Design and implementation of a near maximum likelihood decoder for Cortex codes

Cédric Marchand, Mohamed Ben Hammouda, Yvan Eustache, Laura Conde-Canencia, Emmanuel Boutillon
Université Européenne de Bretagne, Lab-STICC, CNRS, UBS, BP 92116 56321 Lorient, France.
Email: name.surname@univ-ubs.fr

***Abstract*—Cortex codes are an emerging family among the rate-1/2 self-dual systematic linear block codes with good distance properties. This paper investigates the challenging issue of designing an efficient Maximum Likelihood (ML) decoder for Cortex codes. It first reviews a dedicated architecture that takes advantage of the particular structure of this code to simplify the decoding. Then, we propose a technique to improve the architecture by the generation of an optimal list of binary vectors. An optimal stopping criterion is also proposed. Simulation results show that the proposed architecture achieves an excellent performance/complexity trade-off for short Cortex codes. The proposed decoder architecture has been implemented on an FPGA device for the (24,12,8) Cortex code. This implementation supports an information throughput of 300 Mb/s. At a signal-to-noise ratio Eb/No=8 dB, the Bit Error Rate equals 2×10^{-10}, which is close to the performance of the Maximum Likelihood decoder.**

***Index Terms*—Cortex codes, auto-dual codes, VLSI, ML decoding.**

I. Introduction

Nowadays, modern Forward Error Correction (FEC) techniques, such as Low-Density Parity-Check (LDPC) codes [1] approach the limit of the channel capacity, for long code lengths (thousands of bits). Nevertheless, a long FEC code may be not relevant for particular applications, such as mobile phone communications or internet protocols due to latency constraints. For short block length (hundreds of bits or less), LDPC codes showed a low performance due to the increasing density of '1's in the Parity-Check matrices. Turbo-codes [2] achieve near optimal decoding performance for codes longer than a few hundreds bits, but become less appropriate for shorter codes.

The emerging Cortex codes [3], [4] may offer a practical and efficient alternative to the best known iterative decoders, i.e. binary (or non-binary) LDPC and Turbo-Codes for very short frames. Cortex codes were initially proposed by Carlach in [3]. They are systematic rate-1/2 self-dual block codes with large minimum distance. A Cortex encoder combines a very short mother code with a sequence of permutations to produce the parity bits. If the mother code is self-dual, the resulting Cortex code inherits from this self-dual property [4]. Therefore, the $(N = 2K, K)$ parity check matrix of a Cortex code can be written as $H = [P, I]$, where I is the $K \times K$ identity matrix and P a dense $K \times K$ sub-matrix satisfying $P \times P' = I$ (P' denotes the transpose matrix of P). In particular, if $X = (x_1, x_2, \cdots, x_K)'$ is the information vector and $R = (r_1, r_2, \cdots, r_K)'$ the redundancy vector, then $R = P.X$ and $X = P'.R$.

The authors thank Orange Labs for the funding of the study, the "Région Bretagne" and the "European Funds for Regional Development" (FEDER) for funding materials used in the study.

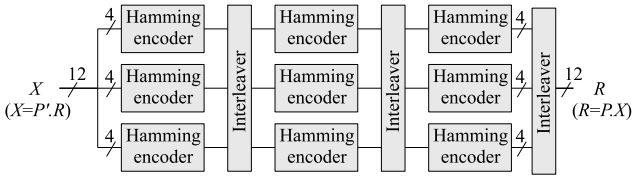

Fig. 1. Architecture of a Cortex encoder with N=24 built from (8,4,4) Hamming codes

Fig. 1 shows an example of a three stage Cortex encoder (24,12,8) also known as the Golay code. The code is based on extended (8,4,4) Hamming codes and interleavers as components. Note that, thanks to the simple network structure, the calculation of R from X (or X from R) requires only $7 \times 9 = 63$ 2-input XOR operations (7 XOR for each extended Hamming code).

Efficient decoding of Cortex codes is a new challenge recently taken up by [5], [6], [7], [8] and work is still to be done to approach the performance of Maximum-Likelihood (ML) decoding at a reasonable cost.

The remainder of the paper is organized as follows: Section II presents the construction of Cortex Codes and gives an overview of the existing Cortex decoders. Section III depicts the proposed decoder architecture. Section IV first shows the synthesis results and BER measurement for the Golay code. Then, a stopping criterion is presented with results in terms of throughput increase.

II. Cortex code decoder

This section presents a brief state-of-the-art of Cortex code decoding. We particularly focus on the method presented in [7]. Then, we propose to modify this architecture in order to improve the decoder performance.

A. ML decoding

Let us consider a (N,K) binary linear code $\mathbf{C}$ and let $C = (c_1, c_2, \cdots, c_N)$ be a codeword of $\mathbf{C}$. For BPSK transmission, C is mapped into the bipolar sequence $Y = (y_1, y_2, \cdots, y_N)$ with $y_i = (-1)^{c_i} \in \{\pm 1\}$. After transmission, the received sequence at the output of the sampler in the demodulator

is $Z = (z_1, z_2, \cdots, z_N)$ with $z_i = y_i + w_i$, where for $1 \leq i \leq N$, w_i's are statistically independent Gaussian random variables with zero mean and variance $\sigma^2 = N_0/2$. The Log-Likelihood Ratio (LLR) associated to the binary symbol c_i is thus $LLR(c_i) = \frac{2z_i}{\sigma^2}$. Assuming that the codewords are equally probable, the ML decoding is reduced to:

$$\hat{C} = \arg\min_{C \in \mathbf{C}} \{P(Z/C)\} \quad (1)$$

Using the transformation in [9], [10], Equation (1) can be replaced by:

$$\hat{C} = \arg\min_{C \in \mathbf{C}} \{\sum_{i=1}^{N} |z_i| \delta(c_i, z_i)\} \quad (2)$$

where $|z_i|$ is the absolute value of z_i and $\delta(c_i, z_i)$ equals 0 if the hard decision $HD(z_i)$ on z_i gives c_i (no transmission error) and equals 1 otherwise (transmission error).

Going back to the auto-dual code, we can separate the function cost due to the K received LLRs of information $(L_X(i)_{i=1..K})$ and K received LLRs of redundancy $(L_R(i)_{i=1..K})$. Let $C_X = (X, R = P.X)$ be a codeword, then the distance $D(C_X)$ between C_X and the received LLRs is defined as: $D(C_X) = D(X) + D(P.X)$, where:

$$D(X) = \sum_{i=1}^{K} |L_X(i)| \delta(x_i, sign(L_X(i))) \quad (3)$$

$$D(P.X) = D(R) = \sum_{i=1}^{K} |L_R(i)| \delta(r_i, sign(L_R(i))) \quad (4)$$

To reduce the complexity of the ML decoding (testing the 2^K codewords), sub-optimal decoding methods have been proposed. The first family of sub-optimal algorithms is based on the exchange of information between processing nodes, such as the Belief Propagation (BP) algorithm. The second family exploits the reliability of the received symbols to search for the most likely codeword in a reduced set of codewords.

B. Iterative algorithm

BP decoding is a soft-input soft-output decoding algorithm relying on the exchange of soft information along the edges of a graph defined by the parity check matrix [11]. The BP algorithm is known to closely approximate the performance of optimal Maximum A Posteriori (MAP) decoding at reduced complexity for codes with sparse parity-check matrices. However, it works poorly with Cortex codes because their parity-check matrices are not sparse.

Different techniques are then investigated. The first one consists in an analog Cortex decoder that replaces the discrete iterations with a continuous processing [5] and shows better performance than an LDPC-like decoder. The second strategy uses a stochastic processing [6] to compute BP, leading to a decoding performance at 0.8 dB from the ML decoding for the (32,16,8) Cortex code.

C. Reduced search algorithms

Reduced search algorithms are based on the reduction of the space of search from $\mathbf{C}$ to a subset $\mathbf{C}_Z$ of codewords that could be close to the received vector Z. Several strategies could be applied: For example, modification of the value of the least reliable bits of the received codeword and perform a decoding algorithm to search for a codeword (known as Chase's algorithm [9]). Another method [12] encodes the K most reliable bits using a modified generator matrix deduced by Gaussian elimination. In [7] and [8], the authors exploit the auto-duality of Cortex codes to create two lists of codewords: the first list is generated from the least reliable information bits X and the second one from the least reliable redundant bits R. Just by simple encoding operations, this method adds diversity in the search of candidate codewords, leading to a very good decoding performance. In contrast, the Chase's algorithm [9] requires several explicit algebraic decoding operation, while the method proposed by Fossorier et al. in [12] requires a Gaussian elimination of the encoding matrix.

Since the error pattern generation is symmetrical for information and parity bits, only the former is presented. In [7], [8], the error patterns are generated by determining the first λ bits of smallest reliability and by testing exhaustively the 2^λ possible error patterns among these λ bits (typical values of λ are 3, 4 or 5).

Note that the generated list of codewords is not optimal, since other pattern errors containing other bits can lead to information vectors X of smaller distances (see eq. 2). For example, if $|L_X| = \{|L_X(i)|\}_{i=1..5} = \{0.35, 0.2, 0.1, 0.35, 0.3\}$ and $\lambda = 3$, the 8 pattern errors imply only the bits $\{x_3, x_2, x_5\}$, with a maximum cost of 0.6 for the error pattern "01101". However, the error patterns "10000" and "00010", which both lead to a cost of 0.35, are never tested.

D. Word generator based on minimum distance

In this paper, we propose to overcome this problem by generating the entire sorted list of candidates with the first ρ smallest distances. The idea is to generate explicitly the complete list of codewords $X^{up}(l)$ and its associated smallest increasing distance $D(X^{up})(l)$, for $l = 1 \ldots \rho$, where $l \leq l' \Rightarrow D(X^{up})(l) \leq D(X^{up})(l')$ (and also the symmetrical list $R^{up}(l), l = 1..\rho$). Doing so, we are guaranteed to test only good candidates. It also allows us to define a criterion to stop the decoding process when it becomes useless, thus reducing the average time and power dissipation to decode a codeword (stopping criterion).

III. Decoder architecture

This Section describes the word generator architecture and its integration in the global decoder architecture.

A. Word generator architecture

For the candidate codewords generation, different techniques have already been proposed [13], [10]. In [10], a systolic architecture generates binary vectors for Non-Binary LDPC decoders. This architecture can be efficiently used as

a codeword generator since it produces the codewords in increasing order (in terms of distance). This means that the ρ first half words generated are the closest to the received half word.

The systolic architecture is based on K Processing Elements (PEs) that are serially connected from $j = 1$ to $j = K$. The j^{th} PE receives from the $(j-1)^{th}$ PE the sorted list $(X^{up}_{j-1}(l), D(X^{up}_{j-1}(l))_{l=1..min(2^{j-1},\rho)}$ generated from the first $j-1$ received symbols $\{z_i\}_{i=1..j-1}$. From this list and the z_j value, the j^{th} PE provides to the $j+1^{th}$ PE the sorted list generated from the first j received symbol. After propagating through the K PEs (latency of $2K$ clock cycles), every cycle, the word generator provides a new word $X^{up}(l)$ (or $R^{up}(l)$) and its associated increasing distance $D(X^{up})(l)$ for $l = 1 \ldots \rho$, i.e. $l \leq l' \Rightarrow D(X^{up})(l) \leq D(X^{up})(l')$. The reader is invited to refer to [10] for more details on the hardware implementation of this algorithm.

B. Pipelined architecture

Fig. 2 shows the decoder architecture which is pipelined for high throughput. The enable signal En_{in} is used on rising edge to indicate the start of the decoding of a new word. At the start of the decoding process, K LLRs are loaded in parallel at the two word generator entities. The falling edge indicates the end of the decoding of the current word. When the enable signal is forced to zero, a new word is fetched. The enable signal is propagated through the decoder so that the different elements are reset (FIFO, Memory) in one cycle for the decoding of the next word. During the distance calculation, the distance already computed in the word generator is added to the distance of the other half of the codeword. To reduce the complexity, the distance computation can be performed by adding only the channel LLRs of erroneous bits [9]. These LLRs are read from a FIFO to deal with the word generator delay.

IV. Application case

The Cortex decoder has been simulated and implemented on an FPGA platform for validation purposes.

A. Simulation results

Fig. 3 illustrates the BER performances of fixed point decoders for $N = 24$, 32, 40 and 56 bits. Thanks to hardware emulation on FPGA [14], very low BER values are obtained.

The curve in dash line illustrates simulation with a combination of the $\lambda = 4$ minimal LLRs as in [7]. The number of generated words with the two methods are equal, but the performance is improved with the minimum-distance-based word generation. Note that the number of generated words increase exponentially with N.

B. Synthesis

Table I shows the synthesis results of the implementation of the Cortex (24,12,8) decoder on an FPGA platform containing a Xilinx Virtex 5 XQ5VLX85. Note that the Word generator, Encoder and Distance entities are instantiated twice in the implementation. Most of the complexity of the decoder resides in the word generator. The maximum frequency, after place and route, is 400MHz. For comparison, the decoder implemented in [7] on a Virtex 5 FPGA requires 2905 slice registers and 1114 slice LUTs.

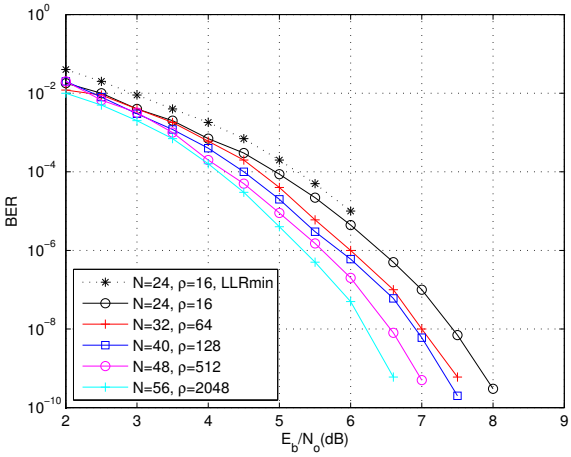

Fig. 3. Fixed point simulation results for $N = 24$, 32, 40, 48 and 56

XQ5VLX85	REG	LUT logic	LUT RAM
decoder	4650	5730	562
Word generator	1611	2450	248
PE_1	154	236	16
PE_{11}	156	243	36
Encoder	25	73	0
Distance	325	242	33
min	120	20	1

TABLE I
Synthesis Results for Cortex (24,12,8) decoder

C. Air throughput without stopping criterion

Table II shows the performance in terms of BER at $E_b/N_0 = 5$ dB for $N = 24, 32, 40, 48$ and 56. The table also shows the number of words generated to reach a BER at 0.1 dB from the ML decoding. The decoding latency is expressed as $L_{dec} = 3(K-1) + log_2(K) + 8 + \rho$. For high frequency, each PE is pipelined in 3 cycles. The word generator latency corresponds to the term $3(K-1)$. The term $log_2(K)$ corresponds to the distance calculation and the term 8 corresponds to the number of pipeline steps in the architecture. Finally, ρ represents the number of tested codewords. For comparison, the decoding latency in [7] is 80 cycles, but

N=	24	32	40	48	56
BER	1.10^{-4}	5.10^{-5}	2.10^{-5}	1.10^{-5}	4.10^{-6}
ρ	16	64	128	512	2048
L_{dec}	61	121	192	594	2142
Mb/s	300	100	62	16	6

TABLE II
Performance and air throughput for $N = 24$, 32, 40, 48 and 56

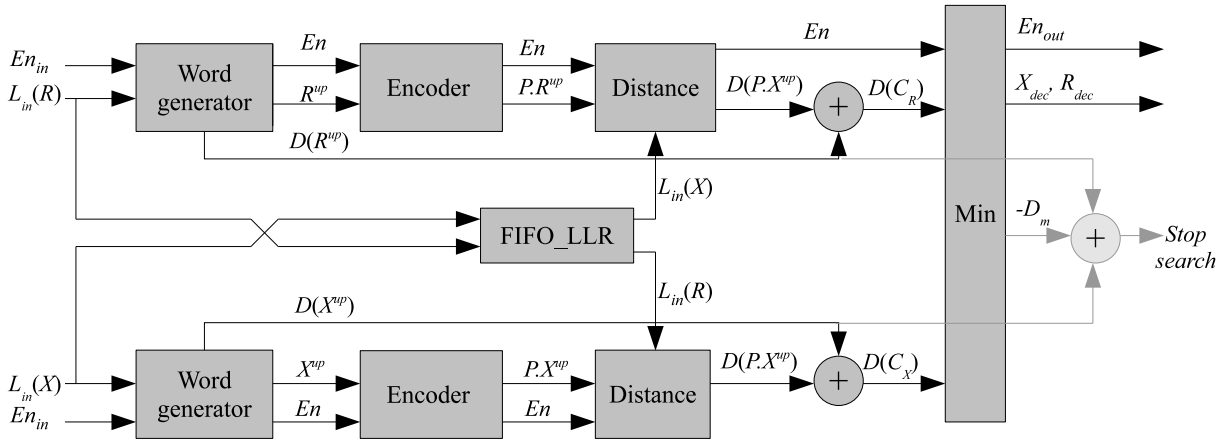

Fig. 2. Code Cortex decoder architecture

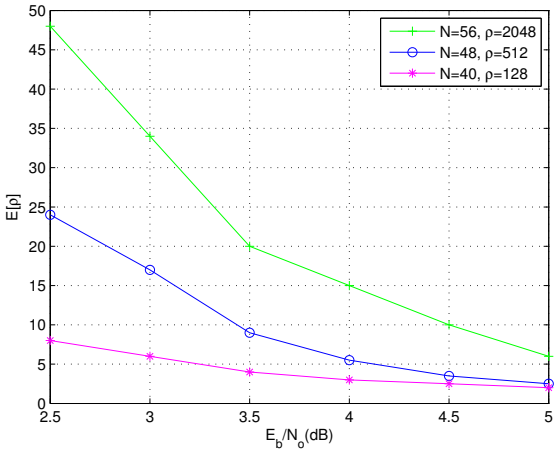

Fig. 4. Average number of generated words before the ML codeword is found as a function of the signal-to-noise ratio

maximum frequency is 72 MHz. Thanks to pipelining, the air throughput is given by $K \times F_{clk}/\rho$. The air throughput is 300 Mb/s for $N = 24$ and 100 Mb/s for $N = 32$. For comparison, air throughput in [7] is 36 Mb/s for $N = 32$, with a latency of 61 cycles.

The increasing ρ value leads to a throughput reduction down to 6 Mb/s for $N = 56$. For $N > 32$, a stopping criterion should be used to reduce the average number of generated words and thus increase the throughput.

D. Optimal stopping criterion

Fig. 4 illustrates the average number of generated codewords $E[\rho]$ before the ML codeword is found. Simulation results are based on a "genius" stopping criterion, i.e., the decoding stops as soon as one of the two word generators provides the ML codeword. For $N = 56$, at $E_b/N_0 = 5$ dB, an ML codeword is found on average after testing 6 words instead of 2048 (thus a 99.7% computation time saving would be obtained). However, proving that a given codeword is the ML codeword is an NP-hard problem.

In practice, our stopping strategy is to stop the decoding process when no better codeword can be found. Let $D_m(l)$ be the minimum distance found after testing the first l codewords. If $D(X^{up}(l)) + D(R^{up}(l)) > D_m(l)$ (stopping criterion), then the decoding process can stop, since the next generated codewords lead to a distance greater than $D_m(l)$.

proof by contradiction: Let us assume that a better codeword can be found for a value $l' > l$. This codeword can be either $C_X(l')$ (hypothesis **H1**) or $C_R(l')$ (hypothesis **H2**). Let us consider first hypothesis **H1**. We have:

$$D(X^{up}(l')) + D(P.X^{up}(l')) < D_m(l) \quad (5)$$

According to the stopping criterion:

$$D(X^{up}(l')) + D(P.X^{up}(l')) < D(X^{up}(l)) + D(R^{up}(l)) \quad (6)$$

Since $D(X^{up}(l')) \geq D(X^{up}(l))$, then

$$D(P.X^{up}(l')) < D(R^{up}(l)) \quad (7)$$

This inequality implies that $R = P.X^{up}(l')$ has already been tested for a value $q \leq l$ and thus, that $D_m(l) \leq D(C_X(l'))$, which is in contradiction with the initial hypothesis **H1**. In the case of hypotheses **H2**, symmetrical arguments also lead to a contradiction, which confirms the proof.

Fig. 5 illustrates the evolution of the distances as a function of l for N=40 and $E_b/N_o = 3$. In this simulation, the stopping criterion stopped the decoding process after 19 generated codewords while the ML codeword is found after 12 codewords.

Fig. 6 shows the average number of words before the stopping criterion detects that an ML codeword has been found.

The main advantage of the proposed stopping criterion resides in its implementation simplicity and the absence of BER performance loss.

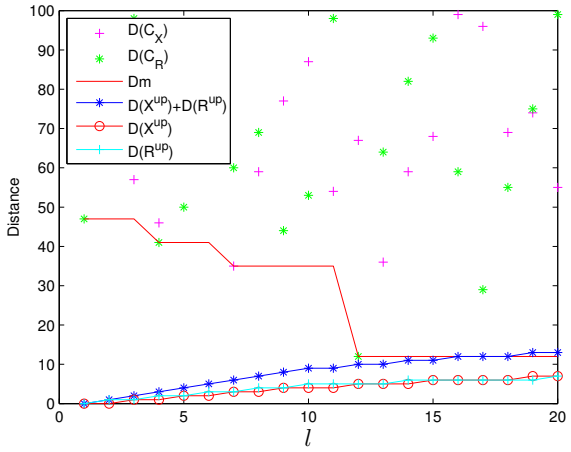

Fig. 5. Distance evolution as a function of l

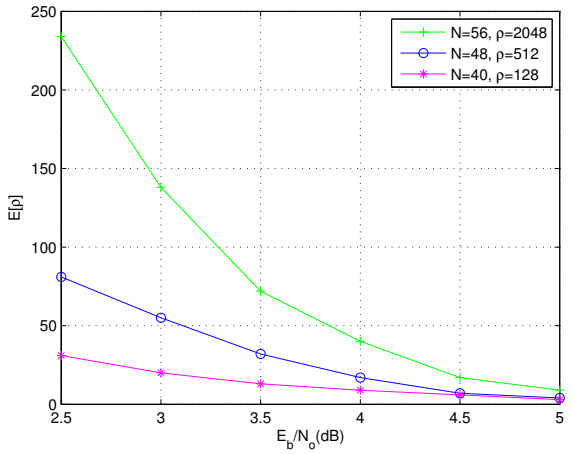

Fig. 6. Average number of generated words using with the stopping criterion

Table III shows the air throughput performance at $E_b/N_o =$ 5 dB for $N = 24$, 32, 40, 48 and 56. The latency of decoding is given by $L_{dec} = 3(K-1)+log_2(K)+8+E[\rho]$. Because of the stopping criterion, the decoding of two consecutive words cannot be pipelined. The air throughput is replaced by $K \times F_{clk}/L_{dec}$ Mb/s.

Note that for $N = 24$, the air throughput is reduced compared to the pipelined implementation without the stopping criterion (Table II) for which the air throughput reaches 300 Mb/s. For $N = 40$, 48 and 56, the stopping criteria makes it possible to keep the air throughput above 100 Mb/s.

N=	24	32	40	48	56
ρ	16	64	128	512	2048
$E[\rho]$	1.5	2	3	5	9
L_{dec}	47	59	73	87	103
Mb/s	101	108	109	109	108

TABLE III
AIR THROUGHPUT FOR $N = 24$,32,40,48 AND 56

V. CONCLUSION

In this paper, we consider the design of efficient Cortex code decoders. An existing soft-decision decoding algorithm is used to exploit the code structure to achieve ML performance. We added a word generator to the architecture and an optimal stopping criterion. We showed that the proposed decoder architecture provides performance which is very close to ML decoding for a fraction of the ML decoding complexity. The implemented pipelined architecture achieves a throughput of 300 Mb/s with $N = 24$ bits. The implementation of a simple stopping criterion provides an efficient solution for $N > 32$. Future work will be dedicated to optimize the hardware implementation of the decoder in terms of area and frequency, as well as the stopping criterion.

REFERENCES

[1] R. G. Gallager, "Low-density parity-check codes," *Information theory,IRE Transaction on*, vol. 8, pp. 21–28, Jan. 1962.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes," in *IEEE International Conference on Communications*, vol. 2, pp. 1064–1070, May 1993.

[3] J.-C. Carlach and C. Vervoux, "A new family of block turbo-codes," in *13th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, (Honolulu, USA), Nov. 1999.

[4] J.-C. Carlach and A. Otamani, "A systematic construction of self-dual codes," *IEEE Transactions on Information Theory*, vol. 49, pp. 3005–3009, Nov. 2003.

[5] J. Perez-Chamorro, F. Sequin, C. Lahuec, M. Jezequel, and L. M. G., "Decoding a family of dense codes using the sum-product algorithm," in *IEEE International Symposium on Circuit and Systems (ISCAS)*, (Taipei, Taiwan), pp. 2685–2688, May 2009.

[6] M. Arzel, C. Lahuec, C. Jego, W. Gross, and Y. Bruned, "Stochastic multiple stream decoding of cortex codes," *IEEE Transaction on Signal Processing*, vol. 59, pp. 3486–3491, 2011.

[7] P. Adde, C. Jego, R. Le Bidan, and J. Perez-Chamorro, "Design and implementation of a soft-decision decoder for cortex codes," in *17th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, pp. 663–666, Dec. 2010.

[8] P. Adde, C. Jego, and R. Le Bidan, "Near maximun likelihood soft-decision decoding of a particular class of rate 1/2 systematic linear block codes," *Electronic Letters*, vol. 47, pp. 259–260, Feb. 2011.

[9] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Transactions on Information Theory*, vol. 18, pp. 170–182, Jan. 1972.

[10] A. A. Ghouwayel and E. Boutillon, "A systolic LLR generation architecture for non-binary ldpc decoder," *IEEE Communications Letters*, vol. 15, pp. 851–853, Aug. 2011.

[11] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transaction on information theory*, vol. 47, pp. 498–519, Feb. 2001.

[12] M. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Transaction on Information Theory*, vol. 41, pp. 1379–1396, Sept. 1995.

[13] A. Valenbois and M. Fossorier, "An improved method to compute lists of binary vectors that optimize a given weight function with application to soft-decision decoding," *IEEE Communication letters*, vol. 5, pp. 456–458, Nov. 2001.

[14] E. Boutillon, Y. Tang, C. Marchand, and P. Bomel, "Hardware discrete channel emulator," in *IEEE International Conference on High Performance Computing and Simulation (HPCS)*, (Caen, France), pp. 452 – 458, June 2010.