Design, Simulation and Hardware Implementation of a Digital Television System: LDPC channel coding

(Invited Paper)

Tarciano F. Pegoraro, Fábio A. L. Gomes, Renato R. Lopes, Roberto Gallo,
José S. G. Panaro, Marcelo C. Paiva, Fabrício A. Oliveira and Fábio Lumertz[1]

***Abstract*— In this paper, we describe a hardware implementation of a low-density parity-check (LDPC) code for the MI-SBTVD Project described in [1], which aims at the development of an advanced Digital Television (DTV) System for the SBTVD Program. We begin the paper by describing the concept of LDPC codes and the design strategies we have used. We also provide some simulation results that show that the proposed code greatly outperforms codes used by other DTV standards. Finally, we provide details of the hardware implementation of the code.**

Keywords - LDPC codes, Digital Television, Implementation of LDPC, SBTVD.

I. Introduction

Low-density parity-check (LDPC) codes were introduced in 1962 by Gallager [2], who also proposed an iterative decoding algorithm for them, known as the sum-product algorithm (SPA). However, LDPC codes laid more or less dormant until the late 1990's, when they were rediscovered by MacKay [3]. Recently, LDPC codes were shown to be able to compete in performance with turbo codes of similar decoding complexity. Besides, they have the advantage of allowing a finer adjustment of the trade-off between performance and decoding complexity [4].

Furthermore, the SPA has been shown to be part of a more general class of algorithms, known as message-passing algorithms, which also includes the *min-sum*. The *min-sum* algorithm performs a little worse than the SPA, but have lower complexity, offering an alternative for resource-limited implementations. Most of the current research on LDPC codes deals with decoder implementation details. Some of the issues faced by system designers are related to the trade-off between memory and processing time, which basically depends on the degree of parallelism used in the implementation, or the trade-off between memory and performance, which is related to the number of bits used for data representation at the receiver.

Another challenge faced by designers of practical LDPC systems is encoding complexity. Richardson and Urbanke [5] proposed an encoding method for a random LDPC code with average complexity of $0,017^2n^2 + O(n)$ for typical irregular codes. Another approach to reduce encoding complexity is to design structured parity-check matrices that permit the use a systematic encoding algorithm. One such case is the family of the so called extended irregular repeat-accumulate (eIRA) codes [6], described later in this paper.

In this article, we detail a practical implementation of an LDPC channel coding system developed for digital television transmission. We offer a complete description of the development process, from the code design to the hardware implementation of the codec. As a starting point, we used eIRA codes, partly inspired by the LDPC codes specified by the DVB-S2 standard [7]. As will be shown in the remainder of this article, many of the constraints that needed to be imposed at the code design level were directly related to the limitations at the hardware implementation level. Therefore, the development process moved back-and-forth from code design to hardware design, until a satisfactory system could be implemented and tested.

The basic concepts of LDPC codes are discussed in more detail in Section II. Section III describes the method used to design structured eIRA codes with optimized performance for the constraints available. The performance of the resulting codes is described in section IV. Section V describes the details for the hardware implementation of the decoder. Finally, Section VI concludes the article.

II. LDPC Codes

LDPC codes are (n, k) binary linear block codes that have a low-density parity-check matrix $\mathbf{H}$. They may be described in terms of a Tanner graph [8], which is a bipartite graph containing variable and check nodes. Each bit in the codeword corresponds to a variable node, and each parity-check equation corresponds to a check node. A variable node is connected to a check node in the Tanner graph if and only if the corresponding codeword bit takes part in the corresponding parity-check equation. For instance, Fig. 1 shows the Tanner graph associated with the $(7, 4)$ Hamming code, whose parity-check matrix is given by

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \quad (1)$$

In Fig. 1, the thick lines correspond to the second row of $\mathbf{H}$.

In [4], it was shown that the performance of an LDPC code is determined by the so-called degree distribution, which we

[1]Tarciano F. Pegoraro, Fábio A. L. Gomes, Renato R. Lopes, Roberto Gallo, Fabrício A. Oliveira and Fábio Lumertz are with the State University of Campinas - UNICAMP, Campinas, SP, Brazil, 13083-852, (tarciano, adrianol, rlopes, lumertz, fabricio@decom.fee.unicamp.br), (gallo@ic.unicamp.br). José S. G. Panaro and Marcelo C. Paiva are with the National Institute of Telecommunications - INATEL, Santa Rita do Sapucaí, MG, Brazil, 37540-000 - (panaro, mcpaiva@inatel.br). This work was supported by FINEP - Financiadora de Estudos e Projetos, by FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo and by Linear Equipamentos Eletrônicos S/A.

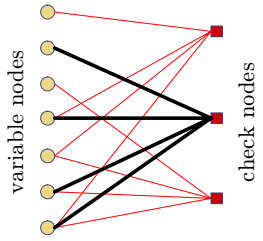

Fig. 1. Tanner graph for the $(7, 4)$ Hamming code.

now define. We first define the degree of a node to be the number of edges connected to it. Let λ_i (resp. ρ_i) be the fraction of edges connected to variable nodes (resp. check nodes) of degree i. Then, $\lambda(x) = \sum_i \lambda_i x^{i-1}$ is the variable node degree distribution, and $\rho(x) = \sum_i \rho_i x^{i-1}$ is the check node degree distribution. Since these polynomials determine the performance of an LDPC code, several algorithms have been proposed to determine the optimal distributions. These will be discussed in Section III-A.

A. Decoding LDPC Codes

The main goal of any decoding algorithm is to determine the maximum a posteriori (MAP) estimates of the transmitted bits, since these minimize the BER. For binary systems, computing the MAP estimate of c_i, the i-th bit of the transmitted codeword, is equivalent to computing the log-likelihood ratio (LLR)

$$L_i = \log \frac{Pr(c_i = 0|\mathbf{r})}{Pr(c_i = 1|\mathbf{r})}, \tag{2}$$

where $\mathbf{r}$ is the received sequence. Then, the MAP estimate is $c_i = 0$ if $L_i > 0$, and $c_i = 1$ otherwise. Computing the LLR precisely is not computationally feasible. However, based on the Tanner graph, it is possible to derive iterative algorithms that compute good approximations for the LLR.

The iterative algorithms are based on an exchange of messages between the variable nodes and the check nodes. Considering the sum-product algorithm (SPA) [3], [2] and a variable node j, the SPA computes the message going from this node to a check node i as

$$v_i = \sum_{j \neq i} u_j. \tag{3}$$

The message from a check node to a variable node is

$$u_i = 2 \tanh^{-1} \left(\prod_{j \neq i} \tanh \left(\frac{v_j}{2} \right) \right). \tag{4}$$

Note that for both types of nodes the outgoing message at a given edge is not a function of the incoming message at that same edge. This means that all messages correspond to extrinsic information, which avoids positive feedback of information.

B. Encoding LDPC Codes

As seen above, decoding LDPC codes is fairly easy. Unfortunately, encoding LDPC codes is not as simple. Indeed,

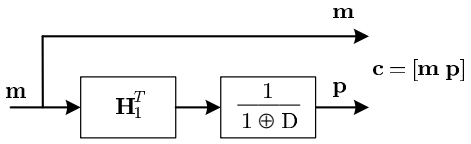

Fig. 2. The encoder for the extended irregular repeat-accumulate code.

assume that we want to obtain the systematic generator matrix $\mathbf{G}$ corresponding to $\mathbf{H}$. We begin by writing $\mathbf{H}$ as

$$\mathbf{H} = [\mathbf{H}_1 \ \mathbf{H}_2], \tag{5}$$

where $\mathbf{H}_1$ and $\mathbf{H}_2$ are $n-k \times k$ and $n-k \times n-k$ binary matrices. Then, $\mathbf{G}$ is given by

$$\mathbf{G} = \left[\mathbf{I}_k \ \mathbf{H}_1^T \mathbf{H}_2^{-T}\right]. \tag{6}$$

Unfortunately, the inverse of a sparse matrix is not sparse, so brute-force encoding using $\mathbf{G}$ is a complex operation.

Several LDPC codes have been proposed that have low-complexity encoding. Normally, this is achieved by imposing some structure on the parity-check matrix. In this paper, we use the extended irregular repeat-accumulate codes (eIRA) of [11]. In these codes, the matrix $\mathbf{H}_1$ is still randomly generated to meet the prescribed degree distributions. The matrix $\mathbf{H}_2$, on the other hand, is given by

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 1 \end{bmatrix}. \tag{7}$$

Using this definition, the multiplication of a vector by $\mathbf{H}_2^{-T}$ can be implemented with a simple accumulator. The resulting encoder for the eIRA is shown in Fig. 2.

C. Capacity of LDPC Codes

For many channels and iterative decoders, LDPC codes exhibit a *threshold of convergence*, *i.e.*, as the block length tends to infinity, an arbitrarily small bit-error probability can be achieved if the noise level is smaller than a certain threshold. For a noise level above this threshold, on the other hand, the probability of bit error is larger than a strictly positive constant. To determine the value of this threshold Richardson and Urbanke developed an algorithm called *density evolution* [4], [9], which iteratively computes the probability density functions (pdf) of the messages.

However, the computation of the threshold and the optimization of $\lambda(x)$ and $\rho(x)$ through density evolution are still computationally intense tasks. A low-cost alternative is based on the observation that the pdf of the messages v can be well approximated by Gaussian mixtures. However, the same is not valid for the messages u. Thus, Ardakani and Kschischang [10] proposed a semi-Gaussian approximation, wherein the pdf of the messages v is computed under the Gaussian-mixture approximation (so that only the means and variances of the

constituent Gaussians need to be evaluated), while the pdf of the messages u is computed applying just one iteration of the density evolution algorithm.

III. Project of Structured eIRA Codes for Digital Television

The project of a structured eIRA LDPC code [6], [11] consists of two stages: the optimization of the degree distributions $\lambda(x)$ and $\rho(x)$ and the construction of the $\mathbf{H}_1$ submatrix of the parity check matrix. In the first stage, the goal is to maximize the noise threshold of the code for a given rate. In the second, the objective is to distribute the "1's" in $\mathbf{H}_1$ respecting the degree distributions, minimizing the small cycles in the Tanner graph and facilitating the storage of $\mathbf{H}_1$. In this section, we describe these two stages in more detail.

A. Optimizing $\lambda(x)$ and $\rho(x)$

The joint optimization of $\lambda(x)$ and $\rho(x)$ is a task with a heavy computational cost. However, it is possible to reduce this complexity using a concentrated degree distribution for $\rho(x)$ without much loss in performance [15]. In this case, the degree of all check nodes can be fixed to d_r and the optimizations is made just over $\lambda(x)$. Employing density evolution with the semi-Gaussian approximation, it is possible to reduce the problem to the following linear program

$$\begin{aligned} &\text{maximize} \quad \textstyle\sum_{i=2}^{d_l} \frac{\lambda_i}{i} \\ &\text{subject to} \quad \begin{cases} \lambda_i \geq 0 \\ \sum_{i=1}^{d_l} \lambda_i = 1 \\ \lambda_1 = \frac{1}{(n-1)} \sum_{i=2}^{d_l} \frac{\lambda_i}{i} \\ \lambda_2 = \frac{2(n-k-1)}{(k+1)} \sum_{i\neq 2}^{d_l} \frac{\lambda_i}{i} \\ \sum_{i=2}^{d_l} \lambda_i g_i(p_{in}) < p_{in} \quad \forall p_{in} \in (0, p_0] \end{cases} \end{aligned} \tag{8}$$

where the constraints imposed by the matrix $\mathbf{H}_2$ are taken into account.

The inputs to this linear program are the channel model, the noise level (threshold), the codeword length n, d_l and $\rho(x) = x^{d_r - 1}$. The optimization process is repeated and the input noise level is changed until the resulting code rate is equal to the required rate r.

As a digital television system requires a quasi-error free reception (BER=10^{-11}), our goal for the design of the LDPC codes was to get the code with the best performance at a BER=10^{-5}. In the overall system, an external Reed-Solomon code will be responsible for minimizing any error-floor and producing a quasi-error-free receiver output.

B. Building the Parity Check Matrix

As a first step to constructing a matrix $\mathbf{H}_1$ of dimension $(n-k) \times k$, a matrix $\mathbf{A}$ is created with dimension $a \times b$, m times smaller then $\mathbf{H}_1$. As will be shown ahead, m is the number o branches that can be processed in parallel by the decoder. The matrix $\mathbf{A}$ is constructed respecting the computed degree distributions $\lambda(x)$ and $\rho(x)$ and minimizing the short cycles in the Tanner graph. This can be done with the progressive edge-growth algorithm [12].

Now, replacing each "0" in $\mathbf{A}$ by a null square matrix of order m and each "1" by permutations of the identity matrix $\mathbf{I}$ of order m, a matrix $\mathbf{B}$ is built with the same dimensions of $\mathbf{H}_1$. The permutation pattern for the "1" can follow the array codes [13]. In this case, the matrix $\mathbf{H}_1$ is obtained by permutations of the rows of $\mathbf{B}$.

The process does not avoid short cycles in the Tanner graph, but greatly reduces them. Furthermore, the regularity in the sub-matrix $\mathbf{H}_1$ allows for a compact storage of the parity matrix $\mathbf{H}$ and reduces the complexity of the codec.

IV. Simulation Results

A DTV receiver must be robust to several kinds of channels. However, as shown in [14], LDPC codes designed for AWGN channels are still robust for other channels. Thus, the codes were designed for the AWGN channel.

The chosen code rates were $1/2$, $2/3$, $3/4$, $5/6$ and $7/8$, and the codeword length was 9792 bits. Employing density evolution with semi-Gaussian approximation as described in Section III, resulted in the degree distributions and threshold of convergence of Table I. The capacity of these degree distributions vary from 0.13 dB to 0.37 dB away from Shannon limit for the binary-input AWGN channel.

TABLE I
Degree Distribution

rate	$1/2$	$2/3$	$3/4$	$5/6$	$7/8$
$\rho(x)$	x^6	x^{11}	x^{16}	x^{23}	x^{25}
λ_1	0.00003	0.00003	0.00002	0.00003	0.00003
λ_2	0.2857	0.1666	0.1176	0.0833	0.0769
λ_3	0.2544	0.3779	0.4118	0.5000	0.6923
λ_5	0.1223				
λ_6					0.2308
λ_9		0.0989			
λ_{10}	0.3197			0.4167	
λ_{11}	0.0180				
λ_{12}		0.3566	0.4767		
thresh. (dB)	0.428	1.236	1.804	2.582	3.211
gap (dB)	0.24	0.13	0.18	0.27	0.37

Monte Carlo simulations were done with the codes generated with the ensembles in Table I. Fig. 3 shows the performance in terms of bit error rate (BER). These codes are 0.7 dB to 1 dB away from Shannon Limit for the binary AWGN channel (considering a BER equivalent to 10^{-5}). We can also see in Fig. 3 that the designed (9792,4896) LDPC code is 3 dB better (at BER=10^{-5}) than the convolutional code of the DVB-T and ISDB-T systems. For these simulations we considered BPSK modulation, AWGN channel and $m = 51$ decoder branches. The BER was computed after 50 wrong codewords, using 50 iterations in the sum-product decoder.

V. Hardware Implementation

Among the different code rates considered for the proposed digital television implementation, we have chosen the $3/4$ rate with a codeword length of 9792 bits. Thus, each codeword will consist of 7344 information bits and 2448 parity bits.

The decoder arrangement could be serial, in the form of a trellis structure, or parallel, in the form of a tree topology.

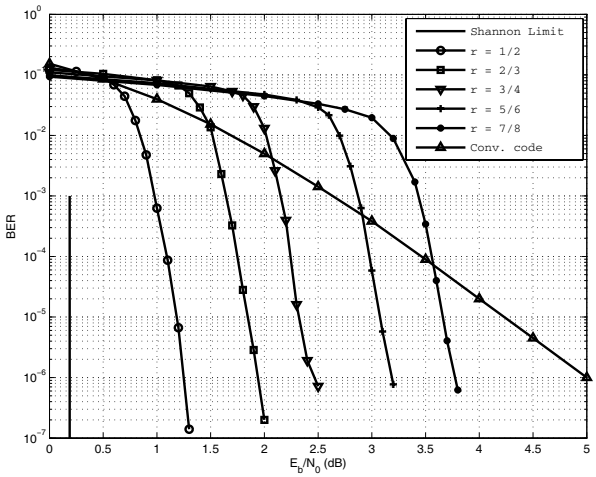

Fig. 3. Performance for the LDPC codes for Digital Television. Also shown are the Shannon limit and the performance of the convolutional code used in DVB-T and ISDB-T, both for rate 1/2.

The trellis structure has been adopted, especially due to the fact that it consumes an area significantly smaller than that of the parallel structure. Next we discuss the structure in more detail.

Perhaps the most significant point in the implementation stage is how to deal with the trade-off between high-speed processing and the device silicon area occupation. The minimum throughput required by the proposed system is 19.33 Mbps. On the other hand, there is the device's physical limitation of the silicon area, which limits the parallelization degree of the implementation.

The encoding and decoding systems were implemented into Field Programmable Gate Arrays (FPGA). The LDPC encoder was developed in Very High Speed Integrated Circuit Hardware Description Language (VHDL – IEEE 1164) using Quartus, a development tool from Altera. The LDPC decoder was implemented using VHDL and System Generator, from Xilinx. The encoder and decoder implementation details are presented next.

A. LDPC Encoder

As seen in Fig. 2, the LDPC encoder multiplies the message by $\mathbf{H}_1^T$ and sends the result through an accumulator. The accumulator is implemented with a single exclusive OR (XOR), which is easy to implement. On the other hand, a significant problem is how to store the matrix $\mathbf{H}_1$, which has dimensions 7344×2448. Fortunately, as discussed in Subsection III-B, it is only necessary to store the indices of the variable nodes connected to the 144 check nodes indexed by multiples of m, which in our case is 51. The variable-node indices are stored in a structure $\mathbf{T}$, which has 112 3-element rows and 32 12-element rows. Since each element in $\mathbf{T}$ is 12-bits long, $\mathbf{H}_1$ is represented using a total of 8.6 kbits, a reduction by a factor of approximately 2000, compared to the full representation.

Furthermore, the structure $\mathbf{T}$ was stored in 15 independent block RAMs (BRAMs), 3 holding the elements of the first 112 rows and 12 holding the elements of the last 32 rows. This allows simultaneous access to all the data necessary at each iteration, during which one parity bit is determined.

B. LDPC Decoder

For a hardware based decoder implementation, the expressions (3) and (4) can be rewritten for better efficiency. The total information available in the variable node v is given by $v = \sum_j u_j$. Then, the variable node messages, shown in (3), can be efficiently computed as

$$v_i = v - u_i, \tag{9}$$

which requires just $2w_v + 1$ additions. For the check-node operation in (4), it is possible to avoid the direct computation of the transcendental function $\tanh$. Assuming a check node with degree two and incoming messages U and V, (4) can be rewritten as

$$\begin{aligned} L(U \oplus V) &= 2\tanh^{-1}\left(\tanh\frac{U}{2}\tanh\frac{V}{2}\right) \\ &= \min\{|U|,|V|\} + z(U,V), \end{aligned} \tag{10}$$

where $z(U,V) = \log\left(\frac{1-\exp^{-|U+V|}}{1-\exp^{-|U-V|}}\right)$ is the *correction term.*

Approximating the correction term by $z(U,V) \equiv 0$, yields the lowest computational complexity, and the *min-sum* algorithm. An expression with a better tradeoff involving computational cost, precision and quantization effect is the *constant approximation* [17], given by

$$z(x,y) = \begin{cases} 0.5 & \text{if } |x| \le 2, |y| > 2|x| \\ -0.5 & \text{if } |y| \le 2, |x| > 2|y| \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

For the whole check node operation, different topologies can be used [16]. The *brute-force* computation of (4) is the most complex. A *parallel implementation* allows the highest throughput at the expense of the largest hardware area and more quantization effects. On the other hand, the *serial implementation* is very robust against quantization effects and requires less hardware resources when synthesized on an FPGA.

For a serial implementation, (4) can be rewritten as

$$u_1 = L(b_2) \tag{12}$$

$$u_i = L(f_{i-1} \oplus b_{i+1}),\ i = 2, \ldots, w_c - 1 \tag{13}$$

$$u_{w_c} = L(f_{w_c-1}) \tag{14}$$

where $f_1 = c_1$, $f_2 = f_1 \oplus c_2$, ..., $f_{w_v} = f_{w_v-1} \oplus c_{w_v}$, $b_{w_v} = c_{w_v}$, $b_{w_v-1} = b_{w_v} \oplus c_{w_v-1}$, ..., $b_1 = b_2 \oplus c_1$ are two sets of random auxiliary variables representing forward and backward message propagations among check nodes. One may note that the evaluation of u_i message by equation (13) requires the determination of the parameters $L(f_1)$, ..., $L(f_{w_v})$, $L(b_1)$, ..., $L(b_{w_v})$. These can be recursively determined replacing the known values for $L(c_1)$, ..., $L(c_{w_v})$ in equation (10). The resulting computational complexity is bounded by $3 \cdot (w_v - 2)$ operations.

The implemented decoder architecture was similar to that used in [18]. As seen in Fig. 4, this implementation has seven

major subcomponents: incoming buffer (inBuffer), output buffer (outBuffer), variable nodes processors (vnp), check node processors (cnp), interleaver and deinterleaver, and a control unit (CR). The last five units form the kernel.

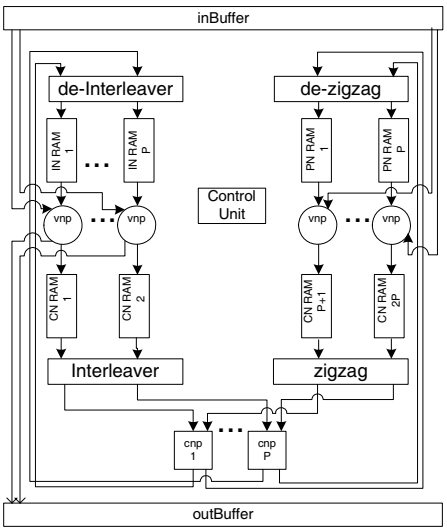

Fig. 4. The LDPC decoder architecture.

The inBuffer is responsible for maintaining all data block available and accessible to the kernel. It has a two block sized circular buffer with serial load and parallel read structure in order to minimize kernel stalls. The vnp compute (9) and the cnp compute (10). Both the interleaver and the deinterleaver perform random row addressing and random column rolling operations. A whole LDPC decoder block is sent in parallel between vnp and cnp through the interleavers and are stored and read from two internal data memories that hold LLR message values.

C. Results

The LDPC encoder was implemented in a Altera Stratix II 2S60 FPGA and developed in VHDL (Very High Speed Integrated Circuit Hardware Description Language). With the repeat accumulate structure of Fig. 2 and the regularity in the matrix $\mathbf{H}_1$, the encoder has a low computational complexity. In fact, only 4% of the device area and 5% of the built-in block RAMs were consumed.

For the decoder, we employed the constant approximation for the correction term and the serial implementation. The decoder was designed for a Xilinx Virtex II XC2V3000 FPGA using VHDL and System Generator, also from Xilinx. The clock frequency for the FPGA board was set to 97 MHz. The resource usage for the decoder were 186 built-in block RAMs (96%) and 9478 slices (61%). The messages were stored with 5 bits (1 for signal, 3 for the integer part and 1 for the fractional part) and, to respect the minimum throughput, the maximum number of iterations was set to 13.

VI. Conclusions

In this paper, we have described the design of an LDPC code and its hardware implementation. We have seen that the eIRA used in this paper indeed allows for a low-complexity encoder implementation, using less than 5% of an FPGA area and memory. The decoder implementation is more complex, occupying most of the device area. Simulation results show that for the AWGN channel and at a BER of 10^{-5}, the proposed code operates within 1 dB of the Shannon limit, and outperforms the convolutional code used in other systems by around 3 dB.

References

[1] José M. C. Brito et al., "Design, Simulation and Hardware Implementation of a Digital Television System: System Overview", The 9th International Symposium on Spread Spectrum Techniques and Applications - ISSSTA-2006, Manaus, Brazil, 2006.

[2] R. G. Gallager, "Low-density parity-check codes,"*IRE Transactions on Information Theory*, vol. IT-8, pp. 21–28, Jan. 1962.

[3] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Information Theory*, pp. 399–431, Mar. 1999.

[4] T. J. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599-618, Fev. 2001.

[5] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001.

[6] Y. Zhang, W. Ryan, and Y. Li, "Structured eIRA codes with low floors," *Proceedings of the International Symposium on Information Theory - ISIT2005*, pp. 174–178, Sep. 2005.

[7] DVB-S.2 Standard Specification, ETSI EN 302 307 V1.1.1 (2005-03). http://webapp.etsi.org/action/PU/20050322/en_302307v010101p.pdf

[8] R. M. Tanner, "A recursive approach to low-complexity codes," *IEEE Trans. Information Theory*, pp. 533–547, Sep. 1981.

[9] T. J. Richardson, A. Shokrollahi, and R. Urbank, "Design of capacity-approaching low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619-637, Fev. 2001.

[10] M. Ardakani and F. R. Kschischang, "A more accurate one-dimensional analysis and design of irregular LDPC codes,"*IEEE Trans. Communications*, vol. 52, no. 12, pp. 2106-2114, Dec. 2004.

[11] M. Yang, W. Ryan, and Y. Li, "Design of efficiently encodable moderate-length high-rate irregular LDPC codes," *IEEE Trans. Communications*, vol. 52, no. 4, pp. 564-571, Abr. 2004.

[12] X. Y. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Communications*, vol. 52, pp. 386–398, Feb. 2005.

[13] J. L. Fan, "Array codes as low density parity check codes,"*in Proc. of the 2nd International Symposium on Turbo Codes and Related Topics*, pp. 543–546, Sep. 2000.

[14] J. Hou, P. Siegel, and L. Milstein, "Performance analysis and code optimization of low density parity-check codes on Rayleigh fading channels," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 5, pp. 924–934, May 2001.

[15] S.-Y. Chung, T. J. Richardson, and R. Urbanke, "Analysis of sum-product decodign of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Information Theory*, vol. 47, pp. 657–669, Feb. 2001.

[16] X.-Y. Hu, E. Eleftheriou, D. Arnold, and A. Dholakia, "Efficient Implementations of Sum-Product Algorithm for Decoding LDPC Codes." *IEEE GLOBECOM2001*, v.2, pp. 1036–1036, 2001.

[17] M. Shen, H. Niu, H. Liu, and J. A. Ritcey, "Finite precision implementation of LDPC coded M-ary modulation over wireless channels," *Conference on Signals, Systems and Computers - ACSSC2003*, v. 1, pp. 114–118, Nov. 2003.

[18] F. Kienle, T. Brack, and N. Wehn, "A synthesizable IP Core for DVB-S2 LDPC code decoding" *Proc. of Design, Automation and Test in Europe*, v. 3, pp. 100–105, 2005.