Codificação LDPC em Sistemas de Televisão Digital

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

***Abstract*— In this paper, we describe the design methodology and hardware implementation of a low-density parity-check (LDPC) code for a digital television (DTV) system. We begin the paper describing LDPC codes and the design strategies we 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.**

***Index Terms*— LDPC codes, Digital Television, FPGA, SBTVD.**

***Resumo*— Neste artigo descreve-se a metodologia de projeto e implementação em hardware de um código de *matriz de paridade de baixa densidade* (LDPC) aplicado a um sistema de televisão digital. Começamos o artigo descrevendo códigos LDPC e as estratégias de projeto empregadas. Também fornecemos alguns resultados de simulação que mostram que o código proposto fornece um desempenho significativamente superior ao empregado em outros padrões de televisão digital. Finalmente, fornecemos detalhes sobre a implementação em hardware do código proposto.**

***Palavras chave*— Códigos LDPC, FPGA, SBTVD, Televisão Digital.**

I. Introdução

Códigos de matriz de paridade de baixa densidade, ou códigos LDPC (do inglês, *low-density parity-check codes*), como são mais conhecidos, foram introduzidos por Gallager em 1962 [1] [2], quando o próprio Gallager também introduziu o seu algoritmo de decodificação, conhecido hoje como algoritmo Soma-Produto (SPA) ou *belief propagation*. Embora introduzido há mais de 3 décadas, o tema permaneceu praticamente estático, sem receber maior atenção da comunidade científica, até a sua "redescoberta" por Mackay [3], no fim dos anos 90. O grande impulso dessa redescoberta foi o progresso dos códigos turbo, outra família de códigos cuja decodificação se baseia em um algoritmo iterativo e que apresenta desempenho próximo ao limite de Shannon. Observou-se desde então que códigos LDPC eram capazes de atingir o mesmo nível de desempenho dos códigos turbo, com praticamente a mesma complexidade, porém com a vantagem de permitir um ajuste mais fino do compromisso entre complexidade e desempenho. Além disso, progressos mais recentes introduziram estruturas especiais de matrizes de paridade, como é o caso dos códigos eIRA (do inglês, *extended Irregular Repeat-Accumulate*) [9], utilizados no projeto aqui apresentado, que permitem reduzir não só a complexidade do processo de decodificação, mas também do processo de codificação.

Tarciano F. Pegoraro, Fábio A. L. Gomes, Fábio Lumertz, Renato R. Lopes, Fabrício A. Oliveira e Roberto Gallo estão vinculados à Universidade Estadual de Campinas - UNICAMP, Campinas, SP, Brasil, 13083-852, (tarciano, adrianol, lumertz, rlopes, fabricio@decom.fee.unicamp.br), (gallo@ic.unicamp.br). Marcelo C. Paiva e José S. G. Panaro estão vinculados ao Instituto Nacional de Telecomunicações - INATEL, Santa Rita do Sapucaí, MG, Brasil, 37540-000.

Outra área onde vários progressos vêm sendo produzidos recentemente é a dos algoritmos de decodificação. A generalização do algoritmo SPA forma a classe dos algoritmos de passagem de mensagens, que, entre outros, inclui o algoritmo Min-Sum (ou Mínimo-Soma, em português). O Min-Sum pode ser visto como uma aproximação do SPA de menor complexidade e com desempenho um pouco pior, porém com grande utilidade em casos onde os recursos computacionais são mais restritos. Uma grande parcela das pesquisas recentes relacionadas a LDPC se concentra no tema dos algoritmos de passagem de mensagens, principalmente no desenvolvimento de versões de menor complexidade, utilizando mensagens compostas por um número reduzido de bits, e que possam ser implementados em hardware, na forma de ASICs (do inglês, *Application-Specific Integrated Circuit*) e FPGAs (do inglês, *Field-Programmable Gate Arrays*). Entre os aspectos que devem ser considerados pelo projetista estão as limitações de memória, área de silício e tempo de processamento.

Outro desafio do projeto de sistemas práticos usando códigos LDPC é a implementação do codificador. Várias propostas [5] [6] foram desenvolvidas com o intuito de reduzir sua complexidade, que a principio é proporcional ao quadrado do comprimento do código, que pode chegar à ordem de dezenas de milhares de bits por bloco. Entre as propostas mais satisfatórias está o uso da classe estendida de códigos de repetição e acumulação irregulares (eIRA), que é usada neste projeto e é descrita na Seção II.

Neste artigo, detalhamos a implementação de um sistema prático de codificação e decodificação LDPC, desenvolvido como parte de um sistema de transmissão para televisão digital. Oferecemos uma descrição completa do processo de desenvolvimento, partindo do projeto do código até a implementação em hardware do codificador e do decodificador. Como ponto de partida, nos baseamos na classe de códigos eIRA, parcialmente inspirados pelos códigos LDPC utilizados na proposta para o padrão DVB-S2 [7]. Como é mostrado no restante do artigo, muitas das restrições impostas ao projeto do código estão diretamente relacionadas às limitações do hardware onde implementamos nossa prova de conceito. Por isso, o processo de desenvolvimento seguiu vários ciclos entre a fase de projeto da matriz de paridade, as simulações de desempenho, e o projeto do hardware, até que pudéssemos chegar a um resultado satisfatório, que pudesse ser testado em um ambiente real.

Os conceitos básicos dos códigos LDPC são discutidos em mais detalhes nas Seções II and III. A Seção IV descreve o método utilizado para projetar códigos eIRA estruturados, com desempenho otimizado para as restrições impostas. Os resultados de desempenho dos códigos projetados são mostrados na Seção V. A Seção VI descreve os detalhes da

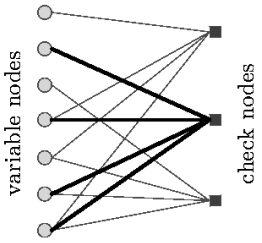


Fig. 1. Grafo de Tanner para o código de Hamming $(7, 4)$.

implementação em hardware do decodificador. Finalmente, a Seção VII apresenta nossas conclusões e comentários sobre o projeto.

II. CÓDIGOS LDPC

Nesta seção descreveremos os códigos LDPC em mais detalhes. Começaremos com uma definição genérica e algumas propriedades destes códigos. Então, discutiremos o processo de codificação e decodificação.

Códigos LDPC (n, k) são códigos de bloco lineares e binários. Desta forma, estes códigos podem ser vistos como um conjunto de vetores binários $\mathbf{c}$ que satisfazem $\mathbf{cH}^T = \mathbf{0}$, onde $\mathbf{H}$ é uma matriz binária $(n-k) \times n$ chamada *matriz de paridade* e as operações de soma e produto são binárias. A característica que define os códigos LDPC é o fato de que sua matriz de paridade é esparsa, ou seja, o número de valores não nulos é muito menor que o número total de valores na matriz $\mathbf{H}$.

Quanto à regularidade, existem dois tipos de códigos LDPC: regulares e irregulares. Em códigos regulares, todas as linhas e colunas de $\mathbf{H}$ têm o mesmo número de uns, embora este número possa ser diferente para ambas. Se o número de uns muda entre colunas e/ou linhas de $\mathbf{H}$, o código é dito irregular.

A decodificação de códigos LDPC assim como algumas com suas características, são descritas em termos de seus grafos de Tanner [8]. Um grafo de Tanner é um grafo bipartido que contém dois tipos de nó: nós de cheque e nós de variável. Qualquer código de bloco linear binário tem um grafo de Tanner correspondente, o qual é construído a partir de sua matriz de paridade $\mathbf{H}$. Cada bit na palavra-código corresponde a um nó de variável, e cada equação de paridade corresponde a um nó de cheque. Um nó de variável é conectado a um nó de cheque no grafo de Tanner se, e somente se, o correspondente bit da palavra-código faz parte da correspondente equação de paridade. A Figura 1 mostra o grafo de Tanner associado ao código de Hamming $(7, 4)$, cuja matriz de paridade é dada por

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

Na Figura 1, as linhas mais grossas correspondem à segunda linha de $\mathbf{H}$. Em [4], é mostrado que o desempenho de códigos LDPC é determinado pela chamada *distribuição de densidades*, que são polinômios que provém uma descrição da estrutura do grafo de Tanner. De fato, define-se o grau de um nó como o número de ramos conectados a ele. Assim, o grau de um nó de variável é o número de equações de paridade do qual o bit correspondente faz parte, enquanto que o grau de um nó de cheque é o número de bits envolvidos na correspondente equação de paridade. Seja λ_i (ρ_i) a fração de ramos conectados a nós de variável (nós de cheque) de grau i. Então,

$$\lambda(x) = \sum_i \lambda_i x^{i-1} \quad (2)$$

é a distribuição de graus dos nós de variável, e

$$\rho(x) = \sum_i \rho_i x^{i-1} \quad (3)$$

é a distribuição de graus dos nós de cheque. Já que estes polinômios são determinantes no desempenho de um código LDPC, vários algoritmos foram propostos para determinar a distribuição ótima. Alguns destes algoritmos serão discutidos na Seção III.

A. Decodificação de códigos LDPC

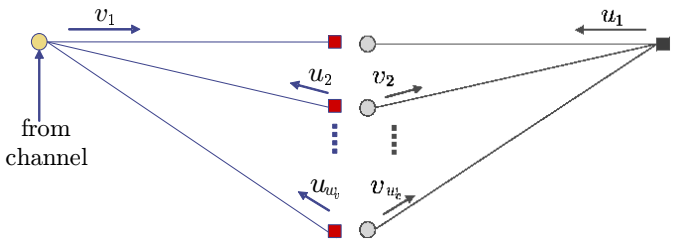
O principal objetivo em qualquer algoritmo de decodificação é determinar uma estimativa da probabilidade máxima a posteriori (MAP) dos bits transmitidos, já que ela minimiza a probabilidade de erro de bit. Para sistemas binários, a estimação MAP de c_i, o i-ésimo bit da palavra-código transmitida, pode ser feita a partir de

$$L_i = \log \frac{Pr(c_i = 0|\mathbf{r})}{Pr(c_i = 1|\mathbf{r})}, \quad (4)$$

onde $\mathbf{r}$ é a seqüência recebida. Assim, a estimação MAP é $c_i = 0$ se $L_i > 0$, e $c_i = 1$ caso contrário. Na literatura, L_i é chamada razão do log das verossimilhanças (LLR, do inglês *log likelihood ratio*). O cômputo exato da LLR é um problema NP completo e, assim, impraticável. Entretanto, baseado no grafo de Tanner, é possível derivar-se algoritmos iterativos capazes de obter uma boa aproximação da LLR.

Os algoritmos iterativos são baseados numa troca de mensagens entre nós de variável e nós de cheque, como ilustrado na Figura 2. Desta forma, a Figura 2(a) mostra o procedimento para um nó de variável de grau w_v. Nesta Figura, nós vemos que o nó de variável combina a informação vinda dos nós de cheque, u_i, $i = 2, \ldots w_v$ com a observação do canal, $-2r_i/\sigma^2$ (considerando uma modulação onde $0 \rightarrow -1$ e $1 \rightarrow 1$), para formar uma mensagem que é enviada novamente para o primeiro nó de cheque, v_1. A Figura 2(b) mostra a operação equivalente em um nó de cheque de grau w_c. A principal diferença entre as figuras está na forma pela qual as mensagens são combinadas, e no fato de que os nós de cheque não tem acesso à informação do canal. A Figura 2 também ressalta o importante conceito de informação extrínseca. Como visto nestas figuras, para ambos os tipos de nó, a mensagem de saída de um dado ramo não é função da mensagem de entrada do mesmo ramo. Isto significa que todas as mensagens correspondem a informação extrínseca, o que impede realimentação positiva de informação.

Talvez o principal algoritmo de decodificação para códigos LDPC seja o algoritmo soma-produto (SPA) [3] [2]. Este algoritmo considera que as mensagens em todos os ramos são independentes, e suas mensagens são os valores "exatos" da informação extrínseca. Esta consideração de independência



(a) Nó de variável para nó de cheque.

(b) Nó de cheque para nó de variável.

Fig. 2. Ilustração do algoritmo *message passing*.

é sempre verdadeira em grafos sem ciclos, para os quais o SPA produz valores exatos para a LLR em um número finito de iterações. Infelizmente, grafos correspondentes a códigos práticos costumam ter ciclos. Mesmo nestes casos, os valores aproximados obtidos pelo SPA são consideravelmente bons, embora ciclos pequenos piorem seu desempenho.

Para um nó de variável j, o SPA computa a mensagem deste nó para um nó de cheque i como

$$v_i = \sum_{j \neq i} u_j. \tag{5}$$

A mensagem proveniente de um nó de cheque para um nó de variável é dada por

$$u_i = 2 \tanh^{-1} \left(\prod_{j \neq i} \tanh \left(\frac{v_j}{2} \right) \right). \tag{6}$$

Claramente, a complexidade do SPA, e de outros algoritmos iterativos baseados em grafos, depende do número de ramos incidente nos nós: poucos ramos significa baixa complexidade. Entretanto, poucos ramos também significa que a matriz de paridade é esparsa. Isto explica porque códigos LDPC podem ser decodificados por algoritmos quase-MAP de baixa complexidade.

B. Codificação de códigos LDPC

Como visto anteriormente, a decodificação de códigos LDPC é relativamente simples. Infelizmente, a codificação de códigos LDPC em geral não é assim tão simples. De fato, considere que queremos obter a matriz geradora sistemática $\mathbf{G}$ correspondente a $\mathbf{H}$. Começamos decompondo $\mathbf{H}$ como

$$\mathbf{H} = [\mathbf{H}_1 \ \mathbf{H}_2], \tag{7}$$

onde $\mathbf{H}_1$ e $\mathbf{H}_2$ são matrizes binárias de dimensões $(n-k) \times k$ e $(n-k) \times (n-k)$. Então, $\mathbf{G}$ é dada por

$$\mathbf{G} = \left[\mathbf{I}_k \ \mathbf{H}_1^T \mathbf{H}_2^{-T} \right]^T. \tag{8}$$

Infelizmente, a inversa de uma matriz esparsa não é esparsa, fazendo com que a codificação por força-bruta empregando $\mathbf{G}$ seja uma tarefa de alta complexidade.

Desta forma, vários códigos LDPC com baixa complexidade de codificação têm sido propostos. Normalmente, a baixa

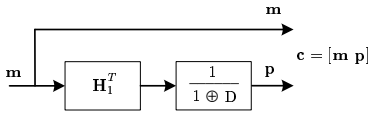


Fig. 3. Codificador para códigos LDPC do tipo eIRA.

complexidade é obtida impondo alguma estrutura à matriz de paridade. Em nosso trabalho, usamos o códigos *extended irregular repeat-accumulate* (eIRA) de [9]. Para estes códigos, a sub-matriz $\mathbf{H}_1$ ainda é aleatoriamente gerada de acordo com uma distribuição de graus pré-determinada. Já a matriz $\mathbf{H}_2$, por outro lado, é dada por

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 1 \end{bmatrix}. \tag{9}$$

Em outras palavras, a diagonal principal e a diagonal abaixo dela contém uns e os elementos restantes são zeros. Empregando esta definição, a multiplicação de um vetor por $\mathbf{H}_2^{-T}$ pode ser implementada com um acumulador simples. O codificador resultante para um código LDPC do tipo eIRA é mostrado na Figura 3.

III. Capacidade de códigos LDPC

Para muitos canais e tipos de decodificadores iterativos, os códigos LDPC apresentam um limiar de operação, ou seja, a medida que o tamanho do bloco tende ao infinito, uma probabilidade de erro de bit arbitrariamente pequena pode ser obtida se o nível de ruído for menor que um certo limiar. Por outro lado, para um nível de ruído acima deste limiar a probabilidade de erro de bit é maior que uma constante não nula. Gallager foi o primeiro a observar este fenômeno para canais simétricos binários (BSC, do inglês *Binary Symmetric Channel*) quando introduziu códigos LDPC regulares [1] [2]. Luby *et al.* generalizaram esta idéia para códigos LDPC irregulares de contrução aleatória [10]. Mais recentemente em [4], Richardson e Urbanke generalizaram estas observações para um grande número de canais de entrada binária, incluindo o AWGN, e vários algoritmos de decodificação, como o SPA. Nesta seção, descreveremos com mais detalhes alguns métodos para se obter este limiar de convergência de códigos LDPC que, como veremos no decorrer deste trabalho, é fundamental para o projeto destes códigos.

O primeiro método a ser descrito é a *evolução de densidades* [4], o qual calcula iterativamente a função densidade de probabilidade (fdp) das mensagens no SPA. Chung *et al.* melhoraram o esforço computacional ao publicarem a versão discreta do algoritmo de evolução de densidades [11] [12], o qual permitiu se chegar a um limiar 0,0045 dB acima do limite de Shannon para canal AWGN e a um código com comprimento de bloco 10^7 e desempenho 0,04 dB acima do limite de Shannon a uma probabilidade de erro de bit (BER, do inglês *Bit Error Rate*) de 10^{-6}.

Ainda assim, o cálculo do limiar e a otimização das distribuições de graus empregando evolução de densidades são tarefas computacionalmente intensas, o que pode inviabilizar seu uso para canais práticos. No entanto, existem alternativas computacionalmente menos complexas e que, dependendo do caso, podem ser suficientemente precisas. Um destes métodos é baseado na aproximação das fdp das mensagens por Gaussianas (no caso de códigos regulares) ou misturas de Gaussianas (para códigos irregulares). Nesta *aproximação gaussiana*, sem sacrificar exageradamente a precisão, a média das densidades Gaussianas (uma quantidade uni-dimensional) pode representar a densidade de probabilidade das mensagens (uma quantidade cuja dimensão é infinita).

Embora a fdp das mensagens enviadas por nós de variável possa ser adequadamente aproximada por distribuições Gaussianas, o mesmo não acontece com a fdp das mensagens enviadas por nós de cheque. Assim, de forma a ter um método computacionalmente menos complexo que o evolução de densidades e mais preciso que com aproximação Gaussiana, Ardakani & Kschischang propuseram o método de *aproximação semi-Gaussiana* [13]. Neste método as fdp das mensagens enviadas por nós de variável continuam sendo aproximadas por Gaussianas ou misturas de Gaussianas. Já as fdp das mensagens enviadas por nós de cheque são obtidas empregando-se apenas uma iteração do algoritmo de evolução de densidades. Curvas de transferência de informação extrínseca (EXIT, do inglês *Extrinsic Information Transfer*) [14] são empregadas então para verificar o maior nível de ruído para o qual o algoritmo iterativo ainda converge, e assim determinar o limiar.

Embora existam outros métodos para a determinação do limiar, descreveremos em mais detalhes os três métodos discutidos anteriormente.

A. Evolução de Densidades Discretas

Considere novamente o valor de LLR v como sendo a mensagem de um nó de variável de grau d_v para um nó de cheque. No decodificador soma-produto, v é igual a soma de todas as LLRs, ou seja,

$$v = \sum_{i=0}^{d_v-1} u_i, \tag{10}$$

onde u_i, $i = 1, \ldots, d_v - 1$ são as LLRs que chegam dos vizinhos dos nós de variável, e u_0 é a mensagem inicial do nó de variável proveniente do canal.

Já mensagem de saída de um nó de cheque pode ser obtida através da relação

$$\tanh\frac{u}{2} = \prod_{j=1}^{d_c-1} \tanh\frac{v}{2}, \tag{11}$$

onde v, $j = 1, \ldots, d_c - 1$ são as LLRs provenientes dos $d_c - 1$ nós de cheque vizinhos, e u é a mensagem de saída do nó de cheque em questão.

De agora em diante, assumimos um código irregular com distribuições de grau, $\lambda(x)$ e $\rho(x)$.

Agora, seja $\mathcal{Q}(w)$ a mensagem w quantizada, isto é,

$$\mathcal{Q}(w) = \begin{cases} \left\lfloor \frac{w}{\Delta} + \frac{1}{2} \right\rfloor \Delta, & \text{se } w \geq \frac{\Delta}{2} \\ \left\lceil \frac{w}{\Delta} - \frac{1}{2} \right\rceil \Delta, & \text{se } w \leq -\frac{\Delta}{2} \\ 0, & \text{c.c.} \end{cases} \tag{12}$$

onde $\mathcal{Q}(w)$ é o operador quantização; Δ é o intervalo de quantização; $\lfloor x \rfloor$ é o maior inteiro menor ou igual a x; e $\lceil x \rceil$ é o menor inteiro maior ou igual a x.

O decodicador soma-produto quantizado é definido como aquele no qual todas as mensagens de entrada e saídas são quantizadas de acordo com (12). Desta forma, no nó de variável temos que $\overline{v} = \sum_{i=0}^{d_v-1} \overline{u}_i$, onde $\overline{v} = \mathcal{Q}(v)$ e $\overline{u}_i = \mathcal{Q}(u_i)$ para $i = 0, \ldots, d_v - 1$. Denomina-se a função massa de probabilidade (fmp) da mensagem $\overline{w}$ por $p_w[k] = \Pr(\overline{w} = k\Delta)$ para $k \in Z$. Assim, p_v relaciona-se com sua fmp de entrada por

$$p_v = \bigotimes_{i=0}^{d_v-1} p_{u_i} \tag{13}$$

onde p_v é a fmp de $\overline{v}$; p_{u_i} é a fmp de $\overline{u}_i$; e $\bigotimes$ é a operação de convolução discreta. Desde que os $\overline{u}_i$ sejam identicamente distribuidos para $1 \leq i < d_v$, a equação anterior pode ser reescrita como

$$p_v = p_{u_0} * \left(\bigotimes^{d_v-1} p_u \right) \tag{14}$$

onde $p_u = p_{u_i}$ e * é a convolução discreta. Isto pode ser calculado eficientemente através da FFT.

No nó de cheque, define-se o operador R como

$$\mathcal{R}(a, b) = \mathcal{Q}\left(2\tanh^{-1}\left(\tanh\frac{a}{2}\tanh\frac{b}{2} \right) \right) \tag{15}$$

onde a e b são mensagens quantizadas. Note que esta operação pode ser feita empregando uma tabela de valores pré-computados, o que faz a evolução de densidades discretas ser computacionalmente mais eficiente. Usando este operador, calcula-se a mensagem $\overline{u}$ de (11) como:

$$\overline{u} = \mathcal{R}(\overline{v}_1, \mathcal{R}(\overline{v}_2, \ldots, \mathcal{R}(\overline{v}_{d_c-2}, \overline{v}_{d_c-1}))) \tag{16}$$

onde assumimos que o processo de decodificação soma-produto discreta nos nós de cheque é feita par-a-par.

Considere $c = \mathcal{R}(a, b)$. A fmp p_c de c é dada por

$$p_c[k] = \sum_{(i,j):k\Delta=\mathcal{R}(i\Delta, j\Delta)} p_a[i] p_b[j]. \tag{17}$$

Abusando da notação, escrevemos isto como $p_c = \mathcal{R}(p_a, p_b)$.

Desde que os p_{v_i} sejam todos iguais, definimos $p_v = p_{v_i}$, para $1 \leq i < d_c$, e escrevemos $p_u = \mathcal{R}(p_v, \mathcal{R}(p_v, \ldots, \mathcal{R}(p_v, p_v), \ldots))$ como $p_u = \mathcal{R}^{d_c-1} p_v$. Definindo $\lambda(p) = \sum_{i=2}^{d_l} \lambda_i \bigotimes^{i-1} p$ e $\rho(p) = \sum_{j=2}^{d_r} \rho_j \mathcal{R}^{j-1} p$ para qualquer fmp p podemos escrever a evolução de densidades discreta como:

Teorema: A evolução de densidades discreta é descrita como

$$p_u^{(l+1)} = \rho(p_{u_0} * \lambda(p_u^{(l)})) \tag{18}$$

onde a fmp inicial $p_u^{(0)}$ tem toda a massa de probabilidade concentrada em zero e l é o número da iteração.

Para executar este algoritmo assume-se inicialmente que a palavra-código toda nula foi enviada. Assim, fixam-se os parâmetros do canal e executa-se o algoritmo iterativamente até que a densidade u tenda a um ponto de massa no infinito (equivalente a uma probabilidade de erro tender a zero) ou o algoritmo convirja para uma densidade tendo uma probabilidade de erro finita.

Definição 1: O limiar de convergência para um código LDPC de distribuição de graus $\lambda(x)$ e $\rho(x)$ é definido como o nível máximo de ruído no qual a probabilidade de erro ainda tende a zero a medida que o número de iterações tende a infinito.

B. Aproximação Gaussiana

Considere um código LDPC irregular com distribuições $\lambda(x)$ e $\rho(x)$. Aproxima-se as fdp de u, u_i, v e v por Gaussianas ou misturas de Gaussianas. Como uma Gaussiana pode ser completamente representada por sua média e variância, podemos manter apenas média e variância durante as iterações. Existe uma condição importante, chamada *condição de simetria* [15], que é preservada durante a evolução de densidades para todas as mensagens, o qual pode ser expressa como $f(x) = f(-x)e^x$, onde $f(x)$ é a fdp de uma mensagem. Aplicando esta condição à aproxição Gaussiana a cada iteração, podemos melhorar a precisão da aproximação. Para uma Gaussiana de média m e variância σ^2, esta condição se reduz a $\sigma^2 = 2m$ e, assim, podemos manter apenas a média.

Assim, a partir da equação (11), a média $m_{v,i}^{(l)}$ da mensagem de saída do nó de variável de grau i na l-ésima iteração é dada por

$$m_{v,i}^{(l)} = m_{u_0} + (i-1)m_u^{(l-1)} \tag{19}$$

onde m_{u_0} é a média de u_0 e $m_u^{(l-1)}$ é a média de u na iteração $(l-1)$. A variância da densidade de saída é dada por $2m_{v,i}^{(l)}$. Além disso, na l-ésima iteração, uma mensagem de entrada em nó de cheque v terá a seguinte mistura de Gaussianas $f_v^{(l)}$ como fdp:

$$f_v^{(l)} = \sum_{i=2}^{d_l} \lambda_i \mathcal{N}(m_{v,i}^{(l)}, 2m_{v,i}^{(l)}). \tag{20}$$

Já em relação aos nós de cheque, com alguma manipulação adicional (veja [11] [12]) chegamos a

$$m_u^l = \sum_{j=2}^{dr} \rho_j \phi^{-1} \left(1 - \left[1 - \sum_{i=2}^{d_l} \lambda_i \phi\left(m_{u_0} + (i-1)m_u^{l-1}\right)\right]^{j-1}\right) \tag{21}$$

onde

$$\phi(x) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_{\mathbb{R}} \tanh\frac{u}{2} e^{-\frac{(u-x)^2}{4x}} du & \text{se } x > 0 \\ 1, & \text{se } x = 0. \end{cases} \tag{22}$$

A mensagem inicial proveniente de qualquer nó de cheque é nula, logo $m_u^{(0)} = 0$. Assim, podemos chegar a seguinte definição:

Definição 2: O limiar de um código LDPC irregular com distribuições de grau $\lambda(x)$ e $\rho(x)$ é o maior nível de ruído (equivalente ao maior valor absoluto de m_{u_0}) em que a média dos valores de LLR das mensagens enviadas de nós de cheque para nós de variável u ainda convirja para ∞ a medida que o número de iterações $l \to \infty$.

C. Aproximação Semi-Gaussiana

No método de aproximação semi-Gaussiana, assim como no método de aproximação Gaussiana convencional, as fdp das mensagens enviadas de nós de variável para nós de cheque, v, é aproximada por uma somatória de gaussianas, ou seja, as equações (19) e (20) ainda são válidas. No entanto, as fdp das mensagens enviadas por nós de cheque para nós de variável u não são mais consideradas como misturas de gaussianas. Agora, a fdp de u é calculada para uma iteração da treliça da Figura 4. Isto pode ser feito através de simulações de Monte Carlo ou através de uma iteração do método de evolução de densidades. Um decodificador LDPC

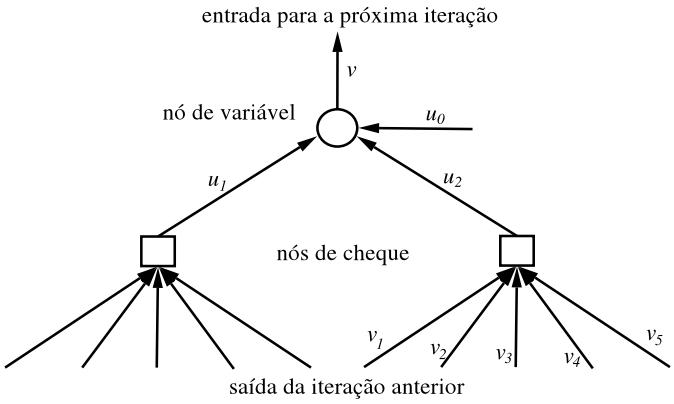


Fig. 4. Treliça da aproximação semi-Gaussiana.

pode ser visto como um bloco que, a cada iteração, usa duas fontes de informação sobre a palavra-código transmitida: a informação proveniente do canal (informação intrínseca I_0) e a informação vinda da iteração anterior (informação extrínseca I_{in}). A partir destas duas fontes de informação, o algoritmo de decodificação tenta produzir uma informação mais confiável sobre a palavra-código transmitida, gerando assim uma nova informação extrínseca I_{out} para a próxima iteração.

Este conceito de evolução da confiabilidade da informação a cada iteração é empregado para a confecção de EXIT charts. Embora empregue-se geralmente a entropia como esta grandeza de informação, Ardakani & Kschischang empregaram, sem perda de desempenho, probabilidade de erro de mensagem, como pode ser visto na EXIT chart na Figura 5. Assim, para cada iteração do algoritmo de decodificação temos que I_{in}, I_0 e I_{out} são associadas, respectivamente, as probabilidades de erro de mensagem p_{in}, p_0 e p_{out}. Assim, a cada iteração, temos $p_{out} = g(p_{in}, p_0)$. A EXIT chart pode ser representada por um gráfico com a função g e sua inversa g^{-1}, como na Figura 5. Cada seta representa uma iteração do decodificador. Muitas das características do decodificador, como número de iterações necessárias para alcançar-se uma

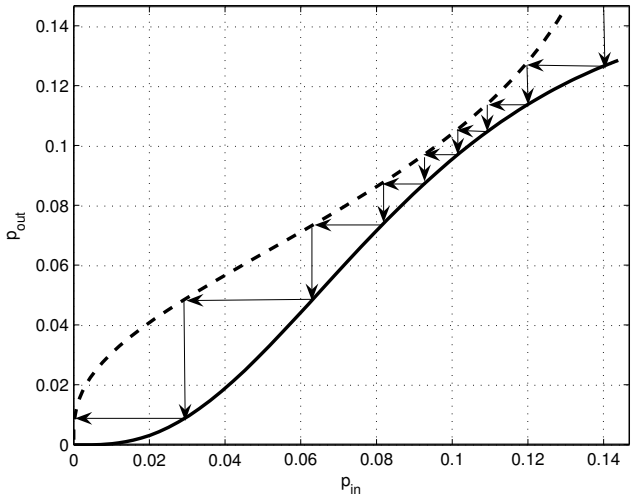


Fig. 5. EXIT Chart.

dada p_{out} e o limiar de convergência podem ser obtidas através de EXIT charts. Note que se o túnel de decodificação (espaço entre g e g^{-1}) estiver fechado, o algoritmo de decodificação converge para uma probabilidade de erro alta. Assim, podemos ter uma outra definição para o limiar de convergência de um código LDPC irregular:

Definição 3: O limiar de convergência para um código LDPC de distribuição de graus $\lambda(x)$ e $\rho(x)$ pode ser definido como o maior nível de ruído no qual o EXIT chart ainda encontra-se aberto.

Para manter a EXIT chart aberta a taxa de erro de mensagem na entrada do decodificador p_{in} deve ser maior que a taxa de erro de mensagem na saída, dada por

$$p_{out} = \sum_{i=2}^{d_l} \lambda_i g_i(x), \tag{23}$$

onde $g_i(x)$ é uma EXIT chart elementar (EXIT chart para nó de variável de grau i).

IV. Metodologia de Projeto para Códigos eIRA Estruturados

O projeto de um código LDPC do tipo eIRA estruturado [9] [6] consiste em duas etapas: a otimização das distribuições de grau $\lambda(x)$ e $\rho(x)$, e a construção da parte $\mathbf{H}_1$ da matriz de paridade. Na primeira etapa, o objetivo é maximizar a capacidade do código. Já na construção de $\mathbf{H}_1$, distribui-se os "1's" respeitando-se as distribuições de grau, minimizando o número de ciclos de pequeno grau no grafo de Tanner e facilitando a representação e armazenamento da matriz de paridade. Este processo leva a códigos LDPC com bom desempenho e com codificadores e decodificadores com complexidade computacional reduzida.

A. Otimização de $\lambda(x)$ e $\rho(x)$

A optimização conjunta de $\lambda(x)$ e $\rho(x)$ é uma tarefa de alta complexidade computacional. Em [9], por exemplo, o autor emprega o algoritmo de evolução de densidades com otimização via evolução diferencial. Embora complexo, este método permite obter códigos para qualquer taxa de codificação.

Uma forma de reduzir a complexidade é utilizar uma distribuição concentrada para $\rho(x)$ [12]. Dessa forma temos

$$\rho(x) = \rho x^{d_r - 1} + (1 - \rho) x^{d_r}, \tag{24}$$

onde seria necessário apenas estimar o valor escalar de ρ. Experimentos realizados em [11] [12] mostraram que não houve degradação de desempenho com o emprego desta forma concentrada de $\rho(x)$.

Outra alternativa para a redução da complexidade da otimização é a substituição do método de evolução de densidade por métodos como a aproximação Gaussiana e semi-Gaussiana. Nestes dois métodos o problema se resume a uma otimização via programação linear. Dentre os métodos estudados, o com aproximação semi-Gaussiana é aquele que fornece a melhor relação entre desempenho e complexidade computacional.

Richardson *et al.* [15] mostraram que a taxa de codificação pode ser expressa em termos de $\lambda(x)$ e $\rho(x)$ da seguinte forma

$$r(\lambda, \rho) = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx} = 1 - \frac{\sum_{i=2}^{d_r} \frac{\rho_i}{i}}{\sum_{i=1}^{d_l} \frac{\lambda_i}{i}}. \tag{25}$$

Eles também mostraram que o número de nós de variável de grau i, $N_v(i)$, pode ser expresso por

$$N_v(i) = \frac{n\lambda_i}{i \int_0^1 \lambda(x)dx} = \frac{n\lambda_i}{i \sum_{j=1}^{d_l} \frac{\lambda_j}{j}} \tag{26}$$

Assim, empregando a EXIT chart e a equação (26), e levando em consideração que para códigos do tipo eIRA $N_v(1) = 1$ e $N_v(2) = (n - k - 1)$, o problema de otimização pode ser formulado pelo seguinte programa linear

$$\begin{aligned} &\text{maximizar} \quad \textstyle\sum_{i=2}^{d_l} \frac{\lambda_i}{i} \\ &\text{sujeito a} \quad \begin{cases} \lambda_i \geq 0 \\ \sum_{i=1}^{d_l} \lambda_i = 1 \\ \lambda_1 = \frac{1}{(n-1)} \sum_{i=2}^{d_l} \frac{\lambda_i}{i} \\ \lambda_2 = \frac{2(n-k-1)}{(k+1)} \sum_{i \neq 2}^{d_l} \frac{\lambda_i}{i} \\ \sum_{i=2}^{d_l} \lambda_i g_i(p_{in}) < p_{in} \quad \forall p_{in} \in (0, p_0] \end{cases} \end{aligned} \tag{27}$$

As entradas deste programa linear são o modelo do canal, o nível de ruído (limiar), o comprimento da palavra-código n, d_l e $\rho(x) = x^{d_r - 1}$. O valor inicial para o nível de ruído pode ser lijeiramente superior ao limite de Shannon para canal AWGN. Caso a taxa de codificação resultante da maximização (27) seja diferente da especificada para o sistema (r), o nível de ruído deve ser incrementado e o processo de otimização repetido. O nível de ruído para o qual a maximização (27) resulta em uma taxa de codificação igual a r é considerado como o limiar de convergência do código.

Este método de otimização das distribuições de graus é uma das contribuições deste trabalho, já que é o primeiro trabalho onde é empregado aproximação semi-Gaussiana no projeto de códigos LDPC do tipo eIRA.

B. Construção da Matriz de Paridade

A matriz $\mathbf{H}_1$ poderia ser contruída aleatorimante respeitando-se as distribuições de grau $\lambda(x)$ e $\rho(x)$ para nós de grau maior que 2. Ou então, diretamente através do algoritmo *progressive edge-growth* (PEG) [16], de forma a minimizar a ocorrência de pequenos ciclos no grafo de Tanner. No entanto, a memória necessária para armazerar-se $\mathbf{H}_1$ seria significativamente grande. Uma forma de diminuir a quantidade de memória para representar $\mathbf{H}_1$ é acrescentar a ela alguma regularidade.

O primeiro passo para construir a matriz $\mathbf{H}_1$ de dimensão $(n-k)\times k$ é criar uma matriz $\mathbf{A}$ de dimensão $a \times b$, M vezes menor que a dimensão de $\mathbf{H}_1$. M é o número de ramos que podem ser processados em paralelo no decodificador, e deve ser projetado levando-se em consideração o compromisso entre a taxa efetiva de dados e a área em hardware ocupada pelo decodificador. A matriz $\mathbf{A}$ pode ser construída com o algoritmo PEG.

A seguir, constrói-se uma matriz $\mathbf{B}$ com as mesmas dimensões de $\mathbf{H}_1$ substituindo cada "0" em $\mathbf{A}$ por matrizes nulas quadradas de ordem M e cada "1" por $\mathbf{J}^{ab}$, onde $\mathbf{J}$ é uma permutação da matriz identidade de ordem M, como na equação (28) para $M = 5$. Matrizes $\mathbf{J}$ e suas permutações são empregadas em códigos LDPC do tipo *Array Codes* [17].

$$\mathbf{J} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{28}$$

Assim, a matriz $\mathbf{H}_1$ pode ser obtida através da seguinte permutação de linhas da matriz $\mathbf{B}$

$$\mathbf{H}_1 = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_{M+1} \\ \vdots \\ \mathbf{b}_{(a-1)M+1} \\ \mathbf{b}_2 \\ \mathbf{b}_{M+2} \\ \mathbf{b}_{(a-1)M+2} \\ \vdots \\ \mathbf{b}_{n-k} \end{bmatrix} \tag{29}$$

onde $\mathbf{b}_i$ é a i-ésima linha da matriz $\mathbf{B}$.

Este processo reduz o número de pequenos ciclos no grafo de Tanner, embora não os elimine por completo. Além disso, a regularidade na sub-matriz $\mathbf{H}_1$ permite uma representação compacta da matriz de paridade $\mathbf{H}$ e reduz a complexidade do codec.

Como um sistema de televisão digital requer uma recepção quase livre de erros (BER=10^{-11}), nosso objetivo para o projeto de códigos LDPC é obter o código que fornece o melhor desempenho em termos de E_b/N_0 para uma BER=10^{-5}. No sistema como um todo, um código externo como o Reed-Solomon eliminaria por completo um possível *error-floor* e levaria o sistema a uma situação quase livre de erros na saída do receptor.

V. Resultados de Simulação

O receptor de Televisão Digital deve ser robusto a vários tipos de canal. Para recepção fixa em uma área rural o canal pode ser puramente AWGN. Por outro lado, para um receptor móvel o canal pode ser Rayleigh. Assim, o esquema de codificação de canal deve ser capaz de se adaptar a todas estas condições. Como visto em [18], códigos LDPC irregulares projetados para canal AWGN são também robustos para outros tipos de canal. Assim, o canal AWGN foi escolhido para o projeto dos códigos.

Para o Sistema de Televisão Digital proposto foram escolhidas cinco taxas de codificação, $1/2$, $2/3$, $3/4$, $5/6$ and $7/8$, e palavras-código de 9792 bits. Empregando evolução de densidades com aproximação semi-Gaussiana como descrito anteriormente resultou nas distribuições de graus e limiares de convergência da Tabela I. A capacidade de códigos LDPC com estas distribuições de graus ficou a uma distância de $0,13$ dB a $0,37$ dB do limite de Shannon para canais AWGN binários.

TABELA I
DISTRIBUIÇÕES DE GRAU

	$1/2$	$2/3$	$3/4$	$5/6$	$7/8$
$\rho(x)$	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		
limiar (dB)	0,428	1,236	1,804	2,582	3,211
gap (dB)	0,24	0,13	0,18	0,27	0,37

Simulações de Monte Carlo foram feitas com os códigos gerados a partir das distribuições de grau da Tabela I. A Figura 6 mostra o desempenho em termos de taxa de erro de bit. Os códigos obtidos ficam de $0,7$ dB a 1 dB do limite de Shannon para canais AWGN binários (considerando uma BER de 10^{-5}). Como pode ser visto na Figura 7, o código LDPC (9792,4896) projetado pela nossa metodologia é 3 dB melhor (a BER=10^{-5}) que o código convolucional empregado nos sistemas DVB-T e ISDB-T. Para estas simulações considerou-se modulação BPSK, canal AWGN e $M = 51$ ramos em paralelo no decodificador. A BER foi computada depois de 50 palavras-código erradas, considerando-se um máximo de 50 iterações para o decodificador soma-produto.

VI. Implementação em Hardware

Dentre as diferentes taxas de codificação consideradas para o sistema de televisão digital proposto, para a implementação em hardware optou-se pela taxa $3/4$ com palavras-código de 9792 bits. Assim, cada palavra-código consiste em 7344 bits de informação e 2448 bits de paridade.

O codificador e o decodificador LDPC foram implementados em *Field Programmable Gate Arrays* (FPGA). O codificador foi desenvolvido em *Very High Speed Integrated Circuit*

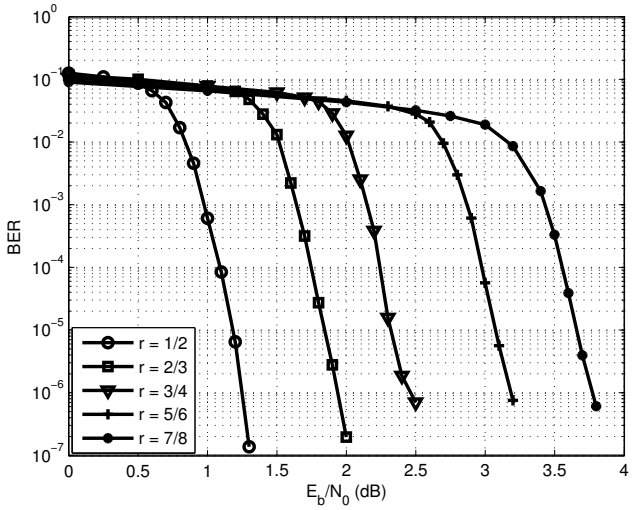


Fig. 6. Desempenho dos códigos LDPC para o STVD.

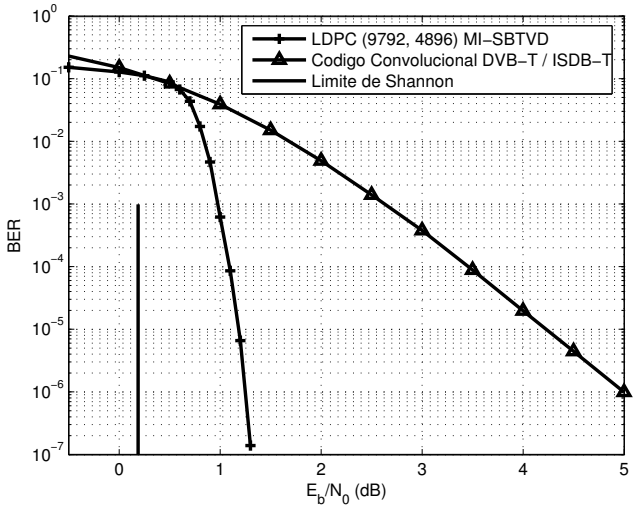


Fig. 7. Comparação entre o código LDPC (9792,4896) e o código convolucional dos sistemas DVB-T e ISDB-T.

Hardware Description Language (VHDL – IEEE 1164) utilizando a ferramenta de desevolvimento Quartus, da Altera. O decodifcador LDPC foi implementado em VHDL e System Generator, da Xilinx.

No decodificador, as operações em nós de cheque podem ser feitas de forma serial com uma estrutura em treliça, ou em paralelo na forma de uma estrutura em ramos. A estrutura em treliça foi adotada devido ao fato de consumir significativamente menos recursos da FPGA que a implementação com estrutura em ramos.

Talvez a questão mais importante na implementação em hardware seja o compromisso entre a área ocupada no dispositivo e a velocidade de processamento. O taxa efetiva mínima requerida pelo sistema proposto é de 19,33Mbps. Por outro lado, existe a limitação física do dispositivo FPGA em área de silício, que limita o grau de paralelismo da implementação.

Os detalhes da implementação do codificador e decodificador LDPC serão discutidos a seguir.

A. Codificador LDPC

Conforme visto na Figura 3, o codificador LDPC multiplica o bloco da mensagem por $\mathbf{H}_1^T$ e envia o resultado para um acumulador. O acumulador é obtido através da simples operação OU-Exclusivo (XOR), sendo assim facilmente implementado. Por outro lado, um grande problema a ser considerado é como armazenar a matriz $\mathbf{H}_1$, de dimensões 7344×2448. Felizmente, conforme discutido na Subseção IV-B, é necessário armazenar apenas os índices dos nós de variável conectados aos 144 nós de cheque indexados por múltiplos de m, que, no nosso caso, é igual a 51. Os índices dos nós de variável são armazenados em uma estrutura que possui 112 linhas de 3 elementos e 32 linhas de 12 elementos, aqui denominada de estrutura $\mathbf{T}$. Uma vez que cada elemento desta estrutura $\mathbf{T}$ possui 12 bits, $\mathbf{H}_1$ pode ser representada usando-se apenas 8.6 Kbits, uma redução de aproximadamente 2000 vezes, se comparado com a representação original completa.

Assim, a estrutura $\mathbf{T}$ foi armazenada em 15 blocos de memória independentes (BRAMs), sendo que 3 armazenam os elementos das primeiras 112 linhas e 12 armazenam os elementos das últimas 32. Desta forma, torna-se possível acessar simultaneamente todos os dados necessários para cada iteração, permitindo a saída serial dos bits de paridade imediatamente após a chegada do último bit da mensagem.

B. Decodificador LDPC

Para uma implementação em hardware do decodificador, as expressões (5) e (6) devem ser reescritas de modo mais eficiente. A informação total disponível no nó de variável v é dada por $v = \sum_j u_j$. Assim, as mensagens provenientes de nós de variável, dadas pela equação (5), podem ser computadas de forma eficiente por

$$v_i = v - u_i, \tag{30}$$

o que requer apenas $2w_v + 1$ adições. Para as operações nos nós de cheque em (6) é possível evitar o cômputo direto da função transcendental $\tanh$. Assumindo um nó de cheque com grau dois e sendo U e V as mensagens que chegam a este nó, (6) pode ser reescrita como

$$\begin{aligned} L(U \oplus V) &= 2\tanh^{-1}\left(\tanh\frac{U}{2}\tanh\frac{V}{2}\right) \\ &= \min\{|U|,|V|\} + z(U,V), \end{aligned} \tag{31}$$

onde $z(U,V) = \log\left(\frac{1-\exp^{-|U+V|}}{1-\exp^{-|U-V|}}\right)$ é chamado de *fator de correção*.

Aproximando o fator de correção por $z(U,V) \equiv 0$ obtemos uma operação com reduzida complexidade computacional, dando origem ao algoritmo *min-sum*. Da mesma forma, uma expressão com um melhor compromisso entre custo computacional, precisão e quantização é a *aproximação constante* [19], dada por

$$z(x,y) = \begin{cases} 0.5 & \text{if } |x| \le 2, |y| > 2|x| \\ -0.5 & \text{if } |y| \le 2, |x| > 2|y| \\ 0 & \text{caso contrário.} \end{cases} \tag{32}$$

Para a operação nos nós de cheque como um todo, diversas topologias podem ser empregadas [20]. A topologia mais complexa é a *força bruta*, a qual computa o valor da equação (6). Já a *implementação paralela* permite a maior taxa efetiva de dados ao custo de uma maior ocupação em hardware, além de apresentar grande sensibilidade aos efeitos de quantização. Por outro lado, a *implementação serial* é muito robusta em termos de efeitos de quantização e requer um menos recursos de hardware quando sintetizada em uma FPGA.

Para a implementação serial, (6) é reescrita como

$$u_1 = L(b_2) \tag{33}$$

$$u_i = L(f_{i-1} \oplus b_{i+1}),\ i = 2, \ldots, w_c - 1 \tag{34}$$

$$u_{w_c} = L(f_{w_c-1}) \tag{35}$$

onde $f_1 = c_1$ e $f_2 = f_1 \oplus c_2$, ..., $f_{w_v} = f_{w_v-1} \oplus c_{w_v}$, $b_{w_v} = c_{w_v}$, $b_{w_v-1} = b_{w_v} \oplus c_{w_v-1}$, ..., $b_1 = b_2 \oplus c_1$ são dois conjuntos de variáveis aleatórias auxiliares que representam o fluxo de mensagens diretas e reversas entre nós de cheque. O cômputo da mensagem u_i pela equação (34) requer a obtenção dos parâmetros $L(f_1)$, ..., $L(f_{w_v})$, $L(b_1)$, ..., $L(b_{w_v})$. Estes podem ser obtidos recursivamente substituindo-se os valores conhecidos por $L(c_1)$, ..., $L(c_{w_v})$ na equação (31). A complexidade computacional resultante é de $3 \cdot (w_v - 2)$ operações.

A arquitetura empregada em nosso projeto é similar a empregada em [21]. Como visto na Figura 8, esta implementação tem sete subcomponentes principais: buffer de entrada (inBuffer), buffer de saída (outBuffer), processadores dos nós de variável (vnp), processadores dos nós de cheque (cnp), entrelaçador e desentrelaçador, e uma unidade de controle (CR). As últimas cinco unidades formam o kernel do decodificador.

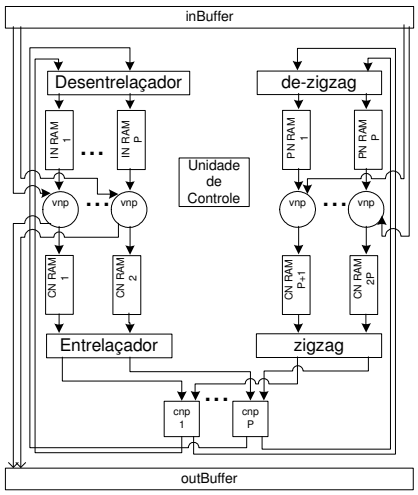


Fig. 8. Arquitetura do decodificador LDPC

O inBuffer é responsável por manter todos os blocos de dados disponíveis e acessíveis ao kernel. De forma a minimizar atrasos no kernel, ele consiste em dois blocos de buffer circulares de escrita serial e leitura paralela. vnp e cnp computam, respectivamente, (30) e (31). Ambos entrelaçador e desentrelaçador executam endereçamento aleatório em linhas e deslocamento em colunas. Um bloco completo do decodificador LDPC é enviado em paralelo entre vnp e cnp através do entrelaçador e são armazenados e lidos dos dois blocos de memórias internas com os valores das mensagens de LLR.

C. Resultados

O decodificador LDPC foi implementado em um dispositivo Altera Stratix II 2S60 e desenvolvido em VHDL. Com a estrutura da Figura 3 e a regularidade na matriz $\mathbf{H}_1$, o codificador teve um baixo custo computacional. De fato, somente 4% da área do dispositivo e 5% dos blocos internos de memória RAM (BRAMs) foram ocupados.

Para o decodificador empregou-se aproximação constante para o fator de correção e uma implementação serial. O decodificador foi implementado em um disposistivo Xilinx Virtex II XC2V3000 empregando VHDL e System Generator, também da Xilinx. A frequência do relógio para o dispositivo FPGA foi de 97 MHz. Em recursos da FPGA formam empregados 186 BRAMs (96%) e 9478 slices (61%). As mensagens foram armazenadas em ponto fixo com 5 bits (1 para o sinal, 3 para a parte inteira e 1 para a parte fracionária) e, de forma a respeitar a taxa efetiva de dados mínima, o número máximo de iterações foi definido como 13.

VII. Conclusões

Neste artigo descrevemos a metodologia de projeto de um código LDPC e sua implementação em hardware. Vimos que o código LDPC do tipo eIRA empregado neste trabalho permite a implementação de um codificador de baixa complexidade computacional, empregando menos de 5% de uma FPGA do tipo Altera Stratix II. A implementação do decodificador é mais complexa, ocupando quase que a totalidade de uma FPGA do tipo Xilinx Virtex IV. Os resultados de simulação mostraram que, para uma BER de 10^{-5}, o código proposto opera em torno de 1 dB do limite de Shannon para canais AWGN, e fornece uma vantagem de 3 dB em relação ao código convolucional empregado em outros sistemas de televisão digital.

Agradecimentos

Este trabalho foi financiado pela Financiadora de Estudos e Projetos (FINEP), Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) e pela Linear Equipamentos Eletrônicos S/A.

Referências

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

[2] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[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 e R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Informormation Theory*, vol. 47, pp. 599-618, Fev. 2001.

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

[6] Y. Zhang, W. Ryan e Y. Li, "Structured eIRA codes with low floors," *Proceedings of the International Symposium on Information Theory - ISIT2005*, pp. 174–178, Set. 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, Set. 1981.
[9] M. Yang, W. Ryan e 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.
[10] M. Luby, M. Mitzenmacher, A. Shokrollahi e D. Spielman, "Analysis of low density codes and improved designs using irregular graphs," *in Proc. 30th Annual ACM Symp. Theory on Computing*, pp. 249–258, 1998.
[11] S.-Y. Chung, G. D. Forney Jr., T. J. Richardson e R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon Limit," *IEEE Communications Letters*, vol. 5, no. 2, pp. 58-60, Fev. 2001.
[12] S.-Y. Chung, T. J. Richardson e 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, Fev. 2001.
[13] M. Ardakani e 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, Dez. 2004.
[14] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Communications*, vol. 49, pp. 1727-1737, Out. 2001.
[15] T. J. Richardson, A. Shokrollahi e R. Urbank, "Design of capacity-approaching low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619-637, Fev. 2001.
[16] X. Y. Hu, E. Eleftheriou e D. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Communications*, vol. 52, pp. 386–398, Fev. 2005.
[17] 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, Set. 2000.
[18] J. Hou, P. Siegel e 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, Maio 2001.
[19] M. Shen, H. Niu, H. Liu e 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.
[20] X.-Y. Hu, E. Eleftheriou, D. Arnold e A. Dholakia, "Efficient Implementations of Sum-Product Algorithm for Decoding LDPC Codes." *IEEE GLOBECOM2001*, v.2, pp. 1036–1036, 2001.
[21] F. Kienle, T. Brack e 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.

Tarciano Facco Pegoraro é doutorando da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas (UNICAMP) e engenheiro de software da Motorola Industrial Ltda, em Jaguariúna, SP. Recebeu os títulos de Engenheiro Eletricista pela Universidade Federal de Santa Maria (UFSM) em 1998 e Mestre em Telecomunicações e Telemática pela UNICAMP em 2000. De 2000 a 2004 atuou como engenheiro de software para redes CDMA na Ericsson Telecomunicações do Brasil e Nortel Networks. Tem interesse nas áreas de processamento digital de sinais e comunicações digitais em geral, especialmente códigos corretores de erros e codificação espaço–temporal.

Marcelo Carneiro de Paiva nasceu em Ipameri, GO, em março de 1979. Formou-se Técnico em Eletrônica pela ETE"FMC" (Escola Técnica de Eletrônica Francisco Moreira da Costa) em 1998 e Engenheiro Eletricista com ênfase em Telecomunicações pelo INATEL (Instituto Nacional de Telecomunicações) em 2003. Atualmente, cursa o Mestrado no INATEL. Trabalha com pesquisa e desenvolvimento de sistemas de transmissão em TV Digital desde 2003.

Fábio Adriano Lisboa Gomes nasceu em Picos, PI, em 07 de março de 1975. Recebeu os títulos de Engenheiro Eletricista e Mestre em Engenharia Elétrica, ênfase em Computação, pelo Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Norte em 1998 e 2000, respectivamente. Desde 2003 é professor do Departamento de Informática das Faculdades Hoyler, campus de Hortolândia. Desde 2004 coordena o desenvolvimento de Aplicativos para WEB e para Dispositivos Móveis da empresa TiSoft, nas plataformas J2EE e J2ME. De 2005 a 2006 foi coordenador científico de projeto PIPE/FAPESP relacionado ao desenvolvimento de Sistemas para Dispositivos Móveis. Tem interesse nas áreas de Processamento digital de sinais, Engenharia de software, Tecnologias Orientadas a Objetos, Plataformas de desenvolvimento para WEB 2.0, Linguagens de programação e Automação de testes de software.

Fábio Lumertz é doutorando do departamento de comunicações (DECOM) da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas (UNICAMP) e membro do ComLab, laboratório de pesquisas em comunicações digitais e WebLabs vinculado ao mesmo departamento. Graduou-se em engenharia elétrica com ênfase em telecomunicações na Pontifícia Universidade Católica do Rio Grande do Sul no ano de 2003. Como aluno e pesquisador da Unicamp, participou do projeto do Sistema Brasileiro de Televisão Digital (SBTVD) na proposta de modulação inovadora.

Renato R. Lopes Recebeu os títulos de Engenheiro Eletricista e Mestre em Engenharia Elétrica pela Universidade de Campinas (UNICAMP) em 1995 e 1997, respectivamente. Ele é PhD em Engenharia Elétrica pelo Instituto de Tecnologia da Georgia (GeorgiaTech), EUA, onde também concluiu, em 2001, um mestrado em matemática. De 2003 a 2005 fez pós-doutorado na Faculdade de Engenharia Elétrica e de Computação da UNICAMP, onde é professor assistente desde 2006. Tem interesse na grande área de teoria das comunicações, incluindo equalização e estimação de canal, códigos corretores de erros, e receptores iterativos.

Fabrício C. de A. Oliveira nasceu em São Paulo, SP, em 06 de setembro de 1976. Possui os títulos de: Engenheiro Eletricista, modalidade Eletrônica, pela Universidade Federal de Pernambuco (2000), Mestre em Engenharia Elétrica, com ênfase em Telecomunicações, pela Universidade Estadual de Campinas (2002). Atualmente cursa o doutorado na Universidade Estadual de Campinas. É pesquisador nas áreas de teoria da informação, compressão de imagens e vídeo e codificação de canal para sistemas de comunicação. Atuou nos projetos de compressão de vídeo usando H.264, transcodificação de vídeo digital e modulação durante o desenvolvimento do sistema brasileiro de televisão digital. Seus principais interesses incluem o modelamento teórico de sistemas de comunicação e as aplicações práticas da teoria da informação.

Roberto Gallo nasceu em Campinas, SP, em 17 de maio de 1978. É Engenheiro de Computação (IC - UNICAMP, 2001) e Mestre em Ciência da Computação (IC - UNICAMP, 2003). É especialista no projeto e desenvolvimento de arquiteturas para hardware reconfigurável (FPGA), em SoCs. Atualmente é aluno de Doutorado em Ciência da Computação (IC - UNICAMP), na área de processadores criptográficos. De 2004 a 2006 foi coordenador científico de projeto PIPE FAPESP na área de hardware criptográfico baseado em sistemas reconfiguráveis. Possui experiência de mais de seis anos no desenvolvimento de sistemas digitais, tendo produzido diversos *IP Cores* de processadores e aceleradores criptográficos. Áreas de interesse: hardware reconfigurável (FPGA), sistemas digitais de larga escala, criptografia.