPC decoder with the original parallelism 360 is too high in practice, as mentioned before. Reducing the parallelism level by sub-matrix split is a good way to reduce the hardware cost at the price of reducing the throughput. Let us denote the original parallelism level as M and the number of the reduced parallelism as D. The number D can be a fraction of M. In our case, M=360 and D can be 180, 120, 90 or 45. The choice of the number D depends on the required throughput, the clock frequency, the number of iteration and the required hardware resources.

The target of the sub-matrix split is to make sure that each group of D bit nodes connected to a group of D parity check nodes can be accessed during the same time or in the same data bus.

Fig. 3.10 gives an example of sub-matrix split. In this case, M is equal to 9, D is equal

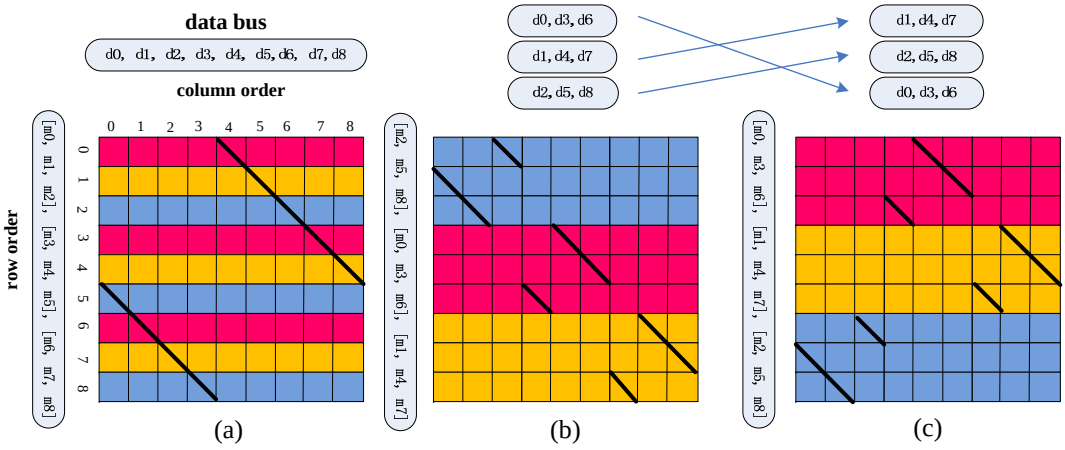


Figure 3.9 — Process of split a RAM word simultaneously with the sub-matrix split

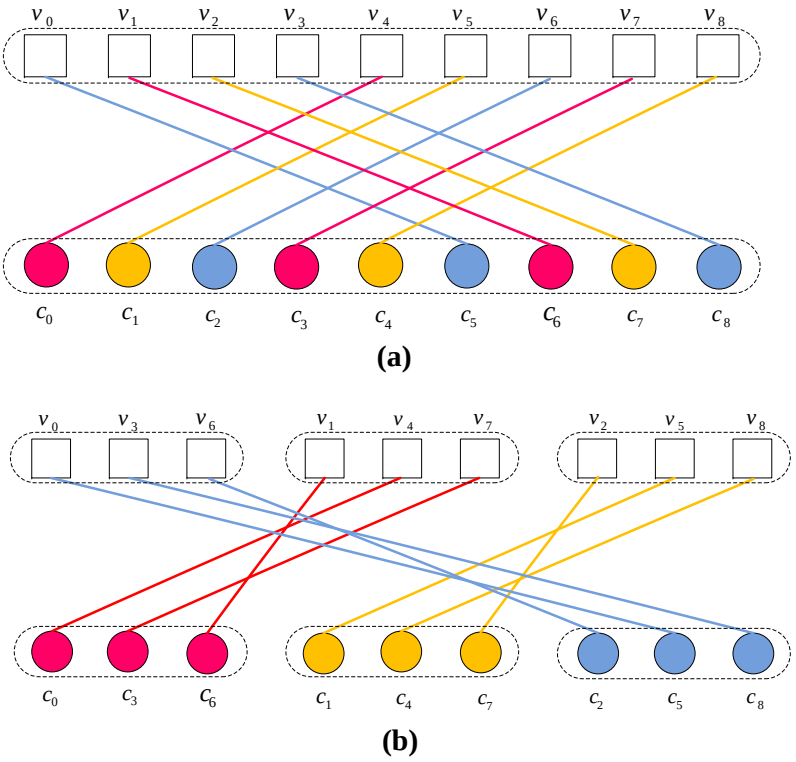


Figure 3.10 — An example of how to split the sub-matrix

to 3 and $Perm$ is equal to 5. $Perm$ is the row index of the first column of the diagonal. To do the sub-matrix split, D bit nodes are grouped. Each of them with an equal distance of M/D. In Fig. 3.10(a), the three groups of bit nodes are $(0, 3, 6)$, $(1, 4, 7)$ and $(2, 5, 8)$. The check node group connected to the bit node group also has the property that each of the check nodes in one group has an equal distance of M/D. Then the 9x9 sub-matrix can be split into three 3x3 sub-matrices.

The matrix split is based on the bit node position. It means that the groups of bit nodes are in sequence. However the corresponding order of groups of check nodes may not be sequential. So the sub-matrices have to be re-ordered to make sure that the check node group becomes sequential if necessary. The permutation among the sub-matrices is computed as: $mod(Perm, (M/D))$. In Fig. 3.10(b), the permutation among the three sub-matrices is equal to 2. Each of the smaller sub-matrices have their own new $Perm$. The way to compute the $Perm$ value is described as:

Algorithm 8 Perm value for each split sub-matrix

1: **For** $i = 0 : M/D$

2: $Perm_i = int[(Perm + i)/(M/D)]$

3.7.2 Avoiding message passing inefficiency caused by double diagonal sub-matrices

The *Double Diagonal Sub-Matrix* (DDSM) induces message update conflicts, which is a critical problem in the design of a LDPC decoder for the DVB-T2 standard. Some solutions have been proposed for the horizontal layered LDPC decoder, however no previous work has mentioned about the corresponding solution for a vertical shuffled LDPC decoder. We are going to detail the problem and propose the solution in the case of a vertical shuffled LDPC decoding.

3.7.2.1 Message update conflict problem

Fig. 3.11 shows a sub-matrix that contains two diagonals. We call it *Double Diagonal Sub-Matrix* (DDSM). As mentioned in Chapter 1, the parity check matrices of DVB-T2 LDPC codes contain this kind of sub-matrices. There are even triple diagonal sub-matrices for the LDPC code with a code length of 64800 with a code rate of $R = 4/5$ LDPC code.

If paralleled LDPC decoder is considered, the double or triple diagonal sub-matrices induce message update conflicts, that affect BER performance.

Let us take the horizontal shuffled Min-Sum decoding algorithm as an example to illustrate this problem. The message sent from the check node group to the bit node group for the first

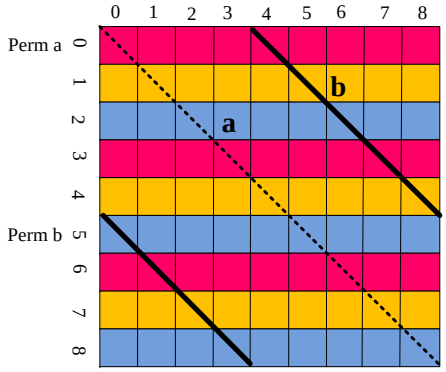


Figure 3.11 — A double diagonal sub-matrix

diagonal is denoted as $[E_{mn^a}]^P$ and the message sent from the bit node group to the check node group for the first diagonal is denoted as $[T_{mn^a}]^P$. The corresponding message for the second diagonal is denoted as $[E_{mn^b}]^P$ and $[T_{mn^b}]^P$. In the traditional message passing schedule, the *a posteriori* message provided by the first diagonal is over written by the *a posteriori* message provided by the second diagonal. This conflict is equivalent to deleting the first diagonal in the DDSM, which leads to performance degradation.

The traditional update process of the *a posteriori* message T_n by these two diagonals is listed as:

$$[T_{mn^a}^{(t)}]^P = [T_n^{(t-1)}]^P - [E_{mn^a}^{(t-1)}]^P \tag{3.34}$$

$$[T_n^{(t)}]^P = [T_{mn^a}^{(t)}]^P + [E_{mn^a}^{(t)}]^P \tag{3.35}$$

$$[T_{mn^b}^{(t)}]^P = [T_n^{(t-1)}]^P - [E_{mn^b}^{(t-1)}]^P \tag{3.36}$$

$$[T_n^{(t)}]^P = [T_{mn^b}^{(t)}]^P + [E_{mn^b}^{(t)}]^P \tag{3.37}$$

The message passing inefficiency is caused by two aspects. One is that the *a priori* message $[T_{mn^a}^{(t)}]^P$ and $[T_{mn^b}^{(t)}]^P$ are both derived from the same $[T_n^{(t-1)}]^P$ message. This leads that the *a priori* message $[T_{mn^b}^{(t)}]^P$ can not benefit from the *a posteriori* message $[T_n^{(t)}]^P$ provided by the first diagonal. Second is that both $([T_{mn^a}^{(t)}]^P + [E_{mn^a}^{(t)}]^P)$ and $([T_{mn^b}^{(t)}]^P + [E_{mn^b}^{(t)}]^P)$ provide the feedback information, holding the information provided by their own computation without any additional information provided by the other diagonal.

Unlike the horizontal shuffled decoding algorithm, the update conflict is present in the update process for the check node information (M_0,P_0,M_1, P_1 and α) of the vertical shuffled

LDPC decoding algorithm. Let us take the update of α as an example. The $[\alpha_m]^P$ is first updated based on the feedback *a priori* message $[sgn\left(T_{mn^a}^{(t)}\right)]^P$ from the first diagonal.

$$[\alpha_m]^P = [\alpha_m]^P \cdot [sgn\left(T_{mn^a}^{(t-1)}\right)]^P \cdot [sgn\left(T_{mn^a}^{(t)}\right)]^P \tag{3.38}$$

However, the $[\alpha_m]^P$ get the updated value based on the feedback *a priori* message $[sgn\left(T_{mn^b}^{(t)}\right)]^P$ from the second diagonal simultaneously and independently from equ. (3.38).

$$[\alpha_m]^P = [\alpha_m]^P \cdot [sgn\left(T_{mn^b}^{(t-1)}\right)]^P \cdot [sgn\left(T_{mn^b}^{(t)}\right)]^P \tag{3.39}$$

Fig. 3.12 illustrates how the update conflict impacts the decoding performance both for horizontal and vertical shuffled decoding algorithms. The vertical shuffled algorithm is more sensitive to the update conflict. Indeed, α represents the syndrome information. Any modification to the syndrome estimation leads to a sign inverse in the computation of the *extrinsic* information. The summation of *extrinsic* messages with sign inversion is more sensitive to modification of the sign of the *a posteriori* message.

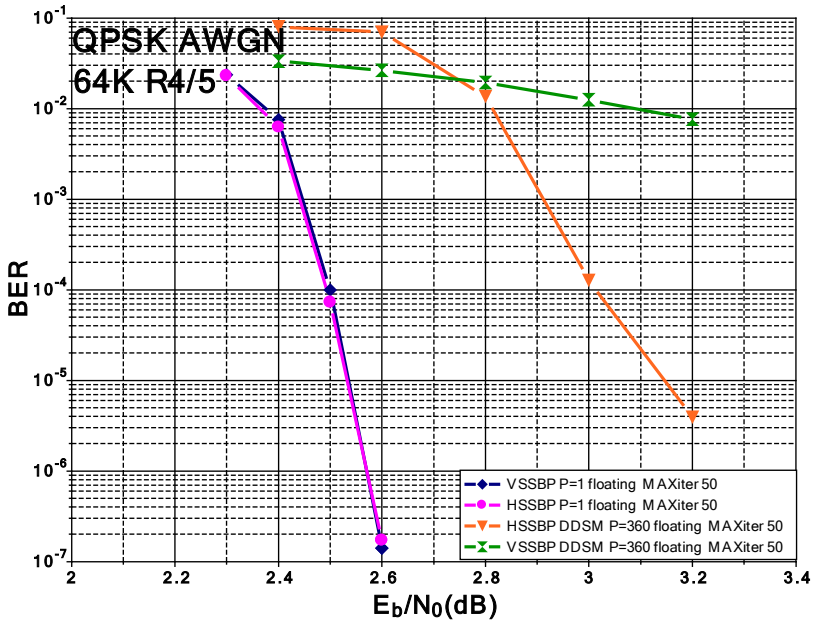


Figure 3.12 — Performance comparison for 64K LDPC with a code rate of R=4/5 over an AWGN channel and a QPSK constellation

3.7.2.2 Methods to avoid message update conflict for horizontal shuffled decoding algorithm

Two kinds of solutions can be applied to reduce the performance degradation caused by DDSM. One solution is based on sub-matrix split of the parity-check matrix and the other

solution is based on the modification of the decoding algorithm and consequently the architecture of the decoder.

The sub-matrix split is an efficient way to reduce the number of DDSMs, but this can not guarantee the removal of all DDSMs. The number of reduced DDSMs is constant and depends on the parity check matrix. If the $Perm$ of the two double diagonals satisfy equ. (3.40), the double DDSMs still exists in the split sub-matrices. Consequently the residual number of the DDSMs for a constant parallelism can be evaluated.

$$mod\left(mod\left((Perm_a - Perm_b), D\right), \left(\frac{M}{D}\right)\right) = 0 \tag{3.40}$$

parallelism	R1/2	R3/5	R2/3	R3/4	R4/5	R5/6
360	8	32	12	23	31	35
180	4	19	5	10	13	21
120	2	16	4	8	15	12
90	2	8	2	3	6	13
45	0	2	0	3	3	5

Table 3.3 — Number of DDSMs for different parallelism level for 64K LDPC codes

Table 3.3 lists the number of the residual DDSMs for different code rates of 64K LDPC with different parallelism levels. Fig. 3.13 illustrates the density of the DDSM in the parity-check matrix of the 16K LDPC with a code rate of R=1/2 with different parallelism levels. The number of DDSMs is reduced from 8 to 1 with parallelism of the parity-check matrix decreased from 360 to 90.

However, even for a DVB-T2 decoder with a parallelism level of 45, there are still some residual DDSMs. The authors in [51] proposed to extend the parity-check matrix to reduce the residual DDSM in order to avoid any modification of the decoding algorithm for a horizontal shuffled decoder. Moreover, the authors in [52] proposed a way to improve the performance of the decoder for the LDPC code with DDSM, as shown in equ. (3.41). In this case, some modification of the architecture is mandatory.

$$[T_n^{(t)}]^P = [T_{mn^a}^{(t)}]^P + [E_{mn^a}^{(t)}]^P + \left([E_{mn^b}^{(t)}]^P - [E_{mn^b}^{(t-1)}]^P\right) \tag{3.41}$$

$$= [T_n^{(t-1)}]^P + \left([E_{mn^a}^{(t)}]^P - [E_{mn^a}^{(t-1)}]^P\right) + \left([E_{mn^b}^{(t)}]^P - [E_{mn^b}^{(t-1)}]^P\right) \tag{3.42}$$

The update of the *a posteriori* message $T_n^{(t)}$ is based on the summation of the differences of *extrinsic* message of the two diagonals. In this case, $[E_{mn^b}^{(t-1)}]^P$ have to be memorized and a selection between the *a posteriori* message update from the first diagonal equ. (3.35) and the *a posteriori* message update from the second diagonal equ. (3.41) is also necessary.

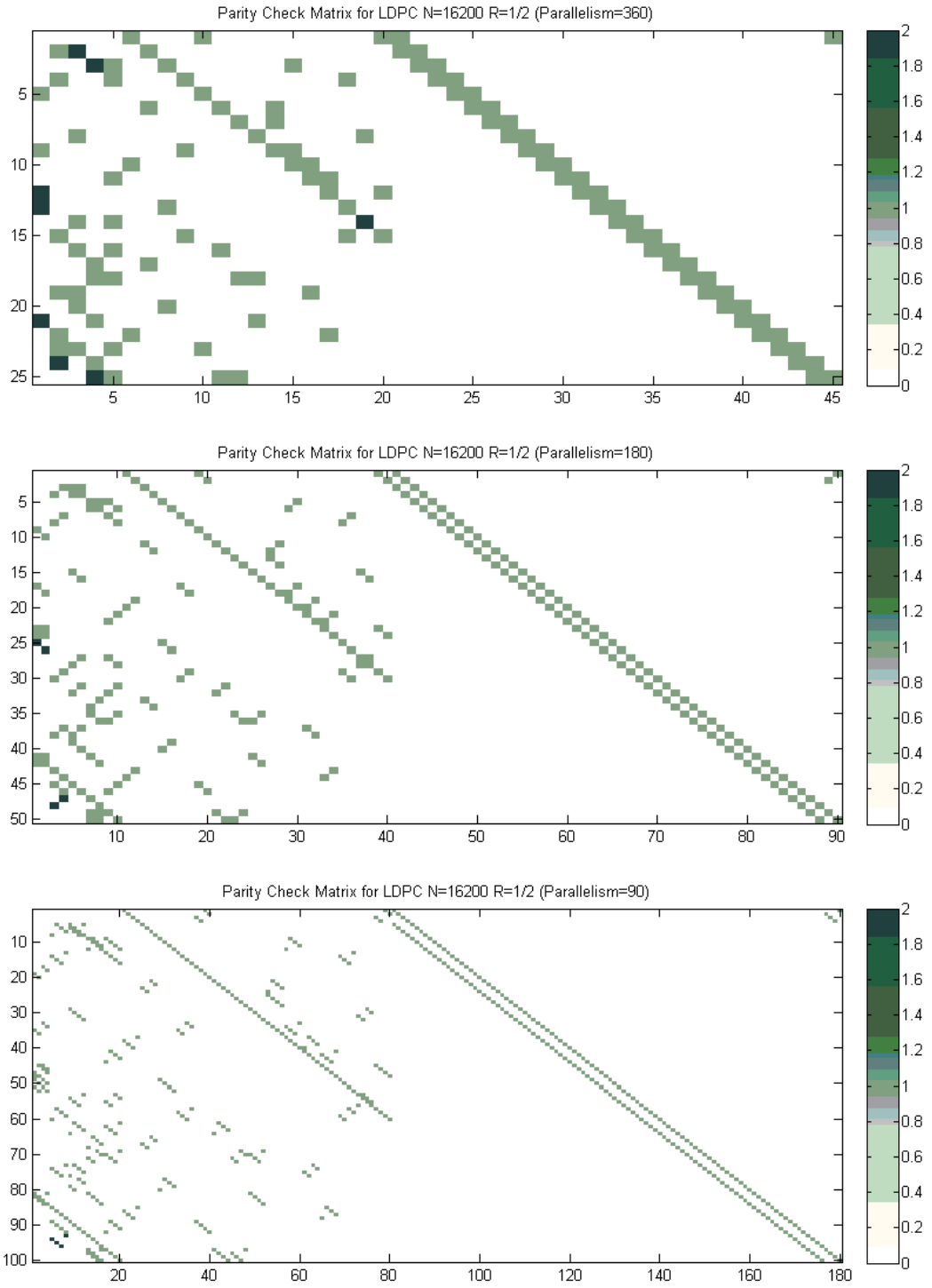
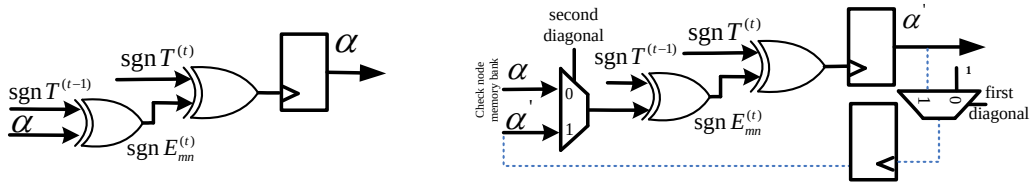


Figure 3.13 — Parity check matrix with double diagonal sub-matrices of 16K LDPC with a code rate of R=1/2

3.7.2.3 Methods to avoid message update conflict for vertical shuffled decoding algorithm

One important target of the check node processor design is to overcome the message update conflicts after sub-matrix split processing. Fig. 3.14 and Fig. 3.15 illustrate the proposed architecture of the check node update processor, that avoids message update conflicts.



(a) without conflict avoiding (b) with conflict avoiding

Figure 3.14 — Architecture of α update for the vertical shuffled normalized Min-Sum LDPC decoder

Let us take the α_m update processor as an example to illustrate the proposal. If there is no double diagonal, then the latest updated α_m is achieved by the multiplication of:

1.) α_m: information read out from the check node information bank

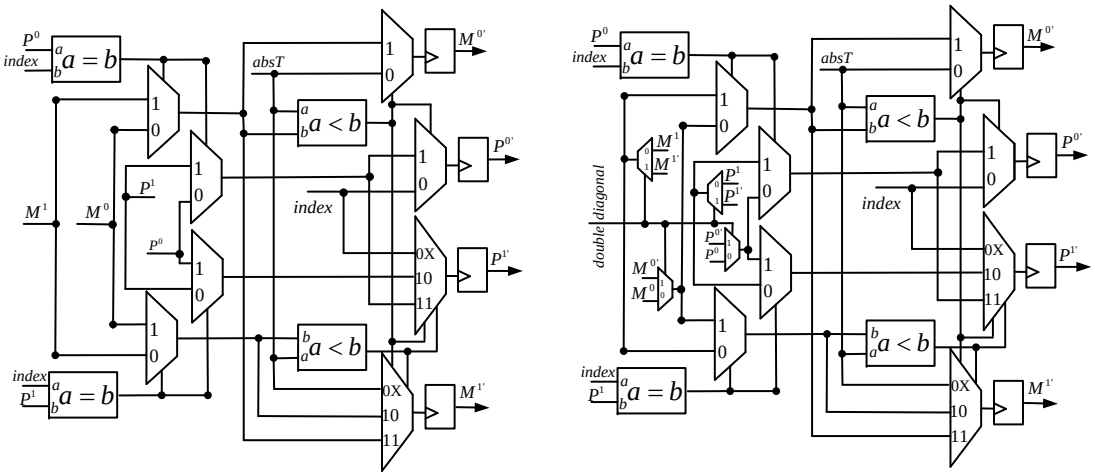
2.) $sgn(T_{mn}^{(t-1)})$: information read out from SgnT RAM

3.) $sgn(T_{mn}^{(t-1)})$: information fed back from the bit node processor

A register for α_m is assigned. Then the updated α_m is memorized and written back to the check node memory bank. For any double diagonal sub-matrix, the update of α_m for the first diagonal, that is denoted as temporary $\alpha^{'}$, follows the way mention previously. However this is not the final value. The update of α_m for the second diagonal uses the registered $\alpha^{'}$ and $sgn(T_{mn^b}^{(t)})$, $sgn(T_{mn^b}^{(t-1)})$. Only this final α_m is written back to the check node memory bank. So the update process for the final α_m can be computed as:

$$[\alpha_m]^P = \underbrace{[\alpha_m]^P \cdot [sgn\left(T_{mn^a}^{(t-1)}\right)]^P \cdot [sgn\left(T_{mn^a}^{(t)}\right)]^P}_{[\alpha_m']^P} \cdot [sgn\left(T_{mn^b}^{(t-1)}\right)]^P \cdot [sgn\left(T_{mn^b}^{(t)}\right)]^P \quad (3.43)$$

The main idea of designing a conflict free message update architecture for a check node processor resides in taking advantage of a temporary value (from the update of the first diagonal) in order to complete the final update. This principle is also applied to M_0,P_0,M_1 and P_1, which are shown in Fig. 3.15.



(a) without conflict avoidance (b) with conflict avoidance

Figure 3.15 — Architecture of MP information update for the vertical shuffled normalized Min-Sum LDPC decoder

3.7.3 Avoiding memory access conflict caused by a pipeline architecture

A horizontal decoding algorithm uses the dependency between each layer, that enables the usage of the latest updated *a posteriori* information to accelerate the convergence speed. The same principle is applied for the vertical shuffled decoding algorithm. It uses the latest updated check node information α_m and β_m to speed up convergence. Therefore, the design of a vertical shuffled LDPC decoder should make sure that each sub-iteration can benefit from the latest updated check node information α_m and β_m.

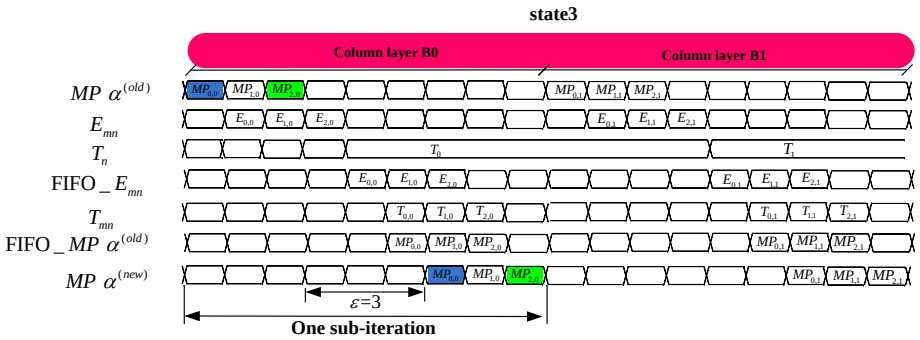


Figure 3.16 — Timing schedule of one iteration for SISOA and SISOB processor without pipeline for a vertical shuffled Min-Sum LDPC decoder

A non-pipelined timing schedule for the check node and bit node processor of the proposed VSS decoder is shown in Fig. 3.16. First the check node processor SISOB sequentially reads the check node information M_0,P_0,M_1,P_1 and α from the check node memory bank and feeds them into a FIFO RAM serially. At the same time, SISOB processor uses this information

to generate the *extrinsic* information E_{mn} and feeds E_{mn} into a FIFO RAM and SISOA processor. The SISOA processor provides the *a posteriori* information after bit node processor receives all the dv_n *extrinsic* information, where dv_n means the bit node degree of the n^{th} bit. In the following dv_n time periods, the *extrinsic* information E_{mn} is read sequentially from FIFO RAM and the *a priori* information is generated by subtracting E_{mn} from T_n simultaneously. The permuted *a priori* information T_{mn} is then fed back to the check node processor. Finally, the check node processor uses it and reads the corresponding check node information to update the check node information at the same. In this non-pipelined schedule, one sub-iteration needs $2dv_n + \epsilon$ time periods, where $\epsilon = 3$ is the delay of one sub-iteration process. The corresponding throughput is given by:

$$D = \frac{K_{ldpc} \cdot f_0}{T_{iter} \cdot N_{iter}} \tag{3.44}$$

where K_{ldpc} is the number of information bits, f_0 is the working frequency of the decoder, N_{iter} is the iteration number and T_{iter} is the total executive cycles necessary for one iteration. For the non-pipelined schedule, T_{iter} is equal to

$$T_{iter} = \sum_{i=0}^{i<(K_{ldpc})/P} 2dv_i + \epsilon \cdot (K_{ldpc})/P \tag{3.45}$$

Most of LDPC decoders use pipeline technique to increase the throughput and the architecture efficiency. However introducing pipeline may cause memory access conflict problems. The check node messages in the check node memory bank may be accessed before being updated. Fig. 3.17 depicts the timing schedule for the check node and bit node processors with pipeline. The delay from the reading of the check node information to the update of the check node information is $dv+3$. During this period, any usage of the check node information MP_m, α_m leads to performance degradation.

Adding idle time before the usage of check node information to wait for the update of required check node information is one method to solve the problem. However this waiting time for each conflict event is totally different. Therefore it is not suitable for hardware implementation.

Let cad_j^i denote the address of the i^{th} check node group connected to the bit node group j. If the check node information with address cad_j^i is read out more than once before the update of this check node information, then there is a memory access conflict. Actually, the condition that leads to a conflict-free memory access is:

$$\{cad_j^i\} \quad \bigcap \quad \{(dv^i + \epsilon) \quad clk \quad delay \quad of \quad cad_j^i\} = \phi \tag{3.46}$$

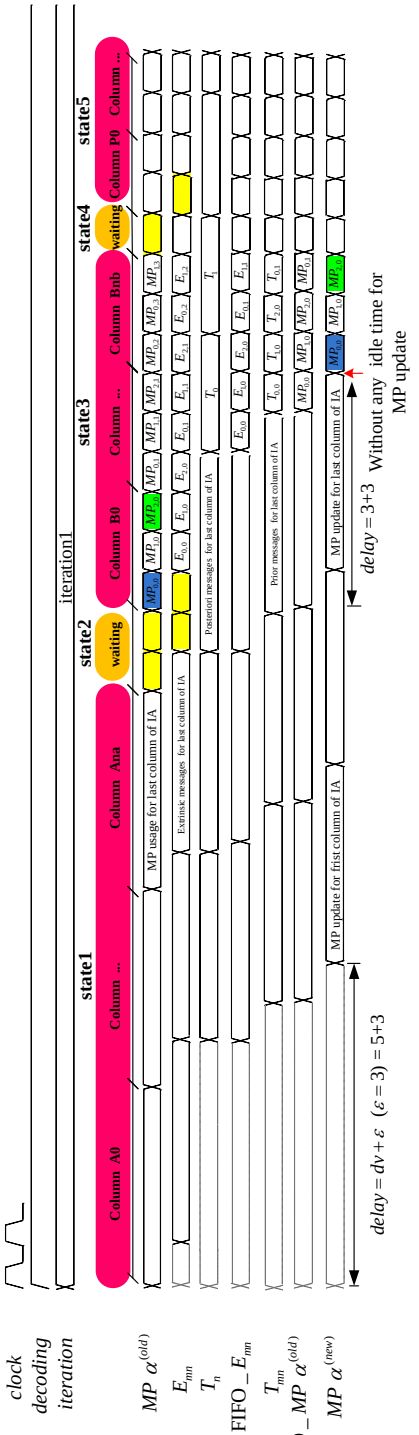


Figure 3.17 — Timing schedule of one iteration for SISOA and SISOB processors with pipeline

The bit nodes are divided into three parts according to the value of bit node degree. **IA** belongs to the information bit nodes. Its bit node degree is bigger than 3 and the value is a function of the code rate. **IB** also belongs to the information bit nodes. Its bit node degree is constant and is equal to 3. **Parity** denotes the parity bit nodes, with a constant bit node degree of 2. The delay of the three parts is different. A lower delay time leads to a lower chance of having memory conflict event.

Rovini [53] and Marchand [51] have proposed to re-arrange the schedule carefully to avoid or minimize the idle time for the horizontal shuffled decoding. In our study, we proposed a memory access free schedule with negligible idle time for vertical shuffled LDPC decoder. The method illustrated in Fig. 3.18 consists of three steps, as follows:

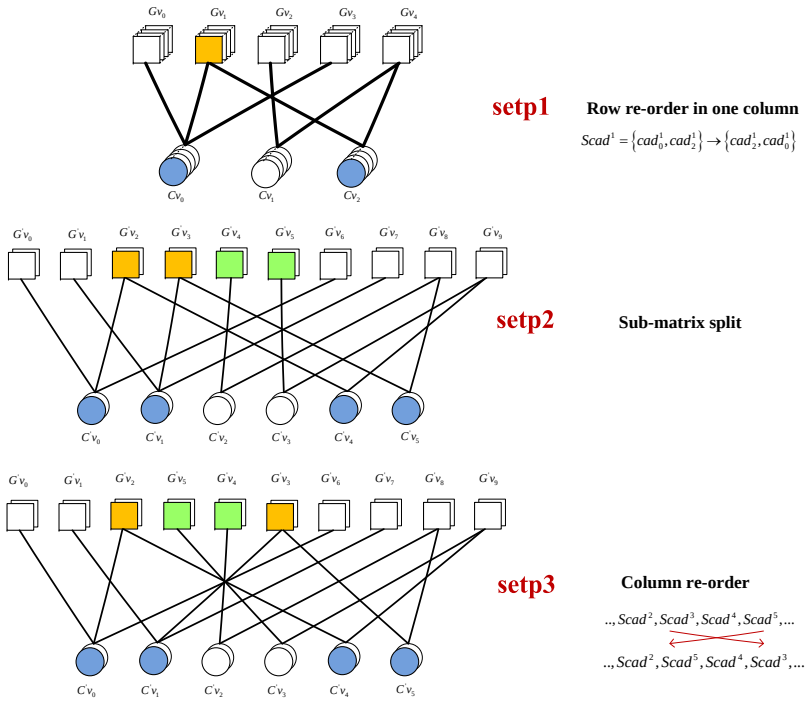


Figure 3.18 — Steps of the proposed rescheduling for one example

1. An elaborate re-arrangement of the cad_j^i following the principle of equ. (3.46) is carried out among each set $Scad^i$. Where cad_j^i corresponds to the j^{th} address of the check node group connected to the i^{th} bit node group for a group size of 360 and $Scad^i$ is the set of cad_j^i for a bit node group i.

2. The sub-matrix split is carried out. Then the number of memory access conflict events is decreased. An architecture with a smaller parallelism level has less risk of memory access conflicts due to pipeline.

3. After these two steps, there may exist remaining conflict events. Finally, a re-arrangement

is applied to the set of $\{Scad^{i0}, Scad^{i1}, \cdots, Scad^{in}\}$. It means a re-order of columns of the parity check matrix. This step introduces some additional hardware resources to manage the input/output accesses of bit node processors. However, these resources induces low additional hardware cost.

The schedule re-arrangement for the horizontal shuffled decoder can be carried out both for the different addresses belonging to check node group and among the sets of check node groups. However for the vertical shuffled decoder, the arrangement among the address sets of bit node groups (step 3) is forbidden, unless it follows by a corresponding re-order of the codeword.

Let us denote the parity check matrix $H = [h_1, h_2, h_3, \cdots, h_n]$, where h_i means the i^{th} column of the the parity check matrix. For any linear code, the codeword $C = [c_1, c_2, c_3, \cdots, c_n]$ should satisfy the syndrome equation : equ. (3.47). So any re-arrangement of the column order of the parity check matrix by π function requires a corresponding codeword re-arrangement by π.

$$[h_1, h_2, h_3, \cdots, h_n]^\intercal \cdot [c_1, c_2, c_3, \cdots, c_n] = 0 \tag{3.47}$$

$$[\pi(h_1, h_2, h_3, \cdots, h_n)]^\intercal \cdot [\pi(c_1, c_2, c_3, \cdots, c_n)] = 0 \tag{3.48}$$

In conclusion: different from the scheduling scheme for horizontal shuffled decoder, the proposed method consists of three steps: re-arrangement of the addresses in the set for each bit node group; sub-matrix split and re-arrangement among the sets of addresses for the bit node groups if necessary. The proposed method guarantees no memory access conflicts caused by a pipeline architecture. Moreover, it limits the time of T_{iter} to a minimum number as:

$$T_{iter} = \sum_{i=0}^{i<(K_{ldpc})/P} dv_i + \underbrace{dv_I A - dv_I B}_{Tstage2} + \underbrace{dv_I B - dv_P}_{Tstage4} + \epsilon \tag{3.49}$$

Each iteration is divided into 5 stages, as shown in Fig. 3.19. The waiting time for stage i is equal to $dv_{stage(i-1)} - dv_{stage(i+1)}$, in order to avoid the check node memory bank update conflict between $stage(i-1)$ and $stage(i+1)$. Consequently, the number of idle time for one iteration in our conflict-free scheduling is $(dv_I A - dv_I B) + (dv_I B - dv_P)$.

3.7.4 Logic synthesis results

The proposed LDPC decoder was designed by VHDL and implemented onto a FGPA board with Virtex 5 LX330 device. The implementation does not support all the code rates of the

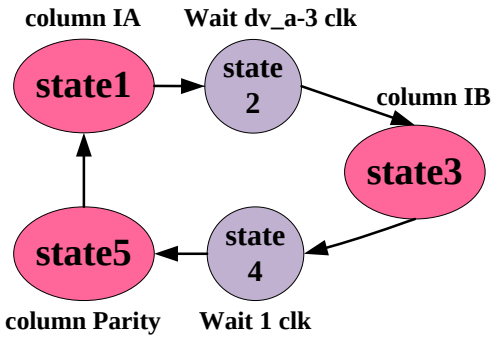


Figure 3.19 — Finite state machine of one iteration

two different code lengths. For the moment, the decoder supports 64K LDPC codes with code rate of R=3/4, R=4/5 and 16K LDPC codes with a code rate of R=3/4. Table 3.4 lists the logic synthesis results of the proposed LDPC decoder after place and route. The maximum frequency f_0 of the decoder can reach 80Mhz after place and route. This results leads to a throughput of 80Mbps for 64K LDPC code with a code rate of R=4/5 with a maximum iteration number of 20.

Number of Slices	14958 out of 51840 28%
Number of Slice Registers	12404 out of 207360 5%
Number of Slice LUTS	39713 out of 207360 19%
Number of Slice LUT-Flip Flop pairs	44809 out of 207360 21%
Number of RAMB18X2s	2 out of 288 1%
Number of RAMB18X2SDPs	6 out of 288 2%
Number of RAMB36SDP_EXPs	33 out of 288 11%
Number of RAMB36_EXPs	46 out of 288 15%

Table 3.4 — Virtex5 LX330 device logic synthesis results after place and route

3.8 Prototype of a simplified DVB-T2 transceiver system

To demonstrate the performance of the proposed vertical shuffled Min-Sum LDPC decoder, a prototype of the transmission system without OFDM modulation on the DVB-T2 standard is implemented in a Xilinx Virtex5 LX330 device on a target board. The performance test was carried out for three different kinds channel: AWGN channel, fading channel and fading channel with erasures.

3.8.1 Simplified DVB-T2 transceiver system

A simplified DVB-T2 system including the transmitter, channel emulator and receiver was implemented to verify and validate the efficiency of the proposed demapping and decoding algorithm and the corresponding architectures. The block diagram of the prototype of the simplified DVB-T2 system is given in Fig. 3.20.

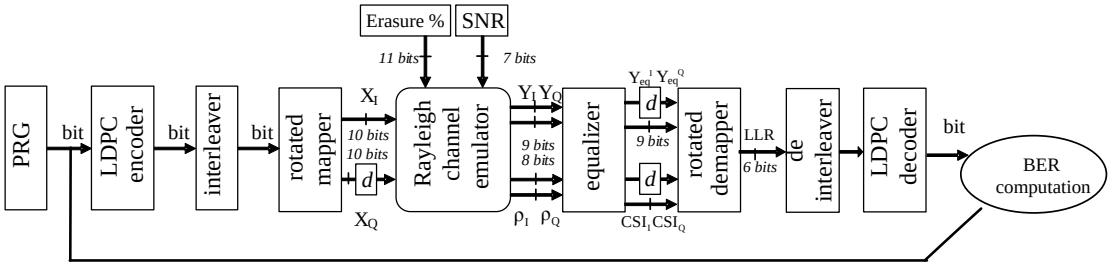


Figure 3.20 — Block diagrams of a simplified DVB-T2 transceiver system

3.8.2 Transmitter elements

The transmitter consists of the pseudo random generator, LDPC encoder, bit interleaver and mapper.

3.8.2.1 Pseudo random generator

At the transmitter side, a *Pseudo Random Generator* (PRG) is assigned to generate the random binary source by using *Linear Feedback Shift Register* (LFSR) with the prime polynomial equation: $p = x^{31} + x^{30} + x^{29} + x^9 + 1$. Table 3.5 lists the input and output ports of the PRG module and gives the quantization bits and the description of each signal. In order to avoid the repetition, we do not list clk and rst signals in the following Tables of the ports definition.

Signal	I/O	Quantization	Description
clk	I	1	system clock
rst	I	1	reset
en	I	1	work enable
psa	I	1	psa:1 random bits psa:0 output zero
out	O	1	bit stream
out_en	O	1	out enable

Table 3.5 — Ports definition of PRG

3.8.2.2 LDPC encoder

The LDPC encoder takes the random binary information bit stream to generate the parity check bits using the encoding algorithm, which has been detailed in Chapter 1. The corresponding architecture is illustrated in Fig. 3.21 and the ports definitions are listed in Table 3.6.

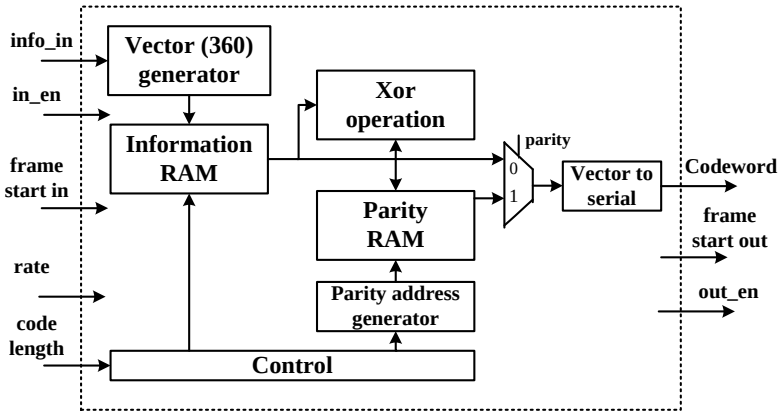


Figure 3.21 — Architecture of the LDPC encoder

Signal	I/O	Quantization	Description
codelen	I	1	0: 16K LDPC 1:64K LDPC
rate	I	3	1: R1/2; 2: R3/5; 3: R2/3; 4: R3/4; 5: R4/5; 6: R5/6
frame_str_in	I	1	frame start lasts one clock
info_in	I	1	uncoded bit stream
in_en	I	1	enable of the uncoded bit stream
frame_str_out	O	1	frame start lasts one clock
cdw_out	O	1	codeword
out_en	O	1	enable of the codeword

Table 3.6 — Ports definition of LDPC endcoder

3.8.2.3 Bit interleaver

The bit interleaver consists of parity interleaving followed by column twist interleaving. The input of the parity interleaver is denoted as λ, the output of the parity interleaver is denoted as u and the output of the column twist interleaver is denoted as v.

In the parity interleaving part, parity bits are interleaved as follows:

$$\mu_{K_{ldpc}+360\cdot t+s} = \lambda_{K_{ldpc}+Q_{ldpc}\cdot s+t} \qquad 0 \le s < 360, 0 \le t < Q_{ldpc} \tag{3.50}$$

where Q_{ldpc} is defined in Chapter 1. The value of Q_{ldpc} changes with different code rates and different length of LDPC codes.

In the column twist interleaving part, the data bits u_i from the parity interleaver are serially written into the column-twist interleaver column-wise and serially read out row-wise as shown in Fig. 3.22. Actually, the write start position of each column is twisted by t_c which depends on different code rates. This interleaver is described by the following:

The input bit u_i with index i, for $0 \le i < N_{ldpc}$, is written to column c_i, row r_i of the interleaver,where:

$$c_i = i \quad div \quad N_r \qquad r_i = i + t_c \quad mod \quad N_r$$

The output bit v_j with index j, for $0 \le i < N_{ldpc}$, is read from row r_j, column c_j, where

$$r_j = j \quad div \quad N_c \qquad c_j = j \quad mod \quad N_c$$

where N_r means the number of rows of the interleaver block and N_c means the number of columns of the interleaver, which satisfy this constraint $N_c \cdot N_r = N_{ldpc}$. Both of them are constant values for different code rates, that are given in Fig. 3.23.

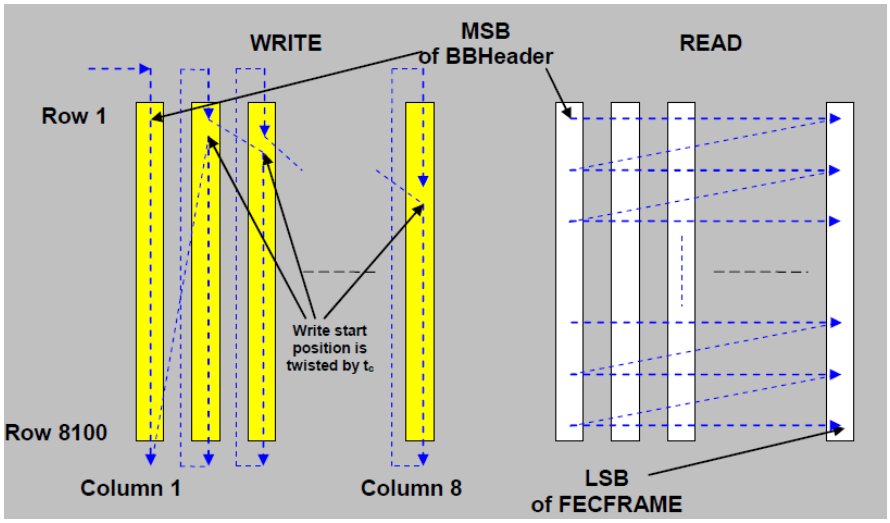


Figure 3.22 — Column twist interleaving scheme for 16K LDPC code and 16-QAM constellation

The parity interleaver is in the form of the parity bits re-order as detailed in the Chapter 1 to make the parity check matrix quasi-cyclic. According to the encoding algorithm with the parallelism level of 360, the 360 parity bits in any data bus of the parity RAM have been interleaved in the form of equ. (3.50). Therefore, no additional hardware resource is necessary

Modulation	Columns N_c	N_{ldpc}	Twisting parameter t_c															
			Col. 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16-QAM	8	64 800	0	0	2	4	4	5	7	7	-	-	-	-	-	-	-	-
		16 200	0	0	0	1	7	20	20	21	-	-	-	-	-	-	-	-
64-QAM	12	64 800	0	0	2	2	3	4	4	5	5	7	8	9	-	-	-	-
		16 200	0	0	0	2	2	2	3	3	3	6	7	7	-	-	-	-
256-QAM	16	64 800	0	2	2	2	2	3	7	15	16	20	22	22	27	27	28	32
	8	16 200	0	0	0	1	7	20	20	21	-	-	-	-	-	-	-	-

Figure 3.23 — Column twisting parameters

to do the parity interleaving. Indeed, a serial output of the data bus in the parity RAM can provide the parity interleaved bits.

The column twist interleaver is mainly composed of an interleaver RAM and the RAM control logics. The RAM is a real two ports RAM, which supports reading and writing at the same time in the case of no address conflict. The size of the RAM is 129600x1, which is divided into two parts. The first 64800 addresses belong to the first part and the later 64800 addresses belong to the second part. These two parts of RAM work as two separate RAMs to take charge of the reading and writing separately. The control part generates the reading and writing addresses based on the column twist definition in the standard. The corresponding architecture is illustrated in Fig. 3.24 and the ports definitions are listed in Table 3.7.

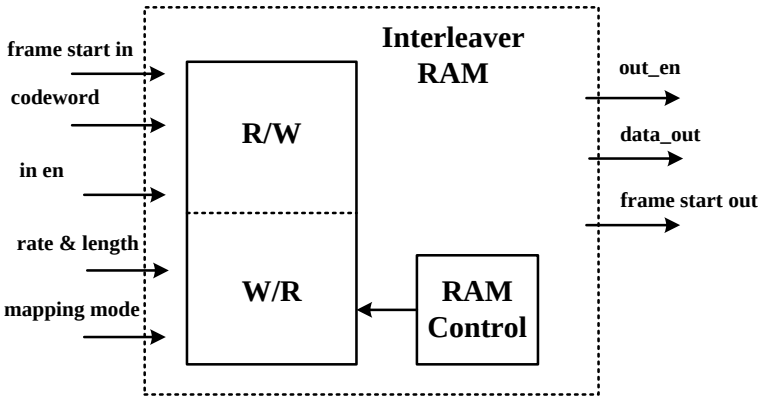


Figure 3.24 — Architecture of the interleaver

3.8.2.4 Mapper

The flexible mapper is composed of the traditional mapper and the rotation computation, that has been detailed in Chapter 2. 10 bits are assigned for each component of the symbol (1 bit for sign, 1 bit for the integral part and 8 bits for the fractional part). The 8 bits can provide a precision of 1/256 in the case of rotated 256-QAM. In order to support the timing constraint of the proposed demapper, the channel input symbol lasts for ten clock periods.

Signal	I/O	Quantization	Description
codelen	I	1	0: 16K LDPC 1:64K LDPC
mapping_mode	I	2	0: QPSK 1:16-QAM 2: 64-QAM 3: 256-QAM
frame_str_in	I	1	frame start lasts one clock
codeword_in	I	1	codeword bit stream
in_en	I	1	enable of the coded bit stream
frame_str_out	O	1	frame start lasts one clock
cdw_out	O	1	codeword
out_en	O	1	enable of the codeword

Table 3.7 — Ports definition of column twist interleaver

The corresponding architecture is illustrated in Fig. 3.25 and the ports definitions are listed in Table 3.8.

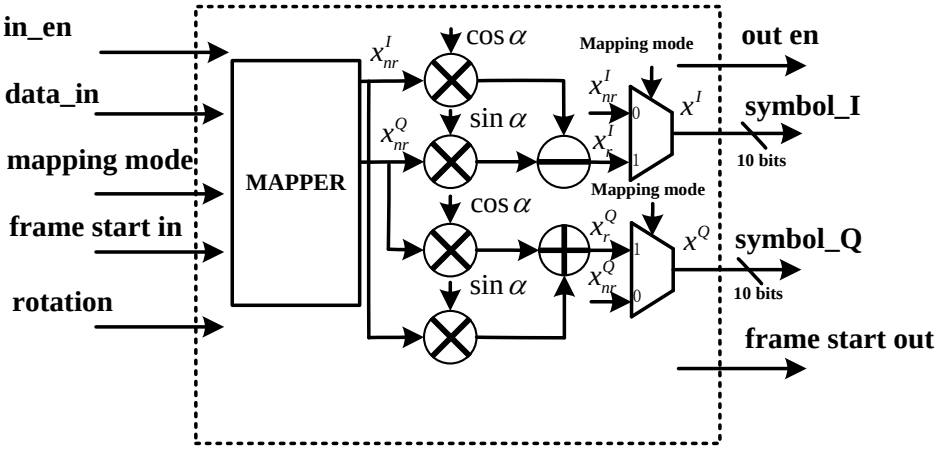


Figure 3.25 — Architecture of the mapper

The FPGA logic synthesis results of the transmitter are detailed in Table 3.9.

3.8.3 Channel emulator

The designed channel emulator supports three different kinds of channels: AWGN, Rayleigh fading without erasure and Rayleigh fading with erasures. The corresponding mathematical models have been detailed in Chapter 1. The channel emulator is obtained based on four Gaussian generators by using the Wallace method [39]. The channel attenuation is based on equ. (3.51) and the in-phase and quadrature components of the output symbol are obtained based on equ. (3.52) and equ. (3.53).

Signal	I/O	Quantization	Description
mapping_mode	I	2	0: QPSK 1:16-QAM 2: 64-QAM 3: 256-QAM
rotation	I	1	0: non-rotated 1: rotated
frame_str_in	I	1	frame start lasts one clock
data_in	I	8	codeword bit stream
in_en	I	1	enable of the coded bit stream
frame_str_out	O	1	frame start lasts one clock
symbol_I	O	10	in-phase of mapped symbol
symbol_Q	O	10	quadrature of mapped symbol
out_en	O	1	enable of the codeword

Table 3.8 — Ports definition of the mapper

Number of Slice Registers	2010 out of 207360 (0%)
Number of Slice LUTS	3235 out of 207360 (1%)
Number of RAM	5 out of 288 (1%)
Frequency	254Mhz

Table 3.9 — Device occupation after logic synthesis for the transmitter

$$H_{attenuation} = \sqrt{V1^2_{Gaussian} + V2^2_{Gaussian}} \tag{3.51}$$

$$Y_I = X_I \cdot H_{attenuation} + V3^2_{Gaussian} \tag{3.52}$$

$$Y_Q = X_Q \cdot H_{attenuation} + V4^2_{Gaussian} \tag{3.53}$$

The SNR ratio is configurable by the input signal sigma, that denotes the square root of the variation of the Gaussian noise. The percentage of the erasure is also configurable by the input signal Pe. If the signal is labelled as erased, the output symbol is forced to zero. The main blocks are illustrated in Fig. 3.26 and the ports definitions are listed in Table 3.10. This channel emulator is designed by one PhD candidate in our lab [54]. The quantization of the output symbols is fixed. So the precision of the channel output hence the following receiver is limited by the number of the quantization bits. The FPGA logic synthesis results of the channel emulator are shown in Table 3.11.

3.8.4 Receiver elements

The receiver consists of the equalizer, demapper, bit de-interleaver and LDPC decoder.

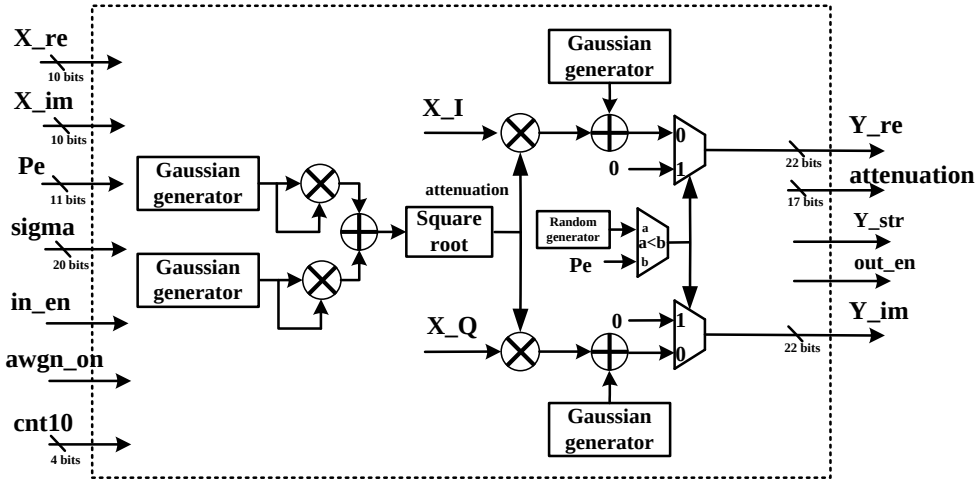


Figure 3.26 — Architecture of the channel emulator

Signal	I/O	Quantization	Description
awgn_on	I	1	0: Fading 1: AWGN
in_en	I	1	enable of the coded bit stream
cnt10	I	4	counter for symbol duration
sigma	I	20	sigma for Gaussian noise
Pe	I	11	Percentage of erasures
rho	I	17	channel attenuation
X_re	I	10	in-phase of mapped symbol
X_im	I	10	quadrature of mapped symbol
Y_str	O	1	channel output start
Y_re	O	22	in-phase of mapped symbol
Y_im	O	22	quadrature of mapped symbol
out_en	O	1	enable of the codeword

Table 3.10 — Ports definition of the channel emulator

Number of Slice Registers	4675 out of 207360 (2%)
Number of Slice LUTS	4082 out of 207360 (1%)
Number of RAM	4 out of 288 (1%)
Frequency	61Mhz

Table 3.11 — Device occupation after logic synthesis for the channel emulator

3.8.4.1 Equalizer

An equalization is assigned before the demapper to provide the equalized symbol and CSI information. In the proposed system, we assume that the equalizer knows perfect channel information. So the attenuation and sigma signals from the channel emulator are directly connected to the equalizer. The corresponding architecture is illustrated in Fig. 3.27 and the ports definitions are listed in Table 3.12.

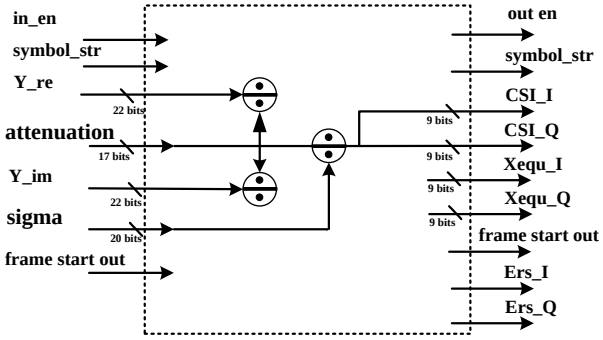


Figure 3.27 — Architecture of the equalizer

Signal	I/O	Quantization	Description
awgn_on	I	1	0: Fading 1: AWGN
in_en	I	1	enable of the coded bit stream
cnt10	I	4	counter for symbol duration
mapping_mode	I	2	0: QPSK 1:16-QAM 2: 64-QAM 3: 256-QAM
Y_re	I	22	in-phase of mapped symbol
Y_im	I	22	quadrature of mapped symbol
sigma	I	20	sigma for Gaussian noise
Pe	I	11	Percentage of erasures
rho	I	17	channel attenuation
Xequ_str	O	1	enable of the codeword
out_en	O	1	enable of the codeword
Xeq_I	O	9	in-phase of equalized symbol
Xeq_Q	O	9	quadrature of equalized symbol
CIS_I	O	9	in-phase of CSI
CSI_Q	O	9	quadrature of CSI
ERS_I	O	1	in-phase 0: not erased 1: erased
ERS_Q	O	1	quadrature 0: not erased 1: erased

Table 3.12 — Ports definition of the equalizer

3.8.4.2 Demapper

The architecture of the proposed demapper has been previously detailed in chapter 2. Importantly, the input complex symbol and according CSI last for ten system clocks. The ports definitions are listed in Table 3.13.

The input and output schedule in the case of QPSK is illustrated in Fig. 3.28. The delay of LLR generation for all the mapping modes is equal to 14 clock cycles and the LLRs belonging to each channel input symbol has an interval of 10 clock cycles.

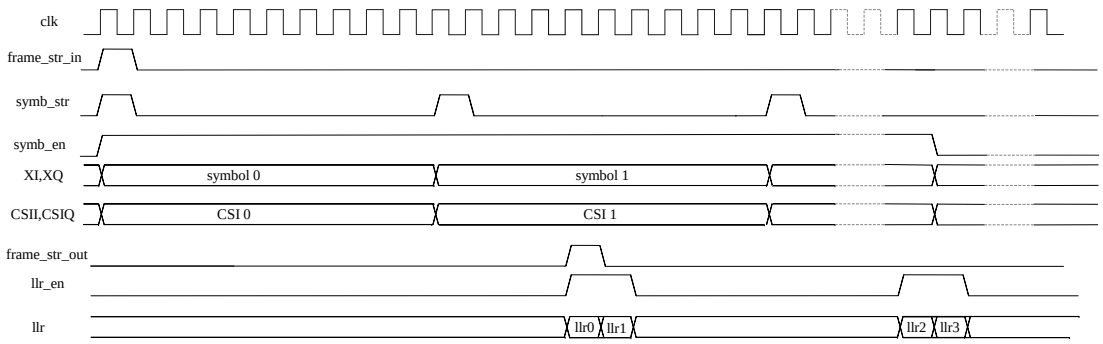


Figure 3.28 — IO schedule of the demapper

Signal	I/O	Quantization	Description
frame_str_in	I	1	start of the frame
symb_str	I	1	start of the input symbol
symb_en	I	1	enable of the coded bit stream
mapping_mode	I	2	0: QPSK 1:16-QAM 2: 64-QAM 3: 256-QAM
rotation	I	1	0: non-rotated 1: rotated
symb_I_in	I	9	in-phase of equalized symbol
symb_Q_in	I	9	quadrature of equalized symbol
CIS_I_in	I	9	in-phase of CSI
CSI_Q_in	I	9	quadrature of CSI
eras_I	I	1	symb_I_in 1: erased 0: not erased
eras_Q	I	1	symb_Q_in 1: erased 0: not erased
rame_str_out	O	1	start of the frame
llr_en	O	1	enable of llr
llr	O	6	llr

Table 3.13 — Ports definition of the demapper

3.8.4.3 Bit de-interleaver

The bit de-interleaver consists of the bit de-interleaver and parity de-interleaver. The parity de-interleaver has the same form of the parity bits re-order detailed in the Chapter 1, that is in order to make the parity check matrix quasi-cyclic. In order to reduce the memory size and the system delay, this part of the function is embedded in the I/O processor of the LDPC decoder.

The column twist de-interleaver shares the same architecture as the column twist interleaver. Only a little change of the address generation is necessary in the control part. The posts definitions are illustrated in Fig. 3.29.

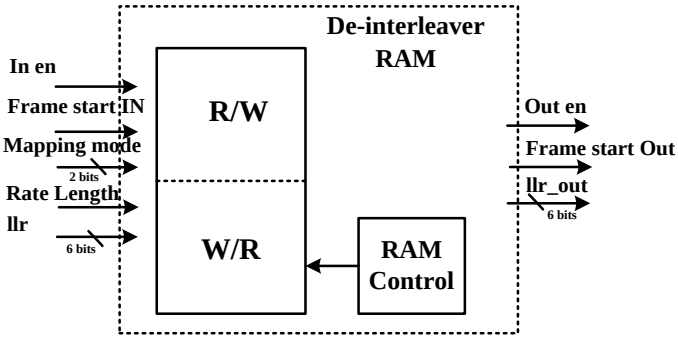


Figure 3.29 — Architecture of the de-interleaver

3.8.4.4 LDPC decoder

The architecture of the proposed VSS Min-Sum LDPC decoder has been detailed in the previous section. The corresponding ports definitions are listed in Table 3.14. The input and output timing is illustrated in Fig. 3.30.

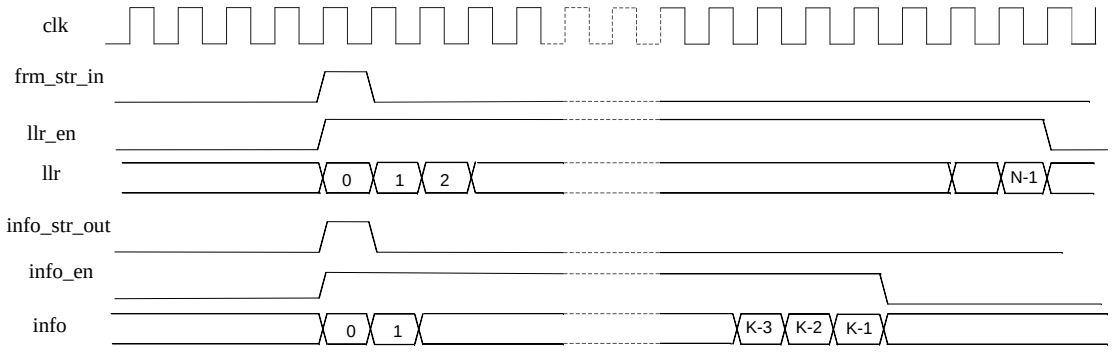


Figure 3.30 — IO timing of the proposed LDPC decoder

The FPGA logic synthesis results of the proposed receiver is shown in Table 3.15.

Signal	I/O	Quantization	Description
frame_str_in	I	1	start of the frame
rate	I	3	1: R1/2; 2: R3/5; 3: R2/3; 4: R3/4; 5: R4/5; 6: R5/6
mapping_mode	I	2	0: QPSK 1:16-QAM 2: 64-QAM 3: 256-QAM
MaxIter	I	5	Maximum iteration number
llr_en	I	1	enable of llr
llr_in	I	6	llr
rame_str_out	O	1	start of the frame
info_en	O	1	enable of decoded information
info	O	1	decoded information

Table 3.14 — Ports definition of the LDPC decoder

Number of Slice Registers	18029 out of 207360 (8%)
Number of Slice LUTS	41032 out of 207360 (19%)
Number of RAM	84 out of 288 (29%)
Frequency	113Mhz

Table 3.15 — Device occupation after logic synthesis for the receiver

3.8.5 System platform

The prototype was implemented onto one Xilinx FPGA chip on a logic emulation board (DN9000K10PCI) composed of 6 Xilinx Virtex 5 LX330 devices. With this board, appropriate communication controllers are available and can be added to the design in order to read/write various output/input memories from a host computer using a USB interface.

The system environment is illustrated in Fig. 3.32.

1. The whole system is implemented onto one FPGA device on DN9000K10PCI emulation board.

2. The host computer is connected to the board by PCI, it is in charge of setting up the various parameters by key board and getting the BER results from emulation board.

3. On the screen, the graphical user interface is in charge of display the configuration, the performance results and the corresponding performance curve.

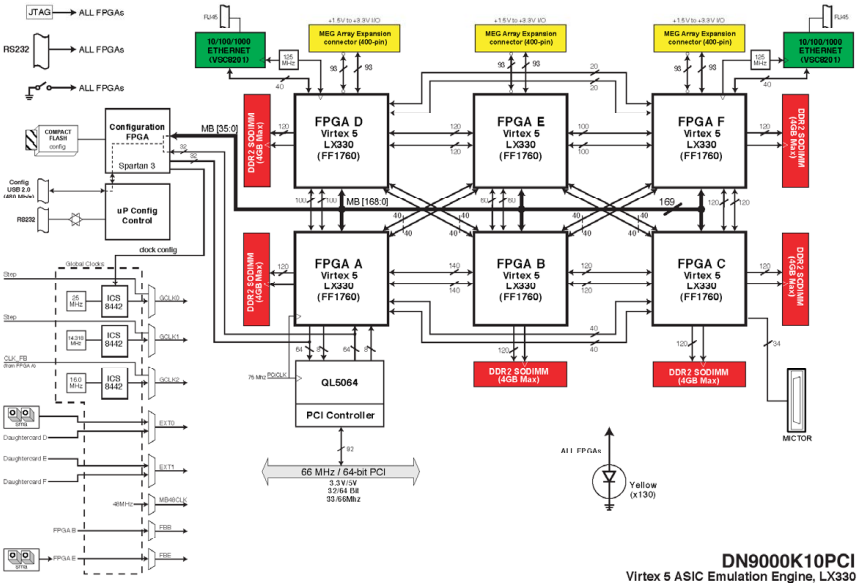


Figure 3.31 — Block diagrams of DN900K10PCI board

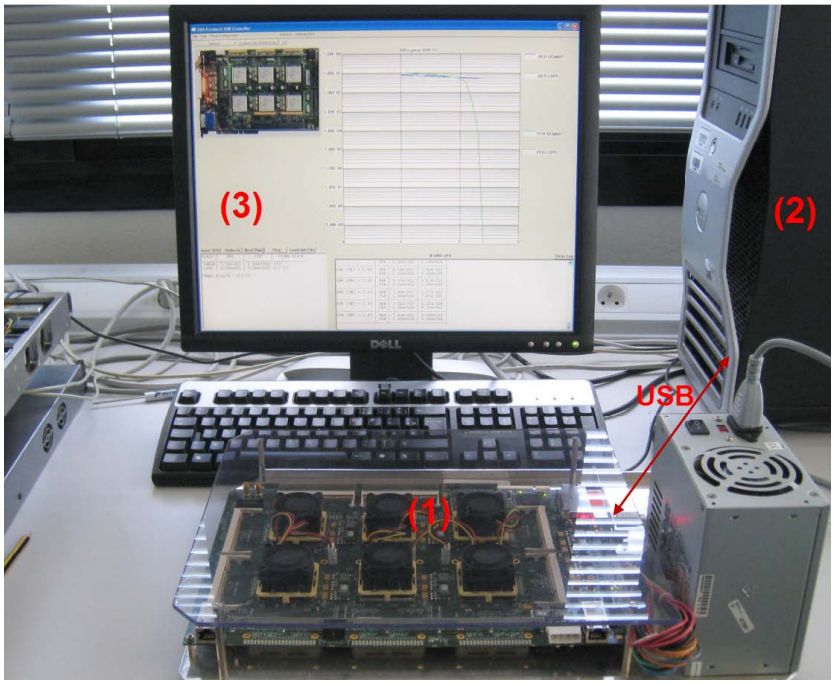


Figure 3.32 — Real development environment

3.8.6 Performance

The performance test of two kinds of LDPC decoders (64K Rare 3/4 and 64K Rate 4/5) are carried out for eight different modulation modes over AWGN and fading channel with and without erasures.

The measured performance in terms of BER over AWGN channel is illustrated in Fig. 3.33.

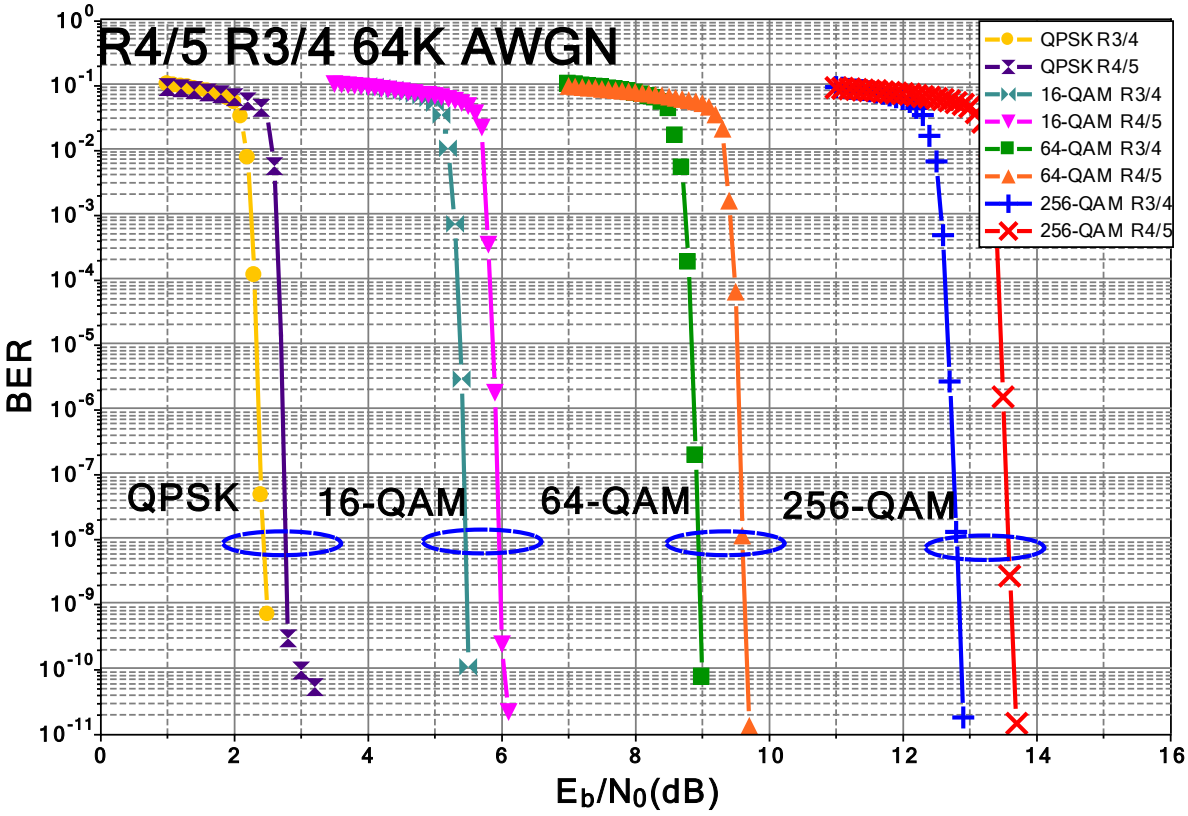


Figure 3.33 — The system performance over AWGN channel for 64K LDPC

The performance comparison between the prototype and fixed point simulation for 64K LDPC with a code rate of R=4/5, which is the FEC mode of Finnish broadcast system, over fading channel are shown in Fig. 3.34 and Fig. 3.35 for the non-rotated and rotated constellation. The performance comparison of the same FEC mode for fading channel with 15% erasures are shown in Fig. 3.36 and Fig. 3.37 for the non-rotated and rotated constellation, respectively.

From the performance results, we can observe that lower constellations provide more performance gain from the signal space diversity. For 64K LDPC code with a code rate of R=4/5, over fading channel, the performance gain from signal space diversity for QPSK is around 1.5 dB and for 16-QAM is around 0.5 dB at BER equals 10^e-6 while there is almost no gain achieved for 63-QAM and 256-QAM. For 64K LDPC code with a code rate of R=4/5, over fading channel with 15% erasures, the performance gain from signal space diversity for QPSK is around 11.5 dB, for 16-QAM is around 10 dB, for 64-QAM is around 9 dB and for 256-QAM is around 7 dB at BER equals 10^e-6.

The tested performance of this prototype is very close to the fixed point simulation results. It verifies the efficiency of the proposed demapper and LDPC decoder.

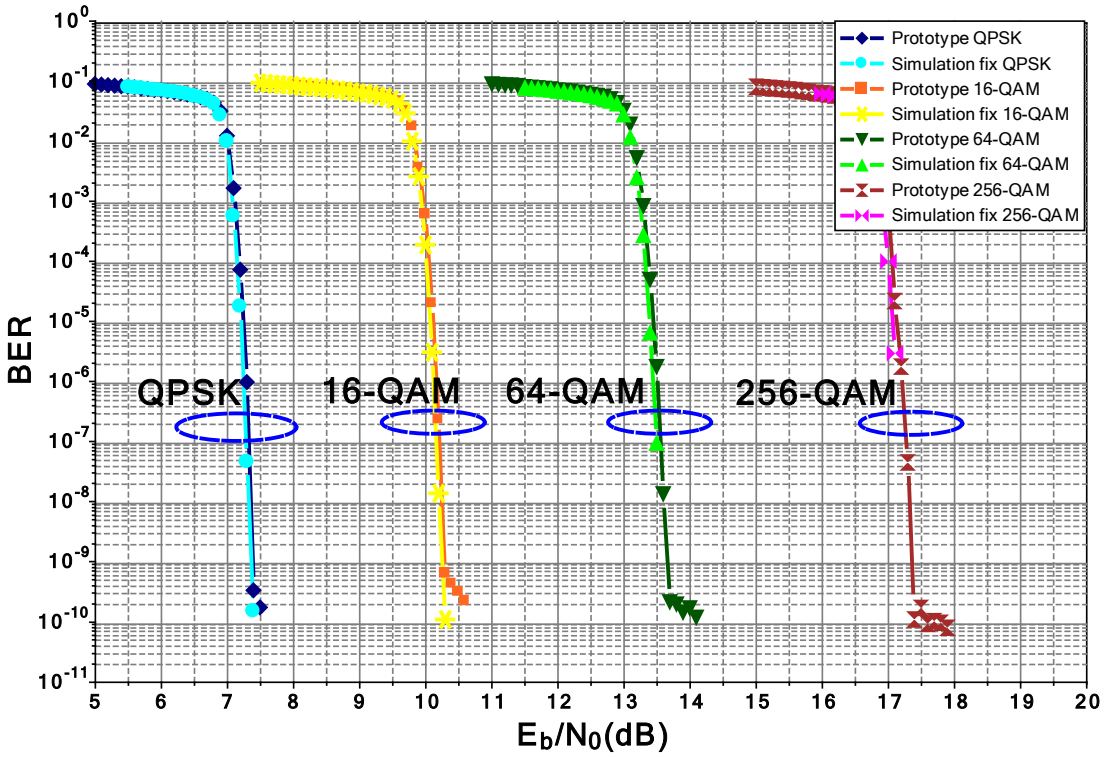


Figure 3.34 — The system performance over fading channel for 64K LDPC with a code rate of R=4/5 with non-rotated constellation

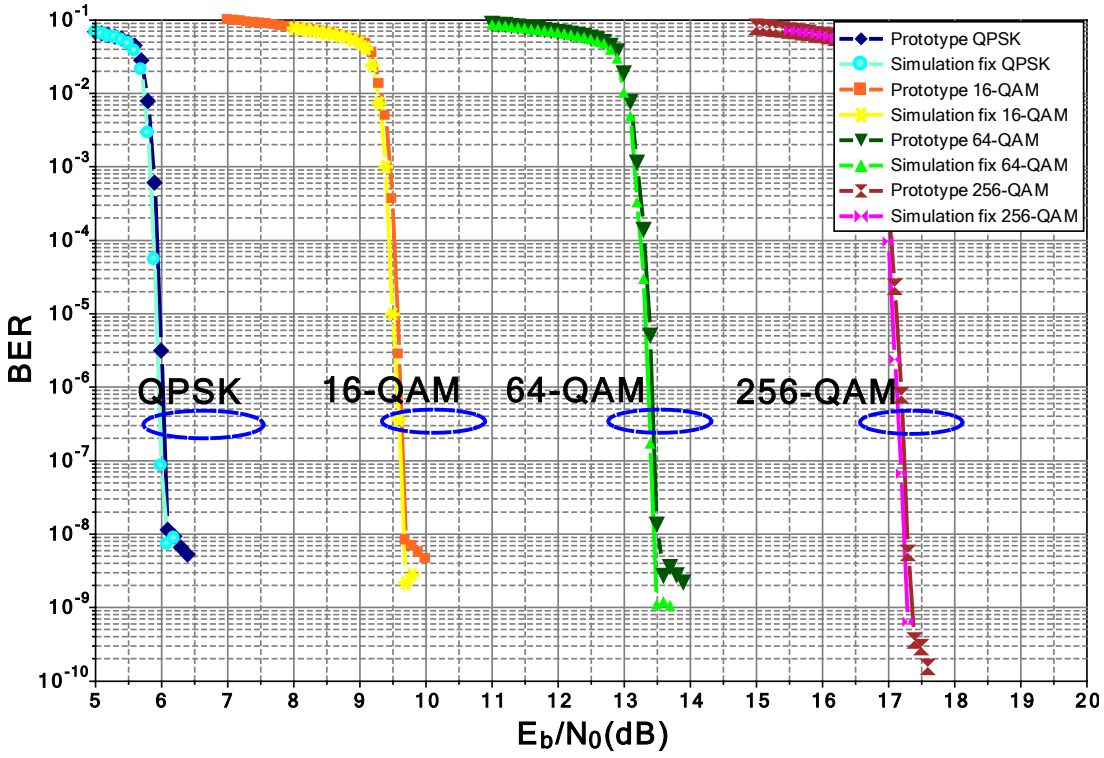


Figure 3.35 — The system performance over fading channel for 64K LDPC with a code rate of R=4/5 with rotated constellation

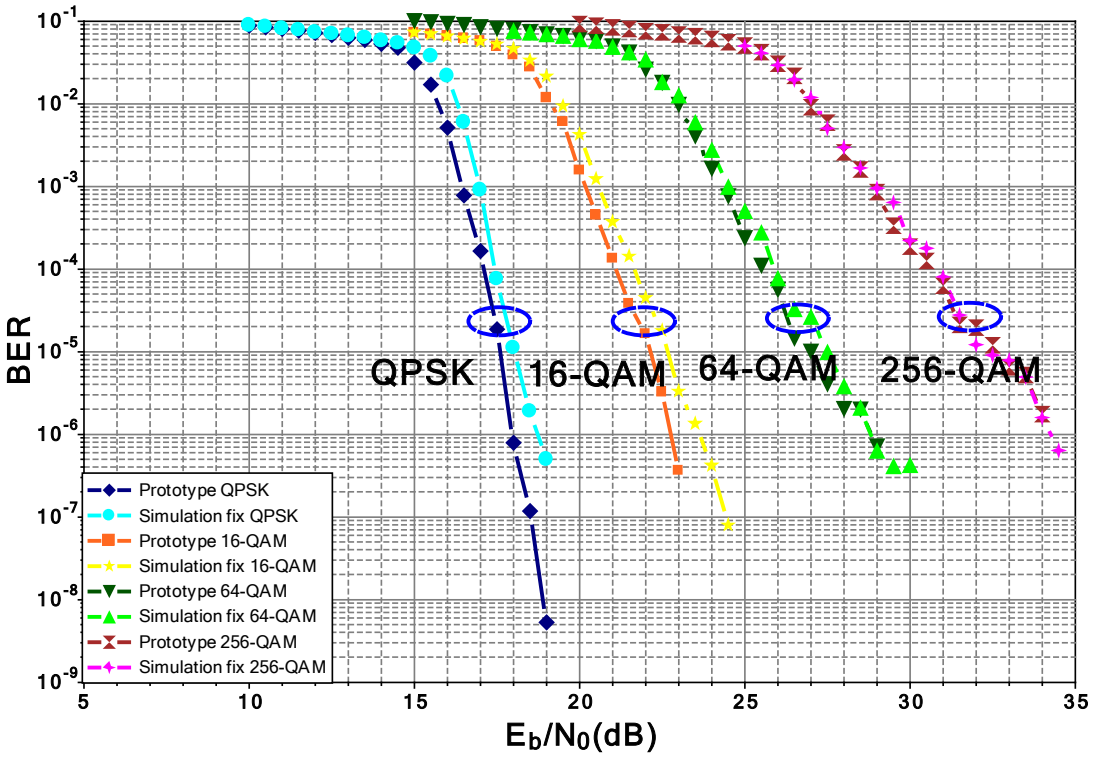


Figure 3.36 — The system performance over fading channel with 15% erasures for 64K LDPC with a code rate of R=4/5 with non-rotated constellation

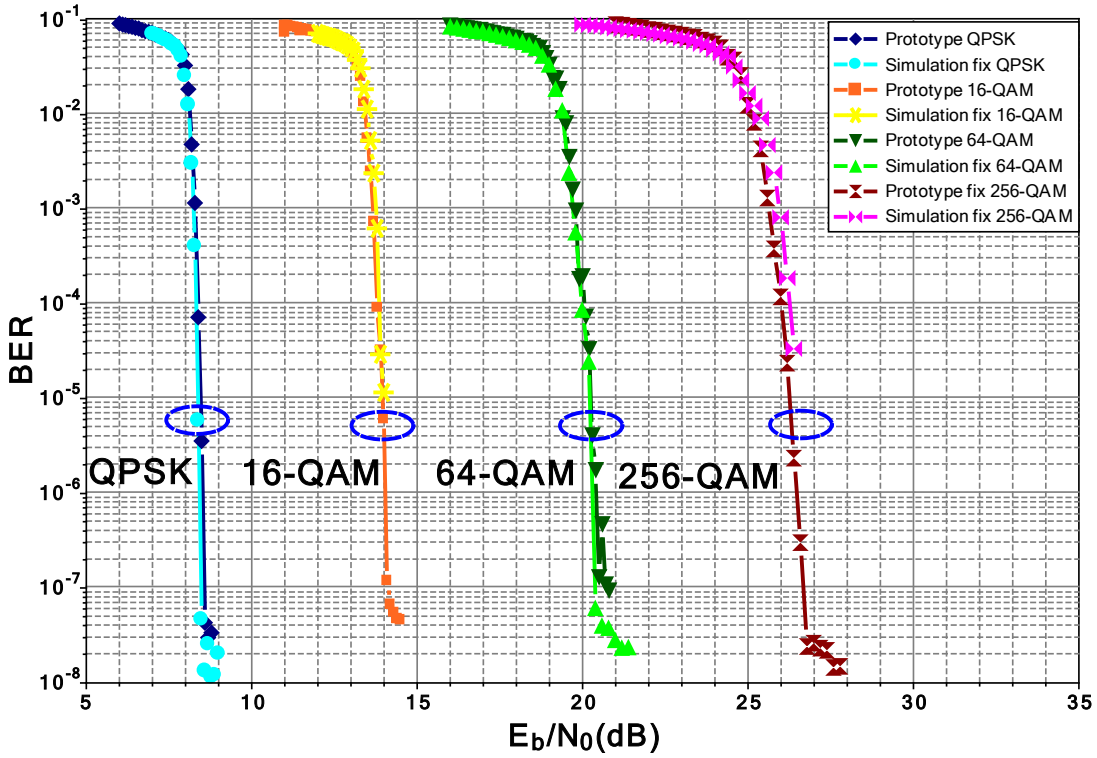


Figure 3.37 — The system performance over fading channel with 15% erasures for 64K LDPC with a code rate of R=4/5 with rotated constellation

3.9 Integration of the demapper and decoder in a complete DVB-T2 system

Our work is sponsored by an European project calle SME42, which is built up by TeamCast (a French company), SIDSA (a Spanish enterprise) and Telecom Bretagne, with the aim to perform the world's first experimentation of the brand new second generation standard for Digital Terrestrial TV. In the SME42 project TeamCast has developed a DVB-T2 modulator, channel emulator and a DVB-T2 demodulator. The proposed demapper and VSS Min-Sum LDPC decoder were also integrated into a real DVB-T2 receiver which is designed by TeamCast. A complete DVB-T2 transmission chain is set up with a DVB-T2 modulator, channel emulator and the receiver, in order to test the performance of the proposed architectures for the DVB-T2 system.

3.9.1 System platform

A complete DVB-T2 transmission chain provided by TeamCast is shown in Fig. 3.38.

1. Power4-T2: a DVB-T2 modulator supports complex modulation schemes.
2. Channel emulator provides configurable AWGN channel, fading channel with and without erasures plus the Doppler shift.
3. Demod4-T2: a DVB-T2 reference demodulator provides a means to test the system performance for basic system configuration.
4. A laptop is in charge of the configuration of channel emulator, such as SNR value; the channel type; number of multipath; the delay, the amplitude and the arriving angle of each path and the value of Doppler shift. The computer is connected to the channel emulator by Ethernet.
5. A comprehensive software Controlcast is installed on this computer, which is in charge of the configuration of the modulator and demodulator and also in charge of display the system configuration, measurement traces, signal spectrum and constellation. The connection between the computer and the demodulator is done by Ethernet protocol.

3.9.2 Prototype and performance

The demodulator board of TeamCast includes 4 FPGA chips with model Virtex5 LX110. The first three devices are in charge of the base band converter, FFT, synchronization, equalizer, frequency de-interleaver, time de-interleaver, etc. TeamCast provides the bit streams for the

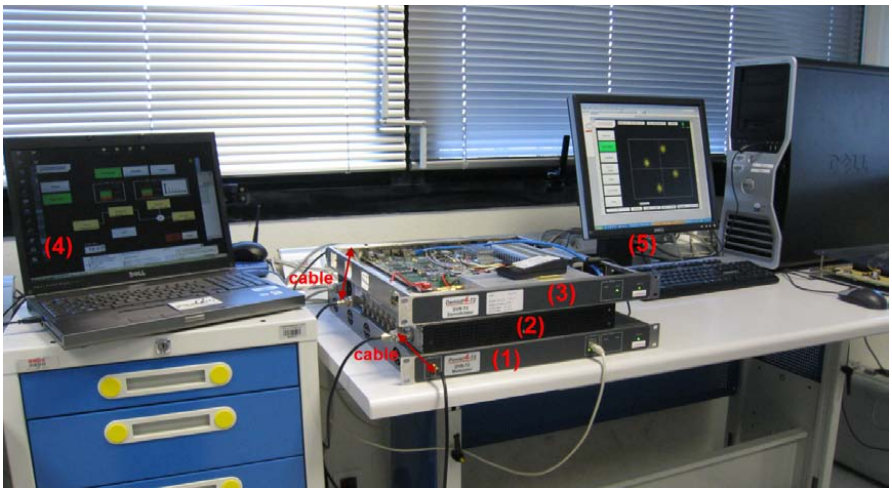


Figure 3.38 — Real development environment

(a) System configuration

(b) Constellation of rotated 256-QAM

Figure 3.39 — Plane of Controlcast

first three chips, but the access of the first three chips are denied. The FEC part is implemented in the fourth FPGA chip. The last chip takes charge of the FEC, which including cell de-interleaver, cyclic removal model, the proposed demapper, de-bit-interleaver, memory RAM, proposed VSS Min-Sum LDPC decoder, make TS and BER calculation, as shown in Fig. 3.40.

The FPGA logic synthesis results of the integration work of the fourth chip are shown in Table 3.16.

Performance test over AWGN channel is carried out for three different FEC modes and for QPSK, 16-QAM and 64-QAM with the system level configuration shown in Fig.3.39.a. The maximum iteration number for LDPC decoder is 31. The corresponding performances are illustrated from Fig. 3.41 and Fig. 3.43, which achieves the expected performance. The integration work for fading channel is still under going.

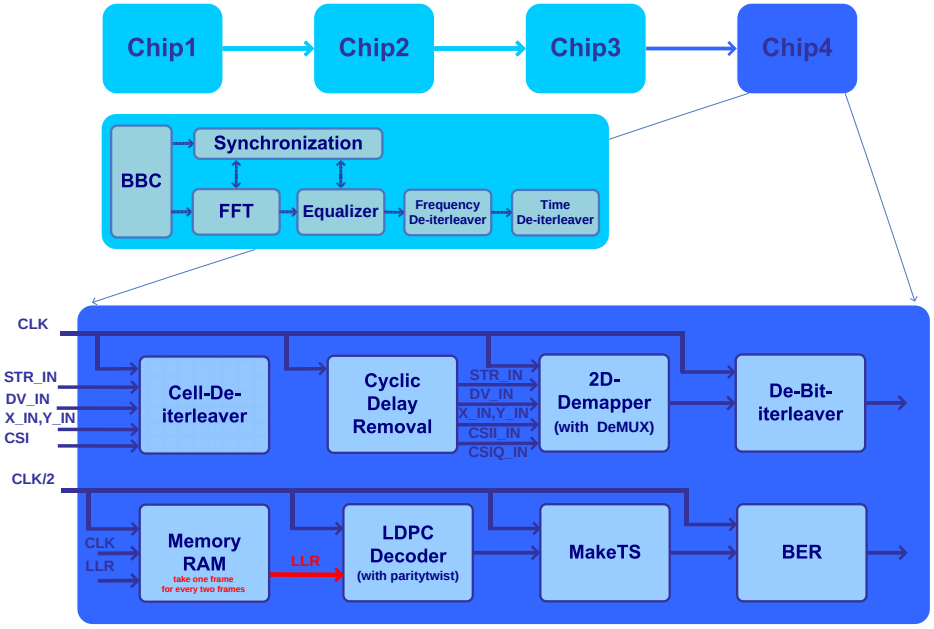


Figure 3.40 — Real development environment

Number of Slice Registers	17916 out of 69120 (25%)
Number of Slice LUTS	44544 out of 69120 (64%)
Number of Block RAM/FIFO	123 out of 128 (96%)
Frequency	55 Mhz

Table 3.16 — Device occupation after logic synthesis for the FEC part of DVB-T2 demodulator

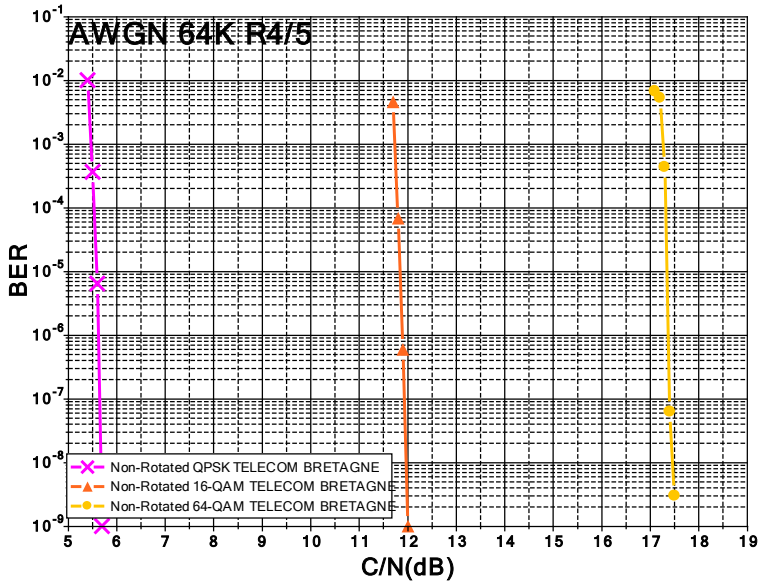


Figure 3.41 — Performance comparison over AWGN for 64K LDPC with a code rate of R=4/5

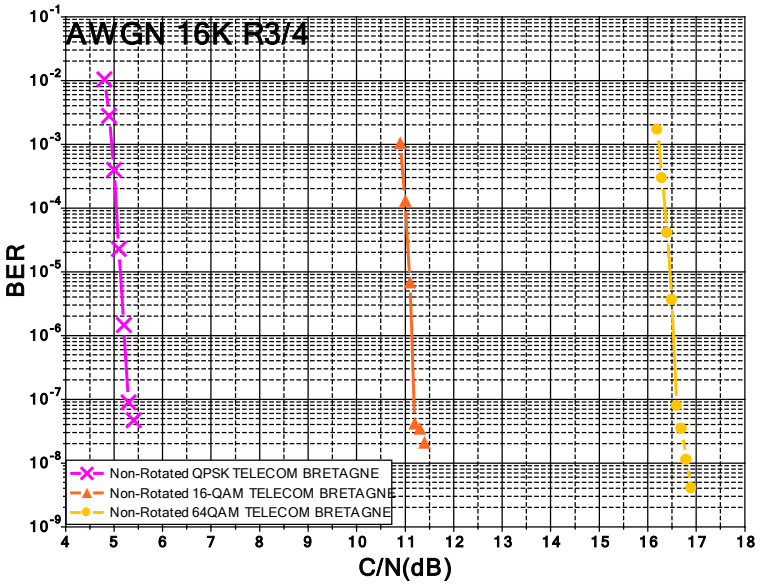


Figure 3.42 — Performance comparison over AWGN for 16K LDPC with a code rate of R=3/4

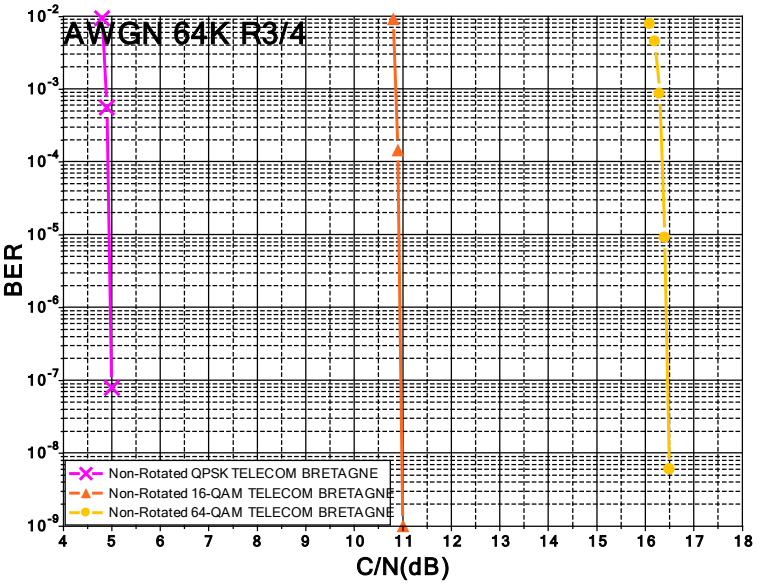


Figure 3.43 — Performance comparison over AWGN for 64K LDPC with a code rate of R=3/4

3.10 Conclusion

The main novelty of this chapter is the proposal of a *vertical shuffled* Min-Sum decoding algorithm and the architecture of the corresponding decoder for the DVB-T2 standard. It facilitates message exchange between the demapper and the decoder for an iterative BICM receiver. To the best of our knowledge, this is the first proposal of a *vertical shuffled* Mim-Sum decoder for DVB-T2. These results were published in [55] and [56].

The proposed *vertical shuffled Min-Sum* decoding algorithm, which uses the latest check node information to speed up the convergence, shows almost the same performance when compared to the *horizontal shuffled* Min-Sum decoding algorithm. The message update conflict problem due to double diagonal sub-matrix and the memory access conflict problem due to pipeline architecture are the two main critical problems for the design of a LDPC decoder of the DVB-T2 standard. Conflicts due to double diagonal sub-matrix is solved by matrix split and a little modification of decoding algorithm. The conflicts due to pipeline has been solved by an elaborate scheduling coupled with sub-matrix split and column re-order if necessary.

The implementation of the proposed LDPC decoder has been carried out onto a Xilinx Virtex XC5VLX330 device. It enables a maximum throughput of 80 Mbits/s for 64K LDPC code with a code rate of R=4/5.

The proposed demapper and VSS Min-Sum LDPC decoder have also been validated by two prototypes of DVB-T2 system. A simplified DVB-T2 system is implemented onto a Virtex 5 LX330 FPGA device. The measured performance is quasi-identical to the fixed point simulation results over AWGN channel and fading channel with and without erasures. Furthermore, the demapper and decoder have also been integrated into a real reference DVB-T2 demodulator. The demodulator with proposed demapper and decoder has more or less the close performance with TeamCast's demodulator for different code rates in different constellation modes, which considerably validates the efficiency of the proposed algorithm and architecture of demapper and decoder.

CHAPTER 4

Design and implementation of an iterative BICM receiver for DVB-T2

The iterative BICM scheme with SSD can provide more performance gain when compared to the non-iterative BICM scheme [57]. Therefore, it is adopted in the DVB-T2 standard and its implementation was encouraged in the implementation guideline of DVB-T2.

The iterative receiver shows attractive performance however it demands high computational complexity. Moreover, the conventional frame-by-frame processing mode causes important additional delay of the message passing between the demapper and decoder due to the block interleaving and de-interleaving process and the state-of-art LDPC decoding. The target of this work is to design a low complexity low latency iterative receiver so that it can completely meet the practical requirements and constraints.

In the following chapter, a novel iterative receiver is proposed. This novel iterative receiver is based on a vertical shuffled LDPC decoding algorithm, which has been presented in Chapter 3. An additional vertical shuffled idea is also applied for the message exchange between the demapper and decoder. This information exchange is performed on sub-frame other than the whole frame. The size of the sub-frame is depending on the desired throughput and code modulation constraints. The proposed receiver is implemented onto an FPGA board. The prototype results demonstrate the efficiency of the proposal. To the best of our knowledge,

this is the first published iterative receiver design for a BICM system dedicated to the DVB-T2 standard.

4.1 Algorithm design for an iterative BICM receiver

The iterative demodulator structure is illustrated in Fig. 4.1. The *a priori* information $P(v^i;I)$, denoted as $P_{apriori}(x_t)$ in this chapter, is computed by combining the interleaved *extrinsic* information $P(c^i;O)$ from the LDPC decoder related to $(m-1)$ bits of the 2^m-QAM constellation.

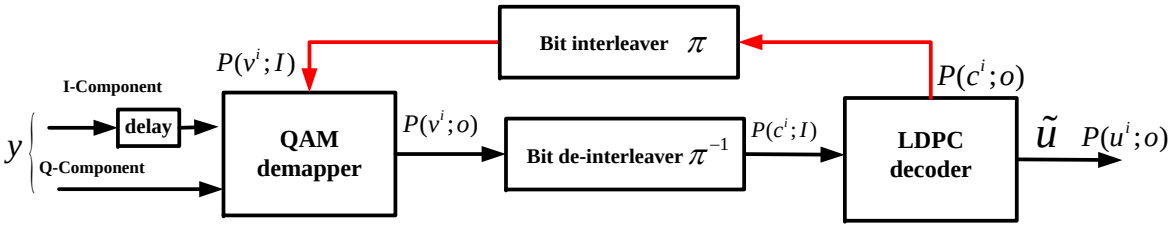


Figure 4.1 — Iterative BICM structure in the DVB-T2 standard

Actually, there are two levels of iteration in an iterative BICM system. One is the local iteration inside the LDPC decoder and the other one is the global iteration between the demapper and decoder. In this chapter, we assume that the LDPC decoder provides the *extrinsic* information to the demapper just after one local iteration, which provides the best performance of the iterative BICM receiver [58]. In other words, one local iteration inside the decoder corresponds to one global iteration between the demapper and decoder.

4.1.1 Demapping algorithm for an iterative BICM receiver

If an iterative demapping is considered, the LLR of bit v_t^i should be calculated in the light of *a priori* knowledge, which is obtained from the LDPC decoder during the previous iteration. Thanks to this knowledge, we should no longer assume that all the constellation bits have the same probability of being 0 or 1. Therefore, an updated LLR computation is provided in equ. (4.1).

$$LLR(v_t^i) = log\left(\frac{\sum\limits_{x_t\in\chi_0^i}\frac{P_{apriori}(x_t)}{\sigma\sqrt{2\pi}}\cdot\exp-\frac{\left|y_t^I-\rho_t x_t^I\right|^2+\left|y_t^Q-\rho_t x_t^Q\right|^2}{2\sigma^2}}{\sum\limits_{x_t\in\chi_1^i}\frac{P_{apriori}(x_t)}{\sigma\sqrt{2\pi}}\cdot\exp-\frac{\left|y_t^I-\rho_t x_t^I\right|^2+\left|y_t^Q-\rho_t x_t^Q\right|^2}{2\sigma^2}}\right) \tag{4.1}$$

where

$$P_{apriori}(x_t) = \prod_{k\neq i;k=0}^{m-1} Pr(b_k), \quad where \quad (b_{m-1}, b_{m-2}, ..., b_0) = x_t \tag{4.2}$$

In the case of iterative process, $P_{apriori}(x_t)$ value for each symbol in the signal space panel for LLR computation for certain bit b_i is computed from *extrinsic* messages provided by the LDPC decoder. It is expressed as a function of the product of probabilities of $(m-1)$ bits (excluding the correspond bit b_i) taking the value 0 or 1. The value of other $(m-1)$ bits are defined by the value of the constellation symbol ($(b_{m-1}, b_{m-2}, ..., b_0) = x_t$).

These probabilities are estimated by the previous iteration of LDPC decoding as equ. (4.3) in probability domain and equ. (4.4) in log domain. The interleaver function and de-interleaver function are defined as π and π^{-1}, respectively.

$$Pr(b_k) = \frac{P_{aposteriori}\left(\pi^{-1}(b_k)\right)}{P_{intrinsic}\left(\pi^{-1}(b_k)\right)} \tag{4.3}$$

$$LLR_{ext}(b_k) = LLR_{aposteriori}\left(\pi^{-1}(b_k)\right) - LLR_{intrinsic}\left(\pi^{-1}(b_k)\right) \tag{4.4}$$

Then the *a posteriori* information for mapping bit b_k is obtained from the $LLR_{ext}(b_k)$ by:

$$Pr(b_k = 0) = \frac{1}{1 + e^{LLR_{ext}(b_k)}} \tag{4.5}$$

$$Pr(b_k = 1) = 1 - Pr(b_k = 0) \tag{4.6}$$

In order to simplify the computational complexity of the demapping algorithm for an iterative process, we divide all $P_{apriori}(x_t)$ by a fixed factor, as described in equ. (4.7). Note that this division has no effect on the BER performance.

$$f_{apriori}(x_t = \underbrace{1111}_{m} | b_i) = \prod_{k=0}^{m} Pr(b_k = 1) \tag{4.7}$$

As a consequence, the computation of the LLR based on Max-Log approximation becomes as:

$$\begin{aligned} LLR(b_k) = &\min_{x_t \in \chi_0^i} \left(-\frac{\left|y_t^I - \rho_t x_t^I\right|^2 + \left|y_t^Q - \rho_t x_t^Q\right|^2}{2\sigma^2} + \sum_{j=0; j\neq k; b_j=0}^{m-1} LLR_{ext}(b_j) \right) \\ &- \min_{x_t \in \chi_1^i} \left(-\frac{\left|y_t^I - \rho_t x_t^I\right|^2 + \left|y_t^Q - \rho_t x_t^Q\right|^2}{2\sigma^2} + \sum_{j=0; j\neq k; b_j=0}^{m-1} LLR_{ext}(b_j) \right) \end{aligned} \tag{4.8}$$

where the summation of LLR_{ext} over j spans all bits b_k that are equal to zero excluding itself.

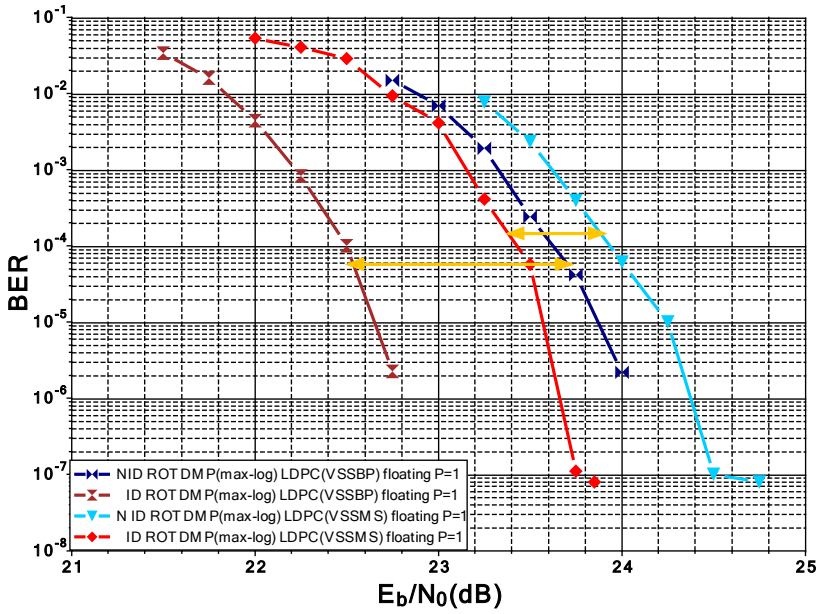


Figure 4.2 — Performance comparison for 64K LDPC with a code rate of R=4/5 over fading channel with 15% erasures and 256-QAM constellation

4.1.2 Decoding algorithm for an iterative BICM receiver

For the targeted iterative receiver, the vertical shuffled LDPC decoding algorithm is more efficient than the traditional horizontal layered decoding algorithm. A detailed description of VSS Min-Sum algorithms has been provided in Chapter 3. The proposed VSS Min-Sum decoding algorithm introduces only a small penalty in terms of performance with respect to VSS BP while greatly reducing decoding complexity. However, in the context of an iterative receiver, the VSS Min-Sum introduces an additional penalty and reduces the expected performance gain. In the simulation case shown in Fig. 4.2, VSS Min-Sum losts 0.6dB performance gain at BER=10^{-6} when compared to VSS BP decoding algorithm.

In fact, in order to increase the performance gain of a Min-Sum based iterative receiver, a decoding algorithm with a higher accuracy is necessary. Consequently, we propose a Min-Sum-3 or denoted as VSS MS3 algorithm, that is described in Algorithm 9. The difference between the VSS MS and VSS MS3 is that the 3rd minimum value and the corresponding index are updated and saved. This leads to an increased accuracy for the second minimum value, therefore increases the performance.

Simulation of different decoding algorithms including BP, Min-Sum and Min-Sum-3 with different parallelism levels have been carried out to estimate the performance gain of iterative receiver over fading channel with 15% erasures.

The required SNR values for different decoding algorithms at 10 e^{-4} and 10 e^{-6} are listed. The corresponding gains from iterative processing (ID) to non-iterative processing

	BP(P=1) floating	BP(P=90) floating	MS(P=1) floating	MS(P=90) floating	MS3(P=1) floating	MS3(P=90) floating	MS(P=90) fixed	MS3(P=90) fixed
NID @10^{-4}	6.90 dB	6.90 dB	7.00 dB	7.05 dB	7.00 dB	7.05 dB	7.15 dB	7.10 dB
ID @10^{-4}	6.15 dB	6.15 dB	6.35 dB	6.43 dB	6.20 dB	6.25 dB	6.70 dB	6.48 dB
Gain @10^{-4}	0.75 dB	0.75 dB	0.65 dB	0.62 dB	0.80 dB	0.75 dB	0.45 dB	0.62 dB
NID @10^{-6}	7.06 dB	7.06 dB	7.18 dB	7.22 dB	7.20 dB	7.20 dB	7.25 dB	7.25 dB
ID @10^{-6}	6.25 dB	6.25 dB	6.50 dB	6.55 dB	6.35 dB	6.35 dB	6.80 dB	6.58 dB
Gain @10^{-6}	0.81 dB	0.81 dB	0.68 dB	0.67 dB	0.85 dB	0.85 dB	0.45 dB	0.68 dB

Table 4.1 — Performance comparison for 64K LDPC with a code rate of R=3/4 and QPSK constellation

(NID) are also listed.

	BP(P=1) floating	BP(P=90) floating	MS(P=1) floating	MS(P=90) floating	MS3(P=1) floating	MS3(P=90) floating	MS(P=90) fixed	MS3(P=90) fixed
NID @10^{-4}	7.94 dB	7.94 dB	8.06 dB	8.08 dB	8.05 dB	8.06 dB	8.30 dB	8.18 dB
ID @10^{-4}	7.22 dB	7.24 dB	7.60 dB	7.62 dB	7.26 dB	7.30 dB	7.95 dB	8.65 dB
Gain @10^{-4}	0.72 dB	0.70 dB	0.46 dB	0.46 dB	0.79 dB	0.76 dB	0.45 dB	0.53 dB
NID @10^{-6}	8.05 dB	8.04 dB	8.25 dB	8.20 dB	8.16 dB	8.18 dB	8.35 dB	8.26 dB
ID @10^{-6}	7.35 dB	7.35 dB	7.66 dB	7.70 dB	7.35 dB	7.36 dB	8.05 dB	8.76 dB
Gain @10^{-6}	0.70 dB	0.69 dB	0.59 dB	0.50 dB	0.81 dB	0.82 dB	0.30 dB	0.50 dB

Table 4.2 — Performance comparison for 64K LDPC with a code rate of R=3/4 and QPSK constellation

Serial decoding denoted as $P = 1$ has no message update conflict problem hence provides the best performance. Thanks to the proposed solution for message update conflict problems detailed in Chapter 3, the parallel decoding algorithm with a parallelism level of 90 achieves almost the same performance gain as serial decoding both for BP, Min-Sum and Min-Sum-3 decoding algorithm.

	BP(P=1) floating	BP(P=90) floating	MS(P=1) floating	MS(P=90) floating	MS3(P=1) floating	MS3(P=90) floating	MS(P=90) fixed	MS3(P=90) fixed
NID @10^{-4}	20.40 dB	20.40 dB	20.57 dB	20.77 dB	20.40 dB	20.60 dB	21.70 dB	20.60 dB
ID @10^{-4}	19.50 dB	19.50 dB	20.18 dB	20.37 dB	19.82 dB	19.83 dB	21.10 dB	21.35 dB
Gain @10^{-4}	0.90 dB	0.90 dB	0.39 dB	0.40 dB	0.58 dB	0.77 dB	0.60 dB	0.75 dB
NID @10^{-6}	20.60 dB	20.60 dB	20.62 dB	20.95 dB	20.59 dB	20.82 dB	21.82 dB	20.80 dB
ID @10^{-6}	19.70 dB	19.70 dB	20.30 dB	20.57 dB	19.92 dB	19.93 dB	21.32 dB	21.55 dB
Gain @10^{-6}	0.90 dB	0.90 dB	0.32 dB	0.28 dB	0.67 dB	0.89 dB	0.60 dB	0.75 dB

Table 4.3 — Performance comparison for 64K LDPC with a code rate of R=3/4 and 256-QAM constellation

Note that the number of quantization bits for channel inputs, LLR and check node information in the fixed point simulation are the same as the number mentioned in the previous chapter.

From Table 4.1 to Table 4.4, we can observe that the iterative receiver with VSS MS3 decoding algorithm outperforms the iterative receiver with VSS MS decoding algorithm by

	BP(P=1) floating	BP(P=90) floating	MS(P=1) floating	MS(P=90) floating	MS3(P=1) floating	MS3(P=90) floating	MS(P=90) fixed	MS3(P=90) fixed
NID @10^{-4}	23.60 dB	23.60 dB	23.95 dB	24.00 dB	23.75 dB	23.82 dB	25.10 dB	24.75 dB
ID @10^{-4}	22.50 dB	22.50 dB	23.50 dB	23.50 dB	22.95 dB	23.05 dB	24.50 dB	23.80 dB
Gain @10^{-4}	1.10 dB	1.10 dB	0.45 dB	0.50 dB	0.80 dB	0.77 dB	0.60 dB	0.95 dB
NID @10^{-6}	23.90 dB	23.90 dB	24.25 dB	24.30 dB	24.05 dB	24.08 dB	25.40 dB	24.95 dB
ID @10^{-6}	22.65 dB	22.72 dB	23.59 dB	23.80 dB	23.20 dB	23.30 dB	24.75 dB	24.10 dB
Gain @10^{-6}	1.25 dB	1.18 dB	0.66 dB	0.50 dB	0.85 dB	0.78 dB	0.65 dB	0.85 dB

Table 4.4 — Performance comparison for 64K LDPC with a code rate of R=4/5 and 256-QAM constellation

0.2dB at a BER of 10^{-6} in fixed point simulation and even higher performance gain can be observed for 256-QAM constellation.

Algorithm 9 Vertical layered MS3 algorithm

{**Initialization: for all bit nodes** n, where $m \in M(n)$ }

$T_{mn}^{(0)} = llr_n$

$\alpha_m^{(0)} = \prod_{n \in N(m)} sgn(llr_n)$

$M_m^0 = min\left(|llr_n|\right), P_m^0 = index\left(M_m^0\right)$

$M_m^1 = secmin\left(|llr_n|\right), P_m^1 = index\left(M_m^1\right)$

$M_m^2 = thirdmin\left(|llr_n|\right), P_m^2 = index\left(M_m^2\right)$

{**Decoding:**}

for $t = 1, .., t_{max}$ {***iteration***}

  for $n = 1, .., N_{ldpc}$ {***sub-iteration***}

    Check node processing

    if $\left(n == P_m^0\right)$

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

    else

      $E_{mn}^{(t)} = \alpha_m \cdot \eta \cdot sgn\left(T_{mn}^{(t-1)}\right) \cdot M_m^0 \ , \ m \in M(n)$

    Bit node processing

    $T_n^{(t)} = llr_n + \sum_{m \in M(n)} E_{mn}^{(t)} \ , \ n \in N(m)$

    $T_{mn}^{(t)} = T_n^{(t)} - \sum_{m \in M(n)} E_{mn}^{(t)} \ , \ n \in N(m)$

    Check node update for next sub-iteration

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

    $M_m^0 = min\left(|T_{mn}^{(t)}|, |T_{mk'}^{(t-1)}|\right), P_m^0 = Index\left(M_m^0\right) \ , \ k' \in N(m) \backslash n$

    $M_m^1 = secmin\left(|T_{mn}^{(t)}|, |T_{mk'}^{(t-1)}|\right), P_m^1 = Index\left(M_m^1\right) \ , \ k' \in N(m) \backslash n$

    $M_m^2 = thirdmin\left(|T_{mn}^{(t)}|, |T_{mk'}^{(t-1)}|\right), P_m^2 = Index\left(M_m^1\right) \ , \ k' \in N(m) \backslash n$

Hard decision according to $T_n^{(t)}$

4.1.3 A joint shuffled demapping and decoding algorithm for an iterative BICM receiver

The conventional iterative receiver processing is executed in a frame-by-frame manner. The demapper gets the channel symbol information to generate the LLRs, that are fed into the de-interleaver simultaneously. The LLRs can only be read out until the de-interleave is fully written. In other words, until the whole frame of LLRs are fed into the de-interleaver RAM. Classically, LDPC decoding algorithm is horizontal layered decoding, in which the *extrinsic* information is accessible only after one complete iteration. The interleaving process is similar to the de-interleaving process, which is based on the frame processing. This kind of schedule induces a large latency for the messages exchange between the demapper and decoder. This is a critical problem for hardware implementation.

To overcome this problem, a vertical shuffled iterative processing is proposed [57]. The main idea is that dividing the whole frame into several sub-frames and applying iterative processing on each sub-frames, which greatly reduces the delay of the message exchange between the demapper and decoder and improves the message exchange efficiency between the demapper and decoder. The proposal is composed of two stages:

1. **Decoding**: based on vertical shuffled paralleled LDPC decoding. Different from the horizontal layered decoding algorithm, in which the *extrinsic* message is available after one iteration, the vertical shuffled decoding schedule can provide a fast generation of the *extrinsic* message of each bit node.

2. **Interleaving/de-interleaving**: based on Look-Up-Table (LUT) memorizing the routing addresses. A memoryless process of the interleaving and de-interleaving function facilitate the message exchange between the demapper and decoder. Combining with the shuffled iterative decoding algorithm, the messages can be sent between demapper and decoder as soon as possible.

Thanks to these two proposals, the classical frame-based one iteration can be decomposed into multiple paralleled sub-frame processing, with a length equalling to the parallelism level. Consequently, both the decoded and demapped *extrinsic* information can be exchanged as soon as it is available.

The shuffled iterative demapping and decoding algorithms is detailed in Algorithm 10. Let us denote Q as the parallelism of the decoder. It can be any factor of 360. Each global iteration of the vertical shuffled iterative processing consists of several sub-iterations, in which the vertical shuffled decoding is applied on one sub-frame, that corresponding to Q bit nodes. During the initialization stage, one main demapper takes the input symbols and processes

Algorithm 10 Shuffled parallel iterative demapping and decoding algorithm

{**Initialization:** }

$ext_n = 0$

{for all bit nodes n, where $m \in M(n)$ }

$T_{mn}^{(0)} = llr_n$

$\alpha_m^{(0)} = \prod_{n \in N(m)} sgn(llr_n)$

$M_m^0 = min\left(|llr_n|\right), P_m^0 = index\left(M_m^0\right)$

$M_m^1 = secmin\left(|llr_n|\right), P_m^1 = index\left(M_m^1\right)$

$M_m^2 = thirdmin\left(|llr_n|\right), P_m^2 = index\left(M_m^2\right)$

for $t = 1, .., t_{max}$ {***iteration***}

{**Decoding:**} **for** $n = 1, 2.., Q$ **where** $n = \pi^{-1}(i)$

Check node processing

if $\left(n == P_m^0\right)$

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

else

$E_{mn}^{(t)} = \alpha_m \cdot \eta \cdot sgn\left(T_{mn}^{(t-1)}\right) \cdot M_m^0$, $m \in M(n)$

Bit node processing

$T_n^{(t)} = llr_n + \sum_{m \in M(n)} E_{mn}^{(t)}$, $n \in N(m)$

$T_{mn}^{(t)} = T_n^{(t)} - \sum_{m \in M(n)} E_{mn}^{(t)}$, $n \in N(m)$

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

Check node update for next sub-iteration

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

$M_m^0 = min\left(|T_{mn}^{(t)}|, |T_{mk'}^{(t-1)}|\right), P_m^0 = Index\left(M_m^0\right)$, $k' \in N(m) \backslash n$

$M_m^1 = secmin\left(|T_{mn}^{(t)}|, |T_{mk'}^{(t-1)}|\right), P_m^1 = Index\left(M_m^1\right)$, $k' \in N(m) \backslash n$

$M_m^2 = thirdmin\left(|T_{mn}^{(t)}|, |T_{mk'}^{(t-1)}|\right), P_m^2 = Index\left(M_m^1\right)$, $k' \in N(m) \backslash n$

{**Demapping:**} **for** $k = \pi(1), .., \pi(Q)$

$$\underbrace{max}_{x_t \in \chi_0^i} \left\{ -\frac{D_{ecd}(x_t)}{\sigma^2} + \sum_{j=0; j \neq i; b_j = 0}^{m-1} ext_j^{(t)} \right\} - \underbrace{max}_{x_t \in \chi_1^i} \left\{ -\frac{D_{ecd}(x_t)}{\sigma^2} + \sum_{j=0; j \neq i; b_j = 0}^{m-1} ext_j^{(t)} \right\}$$

Hard decision according to $T_n^{(t)}$

the LLR serially, with *extrinsic* information set as zero. After the initialization of LDPC decoder(from line 1 to line 8), the Q check node processors firstly generate and pass the *extrinsic* information (from line 12 to line 15) to the bit node processors during each shuffled sub-iteration. Then Q bit node processors take the *intrinsic* information and *extrinsic* information to generate the *a posteriori* information (line 17), *a priori* information (line 18) and *extrinsic* information (line 19). Afterward, this *extrinsic* information is sent back to the demapper through a network, according to the connection between the input channel symbols and the bit nodes. In parallel, Q check node processors update the check node information from the *a priori* information generated by the bit nodes processors (from line 20 to line 24). Finally, Q demappers in maximum use the fed back *extrinsic* information to update the corresponding LLRs (line 25,26), where $D_{ecd}(x_t)$ denotes the squared Euclidean distances from the received symbol to signals in the constellation panel. Some of the updated LLRs are sent back to the LDPC decoder through a network.

4.1.4 Message passing schedules between LDPC demapper and decoder for an iterative BICM receiver

A basic idea of the paralleled vertical shuffled iterative receiver is illustrated in the form of the Tanner graph in Fig. 4.3. The shuffled iterative schedule guarantees immediate message exchange between the demapper and the decoder. There exists several possible schedules, which correspond to the possible combinations between the different parallelism levels of the LDPC decoder and the different partial update strategies for the demapper. A good match between the demapping process and decoder process is mandatory to design an efficient prototype achieving high throughput.

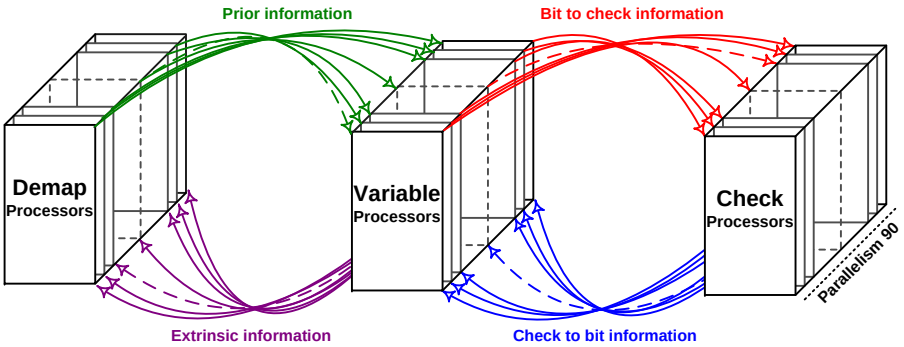


Figure 4.3 — The basic idea of the parallel shuffled iterative receiver

One example of the connection between the channel symbols and the bit nodes with a parallelism level of 4 is illustrated in Fig. 4.4. One *extrinsic* information triggers $(log_2 M - 1)$ LLRs updating of M-QAM constellation. In this case, for a 16-QAM, the bit node processing

of one bit node group provides 4 *extrinsic* information, that leads 12 LLRs updating. The 12 LLRs correspond to 12 bit nodes, that are allocated in the other 3 bit node groups. Therefore, any other bit node group can use the latest LLRs hence takes the best advantage of LLR update. However, in practice, this kind of connection seldom exists. Fig. 4.5 gives another example of the channel symbols and bit nodes connection. In this case, the 4 *extrinsic* messages provided by the first bit node group trigger 12 LLRs update. However these LLRs are sent back to 12 different bit node groups. The update of any bit node group just benefits from one updated LLR. In order to take the best advantage of the updated LLRs, the 12 bit node groups need to process simultaneously, that is not suitable for a hardware implementation. Although the above example is a severe case, this kind of connection may exist. Therefore, finding proper message passing schedule, that suits hardware implementation, is a critical issue for the design of a shuffled iterative receiver.

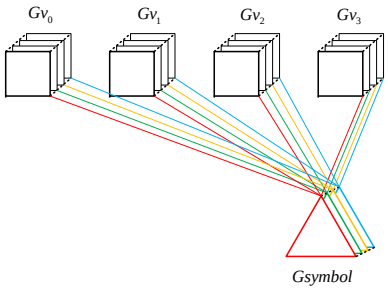


Figure 4.4 — An example of efficient connections between channel symbols and bit nodes

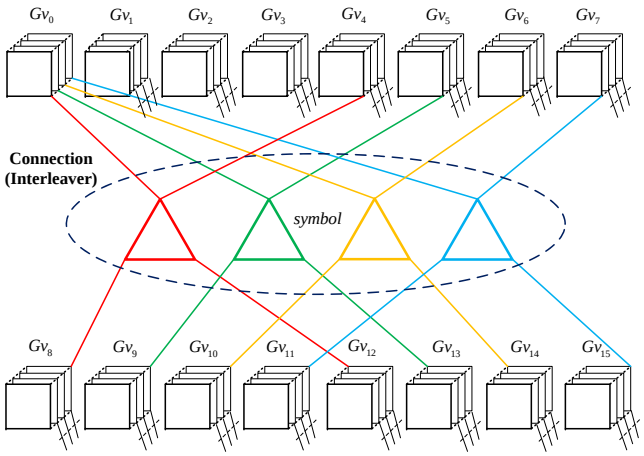


Figure 4.5 — Another example of connections between channel symbols and bit nodes

Three interesting cases are listed in Table 4.5. Schedule A is based on demapper update and adopts a serial schedule between the demapper and the VSS based LDPC decoder. Every processing of the channel input symbol leads to $log_2 M - 1$ bit node update at the initialization stage and one fed back *extrinsic* information triggers $(log_2 M - 1)$ LLRs update. Schedules

B and C are based on VSS LDPC decoder with a parallelism level of 90. In other words, 90 bit node processors get the updated *intrinsic* messages and generate 90 *extrinsic* messages simultaneously that are fed back to 90 demappers in maximum. If all the 90 bits originate from different symbols, then the process requires 90 demappers working in parallel. This clearly represents a worst processing scenario. The difference between Schedule B and Schedule C is in the number of the LLRs that are updated by the demapper during the iterative process.

Schedule	A	B	C
Host	Demapper	Decoder	Decoder
Symbols @ demapper	1	≤ 90	≤ 90
Updated LLRs @ demapper	$log_2 M - 1$	$\leq 90 \cdot (log_2 M - 1)$	90
LDPC parallelism	1	90	90
Bits node processed @ decoder	$log_2 M - 1$	90	90
Extrinsic information feedback @ decoder	$log_2 M - 1$	90	90

Table 4.5 — Message exchange schedules for the shuffled iterative receiver

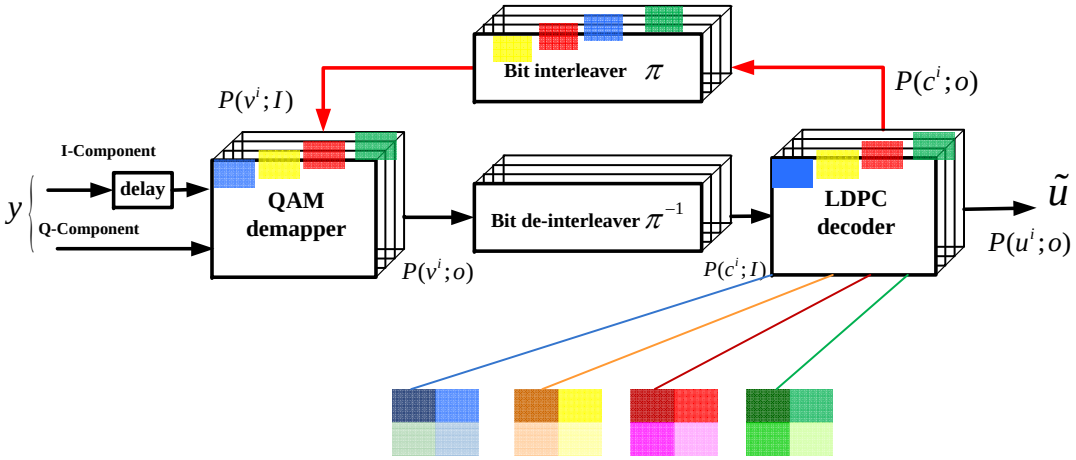


Figure 4.6 — Parallel vertical shuffled iterative BICM receiver

Actually, a LDPC decoder with the parallelism level of 90 can provide 90 *extrinsic* information, that can trigger at most 90 different demappers and generates at most $90 \cdot (log_2 M - 1)$ LLRs. However, the $90 \cdot (log_2 M - 1)$ LLRs can not be written back to the LLR RAM at the same time. In fact, $(log_2 M - 1)$ clock cycles are necessary to write all these values into the LLR RAM.

In the schedule C, we propose that only 90 LLRs are selected and fed back to the LDPC decoder in order to match the decoding process. The LLRs selected to be updated are the ones used by the LDPC decoder during the next sub-iteration. No other LLRs are sent back to the LDPC decoder and participate in the processing of the next bit node group.

Let us take 16-QAM constellation as an example to explain this idea. Fig. 4.6 illustrates

a vertical shuffled iterative receiver with a parallelism level of 4. For one sub-iteration, the decoder first provides 4 *extrinsic* information. This information is routed by a network to the demapper group. At the demapper side, one *extrinsic* information triggers 3 LLRs updating in the case of 16-QAM constellation. If the 4 *extrinsic* information belongs to 4 different channel symbols in one group, then the group of demappers can provide 12 different updated LLRs. However, only 4 LLRs belonging to the second bit node group are selected and sent to the LDPC decoder. The LDPC decoder uses the 4 *extrinsic* information to start the next sub-iteration. The same principle is applied for the following sub-iterations.

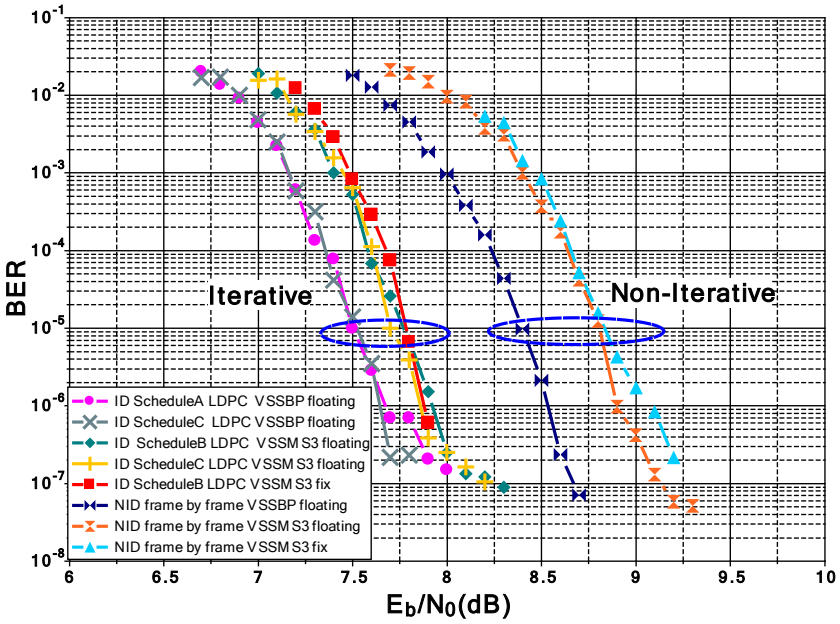


Figure 4.7 — Performance comparison for a 16K LDPC code with a code rate R=4/5 over a fading channel with 15% erasures and a QPSK constellation

Simulations for the different schedules are carried out for 16K LDPC with a code rate of R=4/5 over a fading channel with 15% of erasures and a QPSK constellation. Simulation results shown in Fig. 4.7 show that the proposed schedule C has comparable performance when compared to the optimum schedule A and schedule B both for VSS Belief-Propagation decoding algorithm and the VSS Min-Sum-3 decoding algorithm. Similar results are obtained in Fig. 4.8, for a 64K LDPC code with a code rate of R=4/5 over a fading channel with 15% of erasures and a 256-QAM constellation. It means that the hardware oriented schedule C is appealing for a hardware implementation.

4.2 Design and implementation of an iterative BICM receiver

The proposed iterative receiver with shuffled schedule has several advantages compared to the non-iterative receiver and the non-shuffled iterative receiver [59],[60], [61]. At first, the

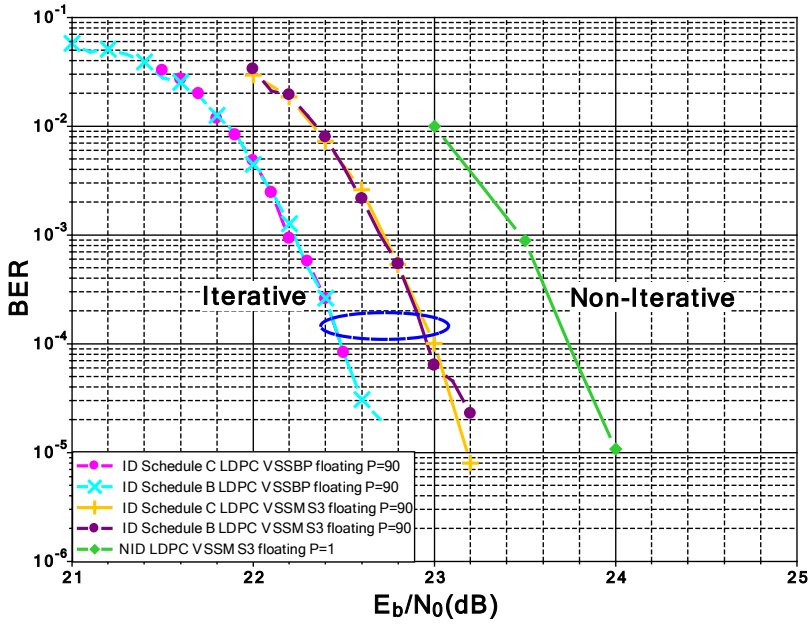


Figure 4.8 — Performance comparison for a 64K LDPC code with a code rate R=4/5 over a fading channel with 15% erasures and a 256-QAM constellation

VSS schedule directly targets updating variable information and facilitates the information exchange between demapper and decoder. At second, the LUT based interleaver 'virtualizes' the usual buffer and reduces both the latency and the hardware requirements in terms of resources. At third, by decomposing the frame into several sub-frames, the iterative processing is applied on each sub-frame. Thus, the latest decoded or demapped information can be rapidly exchanged and therefore accelerates the iterative process convergence.

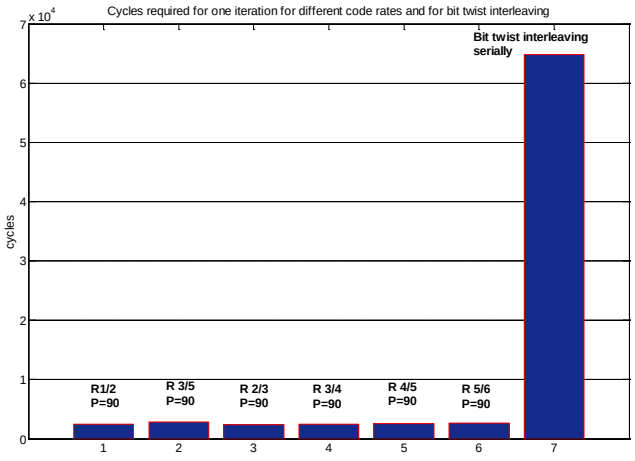


Figure 4.9 — The required time cycles of one decoding iteration for 64K LDPC codes with different code rates and interleaving and de-interleaving

Fig. 4.9 lists the required time cycles for one decoding iteration of a LDPC decoder with a parallelism level of 90 for 64K LDPC codes with different code rates. The required cycles

for interleaving or de-interleaving are also given. The classical interleaving or de-interleaving is a serial process, that does not support for a parallel processing. The time cycles require for a serial interleaving or de-interleaving is around 25 times as the cycles required for one decoding iteration. Even though the interleaving or de-interleaving processing supports for parallel processing with the parallelism level of 90, it still needs 720 cycles, that still greatly slows down the throughput.

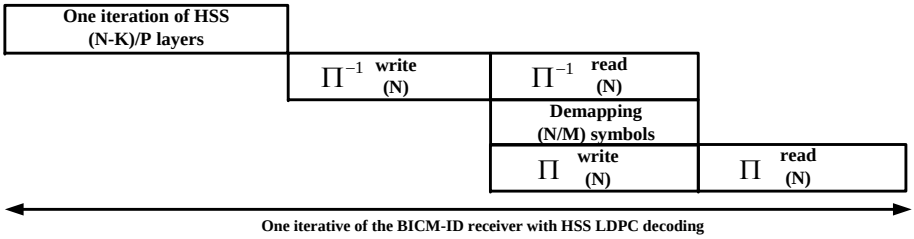


Figure 4.10 — Scheduling of a horizontal iterative receiver

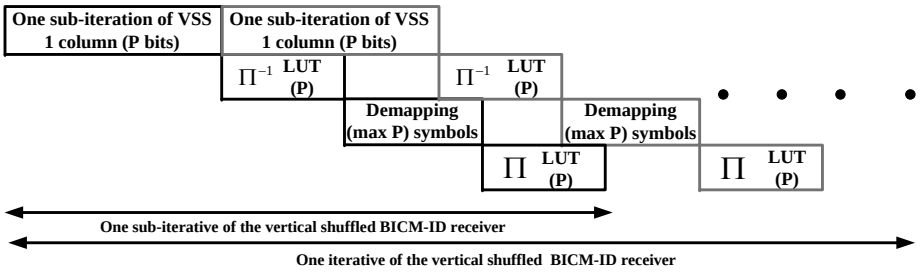


Figure 4.11 — Scheduling of a vertical iterative receiver

Fig. 4.10 and Fig. 4.11 illustrate the iterative processing schedule of the frame-based processing and vertical shuffled processing, respectively. The parity check matrix of 64K LDPC code with a code rate of R=4/5 has 630 nonzero sub-matrices with a dimension of 360x360. To perform one iteration, the frame-by-frame schedule requires $630 \cdot (360/90) + 64800 \cdot 4$ time cycles for an iterative receiver with parallelism of 90, while the iterative shuffled schedule needs only $630 \cdot (360/90) + \delta$ time cycles. δ is the delay of interleaver access plus one demapping process of one sub-iteration. If a LUT is applied in an iterative processing with parallelism of 90, δ is equal to 10 time cycles. It means that the latter only takes around 2% of cycles as the former.

4.2.1 Architecture of an iterative BICM receiver

A parallel vertical shuffled iterative receiver should support parallel LUT routing for interleaving and de-interleaving. The parallelism of the proposed receiver is 90. However, the interleaver in the DVB-T2 standard has not took the parallel access into account. Therefore, for the moment we have considered and implemented as a first step only an iterative receiver

for a QPSK constellation, in which mode there is no interleaving and de-interleaving process in the DVB-T2 standard.

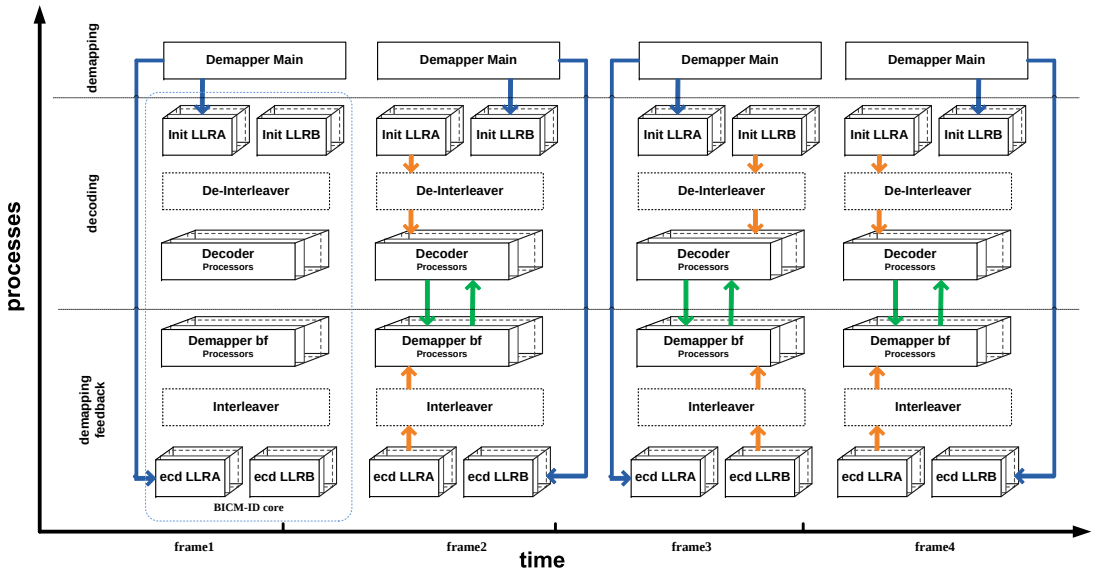


Figure 4.12 — The proposed architecture of the vertical shuffled iterative receiver

The architecture of the proposed iterative receiver is illustrated in Fig. 4.12. One main demapper progressively computes the Euclidean distances and the corresponding LLRs, that are memorized in the ECD RAM and LLR RAM, respectively. Two of those RAMs are allocated. One is in charge of memorizing the input LLRs and Euclidean distance in the initialization process and the other one is in charges of providing the corresponding values in the decoding process. Small modifications are applied on the vertical shuffled Min-Sum LDPC decoder. The bit node processor has to provide the *extrinsic* information by subtracting the updated LLR from the *a posteriori* information. Meanwhile 90 simplified demappers update LLRs by the *extrinsic* information fed back from decoder and the channel input symbol information from the ECD RAM. The updated LLRs are available only after two time cycles of introducing updated *extrinsic* information. In this way, the bit node processor of the LDPC decoder can use the latest updated LLRs without wasting waiting cycles, even for the bits with a check node degree equal to 3.

For a QPSK constellation, the bit node i and $(i+1)$ are associated to the same channel input symbol for information bits, while in order to obtain quasi-cyclic property, the bit $(K_{ldpc}+i)$ and $(K_{ldpc}+i+4)$ are associated to the same input symbols for parity bits for any LDPC code rates with Q as even number.

Fig. 4.14 and Fig. 4.15 illustrate efficient message passing schedule between the demapper and the decoder for the information bits and parity bits, receptively. For the two bit node processors belonging to one channel symbol, the latter bit node processor uses the LLR in-

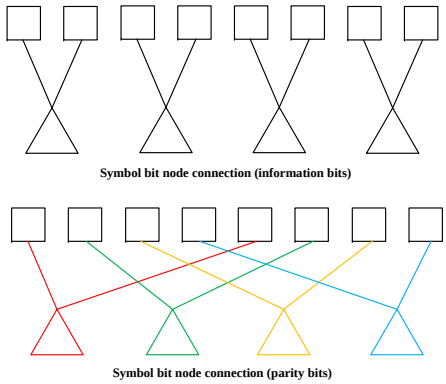


Figure 4.13 — Symbols and bit nodes connection for a QPSK constellation

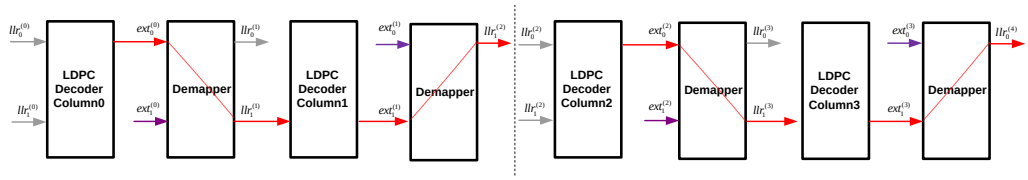


Figure 4.14 — Efficient message passing schedule for the vertical shuffled iterative processing (information bits) for a QPSK constellation

formation updated by the *extrinsic* information provided from the former bit node processor. Therefore, the bit node processors with even number index use the latest LLRs to facilitate an efficient message exchange.

In order to manage the message exchange, few modifications are necessary. Fig. 4.16 gives the corresponding architecture of the message routing between the demappers and the decoders. During the processing of information bit nodes, the bit node processing for the bit node group with odd number index directly gets the updated LLRs generated during the bit node processing of the previous bit node group. As for the bit node processing for the bit node group with even number index, they need to get the LLRs values from the LLR

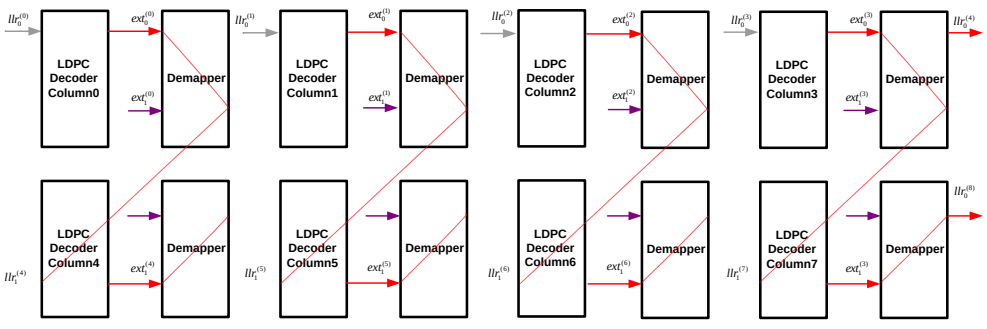


Figure 4.15 — Efficient message passing schedule for the vertical shuffled iterative processing (parity bits) for a QPSK constellation

RAM (re-usage of init RAM) in which the updated LLRs are memorized during the previous iteration. The same timing and message passing schedule can be applied for the processing of parity bit nodes, with a re-order of the bit node processing for every 8 columns as illustrated in equ. (4.9).

$$\begin{aligned}&(K_{ldpc}+i),(K_{ldpc}+i+4),(K_{ldpc}+i+1),(K_{ldpc}+i+5),\\&(K_{ldpc}+i+2),(K_{ldpc}+i+6),(K_{ldpc}+i+3),(K_{ldpc}+i+7)\end{aligned} \tag{4.9}$$

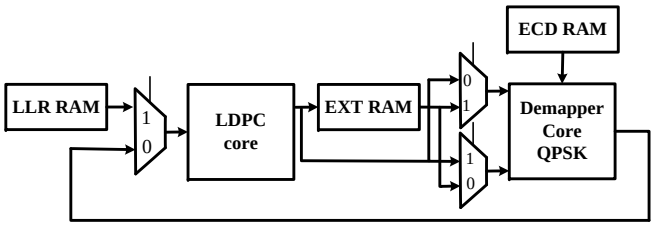


Figure 4.16 — Architecture of the message exchange between the demapper and decoder

4.2.2 The prototyping of the iterative BICM transceiver onto an experimental setup

In order to demonstrate the performance of the proposed shuffled iterative receiver, a prototype of a digital transmission chain that contains an iterative BICM system based on the DVB-T2 standard was implemented onto an FPGA chip on the demo board shown in Fig. 3.31 and Fig. 3.32 in the previous chapter.

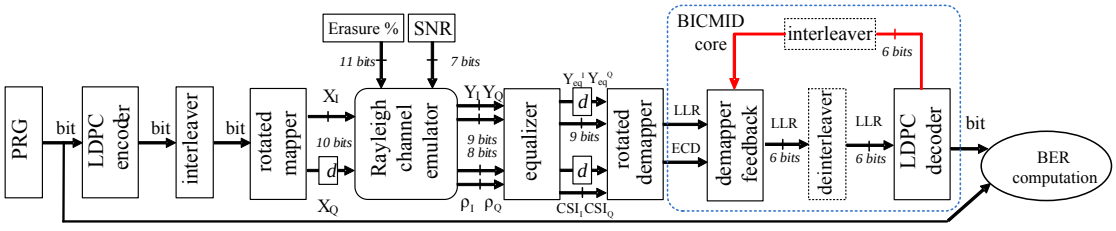


Figure 4.17 — Experimental setup for prototyping of the iterative BICM system for DVB-T2

All the components of the iterative BICM transceiver are illustrated in Fig. 4.17. The BICM-ID receiver is made up of a main demapper and a BICM-ID core. The transmitter and channel emulator are the same as the prototype described in the previous chapter.

The transceiver was synthesized and implemented onto an FPGA board. Computational resources of the BICM-ID MS core takes up about 11% and 44% slice registers and slice

LUTs of a Xilinx Virtex5 VLX330 FPGA device, respectively. If a BICM-ID MS3 core is implemented, 12% slice registers and 52% slice LUTs are necessary. The FPGA logic synthesis results of the proposed BICM-ID core using VSS MS decoding algorithm and VSS MS3 decoding algorithm are shown in Table 4.6 and Table 4.7, respectively.

Number of Slice Registers	23118 out of 207360 (11%)
Number of Slice LUTS	93130 out of 207360 (44%)
Number of Block RAM	179 out of 288 (62%)
Frequency	125 Mhz

Table 4.6 — Device occupation for the BICM-ID core using VSS MS decoding

Number of Slice Registers	26088 out of 207360 (12%)
Number of Slice LUTS	107438 out of 207360 (51%)
Number of Block RAM	193 out of 288 (67%)
Frequency	114 Mhz

Table 4.7 — Device occupation for the BICM-ID core using VSS MS3 decoding

The FPGA logic synthesis results of the corresponding transceivers are shown in Table 4.8 and Table 4.9 respectively. The maximum frequency estimated for the BICM-ID MS core and BICM-ID MS3 core after place and route is 80MHz. It results in a throughput of 107 Mbps, for 64K R=4/5 @ 15 iterations.

Number of Slice Registers	38722 out of 207360 (18%)
Number of Slice LUTS	114857 out of 207360 (55%)
Number of Block RAM	188 out of 288 (65%)
Frequency	98 Mhz

Table 4.8 — Device occupation for the BICM-ID transceiver using VSS MS decoding

Number of Slice Registers	41702 out of 207360 (20%)
Number of Slice LUTS	125578 out of 207360 (60%)
Number of Block RAM	202 out of 288 (70%)
Frequency	89 Mhz

Table 4.9 — Device occupation for the BICM-ID transceiver using VSS MS3 decoding

A comparison of simulated performance in terms of BER is presented in Fig. 4.18. More than 10 dB gain is observed from the BICM-ID VSS MS3 receiver with signal space diversity when compared to the non-iterative classical BICM receiver. Moreover, an additional 0.8dB gain is achieved for the iterative receiver with VSS MS3 decoding algorithm when compared to the non-iterative BICM receiver with signal space diversity.

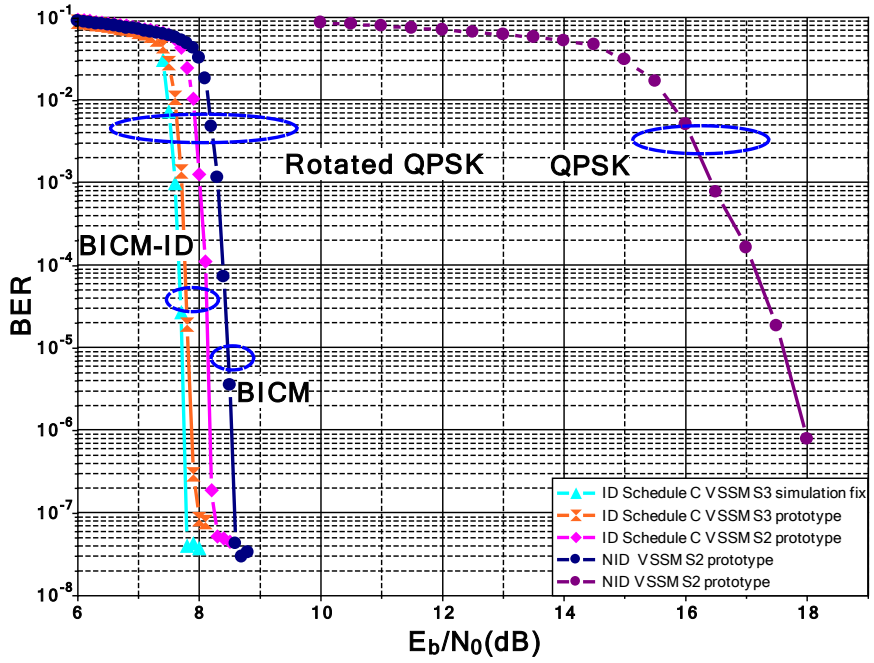


Figure 4.18 — Prototype performance comparison for 64K LDPC with a code rate of R=4/5 over a fading channel with 15% erasures and QPSK constellation

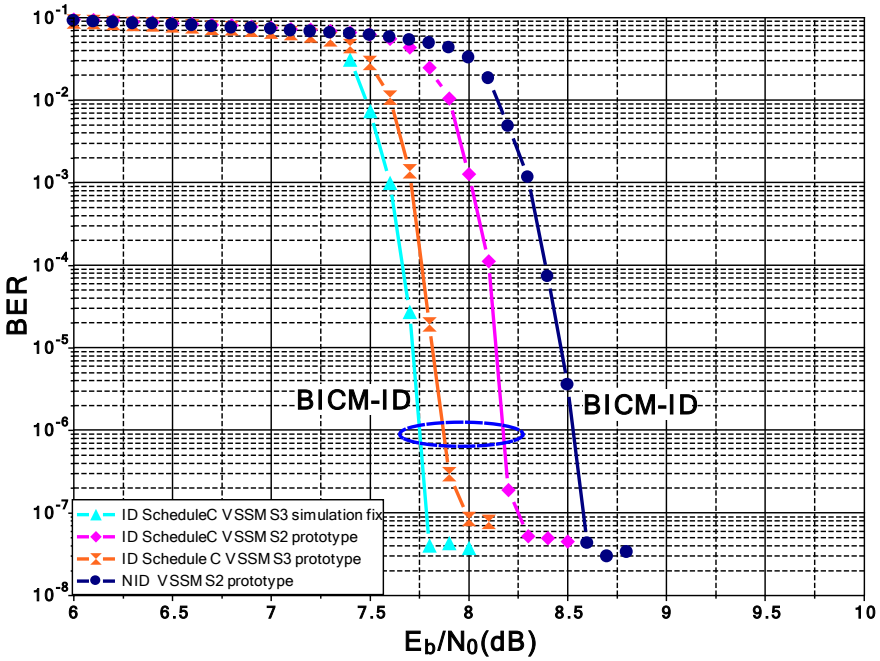


Figure 4.19 — Prototype performance comparison for 64K LDPC with a code rate of R=4/5 over a fading channel with 15% erasures and QPSK constellation

4.3 Conclusion

In this chapter we have proposed a novel iterative receiver with vertical shuffled schedule for the DVB-T2 standard. It enables an efficient information exchange between the demapper and the decoder in an ID context hence greatly reducing the message passing latency. Several schedules of message exchange between the demapper and decoder are studied. The proposed hardware oriented message passing schedule is verified to have almost the same performance as the optimal schedule, that enjoys low complexity and low latency. The work of algorithm level has been published in [57] and the work of hardware implementation level has been published in [59].

FPGA prototypes based on a vertical shuffled schedule with Min-Sum and Min-Sum-3 decoding algorithms have been implemented. The designed iterative receiver achieves high performance gain as expected, that validates the efficiency of the proposed method. To the best of our knowledge, this is the first hardware implementation of a BICM-ID receiver for the DVB-T2 standard.

Conclusion

The work accomplished during this PhD thesis focuses on design and implementation of a DVB-T2 BICM system with low hardware complexity and high throughput both for non-iterative and iterative processing. The obtained results both in algorithm design and hardware implementation have shown the potential of an iterative receiver as a practical and competitive solution for the DVB-T2 standard.

First, we have addressed the issue that the demapping algorithm for rotated QAM constellation should be two-dimensional. It requires large number of multiplications, especially for high order constellations. In order to reduce the computation complexity, we have proposed a *Max-Log two-dimensional demapping algorithm based on sub-region detection*. This algorithm reduces the number of required computations of Euclidean distances. The proposal of *linear approximation of Euclidean distance* further reduces the number of multiplications. Based on these two contributions, a flexible demapper was implemented and tested. It supports all the eight different constellations over an AWGN channel, a fading channel with and without erasures. The demapper can be easily extended to higher order constellations, such as 1024-QAM constellation or 4096-QAM constellation adopted in the DVB-C2 standard.

Then, in order to reduce the latency of the message exchange between the demapper and the LDPC decoder in the case of an iterative receiver, *a Min-Sum vertical shuffled LDPC decoding algorithm* has been proposed to provide a fast generation of extrinsic information. In addition, the methods to *avoid memory access conflicts* due to pipeline and to *avoid message update conflicts* due to double diagonal sub-matrix have been detailed for the vertical shuffled LDPC decoder. A digital communication setup, including source generator, LDPC encoder, bit interleaver, mapper, channel emulator, equalizer, demapper, bit de-interleaver, LDPC decoder and BER calculator, has been implemented onto one FPGA device(Xilinx Virtex 5) on an emulation board (DN9000K10PCI) to verify the algorithm and architecture design of the demapper and LDPC decoder over an AWGN channel and a fading channel with and without erasures. The demapper and LDPC decoder have also been integrated onto a real DVB-T2 demodulator provided by the company Teamcast. The performance measures have been performed in a real environment provided by Teamcast funded by the SME42 project,

which includes a real demodulator, channel emulator and the demodulator. Promising results have shown the efficiency of the demapper and the LDPC decodere.

Afterwards, we have proposed *a joint vertical shuffled iterative demapping and decoding algorithm* to reduce the processing latency of an iterative receiver. The main idea of our proposal relies on dividing the whole frame into sub-frames and applying iterative processing on every sub-frame. This is achieved by using vertical shuffled LDPC decoding to provide a fast generation of extrinsic information and by using a *Look-Up-Table based interleaving/de-interleaving* to provide a fast routing of information between the demapper and the decoder. The *message exchange schedules* have been investigated. An efficient schedule that is suitable for hardware implementation has been detailed and the corresponding shuffled parallel iterative BICM receiver has been designed and implemented onto an emulation board for QPSK constellation as a first step. The design of this iterative receiver takes up around 50% of hardware resources in terms of RAM and logic slice of a Xilinx Virtex 5 LX330 device. The estimated maximum working frequency of the receiver is 80Mhz, that results to a throughput of 107 Mbps for 64K LDPC with a code rate of R=4/5. The measured performance achieves expected performance gains, which validate the efficiency of our proposal. To the best of our knowledge, this is the first hardware implementation of a BICM-ID receiver for the DVB-T2 standard.

Perspectives

The results obtained during this PhD study provide a novel framework for designing an iterative receiver for the DVB-T2 standard. Future works on this subject are expected to design an iterative receiver that supports higher QAM constellation modes and all LDPC code lengths and code rates.

Although the Look-Up-Table based interleaving/de-interleaving can provide a fast routing of information between the demapper and decoder, the interleaver in the DVB-T2 standard does not facilitate parallel processing. This is the main bottleneck of the design of a high throughput vertical shuffled iterative receiver. A new interleaver that supports paralleled processing has to be well designed.

At the presence of extrinsic information, the demapping algorithm does not reduce to a simple geometrical problem any more. Indeed, feedback has to be taken into account and the need to go through the demapper for every iteration can result into an important complexity increase especially for high order constellations. Therefore, further simplification of demapping algorithm is required to reduce the hardware complexity and the processing latency to enable a fast message exchange between the decoder and the demapper.

In the case of an iterative process, the vertical shuffled Min-Sum-3 decoding algorithm outperforms the vertical shuffled Min-Sum decoding algorithm. However, the Min-Sum-3 requires more hardware resources. Future work may be carried out in order to find the best check node processing algorithm at the lowest hardware cost for an iterative receiver.

From the implementation point of view, the way to memorize the channel input symbol, extrinsic information, LLR and the temporary information inside the LDPC decoder should be well organized to avoid memory access conflicts when a paralleled LUT based interleaver/de-interleaver is introduced.

From a long-term perspective, the idea of vertical shuffled iterative structure can also be extended to other blocks of the receiver. A vertical shuffled iterative process can be carried out between the synchronization block or the channel estimation block or etc and the FEC decoder, in order to improve furthermore performance.

Publications

- Meng Li, C.A. Nour, C. Jego, and C. Douillard, "Design of rotated QAM mapper/demapper for the DVB-T2 standard", In Signal Processing Systems (SIPS), 2009 IEEE Workshop on, pages 018 - 023 , oct. 2009 (Published).
- Meng Li, C.A. Nour, C. Jego, and C. Douillard, "Design and FPGA prototyping of a bit-interleaved coded modulation receiver for the DVB-T2 standard", In Signal Processing Systems (SIPS), 2010 IEEE Workshop on, pages 162 -167, oct. 2010 (Published).
- Meng Li, C.A. Nour, C. Jego, and C. Douillard, "Design of an efficient LDPC decoder for bit-interleaved coded modulation receivers", In System On Chip - System In Package (SOC-SIP), June 2010. (Published).
- Meng Li, C.A. Nour, C. Jego, Jianxiao Yang, and C. Douillard, "Efficient iterative receiver for bit-interleaved coded modulation according to the DVB-T2 standard", In Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on, pages 3168-3171, may 2011. (Published).
- Meng Li, C.A. Nour, C. Jego, Jianxiao Yang, and C. Douillard, "FPGA implementation of a shuffled iterative bit-interleaved coded modulation receiver", In ystem On Chip System In Package (SOC-SIP), June 2011. (Published).
- Meng Li, C.A. Nour, C. Jego, Jianxiao Yang, and C. Douillard, "Etude d'un récepteur itératif dédié à un systéme de modulation codée à bits entrelacés pour le standard DVB-T2", In XXIIIe Colloque GRETSI, September 2011. (Published).
- Meng Li, C.A. Nour, C. Jego, Jianxiao Yang, and C. Douillard, "A shuffled iterative bit-interleaved coded modulation receiver for the DVB-T2 standard: Design, implementation and FPGA prototyping", In Signal Processing Systems (SIPS), 2011 IEEE Workshop on, oct. 2011. (Published). **Award best student paper finalist**.

Glossary

AWGN: Additive White Gaussian Noise

BCH: Bose-Chadhui-Hocquenghem
BCJR: Bahl-Cocke-Jelinek-Raviv
BER: Bit Error Rate
BICM: Bit Interleaved Coded Modulation
BICM-ID: Bit Interleaved Coded Modulation- Iterative Decoding

CSI: Channel State Information

DDSM: Double Diagonal Sub-Matrix
DVB: Digital Video Broadcasting
DVB-C2: Digital Video Broadcasting Cable second generation
DVB-S2: Digital Video Broadcasting Satellite second generation
DVB-T2: Digital Video Broadcasting Terrestrial second generation

PDF: Probability Distribution Function

FEC: Forward Error Correcting
FSM: Finite State Machine
FFT: Fast Fourier Transform
FPGA: Field Programmable Gate Array

HD : High Definition
HSS: Horizontal Shuffled Schedule

IRA: Irregular Repeat Accumulate
ISI: Inter Symbol Interference

LDPC: Low Density Parity Check
LLR: Log Likelihood Ratio
LFSR: Linear Feedback Shift Register

MAP: Maximum-A-Posteriori
MFN: Multi-Frequency Network
MISO: Multiple-Input Single-Output
MP: Message Passing
MPEG-4: Moving Picture Experts Group - 4
MS: Min-Sum
MS3: Min-Sum-3

PLP: Physical Layer Pipe
PRG: Pseudo Random Generator

QPSK: Quadrature Phase-Shift Keying

RAM: Random Access Memory
RS: Reed Solomon
Recursive Systematic Convolutional: RSC

SNR: Signal-to-Noise Ratios
SFN: Single-Frequency Network
SOVA: Soft Output Viterbi Algorithm
SSD: Signal Space Diversity

TS: Transport Streams
TCM: Trellis Coded Modulation

VSS: Vertical Shuffled Schedule

List of Figures

List of Tables

Bibliography

[1] DVB-T2. Frame structure channel coding and modulation for a second generation digital terrestrial television broadcasting system. pages 162 –167, 2009.

[2] R. Gallager. Low-Density Parity-Check codes. *Information Theory, IRE Transactions on*, 8(1):21 –28, january 1962.

[3] R.C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control 3 (1)*, page 68¨C79, 1960.

[4] S.M. Alamouti. A simple transmit diversity technique for wireless communications. *Selected areas in communications, IEEE Journal on*, 16(8):1451 –1458, oct 1998.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. In *Communications, 1993. ICC 93. Geneva. Technical Program, Conference Record, IEEE International Conference on*, volume 2, pages 1064 –1070 vol.2, may 1993.

[6] D.J.C. MacKay and R.M. Neal. Near shannon limit performance of low density parity check codes. *Electronics Letters*, 33(6):457 –458, mar 1997.

[7] J. Salz. Digital transmission over cross-coupled linear channels. In *AT&T Technical Journal*, pages 1147–1159, 1985.

[8] J. Boutros and E. Viterbo. Signal space diversity: a power- and bandwidth-efficient diversity technique for the rayleigh fading channel. *Information Theory, IEEE Transactions on*, 44(4):1453 –1467, jul 1998.

[9] I. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 1960.

[10] Elwyn R. Berlekamp. Algebraic coding theory, laguna hills. *McGraw-Hill*, 1984.

[11] J. Massey. Shift-register synthesis and BCH decoding. *Information Theory, IEEE Transactions on*, 15(1):122 – 127, jan 1969.

[12] R.M. Pyndiah. Near-optimum decoding of product codes: block turbo codes. *Communications, IEEE Transactions on*, 46(8):1003 –1010, aug 1998.

[13] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260 – 269, apr 1967.

[14] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *Information Theory, IEEE Transactions on*, 20(2):284 – 287, mar 1974.

[15] G.D Forney. Concatenated codes. *Cambridge, MIT Press*, 1966.

[16] C.E. Shannon. A mathematica theory of communication (Part I). *Bell System Technical Journal*, page 379¨C423, 1948.

[17] C.E. Shannon. A mathematica theory of communication (Part II). *Bell System Technical Journal*, pages 623–656, 1948.

[18] R. Tanner. A recursive approach to low complexity codes. *Information Theory, IEEE Transactions on*, 27(5):533 – 547, sep 1981.

[19] J. Chen, A. Dholakia, E. Eleftheriou, M.P.C. Fossorier, and X.-Y. Hu. Reduced-complexity decoding of LDPC codes. *Communications, IEEE Transactions on*, 53(8):1288 – 1299, aug. 2005.

[20] Z. Juntan and M.P.C. Fossorier. Shuffled iterative decoding. *Communications, IEEE Transactions on*, 53(2):209 – 213, feb. 2005.

[21] M.M. Mansour. A Turbo-decoding message-passing algorithm for sparse parity-check matrix codes. *Signal Processing, IEEE Transactions on*, 54(11):4376 –4392, nov. 2006.

[22] B. Sklar. Rayleigh fading channels in mobile digital communication systems. I. characterization. *Communications Magazine, IEEE*, 35(9):136 –146, sep 1997.

[23] B. Sklar. Rayleigh fading channels in mobile digital communication systems .II. mitigation. *Communications Magazine, IEEE*, 35(7):102 –109, jul 1997.

[24] Charbel ABDEL NOUR. *Iterative mehtods for improving error correcting performance of spectrally efficient coded transmissions*. PhD thesis, 2007.

[25] DVB-T2. Digital video broadcasting (DVB); implementation guidelines for a second-generation digital terrestrial television broadcasting system (DVB-T2). 2009.

[26] G. Ungerboeck. Channel coding with multilevel/phase signals. *Information Theory, IEEE Transactions on*, 28(1):55 – 67, jan 1982.

[27] A.J. Viterbi, J.K. Wolf, E. Zehavi, and R. Padovani. A pragmatic approach to trellis-coded modulation. *Communications Magazine, IEEE*, 27(7):11 –19, jul 1989.

[28] E. Zehavi. 8-PSK trellis codes for a rayleigh channel. *Communications, IEEE Transactions on*, 40(5):873 –884, may 1992.

[29] G. Caire, G. Taricco, and E. Biglieri. Bit-interleaved coded modulation. *Information Theory, IEEE Transactions on*, 44(3):927 –946, may 1998.

[30] S.B. Slimane. An improved PSK scheme for fading channels. In *Global Telecommunications Conference, 1996. GLOBECOM '96. 'Communications: The Key to Global Prosperity*, volume 2, pages 1276 –1280 vol.2, nov 1996.

[31] N.H. Tran, H.H. Nguyen, and Tho Le-Ngoc. Performance of BICM-ID with signal space diversity. *Wireless Communications, IEEE Transactions on*, 6(5):1732 –1742, may 2007.

[32] C. Abdel Nour and C. Douillard. Improving BICM performance of QAM constellations for broadcasting applications. In *Turbo Codes and Related Topics, 2008 5th International Symposium on*, pages 55 –60, sept. 2008.

[33] X. Qiuliang, S. Jian, P. Kewu, Y. Fang, and W. Zhaocheng. Coded modulation with signal space diversity. *Wireless Communications, IEEE Transactions on*, 10(2):660 –669, february 2011.

[34] L. Xiaodong and J.A. Ritcey. Bit-interleaved coded modulation with iterative decoding. *Communications Letters, IEEE*, 1(6):169 –171, nov 1997.

[35] L. Xiaodong and J.A. Ritcey. Trellis-coded modulation with bit interleaving and iterative decoding. *Selected Areas in Communications, IEEE Journal on*, 17(4):715 –724, apr 1999.

[36] L. Xiaogdong and J.A. Ritcey. Bit-interleaved coded modulation with iterative decoding using soft feedback. *Electronics Letters*, 34(10):942 –943, may 1998.

[37] DVB-T2. Frame structure channel coding and modulation for a second generation digital terrestrial television broadcasting system. *ETSI EN 302 755*, 2008.

[38] C. Abdel Nour and C. Douillard. Rotated QAM constellations to improve BICM performance for DVB-T2. In *Spread Spectrum Techniques and Applications, 2008. ISSSTA '08. IEEE 10th International Symposium on*, pages 354 –359, aug. 2008.

[39] C. Wallace. Fast pseudorandom generators for normal and exponential variates. *ACM Trans. Math.Softw*, 22(4):119 –127, 1996.

[40] David J.C. MacKay and Radford M. Neal. Near shannon limit performance of low density parity check codes. *Electronics Letters*, 32:1645–1646, 1996.

[41] A.J. Blanksby and C.J. Howland. A 690-mw 1-gb/s 1024-b, rate-1/2 low-density parity-check code decoder. *Solid-State Circuits, IEEE Journal of*, 37(3):404 –412, mar 2002.

[42] F. Kienle, T. Brack, and N. Wehn. A synthesizable ip core for dvb-s2 ldpc code decoding. In *Design, Automation and Test in Europe, 2005. Proceedings.*

[43] D.E. Hocevar. A reduced complexity decoder architecture via layered decoding of LDPC codes. In *Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on*, pages 107 – 112, oct. 2004.

[44] F. Guilloud, E. Boutillon, J. Tousch, and J.-L. Danger. Generic description and synthesis of LDPC decoders. *Communications, IEEE Transactions on*, 55(11):2084 –2091, nov. 2007.

[45] J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *Information Theory, IEEE Transactions on*, 42(2):429 –445, mar 1996.

[46] J. Erfanian, S. Pasupathy, and G. Gulak. Reduced complexity symbol detectors with parallel structure for ISI channels. *Communications, IEEE Transactions on*, 42(234):1661 –1671, feb/mar/apr 1994.

[47] C. Marchand, L. Conde-Canencia, and E. Boutillon. Architecture and finite precision optimization for layered ldpc decoders. In *Signal Processing Systems (SIPS), 2010 IEEE Workshop on*, pages 350 –355, oct. 2010.

[48] F. Guilloud, E. Boutillon, J. Tousch, and J.-L. Danger. Lambda-min decoding algorithm of regular and irregular LDPC codes. *3nd International Symposium on Turbo Codes and Related Topics*, pages 451–454, sept. 2003.

[49] D.J.C Mackay. LDPC datebase. In *http://www.inference.phy.cam.ac.uk/mackay/codes/data.html.*

[50] R.L. Urbanke. a fast and accurate degree distribution optimizer for LDPC code ensembles. In *http://lthcwww.epfl.ch/research/ldpcopt/.*

[51] C. Marchand, J.-B. Dore, L. Conde-Canencia, and E. Boutillon. Conflict resolution by matrix reordering for DVB-T2 LDPC decoders. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1 –6, 30 2009-dec. 4 2009.

[52] S. Muller, M. Schreger, M. Kabutz, M. Alles, F. Kienle, and N. Wehn. A novel LDPC decoder for DVB-S2 IP. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 1308 –1313, april 2009.

[53] Massimo Rovini, Giuseppe Gentile, Francesco Rossi, and Luca Fanucci. A minimum-latency block-serial architecture of a decoder for ieee 802.11n ldpc codes. In *Very Large*

Scale Integration, 2007. VLSI - SoC 2007. IFIP International Conference on, pages 236 –241, oct. 2007.

[54] Oscar David S.G., A. Matthieu, C. Jego, G. Mauricio, and Antonio G. Design and implementation of a mimo channel emulator onto fpga device. In *IWS'09: XV proyecto Iberchip*, 2009.

[55] Meng Li, C.A. Nour, C. Jego, and C. Douillard. Design and FPGA prototyping of a bit-interleaved coded modulation receiver for the DVB-T2 standard. In *Signal Processing Systems (SIPS), 2010 IEEE Workshop on*, pages 162 –167, oct. 2010.

[56] Meng Li, C.A. Nour, C. Jego, and C. Douillard. Design of an efficient ldpc decoder for bit-interleaved coded modulation receivers. In *System On Chip - System In Package (SOC-SIP)*, June 2010.

[57] Meng Li, C.A. Nour, C. Jego, Jianxiao Yang, and C. Douillard. Efficient iterative receiver for bit-interleaved coded modulation according to the dvb-t2 standard. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 3168 –3171, may 2011.

[58] C.A. Nour and C. Douillard. On lowering the error floor of high order turbo BICM schemes over fading channels. In *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, pages 1 –5, 2006.

[59] Meng Li, C.A. Nour, C. Jego, and C. Douillard. A shuffled iterative bit-interleaved coded modulation receiver for the dvb-t2 standard: Design, implementation and fpga prototyping. In *Signal Processing Systems (SIPS), 2011 IEEE Workshop on*, oct. 2011.

[60] Meng Li, C.A. Nour, C. Jego, Jianxiao Yang, and C. Douillard. FPGA implementation of a shuffled iterative bit-interleaved coded modulation receiver. In *ystem On Chip - System In Package (SOC-SIP)*, June 2011.

[61] Meng Li, C.A. Nour, C. Jego, and C. Douillard. Etude d'un récepteur itératif dédié à un systéme de modulation codée à bits entrelacés pour le standard DVB-T2. In *XXIIIe Colloque GRETSI*, September 2011.