

INFSO-ICT-216203 DAVINCI

D6.1.1

Adapt the already existing simplifications on LDPC $GF(2^q)$ to DaVinci codes

Contractual Date of Delivery to the CEC: 30 06 2008 *(as specified in the contract)*

Actual Date of Delivery to the CEC: 03 07 2008

Author(s): L. Conde-Canencia, A. Al Ghouwayel, E. Boutillon

Participant(s): UBS

Workpackage: WP6

Estimated person months: 4

Security: PU

Nature: R

Version: 0.4

Total number of pages: 20

Abstract: This document first presents a detailed study of the existing algorithms for non-binary LDPC decoding. The level of parallelism necessary to decode the DaVinci codes in real time is then considered. Finally, some original methods for rapid evaluation of the performance of decoding algorithms are presented.

Keyword list: DA VINCI, deliverable, internal report, simplified decoding for non-binary LDPC, parallelism, reduced simulation methods, performance prediction.

Disclaimer:

Executive Summary

This is version v0.4 of the document Deliverable 6.1.1.

This document focuses on the study of the existing decoding algorithms for non-binary LDPC and the complexity analysis of their application to the benchmark DaVinci codes provided in D4.1.1 (first Deliverable of WP4). We also calculate the level of parallelism necessary to decode DaVinci codes in a real time context. Finally, we present some tools for rapid evaluation of the performance of decoding algorithms. These methods will be used in the continuation of WP6.

Authors

Partner	Name	Phone / Fax / e-mail
Université de Bretagne Sud (UBS)		
	L. Conde-Canencia	Phone: +33 297874528 Fax: +33 297874527 e-mail: laura.conde-canencia@univ-ubs.fr
	Ali Al Ghouwayel	Phone: +33 297874568 Fax: +33 297874527 e-mail: ali.al-ghouwayel@univ-ubs.fr
	Emmanuel Boutillon	Phone: +33 297874566 Fax: +33 297874527 e-mail: emmanuel.boutillon@univ-ubs.fr

Table of Contents

1. Introduction	6
2. State-of-the-art in simplified decoding of non-binary LDPC codes.....	6
2.1 A brief history of non-binary LDPC decoding algorithms	6
2.2 BP decoding algorithm	7
2.3 FFT-based BP decoding on $GF(2^p)$	8
2.4 Decoding in the logarithm domain (log-BP)	9
2.5 Simplified decoding in the logarithm domain	9
2.6 Decoding in the bit domain	10
2.7 Conclusion.....	10
3. Level of parallelism required by DaVinci codes.....	12
3.1 Theoretical study	12
3.2 Impact of the decoding algorithm.....	13
3.3 Conclusion.....	13
4. Some methods to evaluate performance of sub-optimal decoding algorithms.....	13
4.1 Reduced Monte-Carlo Simulation (RMCS)	14
4.1.1 Offset simulations	16
4.1.2 Verification of results.....	16
4.1.3 Conclusion	16
4.2 Performance Prediction Distribution Distance (PPDD)-based method	16
4.3 Check node PPDD-based method.....	19
5. Conclusion and further work	19
References	19

List of Acronyms and Abbreviations

BCJR	Bahl Cocke Jelinek Raviv
BP	Believe Propagation
EMS	Extended Min-Sum
FER	Frame Error Rate
FFT	Fast Fourier Transform
IR	Internal Report
LDPC	Low Density Parity Check code
MCS	Monte Carlo Simulation
MS	Min-Sum
PRG	Pseudo Random Generator
RMCS	Reduced MCS
SNR	Signal-to-noise ratio
WP	Workpackage

1. Introduction

The objective of this deliverable is to adapt the already existing simplifications on LDPC $GF(q)$ to DaVinci codes. We will consider here the work of David Declercq on $GF(q)$ -LDPC and estimate the hardware complexity. This sub-task will give a state-of-the-art that will be considered as a reference for the rest of the project.

This deliverable is divided in 4 sections. Section 2 presents a brief state-of-the-art of decoding algorithms for non-binary LDPC codes. For each algorithm, a realistic complexity evaluation is given in terms of number of operations. Section 3 gives a formal description of the minimum level of parallelism (i.e. number of check nodes and variable node processors) required to decode the DaVinci code in real time. Since the level of parallelism depends on the decoding algorithm (not yet selected) and the type of architecture (idem), we only give general formula that will help the joint design of code and architecture. In section 4, we present three different method/tools developed in the first 6 months of WP6 to tune finely the trade-off complexity versus performance of the decoder architecture. Note that a scientific paper is scheduled to present one of these methods in order to disseminate the result in the scientific community.

2. State-of-the-art in simplified decoding of non-binary LDPC codes

$GF(q)$ -LDPC codes have been shown to approach Shannon-coding performance for $q = 2$ and very long code lengths. For moderate and short code lengths, the error performance can be improved by increasing q . However this improvement is achieved at the expense of increased decoding complexity.

After presenting a brief history of non-binary LDPC decoding algorithms, the different techniques will be presented and evaluated in terms of complexity. The objective is to identify the most suitable algorithms to be further investigated for the implementation of the decoder in the following tasks of WP6.

2.1 A brief history of non-binary LDPC decoding algorithms

The straightforward application of the BP algorithm to non-binary LDPC codes has computational complexity dominated by $O(q^2)$ operations for each check sum processing. For this reason, considering $q \geq 16$ results prohibitively complex in terms of implementation. However, as proposed in [MD99] and [BD03], the sum-product step at the parity-check update can be performed in the frequency domain if $q=2^p$. This FFT-Based BP decoding reduces complexity to $O(d_c p q)$. This algorithm was also described in the log domain [SC03], leading to the so-called log-BP-FFT.

A reduced-complexity decoding algorithm for LDPC codes was presented by Declercq and Fossorier in [DF07]. This algorithm, called the Extended Min-Sum (EMS) algorithm, is based on a generalization of the Min-Sum (MS) algorithm used for binary LDPC codes ([ZZB05], [FMI99], [CF02] and [KFL01]). The basic idea is to use only a limited number n_m of reliabilities in the messages at the input of the check node in order to reduce the computational burden of the check node update. With $n_m \ll q$, the complexity of a check node varies in $O(q \log q)$, using log-density-ratio representations of the messages, without sacrificing too much in performance, even for high order field codes, up to $GF(256)$.

A new implementation of the EMS decoder has recently been proposed in [VDVFU07]. A particularity of this new algorithm is that it takes into account the memory problem of the non binary LDPC decoders, together with a significant complexity reduction per decoding iteration. The key feature of the new EMS decoder is to extend the principle of truncating the vector messages to both the check node and data node inputs. The authors truncate the messages from q to n_m values in an efficient way so as to reduce its impact on the performance of the code. Moreover, they introduce an efficient offset correction to compensate for the performance loss. The complexity of the EMS decoder is now theoretically dominated by $O(n_m \log n_m)$, with $n_m \ll q$, which is an important complexity reduction compared to all existing methods [SC03], [WSM04], [DF07]. However, since parallel insertion is needed for reordering the vector messages, hardware complexity is in practice dominated by $O(n_m^2)$; and this is the complexity level that we will consider for our study.

We present by the following a more detailed analysis of each of these algorithms providing the number of operations in each variable and check node per decoding iteration for an elementary step. We will also consider in our study that the variable node degree is constant, $d_v = 2$, for all the codes presented in D4.1.1.

2.2 BP decoding algorithm

In [DF07] the authors introduce a tensorial representation of the messages along the edges in the factor graph representation of the code and transformations of the usual LDPC factor graph in order to write the BP equations in a simple way. This transformation consists of adding variable nodes corresponding to the multiplication of the codeword symbols by their associated nonzero H values. For sake of simplicity, we also adopt the notation considered in [DF07] (Figure 2-1).

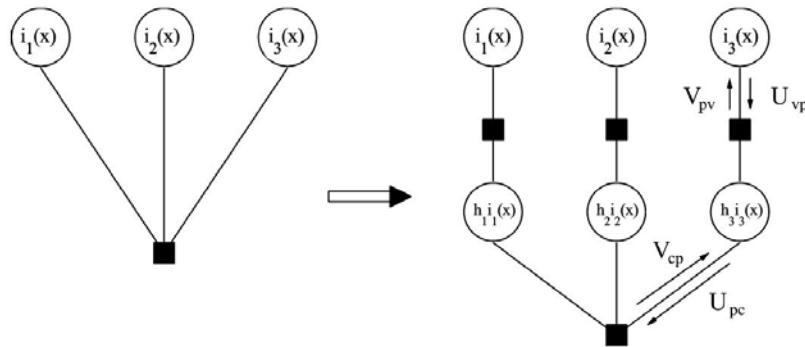


Figure 2-1 : Transformation of the factor graph so as to explicitly represent the nonzero H values (from [DF07])

As in the BP decoding algorithm the check node processing is a convolution of probability densities on $GF(2^q)$, the computational complexity rapidly becomes prohibitively large and the number of basic operations required to compute the parity-check node update (sum-product step) varies exponentially with both the field order q and the parity check degree d_c .

A recursive implementation of the check node update (as presented by Davey and MacKay in [DM98]), can reduce complexity to $O(d_c q^2)$. This backward-forward calculation (same principle as in the BCJR algorithm) for a check node is represented in Figure 2-2 for $d_c = 4$. U_{pi} is the message from permutation node i to the check node and V_{pci} is the message from the check node to permutation node i ($1 < i < d_c$). Note that each \otimes operator performs a direct convolution of two distributions of size q (i.e. q^2 operations). The number of operations needed to implement an elementary step of this algorithm at the variable node and the check node level is presented in Table 2-1, as a function of d_c and particularized for $d_v = 2$ (this characteristic being common to all the benchmark codes from D4.1.1).

We can conclude that, in general, any implementation of the BP algorithm for non-binary LDPC remains thus too complex to allow very-high order fields and q must be limited to 16 for a reasonable hardware implementation.

Table 2-1: Number of operations to perform an elementary step at the check node and variable node in the BP algorithm ($d_v = 2$)

	Number of additions	Number of multiplications	Number of divisions
Variable node	$q-1$	q	q
Check node	$3(d_c - 2) q (q-1)$	$3(d_c - 2) q^2$	0

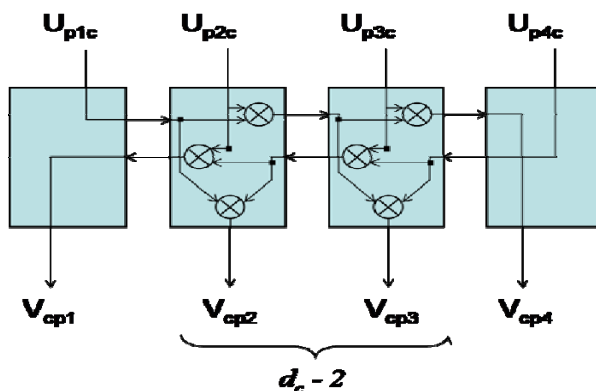


Figure 2-2: Check node backward-forward computation for BP decoding, $d_c = 4$

2.3 FFT-based BP decoding on $GF(2^p)$

In [MD99], the authors propose to perform the computation of the check node update in the frequency domain, which transforms the convolution into a simple product (see also [SC03] and [BD03]). This way, the computational complexity of the check node update is reduced to $O(d_c \cdot p \cdot q)$. Figure 2-3 shows the FFT and the backward-forward calculation for a check node in the FFT-BP decoding algorithm.

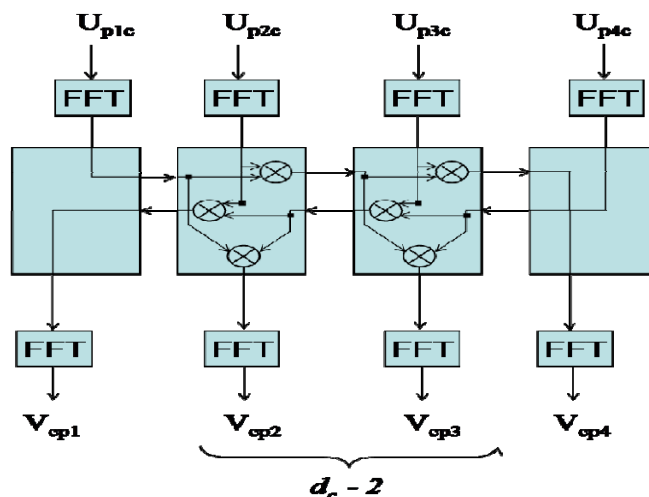


Figure 2-3: Check node backward-forward computation for FFT-based BP decoding, $d_c = 4$

This reduced-complexity BP algorithm allows decoding non-binary LDPC codes in very-high-order fields and for large values of d_c , or equivalently, high code rates, since $R \geq 1 - d_v / d_c$. Table 2-2 gives the number of operations required to perform an elementary step in the FFT BP algorithm ($p = \log_2 q$) for both variable and check nodes. Each \otimes operator performs the convolution of two distributions in the Fourier domain, that is q pairwise multiplications. Each FFT has a complexity of $qp/2$ multiplications and $qp/2$ additions.

Table 2-2: Number of operations to perform an elementary step at the check node and variable node in the FFT-BP algorithm ($d_v = 2$)

	Number of additions	Number of multiplications	Number of divisions
Variable node	$q-1$	q	q
Check node	$d_c qp$	$d_c qp + 3q(d_c-2)$	0

2.4 Decoding in the logarithm domain (log-BP)

The interest of decoding algorithms in the logarithm domain is well-known. First, it transforms the products into simple sums and the normalization of the messages is no longer required. The second reason is that log-domain algorithms are usually more robust to the quantization effects when the messages are stored on a small number of bits [WSM04], [CDEFH05]. The practical decoding algorithms for turbo-codes (max-log-map, max*-log-map) and LDPC codes (MS, logBP) are all expressed in the logarithm domain.

Table 2-3: Number of operations per decoding iteration in the log-BP algorithm

	Number of max*	Number of additions	Number of multiplications
Variable node	$q-1$	q	0
Check node	$3(d_c-2)q(q-1)$	$3(d_c-2)q^2$	0

Table 2-3 presents the number of operations needed to perform an elementary step of the check node and the variable node. The max* operator is defined as: $\max^*(x_1, x_2) = \log(e^{x_1} + e^{x_2}) \approx \max(x_1, x_2)$. Note that this approximation is the one adopted for the simplified algorithms presented in the following section.

Finally, note that the BP FFT algorithm in the logarithm domain combines the advantages of the logarithmic representation for hardware implementation and those of the frequency domain for computational complexity. An analysis of the algorithm in [WSM04] yields that complexity per iteration is $O(Nd, q)$ at the variables nodes and $O(Md, q^2)$ at the check nodes, when the recursive updating of [DM98] is adopted.

2.5 Simplified decoding in the logarithm domain

The algorithms presented in the previous sub-sections are mathematically equivalent and present the same error performance. The main restriction of all these is the limitation on the field order at a reasonable hardware complexity. We present in this section the EMS algorithm which represents an interesting simplified solution for reduced-complexity non-binary LDPC decoding. We focus on two implementations: the one presented in [DF07] and a more recent and simplified one [V07], which mainly constitutes the starting point of the WP6 primary analysis.

The EMS algorithm proposed in [DF07] is a generalization of the MS algorithm used for binary codes, and has the advantage of performing additions only, while using only a limited number of messages for the check-node update. The significant reduction of the complexity of the check node update is due to the fact that only the n_m largest values of the messages at the input of the check node. Another point is that the EMS algorithm can be applied to both regular and irregular non-binary LDPC codes as the simplification is locally performed at the check node.

However, the sub-optimality of the EMS algorithm introduces a performance degradation compared to the BP algorithm. The main reason for this degradation is that the reliabilities computed in the EMS algorithm are overestimated. This causes the suboptimal algorithm to converge too rapidly to a local minimum, which is most often a pseudo-codeword. This behavior has also been observed for binary LDPC codes decoded with MS, as well as for turbo codes decoded with max*-log-map (either parallel or block turbo codes). For binary LDPC codes, a simple technique that is used to compensate for this overestimation is to reduce the magnitude of the messages at the variable-node input by means of a factor or an offset [CDEFH05]. These correction techniques were applied to the EMS algorithm, either using a factor correction or an offset correction [DF07], and simulations showed that the EMS algorithm can approach the performance of the BP-FFT decoder, and even in some cases slightly outperform the BP-FFT decoder.

The complexity of the EMS algorithm is $O(d, n_m q)$ per check node. For values of n_m providing near-BP error performance, this complexity is roughly the same as that of the BP-FFT decoder, but without multiplications or divisions. The EMS algorithm then becomes a good candidate for hardware implementation of non-binary LDPC decoders, because of its reduced complexity and the small or negligible performance degradation it introduces.

A new implementation of the EMS algorithm has recently been proposed by Voicila *et al* in [VDVU07]. This approach aims at reducing both complexity and memory requirements of the EMS non-binary LDPC

decoder. In [DF07], the output messages of the check node are composed of q values, thus the complexity of a single parity check node varies in $O(n_m q)$ and all the messages in the graph are stored with their full representation of q real values, implying high memory requirements. The originality of the new implementation is to store only n_m values in all vector messages (at both variable and check nodes).

As in [DF07], the performance degradation due to the truncation of the messages can be mitigated with a proper compensation of this information loss. The author in [V07] proposes a single scalar offset as correction factor, whose value is optimized with density evolution methods.

Table 2-5 presents the number of operations involved in each variable and check node. In this analysis, we consider practical hardware implementation. n_{op} is the number of necessary steps so that all the n_m values of the output vector are computed. The memory requirements depend linearly on n_m and the complexity is dominated by $O(n_m^2)$, this is why the main challenge is to fix n_m to the minimum value that assures performance approaching the BP decoder.

Table 2-5: Number of operations to perform an elementary step at a variable node (with $d_v = 2$) and a check node with the simplified EMS algorithm

	Number of max	Number of real additions	Number additions in GF(q)
Variable node	$n_m(n_m+2)$	n_m	0
Check node	$3(d_c-2)n_{op}n_m$	$3(d_c-2)(n_{op} + n_m)$	$3(d_c-2)(n_{op} + n_m)$

2.6 Decoding in the bit domain

We have the possibility to perform the whole decoding process in the bit domain. In that case, the size of the message (and thus, memories) will be dramatically reduced from q words to $\log_2(q)$ words. Sub-optimal algorithms like, for example, the Chase-Pyndhia algorithm, can be performed for the check node processor. Theoretically, those types of algorithms are supposed to degrade the performance in the waterfall region but do not affect the error floor region. The idea is too recent to have even rough complexity estimation. In the future, this direction will be studied.

2.7 Conclusion

In this section, we have presented several algorithms for GF(q)-LDPC decoding and a complexity study of their application to the DaVinci benchmark codes provided in D4.1.1. For each algorithm we have detailed the number of operations for an elementary step in a check node and a variable node. This study allowed us to rigorously compare the different algorithms from a hardware perspective.

Tables 2-6 and 2-7 summarize the study presented in Section 2. For each decoding algorithm we provide the number of operations needed to perform an elementary step in both variable and check nodes. In Figure 2-4 we compare the complexity of the FFT-BP and EMS algorithms for the DaVinci benchmark codes (this is $q = 64$ and $d_c = 4, 6, 8, 12$) in terms of number of additions at the elementary check node. The curves show that the EMS algorithm is only interesting for specific values of n_m and these values depend on the code rate (Table 2-8). The value of n_{op} is fixed to $2*n_m$. In practice, values of $n_m < 24$ will be considered for $R=1/2$ and, for $R=5/6$, we will consider $n_m = 8$.

This study motivates us to find new ideas for complexity reduction of the simplified EMS algorithm in order to reduce performance degradation (i.e. if we reduce complexity we can increase n_m and improve performance).

Table 2-6: Number of operations to perform an elementary step in a variable node (with $d_v = 2$) for different decoding algorithms

Decoding algorithm	Number of operations in an elementary step of a variable node				
	Multiplications	Divisions	Max*	Max	Additions
BP	q	q	0	0	q-1
FFT-BP	q	q	0	0	q-1
Log-BP	0	0	q-1	0	q
Simplified EMS	0	0	0	$n_m(n_m+2)$	n_m

Table 2-7: Number of operations to perform an elementary step in a check node (with variable d_c) for different decoding algorithms

Decoding algorithm	Number of operations in an elementary step of a check node			
	Multiplications	Max*	Max	Additions
BP	$3(d_c - 2) q^2$	0	0	$3(d_c - 2) q (q - 1)$
FFT-BP	$d_c qp + 3(d_c - 2)q$	0	0	$d_c qp$
Log-BP	0	$3(d_c - 2)q(q - 1)$	0	$3(d_c - 2)q^2$
Simplified EMS	0	0	$3(d_c - 2)n_{op} n_m$	$3(d_c - 2)(n_{op} + n_m)$ (real) $3(d_c - 2)(n_{op} + n_m)$ (in GF(q))

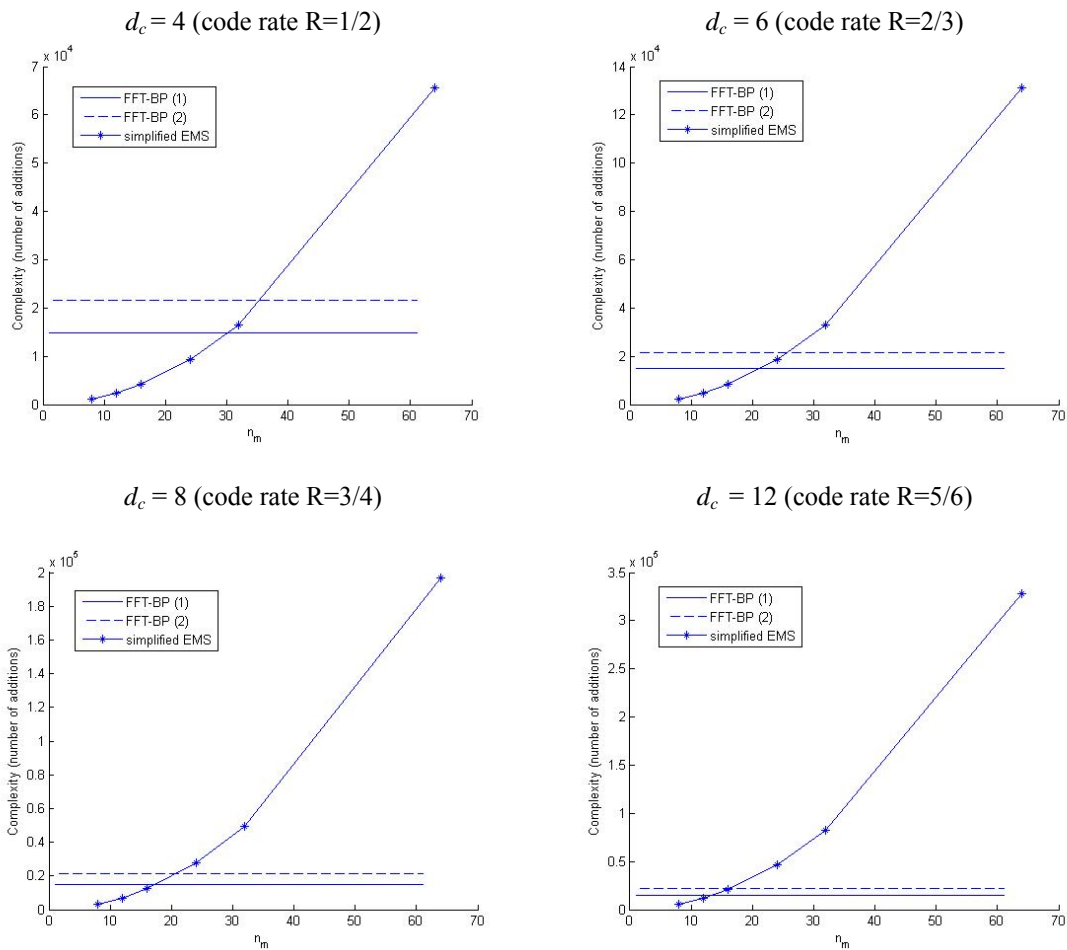


Figure 2-4: Check node complexity comparison in number of additions for the FFT-BP and the EMS algorithms. FFT-BP(1) stands for complexity evaluation considering that 1 multiplication equals 8 additions, idem for FFT-BP(2) 1 multiplication equals 12 additions

Table 2-8: Values of n_m for which the EMS algorithm becomes interesting in terms of complexity

Code rate, R	n_m values
$R = 1/2$	$n_m < 30$
$R = 2/3$	$n_m < 20$
$R = 3/4$	$n_m < 15$
$R = 5/6$	$n_m < 12$

3. Level of parallelism required by DaVinci codes

In this chapter, we give some preliminary results about the level of parallelism required to decode the DaVinci codes. This section is divided in two sub-sections. First we present a theoretical study of the level of parallelism needed as a function of code characteristics, the data throughput and the clock frequency of the decoder. Then, we apply those results to the parameters of the DaVinci codes provided by WP4.

3.1 Theoretical study

In this section, we describe some parameters that characterize the architecture of the non-binary LDPC decoder considering the parallelism aspect.

Let us consider a (d_v, d_c) LDPC defined over $GF(2^p)$ with the following characteristics and specifications (under which the decoder has to be working):

- K , the number of information symbols,
- N , the number of codeword symbols,
- $R = (1 - \frac{d_v}{d_c}) = \frac{K}{N}$, the rate of the code,
- $E = d_v \frac{K}{R}$, the number of edges of the code,
- n_{it} , the maximum number of iterations,
- D_b , the binary throughput. The symbol throughput is defined as $D_s = \frac{D_b}{p}$ (in bit/s),
- $f_c = \alpha D_b$, the clock frequency of the decoder expressed as a multiple of D_b .

The computational power P_c is defined as the average number of edges processed in one clock cycle. The processing of an edge, during an iteration, refers to the transmission of two updated messages through this edge: one from the check node to the variable node, the other from the variable node to the check node (or vice-versa). To fulfil the real time constraint, a codeword should be decoded in the duration of a codeword (or less). The minimum value of P_c can thus be expressed as the ratio between the total number of edge to be processed in a decoding process (i.e. : $E \times n_{it}$) and the total number of clock cycles available during the reception of a new codeword (i.e. $f_c \times (K/D_s) = f_c \times (Kp/D_b)$). Thus:

$$\begin{aligned}
 P_c &= E \times n_{it} \left(\frac{D_b}{K p} \right) \times \frac{1}{f_c} = \frac{d_v K}{R} \times n_{it} \times \left(\frac{D_b}{K p} \right) \times \frac{1}{f_c} \\
 &= \frac{d_v}{p R} \times n_{it} \times \frac{1}{\alpha} [\text{edge / cycle}]
 \end{aligned} \tag{2-1}$$

In the DaVinci benchmark codes, several parameters are already fixed, i.e. $p=6$, and $d_v=2$. Thus, equation (2-1) can be rewritten as:

$$P_c = \frac{1}{3 R} \times n_{it} \times \frac{1}{\alpha} [\text{edge / cycle}] \tag{2-2}$$

Table 3-1 gives the value of P_c for the different code rates R selected in the DaVinci project, the number of iterations fixed to $n_{it} = 8$ and different values of α . This table shows that the level of parallelism is relatively low when speaking about “edge processing”. But, since the decoding is performed in $GF(64)$, an “edge processing” is a rather complex operation. The following section will detail the level of parallelism according to the decoding algorithm.

Table 3-1 : Number P_c of edges to be processed, in average, per clock cycle according to the ratio α between clock frequency and data throughput and the rate R of the code. The number of iterations is fixed to $n_{it}=8$.

	$\alpha=1$	$\alpha=2$	$\alpha=3$	$\alpha=4$
R=1/2	5,34	2,67	1,78	1,33
R=2/3	4	2	1,33	1
R=3/4	3,56	1,78	1,18	0,89
R=5/6	3,2	1,6	1,06	0,8

3.2 Impact of the decoding algorithm

In this section, we determine the number of check node processors (denoted by P) and the number of variable node processors (denoted by V) to process, in average, one edge per clock cycle. Both numbers depend on several parameters:

- the type of decoding algorithm used (see section 3);
- the level of parallelism s used to send the information related to one GF(64) symbol;
- the number e_v of input edges of a variable processor (fixed to 2);
- the number e_c of input edges of a check processor.

First, the decoding algorithm will fix the size N_s of the message carrying the information through an edge. For a BP-like algorithm, $N_s = 64$ and each word of a message contains only the LLR value of the symbol associated to its position. If the EMS algorithm is used, then $N_s = n_m$, and each “word” is then a couple that contains a GF(64) symbol (coded on 6 bits) and its associated LLR value. Finally, for a binary decoder, $N_s = 6$ and each word is the associated LLR of one bit. Thus, if a processor has s entries to receive (respectively send) a given message, this message can be received (respectively sent) in N_s/s clock cycles.

If a check node processor has e_c entries (input and output) to receive/send edge messages, then in N_s/s clock cycles, the check node processor is able to receive/send a maximum of e_c edge messages in N_s/s clock cycles. In a clock cycle, the check node processor will then input/output $e_c s/N_s$ edges. The minimum number P of check node processors required to process P_c edge per clock cycle is then:

$$P = \frac{P_c}{e_c s} \times N_s \quad (2-3)$$

If a check node processor has a single entry ($e_c = 1$) with a level of parallelism 1 ($s = 1$), and the EMS algorithm is used with $n_m=12$ for a rate 1/2 code, $n_{it}=8$ and $\alpha=1$, then the minimum number of check node processors to be implemented is $P=5.34 \times 12 = 64$. The minimum number V of variable node processor can be derived in a similar way.

3.3 Conclusion

In this section, a theoretical study defines the minimum number of check node processors and variable node processors required to fulfill the real time requirements of the DaVinci codes. The level of parallelism is an important factor for the architecture but also for the design of the code. In fact, the problem of multiple memory access and shuffle network can be simplified if the structure of the code integrates a priori the architectural constraints.

4. Some methods to evaluate performance of sub-optimal decoding algorithms

This section describes three fast methods to evaluate the performance of the EMS algorithm when varying the message length n_m and the number of iterations n_{it} . The first method is based on a Reduced Monte-Carlo Simulation where we consider the decoding of a set of frames with initial values of n_m and n_{it} and then re-decoding the resultant erroneous frames with new values of n_m and n_{it} . The second method is based on the prediction of the performance by evaluating a new metric indicating the distance between the probability distributions of the optimal and suboptimal algorithms. This distance is evaluated by processing the output messages of the optimal and suboptimal decoders corresponding to the same inputs.

The third method is based on the same previous approach but instead of treating the inputs and outputs of the whole LDPC decoder, we simply consider the check node and we treat the inputs and the outputs of the check nodes to evaluate the distance between the optimal and suboptimal checks.

4.1 Reduced Monte-Carlo Simulation (RMCS)

The idea is to consider the two principal parameters that impact the performance and the complexity of the LDPC decoder with the EMS algorithm: the parameter n_m and the maximum number of iterations n_{it} . These parameters interact in a nonlinear way, thus the problem of finding a good performance-complexity trade-off seems to be a complex task. Moreover, the evaluation of each configuration (n_m, n_{it}) requires an extensive Monte-Carlo Simulation (MCS) in order to have an accurate estimation of the FER . In order to avoid such MCS, we propose an efficient pragmatic approach leading to the Reduced Monte-Carlo Simulation (RMCS) method. This method is adapted from [BDM07].

Let F be a set of $N=\text{card}(F)$ received codewords at a given SNR. Each frame X of F is sent to a LDPC decoder using the EMS algorithm with parameter (n_{it}, n_m) (noted $\text{EMS}(n_m, n_{it})$ by the following). The result of the decoding process can be either successful or failed. Thus, $\text{EMS}(n_m, n_{it})$ partitions the set F in two distinct sets, the subset $\Phi_d^{n_m, n_{it}}$ of successfully decoded frames of F using $\text{EMS}(n_m, n_{it})$ and the subset $\Phi_u^{n_m, n_{it}}$ of unsuccessfully decoded frames of F using $\text{EMS}(n_m, n_{it})$. Thus,

$$F = \Phi_d^{n_m, n_{it}}(F) \cup \Phi_u^{n_m, n_{it}}(F),$$

Note that the FER can be estimated as:

$$FER = \frac{\text{card}(\Phi_u^{n_m, n_{it}}(F))}{N}$$

Now, assume the (n'_m, n'_{it}) verifies $n'_m \geq n_m$ and $n'_{it} \geq n_{it}$. Then, the decoder $\text{EMS}(n'_m, n'_{it})$ is supposed to be superior to the decoder $\text{EMS}(n_m, n_{it})$, since the number of iterations is greater and the number of messages of the EMS algorithm is also greater. Thus, we can make the hypothesis **H**:

Hypothesis 1: if a frame is correctly decoded with $\text{EMS}(n_m, n_{it})$, then this frame will be also correctly decoded with $\text{EMS}(n'_m, n'_{it})$ if $n'_m \geq n_m$ and $n'_{it} \geq n_{it}$.

By this hypothesis, we consider that the frames successfully decoded with the configuration (n_m, n_{it}) will also be successfully decoded with the configuration (n'_m, n'_{it}) :

$$\begin{aligned} \Phi_d^{n_m, n_{it}}(F) &\subset \Phi_d^{n'_m, n'_{it}}(F) \\ \Phi_u^{n_m, n_{it}}(F) &\supset \Phi_u^{n'_m, n'_{it}}(F). \end{aligned}$$

In order to estimate the Frame Error Rate $FER(n'_m, n'_{it})$ of $\text{EMS}(n'_m, n'_{it})$, it is useless to test all the frames of $\Phi_d^{n_m, n_{it}}$, since, according to hypothesis 1, those frames will be successfully decoded. Thus,

$$FER(n'_m, n'_{it}) = \frac{\text{card}(\Phi_u^{n'_m, n'_{it}}(\Phi_u^{n_m, n_{it}}))}{\text{card}(F)}$$

Since $\text{card}(\Phi_u^{n'_m, n'_{it}})$ can be a several orders of magnitude lower than $\text{card}(F)$, the simulation time to estimate $FER(n'_m, n'_{it})$ can be dramatically reduced compared to a direct Monte Carlo simulation.

This method, named Reduced Monte Carlo Simulation (RMCS) is described bellow.

- **Step 1:** fix $\text{card}(F)$ and the two extreme configurations the smallest and biggest reasonable values for (n_m, n_{it}) (i.e. “worst case” and “best” possible case).
- **Step 2:** start the MCS at a given *Signal to Noise Ratio (SNR)* with the “worst-case” configuration (n_m, n_{it}) . During simulation, if a codeword is not successfully decoded, store its corresponding seed of the Pseudo-Random Generator (PRG). The resultant set of seed will serve to regenerate the same frames to be re-decoded with new (n_m, n_{it}) configuration.

- **Step 3:** iterate all the possible (n_m, n_{ii}) configurations up to the best possible one.
- **Step 4:** choose the configuration (n_m, n_{ii}) giving the best performance-complexity trade-off.
- **Step 5:** perform a classical MCS in order to verify the validity of hypotheses 1 for some (n_m, n_{ii}) configurations.

With this method, the MCS time is dramatically reduced since the test of a new configuration can be a few orders faster than the classical MCS. For example, for a Frame Error Rate (*FER*) equal to 10^{-5} , at least 10^6 frames should be simulated. With the RMCS method, testing a new (n_m, n_{ii}) configuration at the same *FER* (and *SNR*) requires only the decoding of the frames regenerated by the set of PRG whose cardinality is $\ll 10^6$.

In the frame of the DaVinci projet, we experiment this method for a 1/2 rate GF(64) of length 502 for a SNR of 1.8 dB. The initial set F as size of $N=10^6$. The first simulation with EMS(8,8) gives a set $\Phi_u^{12,8}(F)$ of cardinality of 49693 (i.e. an estimate FER(8,8) of $4.9 \cdot 10^{-2}$). Then, starting from $\Phi_u^{8,8}(F)$, the $FER(n_{ii}, n_m)$ for n_{ii} and n_m varying from 8 to 16 have been tested using the RMCS.

In order to further reduce the complexity, we apply recursively the RMCS. For example, to estimate $\Phi_u^{12,9}(F)$, we start directly from $\Phi_u^{12,8}(F)$ instead of the initial $\Phi_u^{8,8}(F)$ set.

Table 4-1 shows the simulation results. As expected, at a fixed n_m the number of erroneous frames decreases with the increase of n_{ii} . On the other hand, increasing n_m will improve the performance by decreasing the frame error rate with only few exceptions. As we remark, there are some cases (figures highlighted in red) where the number of erroneous frames at $n_m + 1$ is higher than that at n_m . For example, at $n_{ii}=11$, the number of erroneous frames at $n_m=23$ is 32 and that at $n_m=24$ is 34. This can be explained by the fact that the arbitrary truncation of messages and the approximated factor of compensation (offset) used in the algorithm do not match the theoretical reasoning at a low number of simulated frames.

Table 4-1: Number of erroneous frames with the reduced decoding approach

n_{ii}	8	9	10	11	12	13	14	15	16
8	49693	24452	13535	8077	5404	3916	2993	2478	2105
9	22917	10717	5388	3006	1878	1341	999	793	691
10	12340	5473	2569	1356	815	536	414	339	291
11	7258	3202	1414	714	411	273	195	150	119
12	4763	2147	893	453	270	164	109	73	62 (<100)
13	3164	1354	609	604	312	185	127	91	60
14	2419	1066	467	254	163	105	72	52	42
15	1827	738	317	168	97	56	38	25	21
16	1349	544	223	120	56	32	22	18	13
17	1020	427	178	78	40	25	19	12	8 (<10)
18	835	367	151	65	25	14	10	5	4
19	693	313	117	45	17	9	7	6	6
20	592	288	116	47	17	9	7	6	6
21	515	274	109	44	18	9	6	4	4
22	455	226	103	37	14	7	4	3	3
23	410	226	94	32	12	5	5	4	3
24	373	214	86	34	14	9	5	4	4
25	348	204	97	39	15	8	5	4	4
26	321	189	91	45	16	6	4	3	3
27	281	136	54	15	9	5	5	3	3
28	262	133	54	13	9	5	5	4	3
29	245	120	53	18	10	6	6	5	5
30	235	127	52	16	8	4	4	3	3
31	234	120	47	14	7	3	3	2	2
32	228	115	52	12	7	5	4	2	2

4.1.1 Offset simulations

In order to determine the best offset that corresponds to each message length n_m , we have simulated the $\Phi_u^{12,8}(F)$ with $n_{it}=8$ and for $n_m=9, \dots, 32$. Table 4-2 shows the simulation results. As we remark, when n_m increases, the corresponding offset decreases and tends to become zero when $n_m=64$.

Table 4-2: Offset values for various message lengths n_m

Offset	n_m
1,5	$8 \leq n_m \leq 13$
1,0	$14 \leq n_m \leq 20$
0.5	$21 \leq n_m \leq 32$

4.1.2 Verification of results

In order to verify the correctness of the reduced RMCS method, we have simulated the same number of frames (10^6 frames used in the creation of the $\Phi_u^{12,8}(F)$) for some values of n_m at different n_{it} to be compared with the corresponding figures of the RMCS.

Table 4-3 Comparison between some figures of classical simulation and RMCS

n_{it}	8		12		16	
	Classical MCS	RMCS	Classical MCS	RMCS	Classical MCS	RMCS
16	1735	1349	71	56	22	13
24	586	373	21	14	8	4
32	471	228	19	7	8	2

As shown in Table 4-3, the results predicted by the reduced decoding approach are close to those obtained with the classical approach. The comparison of these two approaches makes it possible to validate the RMCS method, which we plan to use to design the EMS non-binary LDPC decoder.

4.1.3 Conclusion

The RMCS method significantly accelerates simulation. For example, to test the different configurations ($n_m=32, n_{it}$), we have to simulate at the most 228 frames instead of 10^6 frames. Then an improvement speed by a factor of order 10^4 is obtained. This method was validated by performing the classical MCS for some (n_m, n_{it}) configurations.

4.2 Distance-based method for performance prediction

Non-binary LDPC codes use message vectors that can be represented according to two useful representations: probability vectors and Log Likelihood Ratio (LLR) vectors.

A q -dimensional probability vector $P=(p_0, p_1, \dots, p_{q-1})$ is a vector of real numbers such that $p_i > 0$ for

all i and $\sum_{i=0}^{q-1} p_i = 1$. The LLR vector associated to P is $W=(w_0, w_1, \dots, w_{q-1})$ with

$$w_i = \log \frac{p_i}{p_0}, i = 0, \dots, q-1.$$

In this method we consider the estimation of the distance between the probability distributions of the symbol LLRs of the optimal and suboptimal algorithms. The metric we consider to evaluate this distance is inspired from the entropy principle. Using the optimal algorithm, two databases constituting the probability symbols are created. The first one corresponds to the received codeword intrinsic probabilities and the second corresponds to the extrinsic probability values associated to the decided symbol at the variable nodes. The intrinsic probability database will be processed with the sub-optimal algorithm to generate a set of extrinsic probability databases which will constitute together with the optimal extrinsic probabilities the inputs of our distance evaluation system (Figure 4-1).

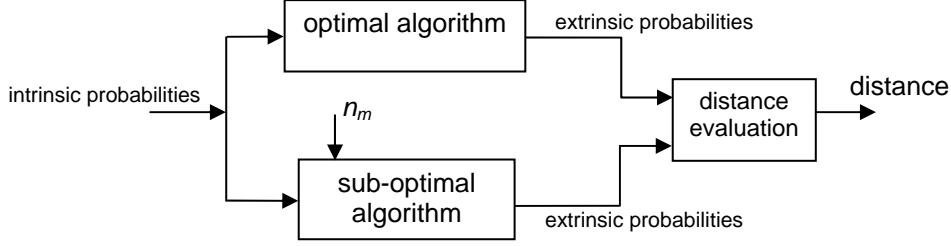


Figure 4-1: diagram block of the distance evaluation system

The algorithm we simulate deals with messages in LLR representations, therefore the probability vectors can be expressed as a function of the LLR values:

$$p_i = \frac{e^{w_i}}{e^{w_0} + e^{w_1} + \dots + e^{w_{q-1}}} \quad (4-1)$$

where w_i is the symbol LLR for $i = 0, \dots, q-1$.

The quantification of the sub-optimality impact is inspired from the measure of the mutual entropy and defined as:

$$d(D, \tilde{D}) = \frac{\sum_{n=0}^{N-1} \sum_{i=0}^{q-1} (p_{ni} - \tilde{p}_{ni}) \left(\log_2 \left(\left| p_{ni} - \tilde{p}_{ni} \right| \right) \right)^\beta}{\sum_{n=0}^{N-1} \sum_{i=0}^{q-1} p_{ni} \log(p_{ni})^\beta} \quad (4-2)$$

where D and \tilde{D} are two sets of N q -dimensional vectors corresponding to the symbol probabilities p_{ni} and \tilde{p}_{ni} of optimal and suboptimal algorithms respectively. The probability values considered here are the values that correspond to the decided symbols at the variable nodes.

We have generated two sets of 100 64-dimensional vectors with the optimal algorithm,. The first set corresponds to the intrinsic probability values of 100 frames successfully decoded with 20 iterations. The second corresponds to the extrinsic probability values representing the decided symbol probabilities at the variable nodes.

The task now is to process the set of intrinsic probabilities with the suboptimal algorithm using different values of n_m and then evaluate the distance between the optimal and suboptimal algorithm for each n_m .

Figure 4-2 shows the variation of the distance with n_m for two values of β : $\beta = 1$ and $\beta = 2$. The curves have an hyperbolic shape, which means that for small values of n_m the distance between the suboptimal and optimal algorithms is important and significantly decreases between two successive values of n_m . The two algorithms become closer when increasing n_m . The objective is then to find the smallest value of n_m that guarantees a small distance between optimal and suboptimal decoding.

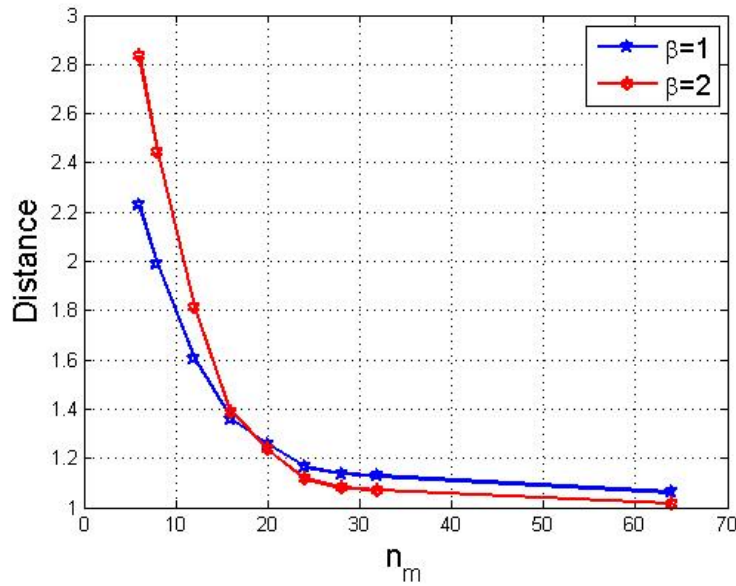


Figure 4-2 : Variation of the distance between the optimal and suboptimal algorithms with n_m

In order to validate this distance-based method, we have established a correlation between the distance function and the FER analysis. For this, we have evaluated the FER log ratio of the suboptimal and optimal algorithms. This FER log ratio is denoted by ΔFER and defined as:

$$\Delta FER = \log_{10} \left(\frac{FER_{sub}}{FER_{opt}} \right) \tag{4-3}$$

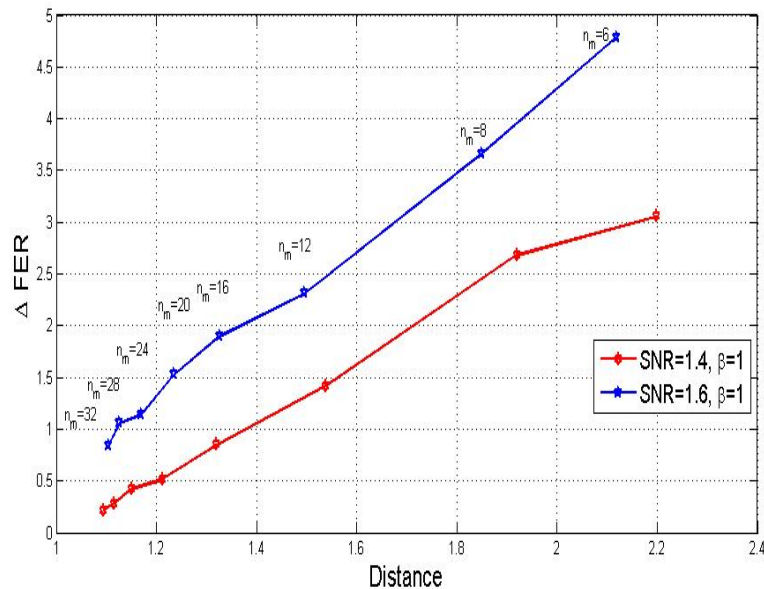


Figure 4-3: Correlation between the variation of the distance and of the FER with n_m , at SNR=1.4 and 1.6 dB.

Figure 4-3 illustrates the variation of the couple $(\Delta FER, d)$ for $n_m=6, 8, 12, 16, 20, 24, 28$ and 32 . It can be seen from this figure that the variation of the distance follows the variation of ΔFER which decreases with the increase of n_m . For example, for SNR=1.4 and $n_m=32$ the distance and the ΔFER have small values (1,05 and 0,2 respectively) and attain values around 3 when $n_m=6$.

This distance-based method is efficient since it rapidly provides a good estimation of the impact of the message length n_m on the decoder performance. Then, using this method one can avoid the extensive MCS.

4.3 Distance-based method for performance prediction at the check node level

With this method, we apply the same previous approach by evaluating the distance at the level of the check node. This permits to evaluate the impact of the sub-optimality independently of the characteristic of the LDPC code. We have created two databases constituting the symbol probabilities at the input and the outputs of the checks nodes associated to the optimal algorithm. The set of input messages is treated with the suboptimal algorithm where various message lengths n_m are considered. This method is under development and some results will soon be provided.

5. Conclusion and further work

In this deliverable, we first presented a brief state-of-the-art of decoding algorithms for non-binary LDPC codes. For each algorithm, a realistic complexity evaluation was given in terms of number of operations. We then presented a study of the level of parallelism required to decode in real time the DaVinci benchmark codes. This theoretical study will help the joint design of code and architecture of the DaVinci codes. Finally, we present preliminary results on the development of “pragmatic” techniques to tune finely the trade-off complexity versus performance of the decoder architecture. A paper that combines some results on the DaVinci codes and an independent study on Duo-Binary turbo codes with the distance-based method for performance prediction will be proposed for publication.

Finally, some work is currently being done on new ideas for simplified EMS decoding of non-binary LDPC. They concern the elementary step of a check node (bubble-check approach) and a new approach to the check node processing (total sum approach).

References

- [BD03] L. Barnault and D. Declercq, “Fast decoding algorithm for LDPC over GF,” *Proc. Inf. Theory Workshop* Paris, France, Mar. 2003, p. 70
- [BDM07] E. Boutillon, C. Douillard, G. Montorsi, “Iterative Decoding of Concatenated Convolutional Codes: Implementation Issues”, *Transactions of the IEEE*, vol. 95, n°6, June 2007
- [CDEFH05] J. Chen, A. Dholakia , E. Eleftheriou , M. Fossorier and X.-Y. Hu “Reduced-complexity decoding of LDPC codes,” *IEEE Trans. Commun*, vol. 53, Issue 8, pp. 1288-1299, Aug. 2005
- [CF02] J. Chen and M. Fossorier, “Density evolution for two improved BP-based decoding algorithms of LDPC codes,” *IEEE Commun. Lett.*, vol. 6, pp. 208, May 2002
- [DM98] M. Davey and D. J. C. MacKay, “Low density parity check codes over GF(q),” *IEEE Commun. Lett.*, vol. 2, no. 6, pp. 165–167, Jun. 1998
- [DF07] D. Declercq and M. Fossorier, “Decoding algorithms for Nonbinary LDPC codes over GF(q)”, *IEEE Trans. Comm.*, Vol. 55, No. 4, April 2007
- [FMI99] M. Fossorier , M. Mihaljević and H. Imai, “Reduced complexity iterative decoding of LDPC codes based on belief propagation,” *IEEE Trans. Commun.*, vol. 47, pp. 673, May 1999.
- [KFL01] F. Kschischang, B. Frey and H.-A. Loeliger, “Factor graphs and the sum product algorithm,” *IEEE Trans. Inf. Theory*, vol. 47, pp. 498, Feb. 2001
- [MD99] D. J. C. MacKay and M. Davey, “Evaluation of Gallager codes for short block length and high rate applications,” *Proc. IMA Workshop Codes, Syst., Graphical Models*, 1999
- [SC03] H. Song and J. R. Cruz, “Reduced-complexity decoding of Q-ary LDPC codes for magnetic recording,” *IEEE Trans. Magn.*, vol. 39, pp. 1081-1087, Mar. 2003

-
- [VDVFU07] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, P. Urard, "Low-complexity, Low-memory EMS algorithm for non-binary LDPC codes," *Proceedings of ICC'2007*, pp. 671-676, Glasgow, UK, June 2007
- [WSM04] H. Wymeersch, H. Steendam, and M. Moeneclaey, "Log-domain decoding of LDPC codes over $\text{GF}(q)$," in *Proc. IEEE Int.Conf. Commun.*, Paris, France, Jun. 2004, pp. 772-776
- [ZZB05] J. Zhao, F. Zarkeshvari and A. H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding LDPC codes," *IEEE Trans. Commun.*, vol. 53, pp. 549, Apr. 2005