Application of Nonbinary LDPC Cycle Codes to MIMO Channels

Ronghui Peng, *Student Member, IEEE,* and Rong-Rong Chen, *Member, IEEE*

***Abstract*— In this paper, we investigate the application of nonbinary low-density parity-check (LDPC) cycle codes over Galois field GF(q) to multiple-input multiple-output (MIMO) channels. Two types of LDPC coded systems that employ either *joint* or *separate* MIMO detection and channel decoding are considered, depending on the size of the Galois field and the modulation choice. We construct a special class of nonbinary LDPC cycle codes called the parallel sparse encodable (PSE) codes. The PSE code, consisting of a quasi-cyclic (QC) LDPC cycle code and a simple tree code, has the attractive feature that it is not only linearly encodable, but also allows parallel encoding which can reduce the encoding time significantly. We provide a systematic comparison between nonbinary coded systems and binary coded systems in both performance and complexity. Our results show that the proposed nonbinary system employing the PSE code outperforms not only the binary LDPC code specified in the 802.16e standard, but also the optimized binary LDPC code obtained using the EXIT chart methods. Through a detailed complexity analysis, we conclude that for the MIMO channel considered, the nonbinary coded systems achieve a superior performance at a receiver complexity that is comparable to that of the binary systems.**

***Index Terms*— MIMO channels, nonbinary LDPC, cycle codes, quasi-cyclic.**

I. Introduction

In recent years, multiple-input multiple-output (MIMO) transmission has been identified as one of the most practical methods to combat fading and to increase the capacity of wireless channels. There has been much research on designing good channel codes such as turbo codes and low-density parity-check (LDPC) codes for MIMO channels. In particular, LDPC codes [1] have attracted substantial interests due to their capacity approaching performance and great flexibility in code design and practical implementation.

Nonbinary LDPC codes are first investigated in [2] where it is shown that nonbinary LDPC codes constructed over higher order Galois fields achieve superior performance than the binary codes for binary symmetric channels and binary Gaussian channels. Applications of nonbinary LDPC codes to Rayleigh fading channels [3], frequency selective channels [4], and MIMO channels [5]–[8] have also been studied. Recently, irregular nonbinary LDPC codes over GF(q) constructed using the progressive edge growth (PEG) algorithm are proposed in [9]. It is shown that as the field order increases, the optimized degree sequence of nonbinary LDPC codes favors a lower average column weight. Furthermore, if the field order is sufficiently large, the optimum graph tends to favor a regular cycle code [10], [11] for which each column of the parity-check matrix contains exactly two nonzero elements.

This motivates us to study the application of nonbinary LDPC cycle codes to MIMO channels. In addition to a detailed comparison of performance and receiver complexity between nonbinary cycle codes and binary codes, we also propose a class of nonbinary codes called the parallel sparse encodable (PSE) codes to reduce the encoding complexity. Each PSE code consists of a quasi-cyclic (QC) LDPC cycle code and a simple tree code. The encoding complexity of the PSE codes is $O(md_c)$, where m is the number of checks in the LDPC code, and d_c is the degree of check node. The QC structure of such codes facilitates parallel encoding which results in significant reduction of encoding time.

Furthermore, we examine the performance of the PSE codes for MIMO systems that employ either *joint* or *separate* (MIMO) detection and (channel) decoding. We refer to such systems as JDD systems or SDD systems. Most work in the literature focuses on JDD systems. In particular, binary coded systems are JDD systems when higher order modulations are used. For nonbinary coded systems, it is shown in [6] that, nonbinary LDPC codes over small Galois field (up to GF(16)) outperform certain binary LDPC codes in JDD systems. However, since the binary LDPC codes used in [6] are not optimized for MIMO channels, they do not serve as accurate performance benchmarks for nonbinary coded systems.

The main contributions of this paper are summarized as follows:

(1) We propose the use of QC nonbinary LDPC cycle code for MIMO channels. Starting from any base QC nonbinary LDPC cycle code, which in general is not sparse encodable, we can construct a PSE code which allows not only linear-time encoding but also parallel implementation. For PSE codes, our encoding method has a much lower complexity than that of the encoding method in [12]. Furthermore, we show that the PSE code achieves a performance that is very close to the base code at a much reduced encoding complexity. Compared to other nonbinary LDPC codes considered in the literature, such as the randomly constructed LDPC codes in [6] and the algebraically constructed codes in [13], the proposed PSE code is more amenable for implementation due to its simple structure.

(2) We take a broad approach by investigating both JDD and SDD systems for MIMO channels. Our results show that PSE codes perform well in both systems. In particular, the proposed SDD system employing the nonbinary PSE code over GF(256) outperforms the JDD system employing an

Manuscript received January 15, 2007; revised May 20, 2007 and August 21, 2007; accepted October 8, 2007. The associate editor coordinating the review of this letter and approving it for publication was A. Gulliver. This work is supported in part by NSF under grant ECS-0547433. The material in this paper was presented in part at the 2006 IEEE Global Telecommunications Conference (Globecom'06).

The authors are with the Dept. of Electrical and Computer Engineering, Univ. of Utah, Salt Lake City, UT, 84112 USA (e-mail: {peng, rchen}@ece.utah.edu).

Digital Object Identifier 10.1109/TWC.2008.070057.

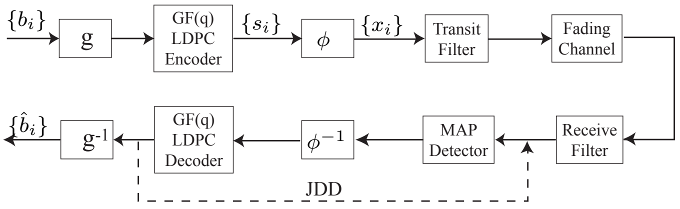


Fig. 1. A schematic block diagram of SDD (without the feedback loop) and JDD (with the feedback loop) systems.

optimized binary LDPC code by 0.37 dB, where the code optimization is done by following a curve fitting approach on EXIT charts proposed in [14]. Due to its highly irregular degree sequence, the encoding complexity of the optimized binary LDPC code is also higher than that of the proposed PSE code. When compared with a more practical QC binary LDPC code defined in the 802.16e standard [15] that is amenable for implementation, the proposed SDD system employing the PSE code achieves a larger performance gain of 0.61 dB.

(3) We conduct detailed complexity analysis to show that performance gain of the nonbinary coded systems is achieved at a receiver complexity that is comparable to that of the binary coded systems. To the best of our knowledge, this is the first work to provide a systematic performance and complexity comparison between the *optimized* binary coded systems with nonbinary coded systems for MIMO channels.

This paper is organized as follows. In Section II, we introduce the system model. Section III describes the proposed PSE nonbinary LDPC cycle codes and discusses code construction and encoding complexity. Performance comparisons and complexity analysis are presented in Section IV. Conclusions are given in Section V.

II. System Model

Fig. 1 shows a block diagram of the nonbinary LDPC coded MIMO system. Assume that the LDPC code is defined over GF(q), where $q = 2^p$. At the transmitter side, a sequence of information bits $\{b_i\}$ is mapped to a sequence of nonbinary symbols in GF(q) (every p bits are mapped to a single nonbinary symbol) through a bit-to-symbol mapper g, before passing to the nonbinary LDPC encoder. Let t denote the number of transmit antennas. At the output of the LDPC encoder, every group of n_0 coded nonbinary symbols $\mathbf{s} = \{s_1, \cdots, s_{n_0}\} \in$ GF(q) is mapped to a group of t constellation symbols $\mathbf{x} = (x_1, \cdots, x_t) = \phi(\mathbf{s})$ through the mapper ϕ. Given the constellation size $M = 2^{m_0}$, we have $p \cdot n_0 = t \cdot m_0$. The sequence of constellation symbols is then passed to the transmit filter and sent through the t transmit antennas. The receiver performs optimal maximum *a posteriori* probability (MAP) detection to compute the prior probabilities for each group of t transmitted constellation symbols. These prior probabilities will then be passed (after the mapper ϕ^{-1}) to the LDPC decoder for iterative decoding. After a finite number of decoding iterations, hard decisions on the nonbinary symbols are made at the output of LDPC decoder, which are then demapped to the sequence of estimated information bits.

When $n_0 = 1$, the MAP detector produces prior probabilities for each GF(q) symbol which can be used directly for nonbinary LDPC decoding over GF(q). Hence, it is sufficient to perform MIMO detection only once followed by channel decoding. This corresponds to a SDD system that performs separate detection and decoding. When $n_0 > 1$, the prior probabilities of the group of n_0 nonbinary symbols are dependent because they are mapped to complex symbols that are transmitted simultaneously. Then it is necessary to pass soft information about the dependent symbols from the LDPC decoder back to the MAP detector to produce updated symbol-wise probabilities. This corresponds to a JDD system that performs joint detection and decoding. As shown in Fig. 1, the JDD system requires a feedback loop from the channel decoder to the MAP detector to allow iterative exchange of soft information.

Let us consider a MIMO channel with two transmit and receive antennas. Suppose that the 16 quadrature amplitude modulation (QAM) is used. An example of the JDD system is with $q = 16$ and $n_0 = 2$. Namely, every *two* GF(16) coded symbols are mapped to two 16 QAM symbols that are transmitted simultaneously through the two transmit antennas. An example of a SDD system is with $q = 256$ and $n_0 = 1$. A *single* coded GF(256) symbol is mapped to two 16 QAM symbols which are transmitted simultaneously.

Next, we explain how the MAP detector shown in Fig. 1 works. The channel model is given by

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \tag{1}$$

where $\mathbf{x} \in \mathcal{C}^{t \times 1}$ is the complex transmitted signal vector that satisfies the component-wise energy constraint $E(\|x_i\|^2) = E_s/t$, and E_s is the total transmitted power, $\mathbf{y} \in \mathcal{C}^{r \times 1}$ is the complex received signal vector, r is the number of receive antennas, $\mathbf{H} \in \mathcal{C}^{r \times t}$ is the channel fading matrix with independent entries that are complex Gaussian distributed with zero mean and unit variance, $\mathbf{n} \in \mathcal{C}^{r \times 1}$ is complex white Gaussian noise with variance σ^2 per dimension. $\mathbf{H}$ is assumed to be known to the receiver but not to the transmitter.

Given each received signal vector $\mathbf{y}$, we perform MAP detection to determine the a *posteriori* probabilities (APP) of each nonbinary symbol $s_j, j = 1, \cdots, n_0$, by computing the log-likelihood-ratio vector (LLRV) over GF(q). Let $\{0, \alpha_1, \cdots, \alpha_{q-1}\}$ denote elements in GF(q). The LLRV of s_j is defined by $\mathbf{z} = \{z_0, z_1, \cdots, z_{q-1}\}$, where $z_i = \ln[p(s_j = 0)/p(s_j = \alpha_i)]$. From equation (1) we have

$$z_i = \ln \frac{\sum_{\mathbf{s}:s_j=0} \exp[-\|\mathbf{y} - \mathbf{H}\phi(\mathbf{s})\|^2/(2\sigma^2)]\mathbf{p}(\mathbf{s})}{\sum_{\mathbf{s}:s_j=\alpha_i} \exp[-\|\mathbf{y} - \mathbf{H}\phi(\mathbf{s})\|^2/(2\sigma^2)]\mathbf{p}(\mathbf{s})} \tag{2}$$

where $\| \cdot \|^2$ denotes the norm square of a vector and $p(\mathbf{s})$ denotes the prior probabilities of $\mathbf{s}$ which are passed from the LDPC decoder. Subsequently, these LLRV values are passed to the LDPC decoder for iterative decoding.

III. Parallel sparse encodable (PSE) nonbinary LDPC cycle codes

In this section, we first discuss the code construction of nonbinary QC LDPC cycle codes. By exploiting the QC structure of nonbinary cycle codes, we then obtain a class of PSE cycle codes that allows not only linear-time encoding but also efficient parallel implementation. Performance of the PSE codes will be examined in Section IV.

A. *Construction of quasi-cyclic nonbinary LDPC codes*

We use the method of quadratic permutation polynomial (QPP) over integer rings [3], [16], [17] to construct nonbinary QC cycle codes. These codes are used as based codes in Section III-C to define PSE codes. First, we apply the QPP method to construct a binary QC code [16], [17]. Then we replace each nonzero binary circulant submatrix in the parity-check matrix of the binary code with a nonbinary circulant permutation matrix to obtain a nonbinary QC code. Analog to the binary case, each row of the nonbinary circulant permutation matrix is the right cyclic-shift of the row above it. The nonzero element δ in the first row of the circulant matrix is randomly chosen from GF(q). We apply this method to construct a rate $1/2$ QC LDPC cycle code over GF(256) using the QPP $f(x) = 17x + 30x^2$. The code length is 300 GF(256) symbols, the dimension of the circulant size $\zeta = 15$, and the check node degree $d_c = 4$. The local girth is 14 for each variable node. We also construct a nonbinary LDPC code over GF(256) based on the PEG algorithm [9], for which one variable node has a local girth of 16, 262 variable nodes have a local girth of 14, and 37 variable nodes have a local girth of 10. Both codes are used in the simulation.

B. *Sparse encoding of binary cycle codes*

Even though it is well-known that binary cycle codes are linearly encodable [10], in this section we provide a proof for this important fact. The encoding method described in the proof will be extended to the encoding of nonbinary cycle codes in Section III-C. Here we represent LDPC cycle codes by *normal graphs* [10], [18] where each row of the parity-check matrix H corresponds to a *vertex* and each column corresponds to the an *edge* whose two *end vertices* correspond to the two rows with nonzero elements in that column.

Theorem 3.1: Binary cycle codes are linearly encodable.

Proof: Since the normal graph of a binary cycle code, denoted by $N(H)$, must be a union of several connected graphs, without loss of generality, it is sufficient to consider a single connected graph G. Assume that H has n columns and m rows, then G has n edges and m vertices. It is well-known that every connected graph contains a *spanning tree*, with any specified vertex as its root [19]. Starting from an arbitrary vertex c_1 in G, let $\mathrm{Tr}(G)$ denote a spanning tree of G with c_1 as the root. Since G contains m vertices, there must be a total of $m-1$ edges in $\mathrm{Tr}(G)$. Let $b_1, b_2, \cdots, b_{n-m+1}$ denote edges in G but not in $\mathrm{Tr}(G)$. The encoding process proceeds as follows: let $b_1, b_2, \cdots, b_{n-m+1}$ correspond to information bits of the code, then the values of the edges in $\mathrm{Tr}(G)$ that are incident to the leaves can be computed since only one edge is unknown at each leaf. Subsequently, by removing all the edges whose values are previously computed, we obtain a new tree. This way we can compute all the edge values level by level until all the edges incident to c_1 are computed. We claim that the check equation corresponding to c_1 is then automatically satisfied. In other words, the vertex c_1 is a redundant check. This is because the summation of all rows in the parity-check matrix of a cycle code equals zero, which means that if all the other $m-1$ checks are satisfied, then the remaining check is also satisfied. ■

The proof above shows that the encoding process of binary cycle codes is equivalent to solving the parity-check equations row by row sequentially with a re-arranged order of the rows in H. We refer to this encoding algorithm as sparse encoding. The codes that can be encoded using sparse encoding are called *sparse encodable* codes. Compared to a similar proof in [20], our proof is more compact due to the use of the well-known spanning tree concept. Also, we do not remove the redundant check as in [20]. This is useful in extending our proof to nonbinary case.

C. *Parallel sparse encoding of nonbinary cycle codes*

Unfortunately, nonbinary cycle codes are not sparse encodable in general. The proof of Theorem 3.1 shows that the root vertex c_1 must be redundant in order for the code to be sparse encodable. This is not necessarily true for nonbinary codes. Therefore, in order to realize sparse encoding for nonbinary cycle codes, one option is to change the code constraint associated with the root vertex c_1. Based on this idea, we propose a novel sparse encoding method for nonbinary QC LDPC cycle codes. Since this method utilizes the QC structure of the LDPC cycle code to facilitate parallel encoding, we refer to it as parallel sparse encoding. We will show that, starting from any base QC nonbinary LDPC cycle code, we can obtain a PSE code consisting of a QC subcode, modified from the base code, and a simple tree subcode. Simulation results in Section IV demonstrate that the resulting PSE code achieves a comparable performance to the base code with much reduced encoding time.

The parallel sparse encoding procedure.

Assume that the parity-check matrix of the base QC code, H, with dimensions $m \times n$, is composed of permutation circulant submatrices of dimensions $\zeta \times \zeta$. We first show that the normal graph $N(H)$ consists of ζ disjoint spanning trees that are isomorphic. We build the spanning trees using the ζ vertices corresponding to the first ζ rows of H as roots. Suppose that we have formed ζ disjoint isomorphic trees each with k levels. Then at the $(k+1)$-th level, we first add all the edges that are incident to the leaves to each of the ζ trees. If cycles appear, then we remove some of these newly added edges while keeping all the new leaves reachable. We say that a vertex in $N(H)$ connects to a circulant submatrix of H if part of its corresponding row belongs to that circulant submatrix. The cycles of H can be categorized as two types: type-I cycle paths that do not cross the vertices connecting to the same circulant submatrix more than twice, and type-II cycle paths that go through several vertices connecting to the same circulant submatrix before returning to the starting vertex. We call those edges in the same tree as *inner edges* and those connecting different trees as *outer edges*. For type-I cycles, we simply remove the same set of edges (under the isomorphism) from each tree. For type-II cycles, we remove outer edges to break the connected graph into ζ disjoint trees. Due to the QC structure of the code, the isomorphism among the ζ trees is still kept after adding the new edges. Therefore, we obtain ζ disjoint trees of $(k+1)$ levels. The trees grow in this way until all the vertices have been reached. After the spanning trees are built, we let the edges not included in the trees be the information symbols. Subsequently, we can perform parallel sparse encoding as described in Theorem 3.1 over each disjoint tree in parallel.

This encoding procedure leads to a modified version of the base QC code. Note that the check equations corresponding to the ζ root vertices are not necessarily satisfied after the values of all edges incident to these vertices are computed. To offer additional protection on the symbols corresponding to these edges, and also to reduce the number of low weight codewords, we add a simple tree code on top of the QC code. The resulting code is a combination of a QC code and a small tree code. Next, we illustrate the parallel sparse encoding method through an example.

Example 3.1: Consider a mother code whose parity-check matrix H_m and corresponding normal graph are shown in Fig. 2 (a).

In the normal graph, the vertex $c_i, i = 1, \cdots, 6$ corresponds to the i-th row of H_m and $b_i, i = 1, \cdots, 9$ corresponds to the i-th column of H_m. A two-layer spanning tree of the normal graph is shown in bold solid lines. Using this mother code, we can construct a base QC cycle code with $\zeta = 3$ as follows.

The parity-check matrix of the base QC code, denoted by H, is obtained by replacing each "0" in H_m with a 3×3 zero matrix, and replacing each "1" in H_m by a nonbinary circulant permutation matrix. In general, the nonbinary circulant permutation matrices are designed to ensure that the resulting nonbinary code has a large girth. Here, for simplicity, we assume that each "1" in the (i, j)-th position (i-th row and j-th column) of H_m, where $(i, j) \neq (4, 9)$, is replaced by a 3×3 identity matrix times $\delta_{i,j}$, where $\delta_{i,j} \in \text{GF}(q)$, and the "1" located at the $(4, 9)$-th position is replaced by $\delta_{4,9}$ times a 3×3 permutation matrix obtained by cyclically shifting each row of the identity matrix to the right by one position.

The normal graph of H is shown in the lower part of Fig. 2 (b). Note that $\{c_i^1, c_i^2, c_i^3\}$ and $\{b_i^1, b_i^2, b_i^3\}$ correspond to equivalent (under the isomorphism) rows and columns of H. Following the PSE procedure, we can identify $\zeta = 3$ disjoint spanning trees each of which is isomorphic to the spanning tree of the mother code. The i-th tree T_i consists of vertices $\{c_j^i, j = 1, \cdots, 6\}$ with c_1^i as the root, and edges $\{b_1^i, b_2^i, b_3^i, b_7^i, b_8^i\}$. In order to form disjoint trees, the remaining edges in the graph, represented by dash lines, are removed to eliminate cycles. Specifically, the inner edges $\{b_4^i, b_5^i, b_6^i\}$ are removed to eliminate the type-I cycles such as $c_1^i \rightarrow c_2^i \rightarrow c_3^i \rightarrow c_1^i$; and the outer edges $\{b_9^i\}$ are removed to eliminate the type-II cycles such as $c_1^1 \rightarrow c_4^1 \rightarrow c_6^2 \rightarrow c_3^2 \rightarrow c_1^2 \rightarrow c_4^2 \rightarrow c_6^3 \rightarrow c_3^3 \rightarrow c_1^3 \rightarrow c_4^3 \rightarrow c_6^1 \rightarrow c_3^1 \rightarrow c_1^1$. The 12 removed edges correspond to information symbols, from which the values of coded symbols can be computed. Note that without the tree subcode, the three root vertices $\{c_1^1, c_1^2, c_1^3\}$ are not necessarily satisfied for nonbinary codes. Hence, we add four additional coded symbols $\{b_{11}, \cdots, b_{14}\}$ and checks $\{c_7, c_8\}$ to offer stronger protection.

The following steps summarize the PSE procedure executed on Fig. 2 (b). Note that in each of the first four steps, encoding is implemented in parallel for each spanning tree $T_i, i = 1, 2, 3$.

Step 1: Let $\{b_4^i, b_5^i, b_6^i, b_9^i, i = 1, 2, 3\}$ be the information bits.

Step 2: For each $i = 1, 2, 3$,
Compute b_1^i using check constraint at c_2^i: $\delta_{2,1}b_1^i + \delta_{2,4}b_4^i + \delta_{2,6}b_6^i = 0$;
Compute b_3^i using check constraint at c_4^i: $\delta_{4,3}b_3^i + \delta_{4,5}b_5^i + \delta_{4,9}b_9^i = 0$;
Compute b_7^i using check constraint at c_5^i: $\delta_{5,7}b_7^i + \delta_{5,6}b_6^i = 0$;
Compute b_8^i using check constraint at c_6^i: $\delta_{6,8}b_8^1 + \delta_{6,9}b_9^3 = 0$, $\delta_{6,8}b_8^2 + \delta_{6,9}b_9^1 = 0$, $\delta_{6,8}b_8^3 + \delta_{6,9}b_9^2 = 0$.

Step 3: Compute $\{b_2^i, i = 1, 2, 3\}$ using check constraint at c_3^i: $\delta_{3,2}b_2^i + \delta_{3,4}b_4^i + \delta_{3,5}b_5^i + \delta_{3,7}b_7^i + \delta_{3,8}b_8^i = 0$.

Step 4: Compute $\{b_{(11+i)}, i = 1, 2, 3\}$ using check constraint at c_1^i: $b_{(11+i)} + \delta_{1,1}b_1^i + \delta_{1,2}b_2^i + \delta_{1,3}b_3^i = 0$.

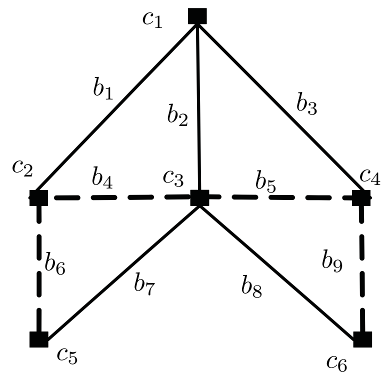
Step 5: Compute b_{11} according to check constraint at c_8 such that $b_{11} + b_{12} + b_{13} + b_{14} = 0$.

D. Encoding complexity of sparse encodable and parallel sparse encodable codes

As discussed above, sparse encoding solves row equations of H sequentially. Its overall complexity is md_c multiplications (denoted by $\otimes$) and $m(d_c - 1)$ additions (denoted by $\oplus$) over GF(q), where d_c is the degree of check nodes. Hence, the encoding process requires $m[d_cT_1 + (d_c - 1)T_2]$ clock cycles, where T_1, T_2 are the clock cycles required for a $\otimes$ and an $\oplus$ over GF(q), respectively. For parallel sparse encoding, the overall encoding time is further reduced to $(m/\zeta)[d_cT_1 + (d_c - 1)T_2]$ for the encoding of QC subcode, plus the encoding time of the tree subcode, which is typically much smaller than the encoding time of the QC subcode. Compared to the generator matrix based encoding scheme, which has a complexity of O(n^2), where n is the code length, the sparse and parallel sparse encoding schemes achieve significant complexity saving since H is a sparse matrix. In [12], Lin *et al.* proposes an encoder for QC LDPC codes which utilizes the QC structure to reduce the density of the generator matrix. This encoder requires $(n - m)[(n - m)/\zeta + 1] \otimes$ and $(n - m)[(n - m)/\zeta] \oplus$ over GF(q). With parallel processing, its encoding time is $\zeta[((n - m)/\zeta + 1)T_1 + ((n - m)/\zeta)T_2]$ clock cycles. In general, since d_c is typically much smaller than ζ, when the code rate is not too low, we will have $d_c < (n/m - 1)\zeta$ so that the proposed parallel sparse encoder has a much lower complexity and shorter encoding time than the encoder in [12]. For instance, for a PSE code constructed from a rate $1/2$ QC GF(256) cycle code with $d_c = 4$, $n = 300$, and $\zeta = 15$, the parallel sparse encoder saves about 60% in complexity and encoding time compared to the encoder in [12]. In [9], a modified PEG algorithm is proposed to construct sparse encodable codes. However, since the codes constructed in [9] are not QC, parallel encoding is not applicable which results in a higher encoding complexity than the proposed parallel sparse encoder.

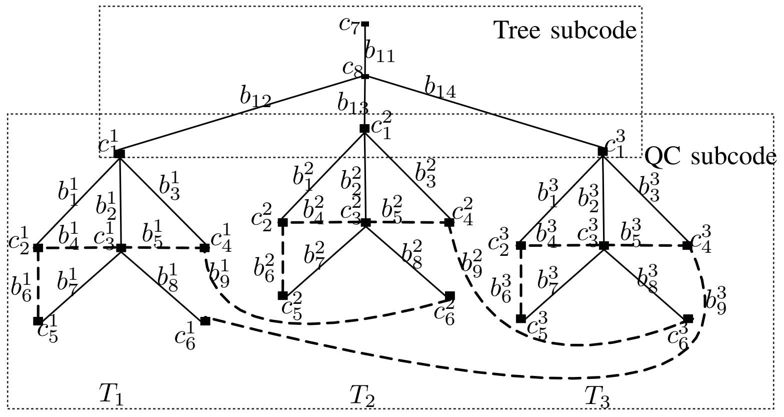
IV. Numerical Results

In this section, we examine the performance of LDPC cycle codes in MIMO channels. We first compare the performance of the proposed PSE cycle codes with those constructed from the PEG algorithm and the QPP algorithm. Then we provide a performance and complexity comparison between nonbinary coded systems employing the PSE codes and binary coded



$$\mathbf{H}_m = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

(a) Normal graph and parity-check matrix of mother code



(b) A PSE cycle code consists of a QC subcode and a tree subcode

Fig. 2. Normal graph representation of a PSE code.

systems employing an optimized binary LDPC code or a QC code in the 802.16e standards.

A. Performance comparisons of different cycle code constructions

We first compare the performance of three LDPC cycle codes over GF(256) constructed using different methods. The PEG code and the QPP code are constructed using the PEG algorithm and the QPP algorithm, respectively, as discussed in Section III-A. The code length is 300 GF(256) symbols and the code rate is $R_c = 1/2$. The PSE code, constructed from the QPP code, consists of a QC subcode and a 5-level tree subcode. It has 165 information symbols, 329 coded symbols, and a rate of 0.502. As discussed in Section III-D, the encoding time of the PEG code and the QPP code, encoded using methods in [9] and [12] respectively, are both much longer than that of the PSE code. Fig. 3 presents the bit-error-rate (BER) and block-error-rate (BLER) performance curves of these three codes for a MIMO channel with two transmit and two receive antennas. The 16 QAM modulation is used. We adopt the same definition of E_b/N_0 as in [21]. The average signal energy per transmitted QAM constellation symbol is E_s/t. Since we assume that the channel matrix has i.i.d. Rayleigh entries and is independent over time, the average signal energy per received antenna is E_s. Hence, the r receive antennas collect total power rE_s, carrying $t \cdot m_0$ coded bits, of $R_c \cdot t \cdot m_0$ information bits. The signal energy per transmitted information bit at the receiver is defined to be $E_b = (r/R_c t m_0) \cdot E_s$, or, expressed in terms of logarithmic SNR measures

$$\left.\frac{E_b}{N_0}\right|_{\text{dB}} = \left.\frac{E_s}{N_0}\right|_{\text{dB}} + 10\log_{10}\frac{r}{R_c t m_0}.$$

To ensure the accuracy of numerical results, we collect at least 100 error blocks for each point in the performance curve. Fig. 3 shows that the QPP code performs slightly better than the PEG code. This may be attributed to the larger girth of the QPP construction. The PSE code, with the shortest encoding time, performs only about 0.05 dB worse than the other codes. This further justifies the effectiveness of PSE codes.

B. Performance comparisons of nonbinary coded systems with binary coded systems

In Fig. 4, we compare the performance of nonbinary and binary LDPC coded systems. The GF(16) PSE code has a code length of 569 symbols (2276 bits) and a rate of 0.502. It is constructed by adding a 5-level tree code to a QC subcode of length 540 and $\zeta = 15$. The GF(256) PSE code has a length of 287 symbols (2296 bits) and a rate of 0.501. It is constructed by adding a 5-layer tree subcode to a QC subcode of length 256 and $\zeta = 16$. These parameters are chosen such that the code lengths are close to that of the rate $1/2$ QC binary LDPC code (2304 bits) defined in the IEEE 802.16e standard [15]. The optimized binary LDPC code has a length of 2304 and

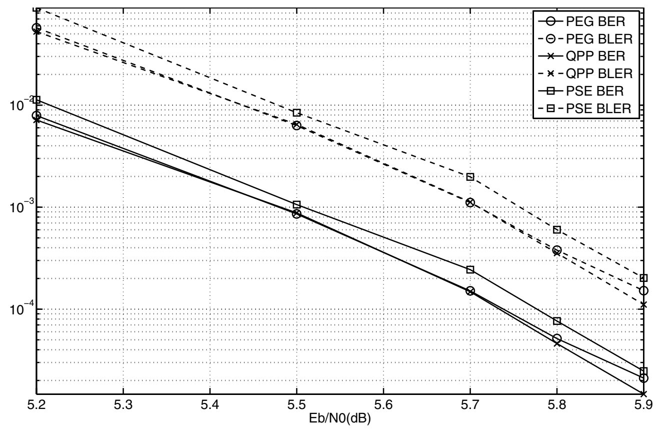


Fig. 3. Performance comparisons of LDPC cycles codes over GF(256). The PEG code and the QPP code are constructed from the PEG and QPP algorithm, respectively. The PSE code is constructed from the QPP code.

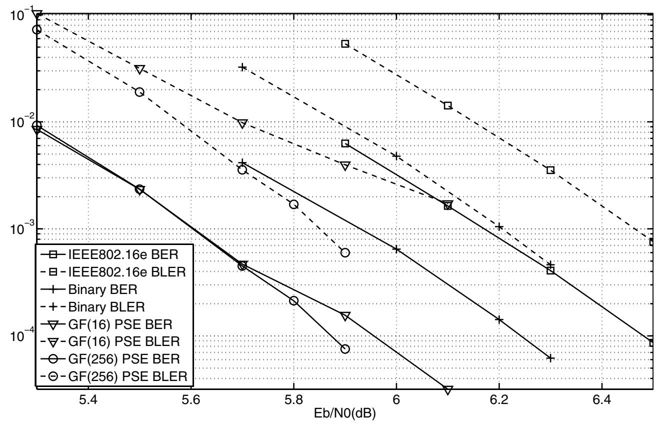


Fig. 4. Performance comparisons of PSE codes over GF(256) (SDD) and GF(16) (JDD), the IEEE 802.16e code (JDD), and the optimized binary LDPC code (JDD). Let d_v and d_c denote the degree sequences of variable nodes and check nodes, respectively. Let $u_v(i)$ denote the fraction of edges that are connected to variables nodes of degree $d_v(i)$, and let $u_c(i)$ denote the fraction of edges that are connected to check nodes of degree $d_c(i)$. The optimized binary LDPC code has parameters $d_v = [2, 3, 7, 8, 23, 24]$, $d_c = [7]$, $u_v = [0.568, 0.298, 0.029, 0.076, 0.012, 0.017]$, $u_c = [1]$. The IEEE 802.16e code has parameters $d_v = [2, 3, 6]$, $d_c = [6, 7]$, $u_v = [0.289, 0.316, 0.395]$, $u_c = [0.632, 0.368]$.

a rate of $1/2$. Its degree distribution is optimized using the EXIT chart approach [14].

Since the complexity of the MIMO detector is much higher than that of each LDPC decoding iteration (a detailed complexity analysis is presented in Section IV-C), for JDD systems employing the GF(2) (binary) and GF(16) codes, we perform multiple super-iterations between the MIMO detector and the channel decoder. In each super-iteration, MIMO detection is performed once followed by five iterations of LDPC decoding. For the SDD system employing the GF(256) code, there is no iterative processing between the MIMO detector and the channel decoder. The decoding process is halted if the decoder converges to a valid codeword or a maximum number of iterations (set to be 40 super-iterations for JDD systems and 150 decoding iterations for SDD systems) is reached.

We consider the same MIMO channel model as described in Section IV-A. Fig. 4 shows that the SDD system employing the PSE GF(256) code performs the best. At BER $= 10^{-4}$, it achieves a performance gain of 0.37 dB compared to the JDD system employing the optimized binary LDPC code. Performance of the IEEE 802.16e code is 0.26 dB worse than that of the optimized binary code and is more than 0.6 dB worse than that of the GF(256) code. We also note that even though the BER curves of the GF(16) code and the GF(256) code are quite close, there is a larger gap between their BLER curves especially at higher SNR. This may be explained as follows. Given the normal graph of a cycle code, one can verify that if there exists a cycle of length w and its parity-check matrix is rank-deficient, then there must exist a codeword of weight w. For cycle codes over smaller Galois fields, there is larger probability that the parity-check matrix of a cycle is rank-deficient when the nonzero elements of H are picked randomly. Hence, there exists many low weight codewords corresponding to the short cycles in the graph and the decoder is more likely to converge to the wrong codeword resulting in undetected errors. We observe that, at 6.1 dB, among the 100 error blocks we collected, 74 error blocks are undetected errors due to low weight codewords. This explains why the GF(16) code does not perform as well in terms of BLER.

C. Complexity analysis of nonbinary coded systems and binary coded systems

In this section, we compare the receiver complexity of nonbinary and binary systems considered in Section IV-B. The basic operations involved are $\oplus$, $\otimes$, logarithm, and exponential. Here we consider two types of $\oplus$ operations: the regular $\oplus$ in the non-logarithmic domain and the sign/logarithmic number system (LNS) $\oplus$ in the logarithmic domain [22]. For the decoding of nonbinary LDPC codes, we refer to the low complexity algorithm in [3] where detailed complexity computations of the decoding algorithm are presented.

We first examine the complexity of the binary system. The complexity of the MIMO detector is computed based on equation (2). The computation of terms $\|\mathbf{y} - \mathbf{H}\phi(\mathbf{s})\|^2$ for all $2^{tm_0} = 2^8$ possible values of $\mathbf{s}$ requires 672 $\otimes$ and 704 $\oplus$ per bit. The bitwise LLR is then computed by performing $2 \times (2^{tm_0}/2 - 1) = 254$ LNS $\oplus$ and 2 regular $\oplus$ per bit. From [3], we find that the binary LDPC decoder requires $75 \times 5 = 375$ $\oplus$ and $34 \times 5 = 170$ LNS $\oplus$ for the five decoding iterations in each super-iteration. Therefore, the complexity of the binary system per super-iteration is $704 + 375 + 2 = 1081$ $\oplus$, 672 $\otimes$, and $254 + 170 = 424$ LNS $\oplus$. Similarly, the complexity of the GF(16) system per super-iteration is 1368 $\oplus$, 672 $\otimes$, 60 LNS $\oplus$ and 320 log/exp. For the GF(256) system, the MIMO detector requires 672 $\otimes$ and 618 $\oplus$ per bit, and the LDPC decoder requires 2176 $\oplus$ and 512 log/exp per bit per iteration [3]. Table I compares the receiver complexity of different systems at BER $= 10^{-4}$. Since an LNS addition requires three $\oplus$ and one look-up table (LUT), and a log/exp (implemented using look-up table) requires one LUT, the overall complexity is counted only in terms of the number of $\otimes$, $\oplus$, and LUTs. We note that since MIMO detection is performed only once in a SDD system, the overall number of $\otimes$ required by the GF(256) system is only $1/4$ of that of the binary system. Even though the GF(256)

TABLE I

COMPLEXITY COMPARISON OF NONBINARY SYSTEMS AND OPTIMIZED BINARY SYSTEM AT BER=1.0×10^{-4}.

	E_b/N_0 (dB)	average # of iter.	total # of multiplication $\otimes$	total # of addition $\oplus$	total # of LUT
GF(2)	6.24	**4** super-iter.	672 * 4 = **2688**	(1081 + 60*3)*4 = **9412**	424*4=**1696**
GF(16)	5.96	**3** super-iter.	672 * 3 = **2016**	(1368 + 60*3)*3 = **4644**	(60+320)*3=**1140**
GF(256)	5.87	**8** decoding iter.	672 * 1 = **672**	(1081 + 424*3)*8 = **18026**	512*8= **4096**

system requires more $\oplus$ and LUTs, its simulation time is comparable to that of the binary system because the latter has more $\otimes$ and each $\otimes$ requires more clock cycles than does an $\oplus$ or a LUT. Hence, the GF(256) system achieves a performance gain of about 0.37 dB over the optimized binary system with a comparable complexity. Table I shows that the GF(16) system also outperforms the binary system at a reduced complexity.

V. CONCLUSION

In this paper, we propose a class of nonbinary LDPC cycle codes for MIMO channels which demonstrates superior performance than the best optimized binary LDPC code. By exploiting the QC structure of nonbinary cycle codes, a novel parallel sparse encoding method is developed to facilitate parallel implementation in addition to linear-time encoding. Compared to the widely studied JDD systems, we show that best performance can be achieved by a SDD system employing a simple PSE cycle code over GF(256). Through explicit performance and complexity comparisons with binary systems, we conclude that the proposed PSE nonbinary cycle codes are good candidates for MIMO channels.

REFERENCES

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

[2] M. C.Davey and D. Mackay, "Low-density parity check codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, pp. 165–167, June 1998.

[3] R. Peng and R.-R. Chen, "Application of nonbinary LDPC codes for communication over fading channels using higher order modulations," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM'06)*, Nov. 2006.

[4] J. Boutros, A. Ghaith, and Y. Yuan-Wu, "Non-binary adaptive LDPC codes for frequency selective channels:code construction and iterative decoding," in *Proc. IEEE Information Theory Workshop*, Chengdu, China, Oct. 2006, pp. 184–188.

[5] G. J. Byers and F. Takawira, "Nonbinary and concatenated LDPC codes for multiple-antenna transmission," in *Proc. 7th AFRICON Conference in Africa*, vol. 1, Sept. 2004, pp. 83–88.

[6] F. Guo and L. Hanzo, "Low complexity non-binary LDPC and modulation schemes communicating over MIMO channels," in *Proc. IEEE Vehicular Technology Conference (VTC'04)*, vol. 2, Sept. 2004, pp. 1294–1298.

[7] R. Peng and R.-R. Chen, "Design of nonbinary LDPC codes over GF(q) for multiple-antenna transmission," in *Proc. IEEE Military Comm. Conf. (MILCOM'06)*, Washington DC, Nov. 2006.

[8] O. Alamri, F. Guo, M. Jiang, and L. Hanzo, "Turbo detection of symbol-based non-binary LDPC-coded space-time signals using sphere packing modulation," in *Proc. IEEE Vehicular Technology Conference (VTC'05)*, vol. 1, Dallas, Texas, Sept. 2005, pp. 540–544.

[9] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inform. Theory*, vol. 51, pp. 386–398, Jan. 2005.

[10] S. Hakimi and J. Bredeson, "Graph theoretic error-correcting codes," *IEEE Trans. Inform. Theory*, vol. 14, no. 4, pp. 584–591, July 1968.

[11] W. W. Peterson and E. J. Weldon, *Error-Correcting Codes*. Cambridge, MA: MIT Press, 2nd ed 1972.

[12] Z. Li, L. Chen, L. Zeng, S. Lin, and W. H. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 54, no. 1, pp. 71–81, Jan. 2006.

[13] S. Lin, S. Song, L. Lan, L. Zeng, and Y. Y. Tai, "Constructions of nonbinary quasi-cyclic LDPC codes: a finite field approach," in *Proc. ITA Workshop 2006*.

[14] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Trans. Commun.*, vol. 52, pp. 670–678, Apr. 2004.

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

[16] O. Y. Takeshita, "A compact construction for LDPC codes using permutation polynomials," in *Proc. IEEE International Symposium on Information Theory (ISIT'06)*, July 2006, pp. 79–82.

[17] ——, "A new construction for LDPC codes using permutation polynomials over integer rings," submitted to *IEEE Trans. Inform. Theory*, June 2005.

[18] J. Forney, "Codes on graphs: normal realizations," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 520–548, Feb. 2001.

[19] R. Diestel, *Graph Theory*, 3rd ed., S. Axler and K. Ribet, eds. Springer, 2005.

[20] J. Lu, J. M. F. Moura, and H. Zhang, "Efficient encoding of cycle codes: a graphical approach," in *Proc. Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, vol. 1, Nov. 2003, pp. 69–73.

[21] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[22] E. E. Swartzlander, Jr., D. V. S. Chandra, H. T. Nagle, Jr., and S. A. Starks, "Sign/logarithm arithmetic for FFT implementation," *IEEE Trans. Comput.*, vol. C-32, pp. 526–534, June 1983.