Turbo-like Decoding Algorithm for Structured LDPC codes

Ajit Nimbalker, Yufei Blankenship and Brian Classon
Motorola Labs - Wireless and Solutions Research,
1301 E. Algonquin Road, Rm:2928,
Schaumburg, IL 60196 USA
Email: {A.Nimbalker, Yufei.Blankenship, Brian.Classon}@motorola.com

Abstract – **This paper presents a high-speed "turbo-like" decoding algorithm for certain structured LDPC codes such as those adopted in IEEE 802.16e and in the draft 802.11n standards. It is shown that after a key modification, such LDPC codes may be processed as Generalized Repeat Accumulate codes, codes which are known to support "turbo-like" decoding. A GRA-like encoder of structured LDPC codes is derived, which in turn leads to the decoding algorithm. It is also shown that the "structured" properties result in an inherent parallelism, leading to an efficient high speed decoder implementation.**

I. INTRODUCTION

Low-density parity-check (LDPC) codes [1] are powerful error-correcting codes that are appearing in standards such as IEEE 802.16e [2] and 802.11n, Digital Video Broadcast, etc. Extensive research in recent years has focused on exploring the theoretical performance of LDPC codes, and on the design of practical encoding/decoding techniques.

LDPC codes are usually decoded via iterative message passing algorithms such as the standard belief propagation (SBP) or the layered BP (LBP) [3]. Although LDPC codes may be viewed as general codes on graphs, additional matrix properties may allow more specific encoding/decoding algorithms. For instance, the class of LDPC codes known as Generalized Repeat Accumulate codes (GRA) allows linear time encoding [4].

By definition, a GRA code is a serial concatenation of several component codes, such as repetition codes, single-parity-check (SPC) codes, and Accumulator (ACC). Therefore, such LDPC codes support both LDPC-like and "turbo-like" decoding algorithms [5]. In general, the parity-check matrix of GRA codes has a *full dual-diagonal parity-check portion* including a weight-1 parity column. However, the weight-1 column (when used in structured LDPC codes) leads to a performance loss and hence, matrices with *partial dual-diagonal parity-check portion* are often preferred in practice, e.g., in IEEE802.16e and 802.11n. These LDPC codes are still easily encodable [6], but it is not clear if such codes still support the "turbo-like" decoding algorithms [5].

This paper describes a method that allows turbo-like decoding of structured LDPC codes. These LDPC codes [7] have parity check matrices (**H**) that comprise of all-zero or shifted identity submatrices, and they also have a *partial dual-diagonal parity portion*. An interpretation that allows a parallelized "turbo-like" decoding (TLD) algorithm of such LDPC codes is presented. TLD can reuse technologies developed for turbo decoders such as log-MAP processors, fixed point analysis, and parallelization techniques, and it can potentially combine the features of LDPC and turbo decoders to achieve high throughput and good performance.

II. BACKGROUND

An LDPC code is specified by a sparse parity-check matrix $\mathbf{H}$, with $\mathbf{H}\mathbf{x}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}}$, where "T" denotes matrix transpose, $\mathbf{0}$ is a zero vector. The codeword is $\mathbf{x}=[\mathbf{s}\ \mathbf{p}]=[s_0, s_1, \ldots, s_{k-1}, p_0, p_1, \ldots, p_{m-1}]$, where $p_0, \ldots, p_{m-1}$ are the parity-check bits; and $s_0, \ldots, s_{k-1}$ are the systematic bits. An $\mathbf{H}$ matrix of an LDPC code is often described by a bipartite graph which also provides a framework for deriving (and visualizing) iterative message passing algorithms. Each 1 in $\mathbf{H}$ defines an edge (i.e., a connection between a variable node and a check node) in the bipartite graph, each column in $\mathbf{H}$ corresponds to a variable node and each row in $\mathbf{H}$ corresponds to a check node.

For example, let an $n = 12$, rate-1/2 code be defined by

$$\mathbf{H}=[\mathbf{H}_s \mid \mathbf{H}_p]=\left[\begin{array}{cccccc|cccccc} 1&0&1&0&0&0&1&1&0&0&0&0\\ 0&1&0&0&1&0&0&1&1&0&0&0\\ 0&0&1&0&0&1&1&0&1&1&0&0\\ 1&0&0&1&0&0&0&0&0&1&1&0\\ 0&1&0&0&1&0&0&0&0&0&1&1\\ 0&0&0&1&0&1&1&0&0&0&0&1 \end{array}\right], \qquad (1)$$

with the left side portion corresponding to k (=6) information bits $\mathbf{s}$, the right side portion corresponding to m (=6) parity-check bits $\mathbf{p}$. By definition, the $\mathbf{H}$ in (1) defines six parity-check equations shown in (2). Since $\mathbf{H}$ is full-rank, and the systematic bits (i.e., x_0 through x_5) are known, the six equations can be solved to obtain the six unknown parity-check bits ($[x_6, x_7, \ldots, x_{11}]$), thus providing the codeword after systematic encoding.

$$\begin{aligned} & & x_7 &= (x_0 + x_2 + x_6) \\ x_8 &+ x_7 &= (x_1 + x_4) \\ x_9 &+ x_8 &= (x_2 + x_5 + x_6) \\ x_{10} &+ x_9 &= (x_0 + x_3) \\ x_{11} &+ x_{10} &= (x_1 + x_4) \\ & x_{11} &= (x_3 + x_5 + x_6) \end{aligned} \qquad (2)$$

From a decoding perspective, turbo-like decoding of an LDPC code is easiest when the *entire parity-check portion* of the **H** matrix is dual-diagonal as shown in (3). In such a case, all the parity-check bits are obtained by a repeat-accumulate structure and this serial concatenation leads to the "turbo-like" decoding algorithm.

$$\mathbf{H}'_p = \underbrace{\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & \ddots & 0 & 0 \\ 0 & 0 & 0 & \ddots & 1 & 0 \\ \vdots & \vdots & \vdots & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathrm{m}} \Bigg\} \mathrm{m} \qquad (3)$$

However, the matrix in (1) is only partial dual-diagonal (since the first column of $\mathbf{H}_p$ is different from the first column in (3)), and it is not clear how to handle LDPC codes of (1) with a GRA-like structure. The following sections show how to modify GRA-like algorithms [5] to handle LDPC codes with partial dual diagonal parity-check portion.

III. A GRA-LIKE ENCODER

This paper illustrates the key ideas using the **H** matrix in (1), and they can be readily extended to other LDPC codes with a partial dual-diagonal parity portion. Considering a systematic encoding, the six equations of (2) can be solved to obtain the parity bits in two steps as follows:

i). The systematic portion of the codeword is used to compute the parity bit corresponding to the non-dual-diagonal portion of $\mathbf{H_p}$, which is the first parity bit x_6 for (1). Simply adding all the parity-check equations in (2) cancels all unknown variables except x_6.

ii). The parity bits corresponding to the partial dual-diagonal portion, which are ($x_7, x_8, x_9, x_{10}, x_{11}$) for (1), are obtained through successive back-substitution (i.e., accumulation) using the parity-check equations in (2).

The rest of the paper considers Step ii), which is a GRA-like structure, leading to the proposed decoding algorithm.

First the input bits [x_0, x_1, x_2, x_3, x_4, x_5, x_6] (including a computed parity bit x_6) are *repeated* according to the number of times each bit appears on the right-hand-side (RHS) of (2). The output of the repetition code is rearranged via an interleaver so that the bits can be grouped in the order they appear on the RHS of (2). The RHS of (2) represents SPC codes, whose outputs are *accumulated* (i.e., the back substitution on the LHS of (2)) using an ACC. The ACC begins and ends in zero-state, and its last output of the ACC is always 0 (thus not transmitted), because the sum of the LHS (and RHS) of (2) is zero.

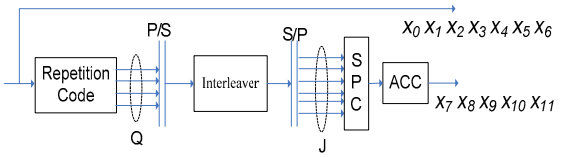

Figure 1. A GRA-like encoder of **H** matrices with a partial dual-diagonal parity portion. The input consists of the information bits and one parity bit (x_0 through x_6). Vector Q contains the repetition factors, and vector J contains the SPC parameters.

While all parity bits are computed using GRA structure in [5], the new method pre-computes the parity bit of the non-dual-diagonal portion (parity bit x_6) in a non-GRA fashion, before applying a GRA-like encoder to compute the remaining parity bits. A block diagram of a GRA-like encoder for the **H** of (1) is shown in Figure 1. The GRA-like encoder may be interpreted as follows (using the notation of [5]).

The input [x_0, x_1, x_2, x_3, x_4, x_5, x_6] passes through a repetition code with a repetition factor Q = [Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6], where input bit x_i is repeated Q_i times. The P/S indicates the bits generated in parallel are converted to serial. An interleaver permutes the output of repetition code before the SPC encoder according to a permutation ρ. The SPC code outputs one bit for every J_i serialized input bits ($J_i \in$ [J_0, J_1, J_2, J_3, J_4, J_5]). The S/P indicates that J_i bits are input to the SPC to obtain a temporary bit u_i, where u_i is equal to the RHS of i^{th} equation in (2). The u_i's are accumulated to obtain remaining unknown parity-check bits.

The exact parameters of the GRA-like encoder may be obtained by partitioning **H** into two parts, $\mathbf{H} = [\mathbf{H_{GRA}}\ \mathbf{H_{p2}}]$, as shown in (4), where $\mathbf{H_{p2}}$ is the partial dual-diagonal parity portion. Note that the columns of $\mathbf{H_{GRA}}$ correspond to the systematic bits and one parity bit.

$$\mathbf{H}_{GRA} = \underbrace{\left[\begin{array}{cccccc:c} 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]}_{k+1}, \quad \mathbf{H}_{p2} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{m-1}, \qquad (4)$$

Parameter Q_i is equal to the number of ones in i^{th} column of $\mathbf{H_{GRA}}$, i = 0, 1,…, k. Parameter J_i is equal to the number

of ones in the i^{th} row of $\mathbf{H}_{\mathbf{GRA}}$, i = 0, 1,…, m-1. The interleaver (ρ) length W is equal to the number of ones in $\mathbf{H}_{\mathbf{GRA}}$. By definition, the i^{th} input bit is permuted to the $\rho(i)^{\text{th}}$ position in the output as a result of the permutation (ρ), which is obtained as follows. Label the ones (i.e., edges) in $\mathbf{H}_{\mathbf{GRA}}$ in a column-wise order starting with the left-most column as shown in the left hand side of (5). These indices sequentially number the edges after repetition and before interleaving. Label the ones in $\mathbf{H}_{\mathbf{GRA}}$ in a row-wise order from the top-most row as shown on the right hand side of (5). These indices sequentially number the edges after interleaving, before being input to the SPC. The permutation (ρ) is given by reading the row-wise label in column-wise order.

$$H_{GRA}=\begin{bmatrix} l_0 & 0 & l_4 & 0 & 0 & 0 & l_{12} \\ 0 & l_2 & 0 & 0 & l_8 & 0 & 0 \\ 0 & 0 & l_5 & 0 & 0 & l_{10} & l_{13} \\ l_1 & 0 & 0 & l_6 & 0 & 0 & 0 \\ 0 & l_3 & 0 & 0 & l_9 & 0 & 0 \\ 0 & 0 & 0 & l_7 & 0 & l_{11} & l_{14} \end{bmatrix} \xrightarrow{\rho} \begin{bmatrix} l_0 & 0 & l_1 & 0 & 0 & 0 & l_2 \\ 0 & l_3 & 0 & 0 & l_4 & 0 & 0 \\ 0 & 0 & l_5 & 0 & 0 & l_6 & l_7 \\ l_8 & 0 & 0 & l_9 & 0 & 0 & 0 \\ 0 & l_{10} & 0 & 0 & l_{11} & 0 & 0 \\ 0 & 0 & 0 & l_{12} & 0 & l_{13} & l_{14} \end{bmatrix} \quad (5)$$

For the (12,6) code of (1), the parameters are Q = [2 2 2 2 2 3], J = [3 2 3 2 2 3], and the interleaver is ρ = [0 8 3 10 1 5 9 12 4 11 6 13 2 7 14], with W=15.

IV. A TURBO-LIKE DECODER

The GRA-like encoder described in the previous section is used to derive a corresponding "turbo-like" decoder whose graphical model with corresponding GRA-parameters is shown in Figure 2. Solid circles indicate repetition nodes (variable nodes corresponding to non-dual diagonal parity portion), the solid squares represent the SPC nodes (or the check nodes) and empty circles represent the variable nodes corresponding to the dual-diagonal parity portion. The non-dual-diagonal parity bit is highlighted to show that it can be treated as a systematic bit during decoding.

A TLD for LDPC codes consists of two component decoders - a repetition decoder which is similar to the variable node update in conventional LDPC decoders, and a combined SPC-ACC decoder (below the interleaver in Figure 2). The SPC-ACC concatenation is equivalent to a 2-state state convolutional code with irregular puncturing (with periods given by vector **J**). Therefore, a trellis-based SPC-ACC decoder can be used as a constituent decoder of a "turbo-like" decoder. Note that as the values of J_i increases, there is increased puncturing in the trellis and hence the resulting SPC-ACC decoder (and the overall TLD) becomes weaker. This property of TLD is further discussed in Section VI.

An iteration of TLD consists of the repetition decoding followed by SPC-ACC decoding (see [5] for trellis update equations). From a graph perspective, the two decoders iteratively exchange extrinsic LLRs related to the edges of $\mathbf{H}_{\mathbf{GRA}}$ via the (de)interleaver. Therefore, the extrinsic message memory is proportional to the number of 1's in $\mathbf{H}_{\mathbf{GRA}}$ which is the interleaver size W. The proposed TLD algorithm updates all edges connected to the systematic bits and one parity bit while the GRA decoder of [5] only updates the edges connected to the systematic bits.

The SPC-ACC processing in TLD is similar yet different from the "check node update" (CNU) in LDPC literature. In TLD, several parity-check equations are linked directly through the ACC. This allows the check nodes to send messages to each other directly during the SPC-ACC decoding. In contrast, in a SBP decoder, parity-check equations do not interact with each other directly.

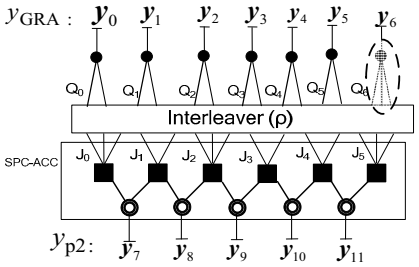

Figure 2. A graphical model of a turbo-like decoder of an LDPC code with a partial dual-diagonal parity portion.

V. STRUCTURED LDPC CODES

Structured LDPC codes are constructed with all-zero submatrix and shifted identity submatrices as building blocks [7]. This enables block-wise or vectorized encoding and decoding which leads to efficient hardware designs [2]. In addition, such codes can also be designed to have a block-wise partial dual-diagonal parity portion, (e.g., in IEEE 802.16e, 802.11n) for easy encoding. This section extends the TLD algorithm of previous sections to structured LDPC codes by deriving an equivalent GRA-like encoder. In particular, the resulting TLD is shown to be highly parallelizable because of the contention-free memory access property of the GRA-like interleaver [8].

A structured LDPC code design starts with a small $m_b \times n_b$ base matrix $\mathbf{H}_{\text{b}}$, makes z copies of $\mathbf{H}_{\text{b}}$, and interconnects the z copies to form a large $M \times N$ binary $\mathbf{H}$ matrix, where $M = m_b \times z$, $N = n_b \times z$. The binary $\mathbf{H}$ matrix is obtained by replacing each 1 in $\mathbf{H}_{\text{b}}$ by a $z \times z$ shifted identity matrix ($\mathbf{P}$), and each 0 in $\mathbf{H}_{\text{b}}$ by a $z \times z$ all-zero matrix. Hence, the $\mathbf{H}$ matrix can also be described by an $m_b \times n_b$ model matrix $\mathbf{H}_{\text{bm}}$, which is obtained by replacing each 0 in $\mathbf{H}_{\text{b}}$ by "–1" (to denote a $z \times z$ all-zero matrix), and by replacing each $h_{i,j}$=1 in $\mathbf{H}_{\text{b}}$ by a shift size $p(i,j)$ to denote a $z \times z$ identity matrix whose columns are cyclically shifted by $p(i,j)$.

For example, the matrix in (1) may be used as a base matrix $\mathbf{H}_{\text{b}}$ to build a model matrix $\mathbf{H}_{\text{bm}}$ in (6). When z=3, $\mathbf{H}_{\text{bm}}$ is converted to a $(6 \times z) \times (12 \times z)$ binary matrix $\mathbf{H}$ by

replacing each –1 with a 3×3 all-zero matrix and each i with $\mathbf{P}_i$, i=0, 1, 2, where P_i is a 3×3 identity matrix whose columns are cyclically shifted to the right by i positions. The resulting **H** matrix has a codeword size N=12×3=36, and an information block size K=6×3=18.

$$\mathbf{H}_{\mathrm{bm}}=\underbrace{\left[\begin{array}{cccccc:c:cccccc} 1 & -1 & 0 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ -1 & 2 & -1 & -1 & 0 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 & 2 & 2 & -1 & 0 & 0 & -1 & -1 \\ 2 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 \\ -1 & 1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\ -1 & -1 & -1 & 0 & -1 & 1 & 0 & -1 & -1 & -1 & -1 & 0 \end{array}\right]}_{n_b}\Bigg\}m_b, \qquad (6)$$

It was shown earlier that base matrix $\mathbf{H}_{\mathrm{b}}$ of (1) can be encoded and decoded using GRA-like structure. If such a base matrix is used to create an **H** matrix by expansion (e.g., as in (6)), then the resulting **H** matrix also has a GRA-like encoder which bears many similarity to that of the base matrix $\mathbf{H}_{\mathrm{b}}$.

Let $\mathbf{S} = [\boldsymbol{S}_0, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_{k-1}]$ and $\mathbf{X} = [\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{n-1}]$ represent the information block and the codeword block, respectively, where each element is a z-bit vector (i.e., size z×1). The blockwise encoding may be done as follows.

i). Fill the systematic portion of codeword with a direct copy of the information bits $[\boldsymbol{S}_0, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_{k-1}]$, i.e., $\boldsymbol{X}_0=\boldsymbol{S}_0, \boldsymbol{X}_1=\boldsymbol{S}_1, \boldsymbol{X}_2=\boldsymbol{S}_2, \ldots, \boldsymbol{X}_{k-1}=\boldsymbol{S}_{k-1}$.

ii). Compute the parity block ($\boldsymbol{X}_k$) related to the non-dual-diagonal parity portion (i.e., by solving the corresponding parity-check equations).

iii). Compute the parity blocks related to the partial dual-diagonal parity portion ($\boldsymbol{X}_{k+1}, \ldots, \boldsymbol{X}_{n-1}$) using a structured GRA-like encoder (block-wise accumulation).

Note that the third step is similar to GRA-like encoding at a blockwise level, which can be divided in z equivalent bitwise counterparts. As illustrated in Figure 3, the encoder consists of z copies of the GRA-like encoder of the base matrix $\mathbf{H}_{\mathbf{b}}$ interconnected by a vector interleaver. The figure assumes that each group of z bits is represented by a column vector.

The main advantage of using a structured LDPC is evident from Figure 3 : highly parallelizable encoding/decoding operations. Note also that the parameters $\boldsymbol{Q}_b$, and $\boldsymbol{J}_b$ of all z copies of structured GRA-like encoder are identical to that of the base matrix. The vector interleaver consists of two stages: i) a permutation (ρ) of the extrinsic LLR vectors that is the same as the base matrix permutation, and ii) a set of shift sizes ($\mathbf{R}_{\mathbf{bm}}$) corresponding to rotation within each extrinsic LLR vector which depends on the model matrix. Referring to Figure 3, the two stages of permutations correspond to column permutations and column rotations, respectively.

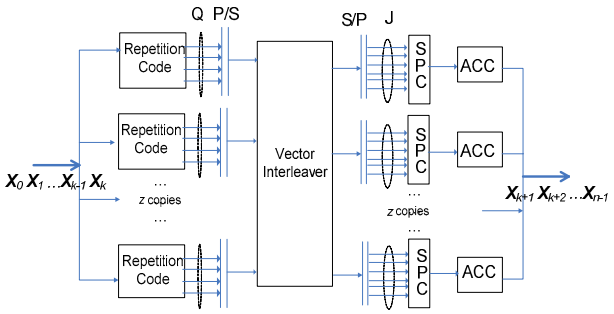

Figure 3. A GRA-like encoder of a structured LDPC code.

For the (36, 18) code of (6), the GRA parameters are identical to those of the base matrix $\mathbf{H}_{\mathbf{b}}$ of (1): $\boldsymbol{Q}_{\mathbf{b}}$ = [2 2 2 2 2 2 3], $\boldsymbol{J}_{\mathbf{b}}$ = [3 2 3 2 2 3], the permutation is ρ = [0 8 3 10 1 5 9 12 4 11 6 13 2 7 14].

The only new parameter required to describe structured GRA-like encoder are the shift values $\mathbf{R}_{\mathbf{bm}}$, which are obtained from the model matrix of (6) by reading the shift sizes in a columnwise order starting from the left hand side of the $\mathbf{H}_{\mathbf{bm,GRA}}$ shown in (7) . This leads to a set of shift sizes given by $\mathbf{R}_{\mathbf{bm}}$=[1 2 2 1 0 1 1 0 0 0 2 1 0 2 0].

$$\mathbf{H}_{bmGRA}=\left[\begin{array}{cccccc:c} 1_0 & -1 & 0_4 & -1 & -1 & -1 & 0_{12} \\ -1 & 2_2 & -1 & -1 & 0_8 & -1 & -1 \\ -1 & -1 & 1_5 & -1 & -1 & 2_{10} & 2_{13} \\ 2_1 & -1 & -1 & 1_6 & -1 & -1 & -1 \\ -1 & 1_3 & -1 & -1 & 0_9 & -1 & -1 \\ -1 & -1 & -1 & 0_7 & -1 & 1_{11} & 0_{14} \end{array}\right] \qquad (7)$$

The TLD of structured LDPC codes can also be performed in a structured (or parallelized) manner, analogous to the structured encoding. The parallelized turbo-like decoder consists of z identical copies of repetition and SPC-ACC decoders that are interconnected through the vector interleaver (z copies of Figure 2). The received LLR values are suitably distributed to the appropriate decoders alike Figure 2.

High speed TLD is achieved by using several (up to z) processors operating in parallel. The LLRs are stored in multiple memories to allow several concurrent read/write operations. In the iterative process, the extrinsic LLRs are exchanged between the processors (through memory operations) according to the vector interleaver.

The vector interleaver of structured LDPC codes can be described as a contention-free (CF) inter-window shuffle (IWS) interleaver [8]. CF interleaving is important for maximizing decoder throughput as it ensures that concurrent read/write operations for the z processors do not result in any memory access contentions, thereby minimizing (de)interleaving latency in the iterative decoding.

The interleaver of a structured LDPC code may be interpreted as a CF interleaver by making the following observation about the two stages of the permutation. The cyclic shift of individual vectors (i.e., column rotation) as specified by $\mathbf{R}_{\mathbf{bm}}$ is equivalent to the inter-window shuffle pattern, while the permutation among the vectors (i.e., column permutation) as specified by ρ is equivalent to the intra-window permutation described in [8].

In general, the CF interleaver can be described as

$$\pi(i) = \rho(i \bmod W) + W\varphi_{\lfloor i/W \rfloor}(i \bmod W), \qquad (8)$$

where vector ρ defines the intra-window shuffling, φ(*i*) defines inter-window shuffling for the i^{th} index of the window. For the structured TLD decoder, the window size is *W* which is the length of the base matrix interleaver ρ, and φ(*i*) is the cyclic shifted index vector with shift size $\mathbf{R}_{\mathbf{bm}}(i)$. For the (36, 12) code of (6), if $\mathbf{R}_{\mathbf{bm}}(i) = 2$, φ(*i*) = (2, 3, …, *z*-1, 0, 1). Mathematically, the inter-window shuffle pattern can be expressed as follows,

$$\varphi_{\lfloor i/W \rfloor}(i \bmod W) = \left(R_{bm}(i \bmod W) + \left\lfloor \frac{i}{W} \right\rfloor \right) \bmod z \qquad (9)$$

and it indicates which window the position *i* is mapped to. In the next section, performance results for TLD and SBP are shown using IEEE 802.16e LDPC codes.

VI. PERFORMANCE

Structured LDPC codes from the IEEE 802.16e are chosen to compare the proposed algorithm with SBP decoding. In the IEEE 802.16e standard, the model matrices of all the code rates (1/2, 2/3, 3/4, 5/6) have 24 columns, while the number of rows is a function of the code rate. Different codeword sizes are obtained by suitably choosing an expansion factor *z*. For example, the rate-1/2 code has a 12×24 base matrix, and with an expansion factor of *z*=24, it results in a 576-bit codeword.

The 20^{th} iteration FER performance of the IEEE 802.16e LDPC codes with rates-1/2, 2/3 and 3/4, and an expansion factor *z*=96 (*N*=2304-bit codeword) is shown in Figure 4 for a BPSK-modulated AWGN channel. The complexity of the SBP and TLD algorithms per iteration is assumed to be similar as the number of equivalent check node updates is the same. Figure 4 indicates that TLD outperforms SBP when the LDPC code has a substantial dual-diagonal parity portion, i.e., at lower code rates.

As the rate increases (e.g., from rate-1/2 to 2/3 or 3/4), the number of dual-diagonal parity columns decreases, and each ACC trellis of the TLD now connects fewer check equations together. This can also be interpreted as increased puncturing in the SPC-ACC trellis, and therefore the performance advantage of TLD performance over SBP is reduced as the rate increases.

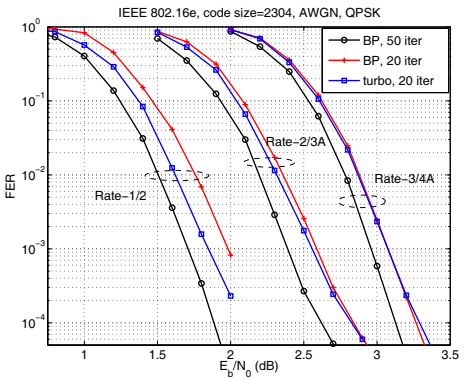

Figure 4. IEEE 802.16e LDPC codes, *N* = 2304 (with *z*=96), *R* = 1/2, 2/3 and 3/4 with flooding schedule for both standard BP and "turbo-like" decoding. For rate-2/3 and 3/4, the codes designated as 2/3A and 3/4A were simulated.

VII. CONCLUSIONS

In this paper, a turbo-like decoding (TLD) algorithm is proposed for structured LDPC codes with partial dual-diagonal parity portion. The encoding and turbo-like decoding algorithm is described for such LDPC codes by deriving a GRA-like structure. It is demonstrated that structured LDPC codes facilitate high speed TLD due to the contention-free property of its interleaver. The performance of the TLD algorithm is compared with standard belief propagation using IEEE 802.16e LDPC codes. It is noted that the proposed algorithm successfully applies the turbo decoding concepts to the decoding of LDPC codes and has the potential of achieving better complexity/performance tradeoffs.

VIII. REFERENCES

[1] R. G. Gallager, "Low density parity check codes," *IRE Trans. Inform. Theory*, IT-8, pp. 21-28, Jan. 1962.

[2] IEEE Std 802.16e-2005, approved Dec 2005, pub. Feb 2006.

[3] M. Mansour, N. Shanbag, "High-throughput LDPC decoders", *IEEE Trans. on VLSI*, vol. 11, pp. 976-996, Dec. 2003.

[4] H. Jin, A. Khandekar, and R. McEliece, "Irregular Repeat-Accumulate Codes," in *Proc. 2nd Int. Symp. Turbo Codes and Rel. Topics*, Brest, pp. 1-8, Sep. 2000.

[5] K. M. Chugg, P. Thiennviboon, G. D. Dimou, P. Gray, and J. Melzer, "A new class of turbo-like codes with universally good performance and high-speed decoding", *MILCOM*, 2005.

[6] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," IEEE Trans. Inform. Theory, vol. 47, pp. 638-656, Feb. 2001.

[7] D. Sridhara, T. Fuja, and R. M. Tanner, "Low density parity check codes from permutation matrices", *Conf. on Inform. Sciences and Sys.,* John Hopkins University, Mar. 2001.

[8] A. Nimbalker, T. K. Blankenship, B. Classon, T. Fuja, and D. J. Costello, Jr, "Inter-window shuffle interleavers for high throughput turbo decoding", in *3rd Int. Symp. On Turbo Codes and Rel. Topics*, Brest, pp. 355-358, Sep. 2003.