

INFSO-ICT-216203 DAVINCI

D4.3

Generalized Non-Binary LDPC Codes: Split-LDPC Codes (or Multi-Binary GLD Codes)

Contractual Date of Delivery to the CEC:	30 June 2009 <i>(as specified in the contract)</i>
Actual Date of Delivery to the CEC:	18 June 2009
Author(s):	David Declercq, Charly Poulliat and Weigang Chen
Participant(s):	ENSEA
Workpackage:	WP4
Estimated person months:	8
Security:	PU
Nature:	R
Version:	1.0
Total number of pages:	18

Abstract: In this deliverable, we introduce a generalization of linear families of non binary LDPC codes. We do not restrict the parity check equations to be defined on a finite Galois field. Instead we allow more general non binary parity check equations which can be interpreted in the scope of finite Abelian groups build from extensions of the binary group F_2 . We present the structure of this new general class of LDPC codes with binary images (or mappings) of the codeword symbols, codeword sub-symbols and parity-check matrix non zero values.

This class of codes is a generalization of the one introduced in the litterature. The theoretical study of this class of codes is at its early stage, and we present in this deliverable only preliminary results. We have however conducted a huge number of Monte-Carlo simulations in order to give the first hints of the Split-LDPC Codes performance compared to the classical non binary Field codes. More theoretical and practical results on this class of codes will be reported in a future DAVINCI-WP4 deliverable, including the hardware implementation issues and their rate-compatible adaptation.

Keyword list: DA VINCI, deliverable, internal report.

Disclaimer:

Report on Generalized NonBinary LDPC Codes: Split-LDPC Codes (or Multi-Binary GLD Codes)

David Declercq, Charly Poulliat and Weigang Chen

Abstract

In this deliverable, we introduce a generalization of linear families of non binary LDPC codes. We do not restrict the parity check equations to be defined on a finite Galois field. Instead we allow more general non binary parity check equations which can be interpreted in the scope of finite Abelian groups build from extensions of the binary group \mathbb{F}_2 . We present the structure of this new general class of LDPC codes with binary images (or mappings) of the codeword symbols, codeword sub-symbols and parity-check matrix non zero values.

This class of codes is a generalization of the one introduced in [6]. The theoretical study of this class of codes is at its early stage, and we present in this deliverable only preliminary results. We have however conducted a huge number of Monte-Carlo simulations in order to give the first hints of the Split-LDPC Codes performance compared to the classical non binary Field codes. More theoretical and practical results on this class of codes will be reported in a future DAVINCI-WP4 deliverable, including the hardware implementation issues and their rate-compatible adaptation.

CONTENTS

I	Definition and Structure of Generalized Non Binary LDPC codes	2
II	Issues related to the class of NB-Split-LDPC codes	5
III	Discussion on the Decoding of NB-Split-LDPC codes	8
IV	Minimum Distance Properties of Designed DaVinci and Split-LDPC Codes	9
	IV-A Component Code Selection	9
	IV-B Code Optimization with Random Permutations of One Component Code	9
V	Frame Error Rate Results with FFT-Belief Propagation	12
	References	14

I. DEFINITION AND STRUCTURE OF GENERALIZED NON BINARY LDPC CODES

In this first section, we will describe in more details what we define as a structure of Generalized Non Binary LDPC code. Both the algebraic definition and the structure of the associated Tanner graph are explained.

An LDPC code is linear block code defined on a very sparse parity-check matrix \mathbf{H} with the dimensions of $M \times N$, which can be defined over the binary Galois field or high order Galois fields. Let $\mathbf{x} = [x_0 \dots x_{N-1}]$ be a codeword. If the code is defined over a finite field $\text{GF}(q)$ with $q = 2^p$, the i -th parity check equation can also be written as

$$\sum_{j:h_{ij} \neq 0} h_{ij}x_j = 0 \text{ in } \text{GF}(q)$$

where h_{ij} are non-zero elements from $\text{GF}(q)$. This definition can be generalized to the case of non-binary code ensembles defined over the general linear group [1] or more generally to the case of codes defined over the finite Abelian group $\mathbb{G}(2^p) = \mathbb{F}_2^p$ [2]. Defined in these kinds of sets, the i -th parity check equation can be written as

$$\sum_j h_{ij}(x_j) = 0 \text{ in } \mathbb{G}(2^{p_2}) \quad (1)$$

where $h_{ij} : \mathbb{G}(2^{p_1j}) \rightarrow \mathbb{G}(2^{p_2})$ is a linear function. This definition is very general and encompasses the classical non binary codes as special cases. Since the functions $h_{ij}(\cdot)$ are linear, they can be represented by a binary input matrix of size (p_1j, p_2) with full column-rank p_1j . In the following, the binary matrix representing the functions $h_{ij}(\cdot)$ will be called *clusters*, and will be denoted H_{ij} . In principle, the values of the column dimensions of the clusters could be different for a single parity check defined as (1), as described in details in [3], [4], and as long as $p_1j \leq p_2, \forall j$. Note that generalized NB-LDPC codes are no longer defined on a single finite field or a single finite group, which makes them difficult to describe algebraically. In order to simplify the presentation of the codes, in this deliverable, we will describe the structure of generalized NB-LDPC codes with the binary image of the parity check matrix \mathbf{H}_{bin} build from the binary (p_1j, p_2) clusters H_{ij} . The binary image \mathbf{H}_{bin} corresponds to a binary linear code which is locally dense (since images of the clusters have a lot of length-4 cycles), but which is cluster-wise sparse. This means that a Tanner graph representing a generalized NB-LDPC code will be sparse only if the edges connecting the symbol nodes and the check nodes represent connexions between the clusters of the functions $h_{ij}(\cdot)$. In such Tanner graph, the symbol nodes represent elements of a finite group $\mathbb{G}(2^{p_1j})$, or equivalently a vector of p_1j bits.

For simplicity, let us call *sub-symbol*, a symbol in $\mathbb{G}(2^{p_1j})$ with $p_1j < p_2$. In case $p_1j = p_2$, the sub-symbol is simply called a *symbol*. We associate to a sub-symbol x_j a binary map composed of p_1j bits. Any choice of mapping is allowed, and we do not discuss in this report the impact of a specific choice of mapping on the implementation of the codec. In the scope of the DaVinci EMS decoder, defining a binary map is equivalent to specifying in which order the sub-symbols (or the symbols) are stored in the chip memory. Therefore, we will choose an arbitrary binary map for each set $\mathbb{G}(2^{p_1j})$. In order to be compliant with codes defined on fields, we choose the binary map corresponding to the Galois field with same number of elements, and defined with the minimal root polynomial of the field.

Interpreting an NB parity check equation with binary images is equivalent to considering a binary component code for which the parity check equation has p_2 rows and $\sum_j p_1j$ columns. Additionally, the linear functions $h_{ij}(\cdot)$ corresponding to (p_1j, p_2) nonzero clusters H_{ij} are one-to-one functions which associate a binary map of p_1j bits to a binary map of p_2 bits. Figure 1 gives the example of a cluster with $(p_1j, p_2) = (3, 4)$ and draw the associated function.

Therefore, the i -th NB-parity-check equation can be interpreted as a binary component code, for which the parity-check matrix H_{cc} is composed of the concatenation of the d_c binary clusters representing the

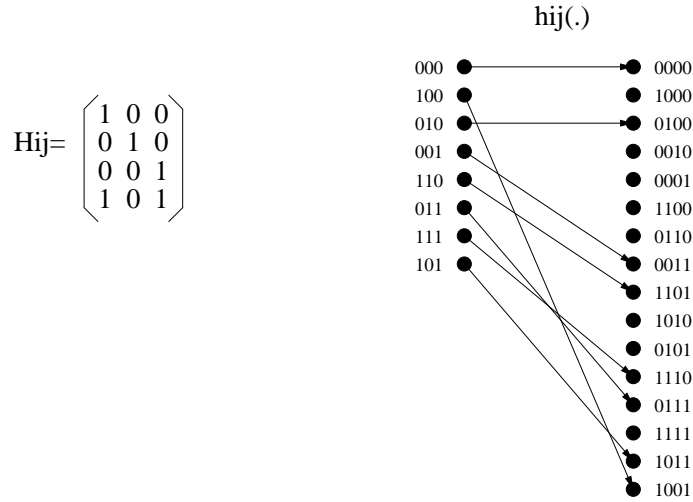


Figure 1. A linear function from $\mathbb{G}(8)$ to $\mathbb{G}(16)$ associated to a full column-rank rectangular cluster. The cluster has full rank equal to 3 and therefore correspond to a one-to-one mapping of vectors of 3 bits into vectors of 4 bits. The mappings of $\mathbb{G}(8)$ and $\mathbb{G}(16)$ correspond to the same binary mappings of the Galois Fields with same cardinality and minimal root polynomial.

$h_{ij}(\cdot)$:

$$H_{cc_i} = [H_{i1}H_{i2} \dots H_{id_c}] \quad (2)$$

Using the above defined notations for the binary clusters and \mathbf{x}_j for the binary mappings of the symbols (or sub-symbols), we can re-write the generalized parity-check equation (1) using a binary vector notation:

$$\sum_{j:\mathbf{H}_{ij} \neq \mathbf{0}} \mathbf{H}_{ij} \mathbf{x}_j^T = \mathbf{0}^T \quad (3)$$

The figure 2 represents the binary images of 2 codes with same parameters. The binary parity check matrix of the component code is then split in d_c clusters of possibly equal or different sizes p_{1_j} . For this reason, we have called this family of generalized NB-LDPC codes “*Split NB-LDPC codes*”. Please note that both the global code rates and dimension are equal for the two codes, as well as the average binary density (number of ones in the parity check matrix \mathbf{H}_{bin}), but the Tanner graphs are different. This means that the algebraic properties of these two codes are different, and so is the behavior of a message passing decoder on the associated Tanner graphs.

There are some extreme cases with this family which correspond to already existing classes of LDPC codes:

- 1) if $p_{1_j} = p_2 \forall j$, the clusters are square matrices and the code is a strictly speaking non binary LDPC code. If the clusters are binary *companion matrices* of the non zeros element of a finite Galois field, then the non-binary code is the usual non binary LDPC code type [5], [15]. If the clusters are more general full rank square matrices, then the LDPC code is a non binary code defined over the general linear finite group $\mathbb{G}(2^{p_2})$ [1], [7].
- 2) if $p_{1_j} = 1 \forall j$, then the clusters are columns vectors of length p_2 bits, and the Split NB-LDPC codes reduce to the class of Generalized Low-Density Codes (GLD codes) introduced and studied in [16]. Indeed, a parity-check is composed of the concatenation of column vectors (2) and form a component code (Hamming or BCH for example) if the linear functions $h_{ij}(\cdot)$ are choosen appropriately. The classes of Split NB-LDPC codes with $p_{1_j} = 1$ and GLD codes are then equivalent, and so are their message passing decoder, based on optimal local extrinsic computation (Belief Propagation equations and BCJR equations are the same in this case), and bit-wise message passing.
- 3) in [6], a special case of Split NB-LDPC codes is proposed, which is different to the extreme cases just presented. In this paper, the authors have proposed to use as component code (2) a classical non binary parity check defined on a Galois field, and then to consider rectangular clusters formed

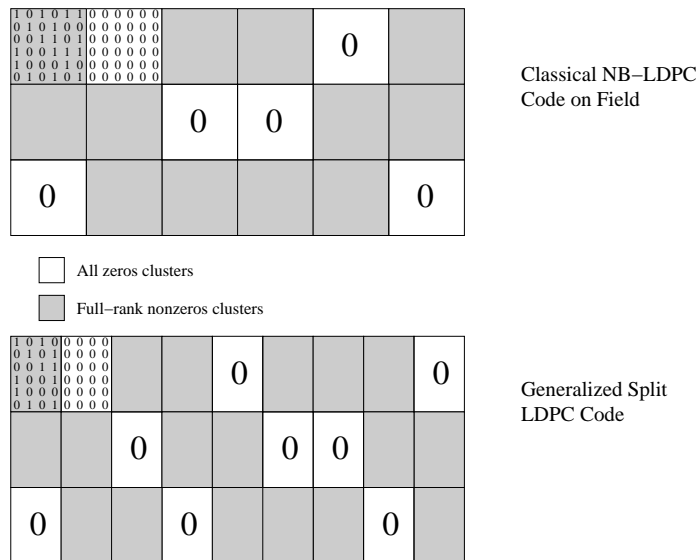


Figure 2. Binary Images of Non Binary LDPC codes with different cluster sizes. Both correspond to the same code rate and codeword length, and both correspond to a strictly regular ($d_v = 2, d_c$) Tanner graph. In the first case, the clusters are squares, and the order of check nodes in the Tanner graph is $d_c = 4$. In the second case, the clusters are rectangular, and the order of check nodes in the Tanner graph is $d_c = 6$ (more dense).

with a set of columns of the field elements compaignon matrices. In this special case, not only the component code is split in clusters, but the binary images of the field non-zero values as well. One can easily see that this is only a special case of the general class presented in this deliverable, both in terms of the component code selection, and in terms of the split structure.

The class of Split NB-LDPC codes is very general, and could lead to very ineteresting codes for which the NB-Tanner graph is sparse enough to allow efficient message passing decoding, together with better algebraic properties than NB-LDPC codes defined on fields. However, this new class is also much more difficult to analyse and to design than binary or non binary LDPC codes. In order to limit the scope of the study, we will consider in this report only 2 cases of parity checks defined with clusters of size $(3, 6)$ and $(4, 6)$. We have drawn on figure 3 the Tanner graphs of a parity check equation in $\mathbb{G}(64)$ split in two different manners. Both correspond to an equivalent binary component code of length $n_b = 24bits$ and dimension $k_b = 18bits$. These parity check equations will be used to build $R = 1/2$ codes with DAVINCI constraints (target codeword lengths). We will not discuss more general Tanner graph structures than the ones on figure 3. More general structure include connexion irregularity at the sub-symbol node side (different than $d_v = 2$), or non-uniform/non-regular split of the component codes. On figure 3, the following two types of Split NB-LDPC codes are considered:

- type \mathcal{C}_2 Each component code is split in $d_c = 6$ sub-symbols composed of 4 bits, that is in $\mathbb{G}(16)$.
- type \mathcal{C}_3 Each component code is split in $d_c = 8$ sub-symbols composed of 3 bits, that is in $\mathbb{G}(8)$.

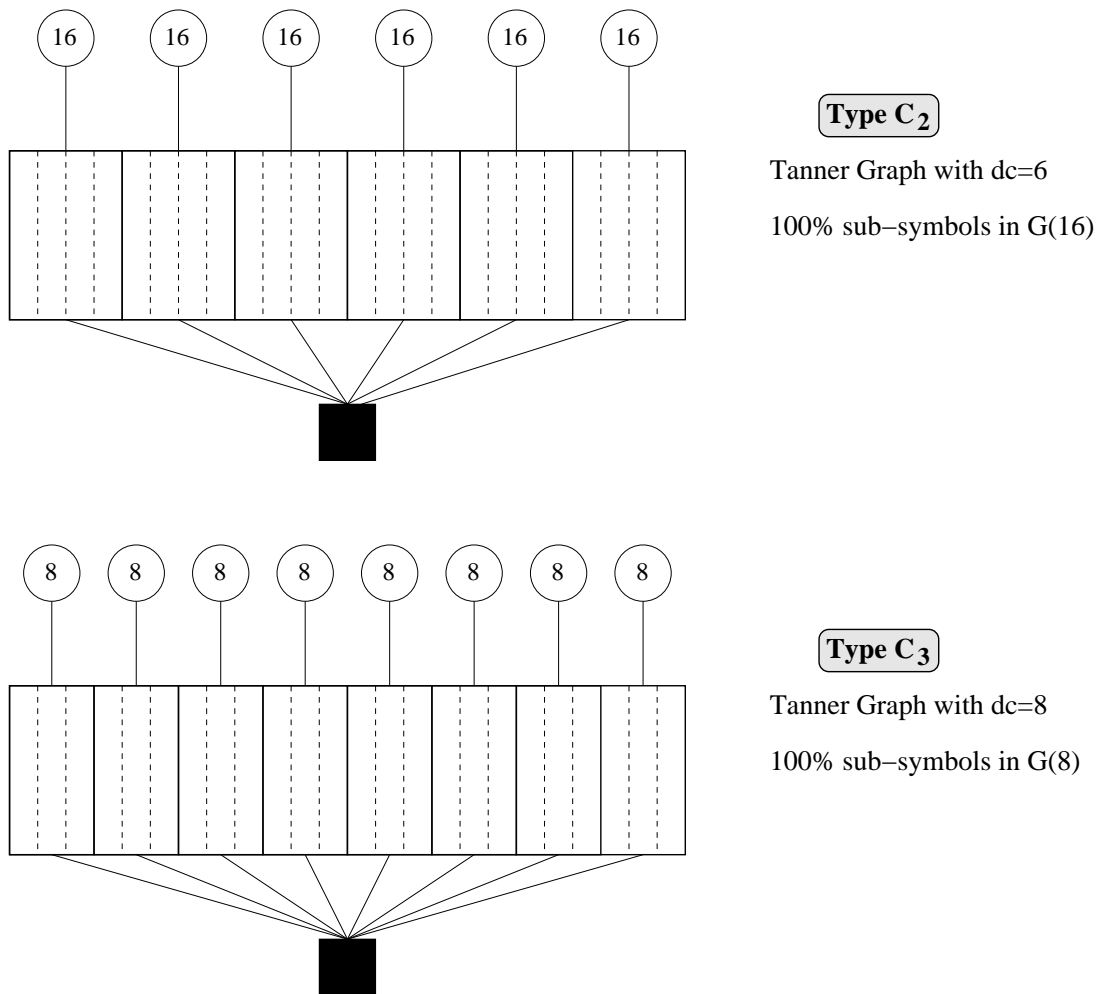


Figure 3. Split Structure of Generalized LDPC codes considered in this Deliverable. Representation of a Single Parity Check Equation.

II. ISSUES RELATED TO THE CLASS OF NB-SPLIT-LDPC CODES

In the first section of this deliverable, we have described the general structure of this new class of generalized non binary LDPC codes. Now we list what we think are the central research questions related to this new class, and we try to foresee the possible advantages in terms of practical implementation and expected performance compared to existing non binary LDPC codes. Some of these issues are treated in this deliverable, and we have made extensive performance simulations reported in section V to illustrate the first results obtained in the DAVINCI project.

a) Message Passing Decoder and Memory Requirements:

It is well known from the literature (see [7], [8], e.g.) that one of the major issue in hardware practical implementation of non-binary decoders is the memory requirements induced by the vector messages composed of q elements used in message passing non binary decoders. Storing all messages leads to a too high storage area on the chip.

The class of NB-Split-LDPC codes solves partially the problem of memory since in the message passing decoder on the NB-Tanner graph, the messages are associated to the variable nodes which represent *sub-symbols* instead of symbols. Therefore, a message in the Tanner graph of a NB-Split-LDPC code has only $2^{p_{1j}}$ entries instead of 2^{p_2} . If we assume that each message value is stored on n_q bits of quantization,

the total number of bits to represent the messages in the Tanner graph is indicated on the table I. We indicate only the memory for storing the messages in one direction of each edge, which is usually the case for efficient hardware implementations of LDPC decoders. As we can see, for a Rate $R = 1/2$ code in $GF(64)$ or $\mathbb{G}(64)$ with $N_b = 576$ bits for example, a field decoder would require $12288.n_q$ bits to represent the messages when only $4608.n_q$ are necessary for the NB-Split-LDPC of class \mathcal{C}_2 and $3072.n_q$ for the NB-Split-LDPC of class \mathcal{C}_3 .

Field	Split \mathcal{C}_2	Split \mathcal{C}_3
$2^p.d_c.M.n_q$	$\frac{3}{2}d_c.2^{\frac{2p-2}{3}}.M.n_q$	$2d_c.2^{\frac{p-2}{2}}.M.n_q$

Table I

MEMORY STORAGE REQUIREMENTS FOR MESSAGES IN A NB TANNER GRAPH. $2^p = 2^{p_2}$ IS THE ORDER OF THE FIELD OR THE GROUP IN WHICH THE PARITY CHECKS ARE DEFINED, M IS THE NUMBER OF PARITY CHECKS, AND d_c IS THE CONNEXION DEGREE OF THE CODE IN THE FIELD.

Note that the NB-Split-LDPC codes are compliant with the low complexity EMS decoder developed in the DAVINCI project. As long as the truncation parameter n_m is lower or equal than the sub-symbol order 2^{p_1j} , the adaptation of the decoder is possible. However, decoding a NB-Split-LDPC code is slightly more complex in terms of the number of elementary operations per decoding iteration. Indeed, since the degrees of the check nodes in a NB-Split Tanner graph is larger than the degree of field codes, we expect an according increase in the computational need. We do not predict the impact of this new class of LDPC codes in terms of hardware design, and this will be the scope of a future task in the DAVINCI project (common to WP4-WP6).

A thorough discussion on the efficient implementation of the *edge update* or *linear function nodes update* is conducted in paper [7]. Although [7] limits the discussion to full rank square clusters, we believe that implementing in the hardware model the generalized linear functions for Split NB-LDPC codes is not an issue.

b) Structured Code Construction for Practical Issues:

The NB-Split-LDPC code and the corresponding Tanner graph should have very “regular” structure in order to simplify the parallelisation of the processors in the hardware model. The following three issues must be addressed in the remaining of the project. None of these issues are solved for now.

- Practical codes should have a linear encoding structure. It has been shown in the literature that non binary cycles codes can be encoded in linear time with proper matrix representation, but this method cannot be applied to NB-Split-LDPC code, even if the NB-Split-LDPC code has a cycle Tanner graph. Other encoding procedure or constraints on the code design need to be discovered to solve this issue.
- The design of efficient quasi-cyclic graphs or codes, using the method of lifted protographs [] should be studied. Indeed, for binary codes, the protograph based LDPC constructions have multiple advantages with respect to the other existing families of LDPC codes. In particular, the designed codes are very robust to small to moderate codeword lengths, and it is easy to implement a parallel hardware model for their decoder. The same kind of advantages are foreseen for NB-LDPC codes and NB-Split-LDPC codes.
- We will study in the future the adaptation of the code design and optimization when only one component code H_{cc} is used for all parity checks of the type (3). Using the same component code, as it is advised in the case of binary GLD codes [16] has the advantage of simplifying the decoder architecture, without sacrificing much in performance.

c) Code Design related to the Non Binary Clusters H_{ij} :

One great advantage of NB-Split-LDPC codes is that the family is larger than for classical NB-LDPC codes. In particular, the fact that the binary parity check matrix \mathbf{H}_{bin} is composed of rectangular clusters instead of square clusters brings more degrees of freedom in the code optimization. The density of ones is less localized for NB-Split-LDPC codes, and as a consequence, we expect more powerful codes, with higher minimum distance than for the first DAVINCI codes. This will have a direct impact on the error floor performance, and also on the probability of undetected errors, *i.e.* instances when the decoder converges to a wrong codeword. Indeed, for short lengths NB-LDPC codes, the minimum distance is so low that a (too) large proportion of decoding errors are due to convergence to the low weight codewords. This behavior will be confirmed in this deliverable by FER simulations.

For the design of short to moderate code, we need to propose a proper finite length code optimization. In this deliverable, we will only consider the NB-Split-LDPC of class \mathcal{C}_2 and \mathcal{C}_3 , and the nonzero cluster optimization will be made from a generalization of approach developed in [5]. This generalized approach is limited since the number of cycles and Stopping Sets grows exponentially with d_c , and since d_c grows when we split with smaller and smaller “sub-symbols”, it becomes impossible to track the local minimum distance as an optimization measure for the overall code minimum distance. This optimization procedure gives however good first insight on the advantages of NB-Split-LDPC codes over classical NB-LDPC codes, as demonstrated in sections IV and V.

Nevertheless, other optimization techniques for this new class of LDPC codes need to be developed. The first step is to develop non binary multidimensionnal EXIT charts or non binary Density Evolution. This is not a trivial generalisation from the binary case, and the existing models proposed in the literature are not very accurate, and therefore not very efficient in terms of code design, especially when targeting moderate codeword lengths. A first version of the NB EXIT charts is presented in deliverable D4.2. Multidimensionnal EXIT charts for multi-edge type Tanner graphs would be also interesting to address optimal puncturing of the designed codes, and full-diversity property when block-fading channel is encountered (as is often the case in MIMO wireless transmission).

IV. MINIMUM DISTANCE PROPERTIES OF DESIGNED DAVINCI AND SPLIT-LDPC CODES

For codes defined over $\text{GF}(q)$, when addressing finite length design, it has been shown in [14] and [5] that selecting carefully the non binary entries of the parity-check matrix can improve the overall performance of the code when compared to randomly chosen coefficients. The selection of the non zero values can impact both on the waterfall and the on error floor. The observed performance gains are dependent of both the field order and the code rate.

In the waterfall region, selecting the edges label row-wise is critical. It is shown in [5] that “best” rows are selected according to their equivalent binary minimum distance and multiplicity of the minimum distance. In addition to that, it has been also shown in [5] that it is possible to lower the error floor by avoiding low weight codewords induced by some algebraic topological structures of the underlying Tanner graph, such as cycles or stopping sets. Choosing properly the edge labels of the stopping sets has a direct influence on the local minimum distance of the code, and therefore on the global minimum distance as well.

In this work, we aim at generalizing the method proposed in [5] to the case of NB-Split-LDPC codes. Before describing in details our design method, let us first introduced the main features. Basically, the proposed finite length design is based on two main steps: (i) building the graph, ie. optimizing the edge connections and (ii) selecting the non zeros entries of H , ie. choosing carefully the application $h_{ij}(\cdot)$ or equivalently the clusters H_{ij} . The first part can be efficiently addressed using some instances of the PEG algorithm [11] or its improved version [12], aiming at maximizing the local girth. Then, non zeros entries are selected carefully to ensure both good waterfall behavior and low error floors. The main difference between [5] and the work presented here is that we will maximize the local minimum distance by changing iteratively the content of binary rectangular clusters instead of binary square images of non-zeros field values.

A. Component Code Selection

In [5], the authors selected the best row entries according to the maximum of d_{\min} using the equivalent binary parity-check matrix of each row. This can be naturally extended to the case of NB-Split-LDPC by considering good codes for each component code H_{cc} , that is a binary code with p_2 rows and $\sum_j p_{1j}$ columns having a good minimum distance and minimal multiplicity low Hamming weights. Using this selection criterion, we can then consider better component codes in $\mathbb{G}(q)$ than in $\text{GF}(q)$. For example, the best d_c -tuples of coefficients for $\text{GF}(64)$ with $d_c = 4$ have minimum distance of 3 in $\text{GF}(q)$ while it is possible to consider component codes with $d_{\min} = 4$ in $\mathbb{G}(64)$. This will have a direct impact on the waterfall of the LDPC code.

This motivates the search for components codes achieving the best bound in terms of minimum distance and multiplicity [13]. For our example, this can be done for example by finding good codes using carefully chosen shortened versions of a (63, 57) Hamming code, or by shortening a (32, 26) extended-Hamming code. This is not the optimal choice, but we will see in the performance results section that this example is sufficient to ensure some gain compared to $\text{GF}(q)$ LDPC codes. Future work will aim at considering the optimum choice for the component codes.

Once a code component or a collection of code components has been selected, we can easily generate other good codes using bitwise permutations. The technique of bitwise permutation will allow to put some diversity in the choice of the nonzero clusters, without changing the component code.

B. Code Optimization with Random Permutations of One Component Code

In this work, we further propose rank-guaranteed random bitwise permutations to expand the possible entries which can construct a component code with good distance properties. With bitwise permutations,

more component codes with good minimum distance can be generated ensuring the diversity of the non zero entries to fully benefit from the optimization procedure. Let us denote $\mathbf{\Pi}$ a bitwise permutation of size $\sum_j p_{1j}$.

$$\begin{aligned} H'_{cc_i} &= [H_{ij_0} \dots H_{ij_k} \dots H_{ij_{d_c, i-1}}] \cdot \mathbf{\Pi} \\ &= [H'_{ij_0} \dots H'_{ij_k} \dots H'_{ij_{d_c, i-1}}] \end{aligned} \quad (4)$$

It is clear from these equations that from the same component code and the same clustering *profile*, one can get different clusters H_{ij_k} and H'_{ij_k} with different properties. The only constraint that we make is to ensure full column rank for all clusters H'_{ij_k} . Then, when considering these permutations, we apply a generalization of the optimization method proposed in [5]. The optimization consists in iteratively selecting the randomly permuted rows and optimize the local minimum distance on the small topologies like cycles or stopping sets, which are build from combination of cycles. For graphs with $d_v = 2$, the above mentioned topological structures are closed structures which are isolated. Consequently, the distance spectrum of those isolated structures contribute directly to the low weight spectrum of the overall code. On figure 5, we have sketched the cases of a length-6 cycle and a stopping set composed of 2 imbricated length-6 cycles.

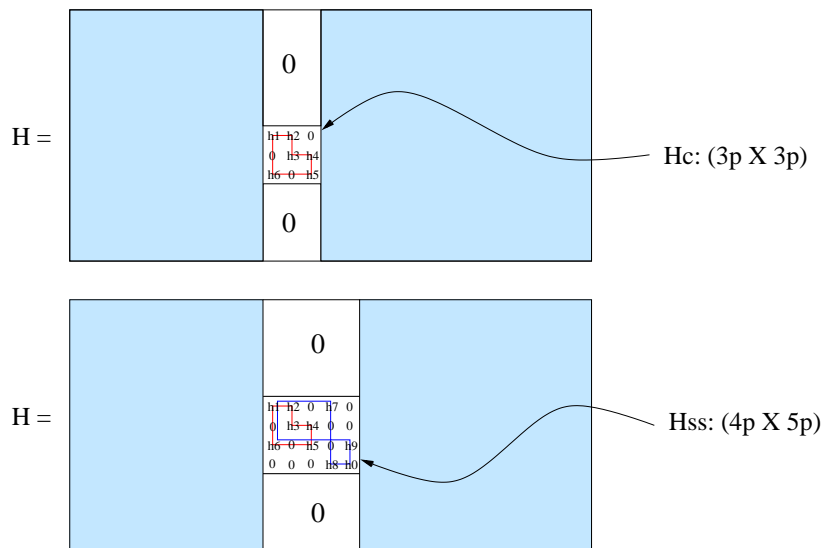


Figure 5. Binary image of a 6-cycle and a stopping set for a $d_v = 2$ cycle code. The stopping set structure is composed of two imbricated 6-cycles and a length-8 cycle. Both structures are isolated and define sub-codes, which distance spectra are closely related to the overall distance spectrum of the code. This figure considers the case of square $(p \times p)$ clusters.

The optimisation procedure follows:

- 1) Build a cycle Tanner graph with $d_v = 2$ with maximum girth g and minimum number of short cycles.
- 2) List all Cycles of length $\in \{g/2, g/2+1, \dots, g/2+\gamma_1\}$ and put them in the set \mathcal{S}_c . List all Stopping Sets of length $\in \{\lceil 3g/4 \rceil, \lceil 3g/4 \rceil + 1, \dots, \lceil 3g/4 \rceil + \gamma_2\}$ and put them in the set \mathcal{S}_{ss} .
- 3) Consider a single component code H_{cc} with good distance properties.
- 4) Random walk Optimization (one iteration):
for each and every row $i = 0 \dots M - 1$:
replace the clusters $[H_{ij_0} \dots H_{ij_k} \dots H_{ij_{d_c, i-1}}]$ by $[H'_{ij_0} \dots H'_{ij_k} \dots H'_{ij_{d_c, i-1}}]$ (cf. equation (4)) iff

$$Cost(H'_{cc_i}) < Cost(H_{cc_i})$$

where $Cost(\cdot)$ is a weighted objective function of the low-weight spectrum computed locally on $\mathcal{S}_c \cup \mathcal{S}_{ss}$. The objective function could vary from one code to another, but in principle $Cost(\cdot)$ decreases if the

low-weight spectrum improves, aiming at an iterative improvement of the global minimum distance.

The optimization of NB-Split-LDPC codes with the above described procedure is less efficient than of Field NB-LDPC codes. The main reason is that the number of considered topologies in the sets \mathcal{S}_c and \mathcal{S}_{ss} grows exponentially with the density of the Tanner graph. Since the density of the Tanner graph increases when the size of the sub-symbols diminishes, it is exponentially more and more time consuming to take into account topologies of the same size (same number of bits) in the computation of the local minimum distance (computation of the cost function $Cost(.)$). Therefore, the more we wish to improve the code's distance properties by splitting the parity checks in a large number of sub-symbols, the more difficult is the optimization of the code.

We have reported on tables II and III the low-weight distance spectra of DAVINCI benchmark codes, with NB-Split-LDPC codes of the same sizes and rates. The spectra have been estimated with the improved impulse method presented in [18]. Although it is clear that the low-weight spectra are improved when the order of the check splitting increases, we can see on these tables that the optimization of NB-Split-Codes is not very efficient at this stage. For example, the minimum distance of benchmark DAVINCI code for $R = 1/2$ and $N_b = 576$ is $D_{min} = 19$ and the NB-Split-LDPC of class \mathcal{C}_3 has the same minimum distance. However, we can see that the spectrum of the NB-Split-LDPC is a lot sparser than the field code. This lower multiplicity of low-weight codewords will have an impact on the error floor and the probability of undetected errors.

For larger codeword lengths, the spectrum improvement could be very important. This is especially the case for rate $R = 1/2$ and $N_b = 2304$. While the minimum distance of field code is strongly limited by the girth of the graph, with a minimum distance of $D_{min} = 24$, the type \mathcal{C}_3 has no codeword of weight lower than 34, and a very sparse spectrum until at least a weight-50. This statement should be taken carefully since the impulse method used to estimate the low weight spectra of sparse codes [18] becomes less and less efficient as D_{min} grows. Those results are therefore only an indication of the NB-Split-LDPC codes behaviors. Future research in the DAVINCI project will tackle the search for better optimization methods for NB-Split-LDPC codes.

	$N_b = 576$ bits	$N_b = 2304$ bits
Field GF(64)	weight 19 → #1 weight 20 → #54 weight 21 → #258 weight 22 → #668 weight 23 → #1474 weight 24 → #3167	weight 24 → #1 weight 25 → #15 weight 26 → #167 weight 27 → #489 weight 28 → #966 weight 29 → #1738
Split \mathcal{C}_2 GF(64)	weight 20 → #2 weight 21 → #5 weight 22 → #13 weight 23 → #25 weight 24 → #59	weight 26 → #5 weight 28 → #10
Split \mathcal{C}_3 GF(64)	weight 19 → #2 weight 22 → #1 weight 23 → #2 weight 24 → #3	weight 34 → #1 weight 55 → #1 weight 56 → #1 weight 59 → #1

Table II
MINIMUM DISTANCE SPECTRA OF GENERALIZED LDPC CODES WITH PARITY CHECKS IN GF(64) - RATE $R = \frac{1}{2}$

V. FRAME ERROR RATE RESULTS WITH FFT-BELIEF PROPAGATION

In this report, we made extensive Monte Carlo simulations in order to show the performance of NB-Split-LDPC codes compared to the benchmark DAVINCI codes. Since there is no EMS-like, low complexity decoder for NB-Split-LDPC codes, we have compared the performance of both families of codes with the FFT-Belief propagation decoder [17]. Results are plotted on figure 6 to 9 for 2 rates $R = 1/2$ and $R = 3/4$, and two codeword lengths $N_b = 576$ and $N_b = 2304$. All simulation points have been plotted with a maximum of $N_{it} = 100$ iterations, and a minimum of 100 recorded frame errors.

The expected behavior of NB-Split-LDPC codes compared to field codes is confirmed. We have indicated in the legends the probability of undetected errors, i.e. the percentage of time the decoder converges to a wrong codeword, meaning that the decoder cannot detect the error. As we can see, for all considered cases, the error floor for field codes is largely due to undetected errors, meaning that the low-weight codewords dominate the error floor behavior, as in the Turbo-codes case. However, for NB-Split-LDPC codes, the number of undetected errors is considerably lowered, especially when the check nodes are split in a large number of sub-symbols (type \mathcal{C}_3 codes). The price to pay to an improvement in the error floor and lower probability of undetected errors is a loss in waterfall. This loss can be relatively large, like in the case of $R = 1/2$ and $N_b = 2304$, but can be small ($R = 1/2$ and $N_b = 576$) or negligible ($R = 3/4$ and $N_b = 576, 2304$).

	$N_b = 576$ bits	$N_b = 2304$ bits
Field GF(64)	weight 9 → #4 weight 10 → #137 weight 11 → #1071 weight 12 → #4851	weight 11 → #1 weight 12 → #20 weight 13 → #423 weight 14 → #1903 weight 15 → #6493
Split C_2 GF(64)	weight 10 → #13 weight 11 → #106 weight 12 → #503	weight 11 → #1 weight 12 → #1 weight 13 → #11 weight 14 → #108 weight 15 → #335
Split C_3 GF(64)	weight 9 → #10 weight 10 → #29 weight 11 → #122 weight 12 → #521	weight 12 → #4 weight 13 → #11 weight 14 → #47 weight 15 → #98

Table III
MINIMUM DISTANCE OF GENERALIZED LDPC CODES WITH PARITY CHECKS IN GF(64) - RATE $R = \frac{3}{4}$

The work on the design and analysis of NB-Split-LDPC codes is still on-going. Especially, the following issues have to be solved:

- Adaptation of the low complexity EMS decoder to this family,
- efficient optimization of finite length NB-Split-LDPC,
- structured representations of the associated Tanner graphs,
- robustness to rate and constellation changes for adaptive coding schemes.

All these issues are under consideration in the DAVINCI project and progress will be reported in future deliverables.

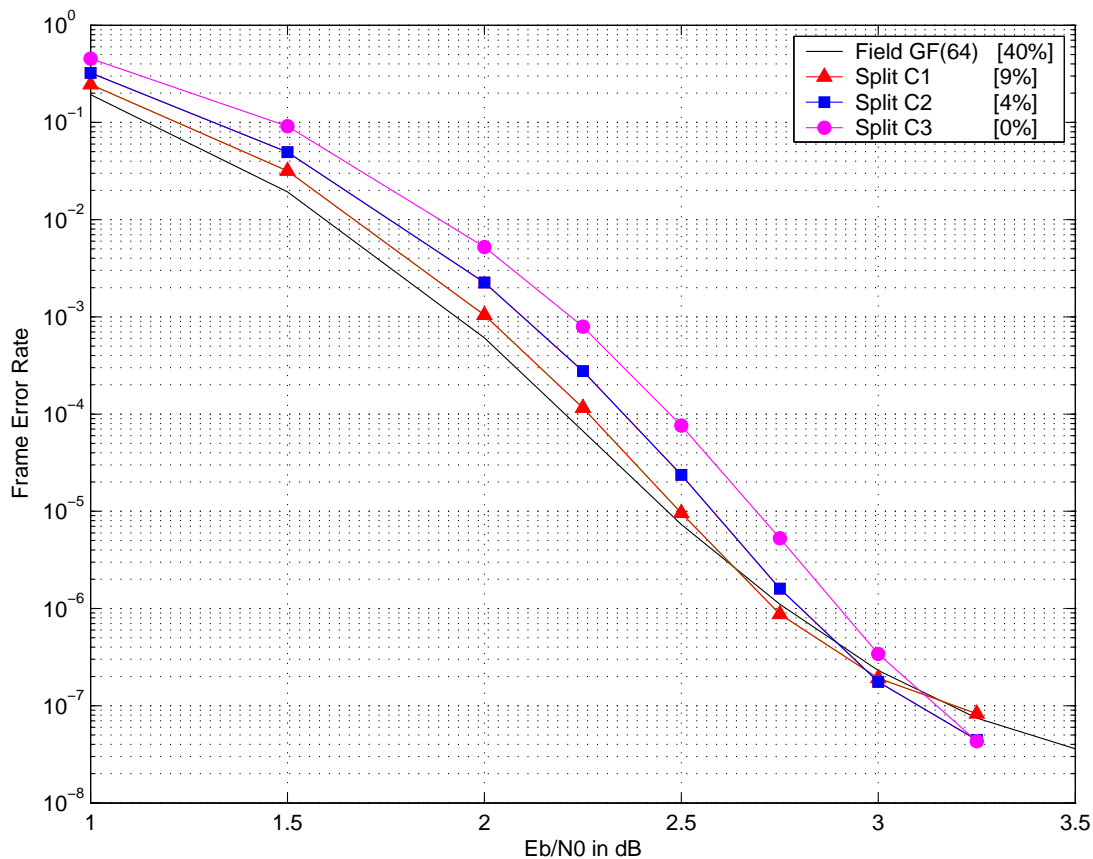


Figure 6. Performance Results of GF(64)-DaVinci codes and

REFERENCES

- [1] V. Rathi and R. Urbanke, "Density evolution, thresholds and the stability condition for non-binary LDPC codes," *IEE Proceedings-Communications*, Vol. 152, No. 6, pp. 1069–1074, Dec. 2005.
- [2] A. Goupil, M. Colas, G. Gelle and D. Declercq, "FFT-based BP Decoding of General LDPC Codes over Abelian Groups," *IEEE Trans. on Commun.*, vol. 55, no. 4, pp. 644–649, April 2007.
- [3] L. Sassatelli and D. Declercq, "Non-binary Hybrid LDPC Codes: structure, decoding and optimization", in the proc. of ITW'06, Chengdu, China, October 2006.
- [4] L. Sassatelli and D. Declercq, "Analysis of Non-binary Hybrid LDPC Codes", in the proc. of ISIT'07, Nice, France, June 2007.
- [5] C. Poulliat, M. Fossorier and D. Declercq, "Design of regular $(2, d_c)$ -LDPC codes over $GF(q)$ using their binary images", *IEEE Trans. on Commun.*, vol. 56, no. 10, pp. 1626–1635, October 2008.
- [6] A. Voicila, D. Declercq, F. Verdier M. Fossorier and P. Urard, "Split Non-binary LDPC Codes", in the proc. of ISIT'08, Toronto, Canada, July 2008.
- [7] W. Chen, C. Poulliat, D. Declercq, L. Conde-Canencia, A. Al-Ghouwayel and E. Boutillon, "Non-Binary LDPC Codes defined over the General Linear Group: Finite Length Design and Practical Implementation Issues", in the proc. of VTC'09 (Special Session FP7-ICT-RAS-Cluster), Barcelona, Spain, April 2009.
- [8] A. Voicila, D. Declercq, F. Verdier, M. Fossorier and P. Urard, "Low-Complexity Decoding for non-binary LDPC Codes in High Order Fields", accepted in *IEEE Trans. Commun.*, 2009
- [9] L. Barnault and D. Declercq, "Fast Decoding Algorithm for LDPC over $GF(2^q)$ ", in the proc. of ITW'03, Paris, France, 2003.
- [10] X.Y. Hu and E. Eleftheriou, "Binary Representation of Cycle Tanner-Graph $GF(2^q)$ codes", *EEE Int. Conf. on Commun.*, Paris, France, June 2004.
- [11] X.-Y. Hu, E. Eleftheriou and D.M. Arnold, "Regular and Irregular Progressive Edge-Growth Tanner Graphs", *IEEE Trans. on Inf. Theory*, Vol. 51, No. 1, pp. 386–398, January 2005.
- [12] A. Venkiah, D. Declercq and C. Poulliat, "Design of Cages with a Randomized Progressive Edge Growth Algorithm", *IEEE Commun. Letters*, vol. 12(4), pp. 301–303, April 2008.
- [13] T. Verhoeff, "An updated table of minimum-distance bounds for binary linear codes," *IEEE Trans. Inform. Theory*, vol. IT-33, no. 5, pp. 665–680, Sept. 1987.
- [14] M.C. Davey and D. MacKay, "Low-Density Parity-Check Codes over $GF(q)$ ", *IEEE Commun. letters*, vol. 2, pp. 165–167, June 1998.
- [15] S. Pfletschinger, A. Mourad, E. Lopez, D. Declercq and G. Bacci, "Performance Evaluation of Non-Binary LDPC Codes on Wireless Channels", in the proc. of ICT Mobile Summit, Santander, Spain, June 2009.

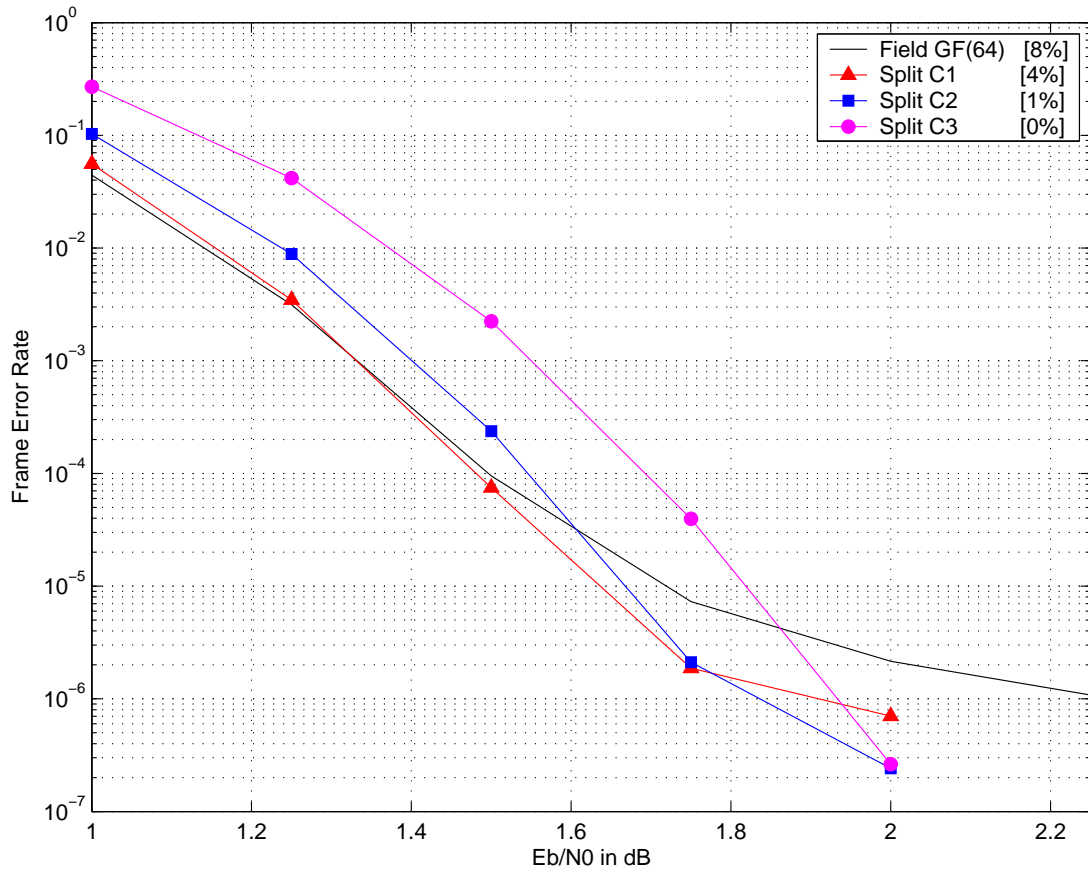


Figure 7. Performance Results of GF(64)-DaVinci codes and

- [16] J.J. Boutros, O. Pothier and G. Zémor, "Generalized low density (Tanner) codes", IEEE International Conference on Communications, vol. 1, pp. 441-445, Vancouver, Canada, June 1999.
- [17] D. Declercq and M. Fossorier, "Decoding Algorithms for Nonbinary LDPC Codes over GF(q)", IEEE Trans. on Commun., vol. 55(4), pp. 633-643, April 2007.
- [18] D. Declercq and M. Fossorier, "Improved Impulse Method to Evaluate the Low Weight Profile of Sparse Binary Linear Codes", in the proc. of ISIT'08, Toronto, Canada, July 2008.

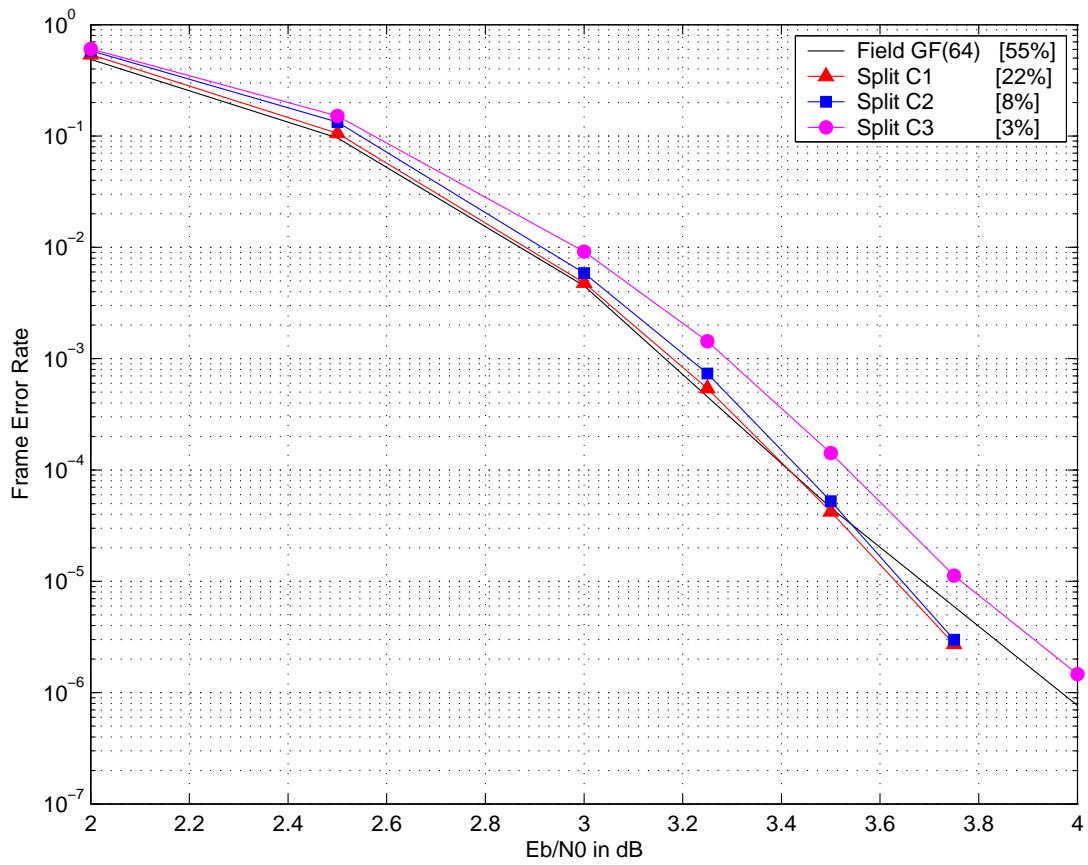


Figure 8. Performance Results of GF(64)-DaVinci codes and

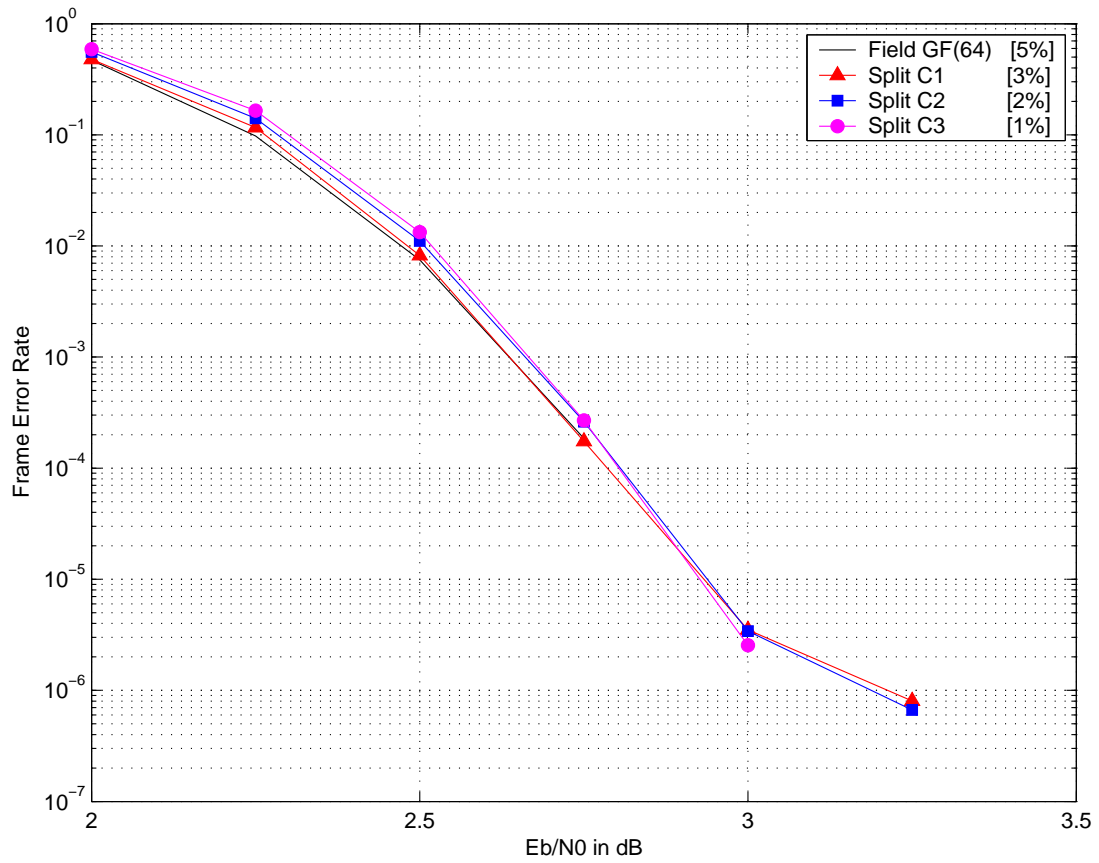


Figure 9. Performance Results of GF(64)-DaVinci codes and