A Low Complexity Sum-Subtract Decoding Algorithm for Non-Binary LDPC Codes over GF(q)

Leonardo Arnone[1], Jorge Castiñeira Moreira[2], Carlos Gayoso[1],
Claudio González[1], and Miguel Rabini[1]

[1]Laboratorio de Componentes, Facultad de Ingeniería,
Universidad Nacional de Mar del Plata, Argentina.
[2]Laboratorio de Comunicaciones, Facultad de Ingeniería,
Universidad Nacional de Mar del Plata, Argentina.
{leoarn,casti,cgayoso,cmgonzal,mrabini,
cmgonzal,mrabini}@fi.mdp.edu.ar
http://www.fi.mdp.edu.ar

Abstract. In this paper, we present a low complexity Sum-Subtract decoder for non-binary LDPC codes defined over $GF(q)$. The performance of this decoding algorithm is similar to that of the Fast Fourier Transform Sum-Product algorithm usually utilized for decoding non-binary LDPC codes. It is a simplified algorithm that can be easily implemented on programmable logic technology such as FPGA devices because of its use of only additions and subtractions, avoiding the use of quotients and products, and of float point arithmetic. The algorithm yields a very low complexity programmable logic implementation of an NB-LDPC decoder with an excellent BER performance.

Key words: Galois fields, non-binary LDPC codes, programmable logic technology

1 Introduction

It is known that binary low density parity check (LDPC) codes achieve rates close to the channel capacity for very long codeword lengths, and they are nowadays being implemented in many modern communication standards.

Bit Error Rate (BER) performance of binary LDPC codes are however not that impressive when the code size is low or moderate, and high order modulation is used for transmission. For these cases, non binary LDPC (NB-LDPC) codes designed in high order Galois fields $GF(q)$ have shown great interest [1], [2]. They can be shown to outer perform equivalent LDPC codes defined over the binary field.

A non-binary LDPC code is simply an LDPC code with a sparse parity check matrix containing elements that could be defined over a finite field. Of particular interest are LDPC codes defined over finite fields of the form $GF(2^m)$, where m is a positive integer greater than 1. In 1998, Mackay presented the idea of LDPC codes over finite fields [3]. They have shown that non-binary LDPC codes can achieve increases in performance over their binary counterparts when the size of the finite field is increased. Mackay also showed how the Sum-Product (SP) algorithm could be extended to decode non-binary LDPC codes, but the resulting complexity made this decoding algorithm be unfeasible. Only non-binary LDPC codes over small finite fields were implemented.

For an NB-LDPC code defined over $GF(q)$, each received symbol can be one of q different elements in $GF(q)$. Therefore the horizontal step becomes more complicated as there are now more possible non-binary sequences to satisfy the parity check equations. Each nonzero element defined over $GF(q)$ in the parity check matrix $\mathbf{H}$ has q probabilities associated with it, instead of two probabilities as in the binary case. These are some reasons for making the extended SP algorithm be of a very high complexity.

As said above, NB-LDPC codes provide a gain over their binary counterpart, but this performance gain comes together with a significant increase of the decoding complexity. NB-LDPC codes are decoded with message passing algorithms as the Sum-Product algorithm, but the size of the messages varies in the order q of the field. Therefore, an implementation of the SP decoder has complexity in $O(q^2)$. A Fourier domain implementation of the SP is possible like in the binary case, reducing the complexity to $O(q\log(q))$, but this implementation is only convenient for messages expressed in the probability domain. Details of this Fast Fourier Transform Sum-Product algorithm, which will be referred to as the FFT-SP algorithm, can be found in [4].

In the horizontal step of the classic SP algorithm it is necessary to find all possible binary sequences that satisfy a parity check equation, determining the probability of each sequence and adding them all together [5]. In general, we can write this as a convolution operation. This implies that the same result can be achieved by replacing the convolution operation with Fourier transform. This is the essential idea behind the FFT-SP algorithm. However, the number of additions and products involved in the FFT-SP algorithm make difficult the implementation of optimal NB-LDPC decoders on low complexity programmable logic.

An NB-LDPC code is defined by a very sparse random parity check matrix $\mathbf{H}$ whose components belong to a finite field $GF(q)$. Decoding algorithms of LDPC codes are iterative message passing decoders based on a factor (or Tanner) graph representation of the matrix $\mathbf{H}$.

As in the case of binary decoders, there are two possible representations for the passing messages: probability weights vectors or log-density-ratio (LDR) vectors. The use of the LDR form for messages has been advised by many authors who proposed practical LDPC decoders. Indeed, LDR values which represent real reliability measures on the bits or the symbols are less sensitive to quantization errors due to finite precision coding of the messages [6].

In this paper we propose a very low complexity Sum-Subtract decoding algorithm for NB-LDPC codes. This algorithm is based on a transformation of the representation of variables of the FFT-SP decoding algorithm, and involves the use of additions and subtracts, and also uses two look-up tables.

A comparison is done between the BER performance of the proposed decoding algorithm, and the BER performance of the FFT-SP decoding algorithm, for a given NB-LDPC code.

Results show that there is no significant difference in BER performance between the optimal and the proposed algorithm. The proposed algorithm is characterized by a very low complexity implementation, thus becoming a better alternative for its programmable logic implementation than the traditional FFT-SP algorithm.

2 Decoding Non-Binary LDPC Codes over $GF(q)$ with Fast Fourier Transform Sum-Product algorithm (FFT-SP)

LDPC codes are a powerful class of linear block codes characterized by a parity check matrix $\mathbf{H}$, which fits the condition $\mathbf{H}\circ\mathbf{c} = \mathbf{0}$ for any codeword $\mathbf{c}$. An LDPC decoder is essentially a decoding algorithm designed for finding a codeword $\hat{\mathbf{c}}$ (an estimate of the codeword $\mathbf{c}$), able to fit the condition:

$$\mathbf{H} \circ \hat{\mathbf{c}} = \mathbf{0}. \tag{1}$$

The LDPC decoding algorithm is described over a bipartite graph depicted considering the relationship between the symbol nodes j, which represent the bits or symbols of the code vector $\mathbf{c}$, and the parity check nodes i, which represent the parity equations described in matrix $\mathbf{H}$. In this iterative process, each symbol node j sends to a parity check node i the estimation $Q_{i,j}(x)$ that this node generates with the information provided by all others parity check nodes connected to it, based on the fact that the parity check node j is in state x, with $x = \left\{ 0,\, 1,\, \alpha \cdots \alpha^{q-2} \right\}$ for NB-LDPC codes.

Then, each parity check node i sends the estimation $R_{i,j}(x)$ to each symbol node j generated with the information provided by the other symbol nodes connected to it, based on the fact that the parity check node i condition is satisfied, if the symbol node j is in state x. This is an iterative process in which information is interchanged between these two types of nodes. This iterative process is stopped when the condition described by Eqn (1) is satisfied. In this case the corresponding decoded codeword is considered a valid codeword. Otherwise, the decoding algorithm stops after a given number of iterations are performed. In this case the decoded word may or may not be a codeword.

Another difference in the Tanner graph of an NB-LDPC code is that non-zero elements of the parity check matrix $\mathbf{H}$ are in the set $\left\{ 0,\, 1,\, \alpha \cdots \alpha^{q-2} \right\}$ and this leads to a modified graph that takes into account this representation. Circulant shift operations over the tensors of probabilities or estimates in the decoding algorithm take into account this structural difference [4], [5].

The FFT-SP decoding algorithm for NB-LDPC codes involves the evaluation of the following steps:

2.1 Initialization

The initialization process is done by setting the values of estimations $Q_{i,j}(x)$ to the a priori probabilities of the symbols $P_j(x)$. The a priori probability $P_j(x)$ is the probability of the j_{th}-symbol node adopting the value of x, with $x = \left\{0,\, 1,\, \alpha \cdots \alpha^{q-2}\right\}$. These values depend on the samples of the received signal from the channel [5].

2.2 Horizontal Step

The horizontal step determines the probability $R_{i,j}(x)$, defined as [4], [5]:

$$R_{i,j}(x) = F^{-1}\left(\prod_{k \in N(i)\backslash j} F\left(Q_{i,k}(x)\right)\right) \tag{2}$$

The set of indexes of all the symbol nodes j related to the parity check node i will be denoted as $N(i)$, and $N(i)\backslash j$ will be the same set but excluding the index j. F is the Fourier transform and F^{-1} is the inverse Fourier transform. For NB-LDPC codes defined over $GF(q)$, F and F^{-1} are obtained by multiplying the tensors products by Hadamard matrices [4], [5].

2.3 Vertical Step

For each i,j the quantities $Q_{i,j}(x)$ are evaluated. Then the pseudo posterior probabilities $Q_j(x)$ are updated.

$$Q_{i,j}(x) = \beta_{i,j}\, P_j(x) \prod_{k \in M(j)\backslash i} R_{k,j}(x) \tag{3}$$

$M(j)$ is the set of sub indexes of all parity check nodes i related to the symbol node j, and $M(j)\backslash i$ will be the same set but excluding the index i. $\beta_{i,j}$ is a normalizing constant such that $\sum Q_{i,j}(x) = 1$. The pseudo posterior probabilities $Q_j(x)$are determined by:

$$Q_j(x) = \beta_j\, P_j(x) \prod_{M(j)} R_{i,j}(x) \tag{4}$$

where again β_j is a normalizing constant such that $\sum Q_j(x) = 1$. Finally, from these pseudo posterior probabilities estimates of the transmitted code word can by found by:

$$\hat{c}_j = \arg\max_x \left(\beta_j\, P_j(x) \prod_{M(j)} R_{i,j}(x)\right) \tag{5}$$

3 Proposed Sum-Subtract Decoding Algorithm (SSD)

The proposed simplification makes this algorithm operate using only additions and subtractions. This simplification makes use of a logarithmic version of the calculations involved in the FFT-SP algorithm described in the previous Section.

We define $P_j(x) = e^{p_j(x)}$, $Q_{i,j}(x) = e^{q_{i,j}(x)}$, $R_{i,j}(x) = e^{r_{i,j}(x)}$ and $Q_j(x) = e^{q_j(x)}$. The NB-LDPC Sum-Subtract decoding algorithm involves the evaluation of the following steps:

3.1 Initialization

It is only necessary to initialize variables $q_{i,j}(x)$ with the values $p_j(x)$.

3.2 Horizontal Step

The Hadamard transform is used to implement the Fourier transform of the product of estimates, which are then multiplied as tensors to obtain a given final tensor, which is then operated by using the inverse Fourier transform, also implemented by performing a Hadamard transform. By applying the proposed exponential operation, the Hadamard Transform operation $F\left(Q_{i,j}\left(x\right)\right) = F\left(e^{q_{i,j}(x)}\right)$ of the horizontal step involves the use of additions and subtractions of terms of the form:

$$e^{q(0)} \pm e^{q(1)} \pm e^{q(\alpha)} \pm \cdots \pm e^{q(\alpha^{q-2})} \tag{6}$$

Additions and subtractions are determined by the operations defined in each row of the corresponding Hadamard matrix, which is used to perform the Fast Fourier transform. Form and size of the corresponding Hadamard matrix depends on the size of the finite field q that is used, but operations in all the cases adopt the form of Eqn. (6).

In order to solve calculations of the form of Eqn. (6), we resort to a recursive application of a basic two-term operation $e^c = e^a \pm e^b$, where c can be calculated by using the expression $c = f(a,b) = \max(a,b) + \ln\left(1 + (-1)^s e^{-|a-b|}\right)$. Here, $s = 0$ for an addition and $s = 1$ for a subtraction. Function f is iteratively applied to give solution to the Hadamard transform operations in the exponential form, usually present in the horizontal step of the algorithm:

$$f_{H(i,j)} = f\left(q_{i,j}(0), f\left(q_{i,j}(1), f\left(q_{i,j}(\alpha), f(\cdots, q_{i,j}(\alpha^{q-2}))\right)\right)\right) \tag{7}$$

Thus,

$$r_{i,j}(x) = f_{H(i,j)}\left(\sum_{k \in N(i)\backslash j} f_{H(i,k)}\right) \tag{8}$$

Function f can be implemented in practice by means of two look-up tables of the form:

$$\begin{aligned} f_+(a,b) &= \max(a,b) + \ln\left(1 + e^{-|a-b|}\right) \quad (s=0) \\ f_-(a,b) &= \max(a,b) + \ln\left(1 - e^{-|a-b|}\right) \quad (s=1) \end{aligned} \tag{9}$$

3.3 Vertical Step

Eqn (3) used in the FFT-SP algorithm can be performed in the exponential mode as:

$$e^{q_{i,j}(x)} = \beta_{i,j}\, e^{p_j(x)} \prod_{k\in M(j)\backslash i} e^{r_{k,j}(x)} = \beta_{i,j} e^{p_j(x)+\sum_{k\in M(j)\backslash i} r_{k,j}(x)} \tag{10}$$

We define an auxiliary variable:

$$c_{i,j}(x) = p_j(x) + \sum_{k\in M(j)\,\backslash i} r_{k,j}(x) \tag{11}$$

so that,

$$e^{q_{i,j}(x)} = \beta_{i,j} e^{c_{i,j}(x)} \tag{12}$$

In order to fit the normalization condition $\sum Q_{i,j}(x) = \sum e^{q_{i,j}(x)} = 1$, $\beta_{i,j}$ should be equal to:

$$\beta_{i,j} = \left(e^{c_{i,j}(0)} + e^{c_{i,j}(1)} + e^{c_{i,j}(\alpha)} + \cdots + e^{c_{i,j}(\alpha^{q-2})}\right)^{-1} \tag{13}$$

then:

$$e^{q_{i,j}(x)} = \frac{e^{c_{i,j}(x)}}{e^{c_{i,j}(0)} + e^{c_{i,j}(1)} + e^{c_{i,j}(\alpha)} + \cdots + e^{c_{i,j}(\alpha^{q-2})}} \tag{14}$$

and thus:

$$q_{i,j}(x) = c_{i,j}(x) - f\left(c_{i,j}(0), f\left(c_{i,j}(1), f\left(c_{i,j}(\alpha), f(\cdots, c_{i,j}(\alpha^{q-2}))\right)\right)\right) \tag{15}$$

where $f\left(c_{i,j}(0), f\left(c_{i,j}(1), f\left(c_{i,j}(\alpha), f(\cdots, c_{i,j}(\alpha^{q-2}))\right)\right)\right)$ allows the normalization condition $\sum Q_{i,j}(x) = \sum e^{q_{i,j}(x)} = 1$ to be fulfilled.

The pseudo posterior probability $q_j(x)$ is similarly evaluated as in the previous step, by defining an auxiliary variable:

$$c_j(x) = p_j(x) + \sum_{i\in M(j)} r_{i,j}(x) \tag{16}$$

such that $q_j(x)$ is equal to:

$$q_j(x) = c_j(x) - f\left(c_j(0), f\left(c_j(1), f\left(c_j(\alpha), f(\cdots, c_j(\alpha^{q-2}))\right)\right)\right) \tag{17}$$

As in the previous case, $f\left(c_j(0), f\left(c_j(1), f\left(c_j(\alpha), f(\cdots, c_j(\alpha^{q-2}))\right)\right)\right)$ allows the normalization condition $\sum Q_j(x) = \sum e^{q_j(x)} = 1$ to be fulfilled.

Finally estimate of the decoded output $\hat{c}_j(x)$ is calculated as:

$$\hat{c}_j(x) = \max_x \left(q_j(x)\right) \tag{18}$$

4 Look-Up Tables Implementation

The performance of the proposed decoding algorithm is set by the characteristics of the look-up tables f_+ and f_- Eqn(9). Assuming that the maximum number of bits used to construct these tables is n, the maximum number of entries of these tables is of size 2^n.

5 Complexity Analysis

In order to do an analysis of the complexity of the proposed algorithm, we define $t = M(j)_{av}$ as the average number of non-zero elements per column, and $u = N(i)_{av}$ as the average number of non-zero elements per row, in the parity-check matrix of the code. Typically we have $u = Nt/M$, such that for an 1/2-rate NB-LDPC code for instance, $M = N/2$, and $u = 2t$.

Using $t = 3$, $u = 6$ and $GF(4)$, the FFT-SP decoding algorithm involves the calculation of $108N$ products, $54N$ sums-subtracts and $24N$ quotients (average).

The proposed SSD algorithm requires $132N$ sums-subtracts, $144N$ accesses to the look-up tables and $144N$ comparisons. In spite of requiring more sums-subtracts than the traditional decoding algorithm, the complexity of the proposed algorithm implementation is highly reduced due to the fact of operating with neither quotients nor products.

6 Simulations Results

Figure 1 shows results obtained by doing a simulation with the NB-LDPC code $C_{LDPC}(12, 8)$ defined over $GF(4)$ in a transmission of 1000 blocks of 12 message non-binary each, decoded using either the proposed Sum-Subtract decoding algorithm (SSD) or the FFT-Sum-Product decoding algorithm (FFT-SP).

Decoding in all the cases uses 5 and 20 iterations. The BER performance of the proposed SSD is also the same as the FFT-SP algorithm. Figure 1 also shows results for the NB-LDPC code $C_{LDPC}(12, 8)$ defined over $GF(256)$.

The BER performance of the schemes that use tables of size 256 or 512 for implementing function f (briefly described in Fig. 1 as SSD $GF(q)$ Table 256 or 512) do not show differences with respect to those which use the ideal function f (briefly described in Fig. 1 as SSD $GF(q)$ Ideal). Therefore, it is possible to implement a low complexity decoding algorithm without significant BER performance loss, by using the proposed logarithmic decoder with two look-up tables of reasonable size.

7 Conclusions

In this paper a low complexity decoder for NB-LDPC codes is proposed. The algorithm has the advantage of significantly reducing the complexity of the original FFT-SP decoding algorithm [4], [5]. The reduction in complexity is because

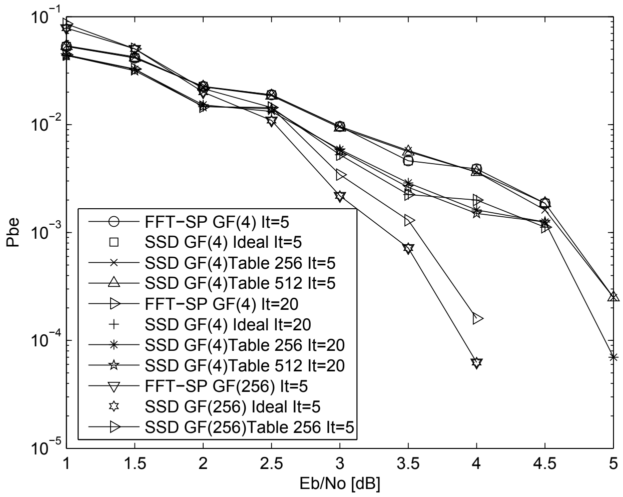

Fig. 1. Simulation of the BER performance of NB-LDPC codes, using the SSD or FFT-SP algorithms, for different look-up tables and numbers of iterations.

the proposed algorithm uses only sums and subtracts, and two look-up tables, avoiding the use of quotients and products, operations that are of high complexity in practical implementations, especially using Field Programmable Logic Arrays (FPGA) technology [7].

References

1. Hu X.Y., Eleftheriou E.: Binary Representation of Cycle Tanner-Graph GF(2q) Codes. In: Proc. IEEE Intern. Conf. on Commun., Paris, France, pp. 528–532, (2004)
2. Poulliat C., Fossorier M., Declercq D.:Design of non binary LDPC codes using their binary image: algebraic properties. In: ISIT'06, Seattle, USA, (2006).
3. Davey M.C, MacKay D.J.C.:Low-Density Parity-Check Codes over GF(q). In: IEEE Communications Letters, vol. 2, No. 6, pp. 165–167, (1998).
4. Declerq D., Fassorier M.:Decoding Algorithms for Non-Binary LDPC Codes over GF(q). In: IEEE Transactions on Communications, vol. 55, No. 4, pp.633–643, (2007)
5. Carrasco R., Johnston M.:Non-Binary Error Control Coding for Wireless Communication and Data Storage. In: Ed John Wiley and Sons Ltd, U.K, (2008)
6. Ping L., Leung W.K.:Decoding low density parity check codes with finite quantization bits. In: IEEE Commun. Lett., pp.62–64, (2000)
7. Altera Corporation, `http://www.altera.com`