Abstract—A novel algebraic construction technique for LDPC convolutional codes (LDPCCCs) based on permutation polynomials over integer rings is presented. The underlying elements of this construction technique are graph automorphisms and quasi-cyclic (QC) codes. The algebraic structure of the obtained LDPCCCs, their encoding and decoding are discussed. These new codes have a special structure, which is favorable for high-rate VLSI implementations. Their performances in the asymptotic case are superior to that of the QC codes from which they are derived at a much lower complexity. 

I. INTRODUCTION

Low-Density Parity-Check (LDPC) codes were invented by Gallager [1] in the early 60’s. The convolutional counterparts of Gallager’s LDPC block codes, namely LDPC convolutional codes (LDPCCCs), were proposed by Jiménez-Felstström and Zigangirov [2]. Contrary to LDPC block codes, a single LDPCCC can be applied to a multitude of frame lengths because of its inherent convolutional structure. Moreover, a pipelined version of the conventional message-passing decoder can be derived. This pipelined decoder alleviates the typical on-chip interconnection problems [3] and has some other advantages regarding its VLSI design. In [4] it was shown that for a comparable BER performance the complexity of a decoder for LDPCCCs is much lower than the complexity of a decoder for block codes. Additionally, LDPCCCs can be easily encoded using shift-registers.

The original construction technique for LDPCCCs presented in [2] consists of unwrapping the parity-check matrices of LDPC block codes. The codes obtained through this technique are of semi-random nature. This randomness presents some disadvantages in terms of storing and accessing the parity-check matrices, encoding data and analyzing the codes. The insertion of an algebraic structure in the LDPCCCs can be exploited to overcome some of these problems and, also, for even more efficient VLSI designs (e.g. parallelization methods, memory architectures, scheduling, etc.).

Several methods for designing structured LDPCCC block codes have been proposed recently (see [5], [6] and the references therein). Among these methods, the quasi-cyclic (QC) constructions proposed by Tanner et al. [5] are particularly interesting because they allow a direct mapping into convolutional codes [7]. However, QC codes with circulant matrices are known to have poor distance properties for higher block lengths. This fact is also reflected in the corresponding convolutional codes [7]. In this context, Takeshita [8] proposed a new code construction method, where the parity-check matrices are defined by permutation polynomials over integer rings. The obtained parity-check matrices (graphs) have automorphisms that can be exploited to rearrange them in a generalized circulant fashion. Although not yet formally proven, the minimum distances of these QC codes seem to grow with the block length. LDPCCCs obtained from this class of QC codes have not been previously considered in the literature and are the subject of this paper.

The remainder of this paper is organized as follows. Section II explains how convolutional codes can be obtained from QC block codes. In Section III, we show how to obtain QC block codes from a class of permutation polynomials over integer rings. The LDPCCCs derived from these QC block codes are defined in Section IV. In the same section, issues like encoding, decoding, scalability and termination of these novel LDPCCCs are discussed. Section V presents several results on performance for our codes. Finally, we summarize our findings in Section VI.

II. LDPC CONVOLUTIONAL CODES FROM QC CODES

LDPC convolutional codes are derived from QC codes by replicating their constraint graphs to infinity [5]. This can be expressed by representing the circulants in the parity-check matrix of a QC code by polynomials. Then, the polynomial parity-check matrix of the corresponding LDPCCC has the form

\[ \mathbf{H}(D) = \begin{bmatrix} p^{(1,1)}(D) & \cdots & p^{(1,K)}(D) \\ \vdots & \ddots & \vdots \\ p^{(J,1)}(D) & \cdots & p^{(J,K)}(D) \end{bmatrix} \]  

(1)

where \( p^{(j,k)}(D) = \sum_{i=1}^{I(j,k)} D^{e(i,j,k)} \). The \( I(j,k) \) monomial entries in \( p^{(j,k)}(D) \) correspond to the identity matrices cyclic-shifted by \( e(i,j,k) \) positions, which are present in each circulant of the QC parity-check matrix.

III. PERMUTATION POLYNOMIALS AND QC CODES

Let \( G = (V \cup C \cup E) \) be a regular bipartite graph with vertex degrees \((\lambda, \rho)\). The sets \( V = \{v_1, \ldots, v_N\} \), \( C = \{c_1, \ldots, c_{N-M}\} \), and \( E = \{e_1, \ldots, e_L\} \), where \( L = N\lambda = (N-M)\rho \), represent the variable nodes, the check

[1] This work was supported by the German ministry of research and education within the project Wireless Gigabit With Advanced Multimedia Support (WIGWAM) under grant 01 BU 370.
nodes and the edges, respectively. Next, we define the sets \( \mathbb{R} = \{r_1, \ldots, r_L\} \) and \( \mathbb{L} = \{l_1, \ldots, l_L\} \), which represent the right and left labels of the edges in \( \mathbb{E} \), respectively. In addition, each \( v_i \) (\( c_i \)) is associated to a subset of \( \mathbb{R} (\mathbb{L}) \), namely \( \mathbb{R}_i = \{r_{(i-1)\lambda+1}, \ldots, r_{(i-1)\lambda+\lambda}\} \) (\( \mathbb{L}_i = \{l_{(i-1)\rho+1}, \ldots, l_{(i-1)\rho+\rho}\} \)). A permutation is defined by a bijective function \( f: \mathbb{R} \rightarrow \mathbb{L} \). In this sense, \( f \) defines the graph connections and, consequently, the parity-check matrix of the code.

In this work we are interested in permutations, which can be expressed through polynomials. More specifically, we will only consider quadratic permutation polynomials (QPP's) over integer rings with the simple form

\[
 f(x) = ax + bx^2. \tag{2}
\]

The conditions that the coefficients \( a \) and \( b \) must satisfy to make \( f \) a permutation polynomial over the ring \( \mathbb{Z}_L = \{0, \ldots, L-1\} \), as well as, the search algorithm for coefficients that maximizes the graph’s girth are described in [8]. As mentioned by Takeshita, the nature of the permutation polynomials results in graph automorphisms. The disjoint sets of variable nodes \( \mathbb{V}_i = \{v_i, v_{i+\beta}, \ldots, v_{i+\frac{N}{\beta} - 1}\} \) with \( i = 0, \ldots, N - \frac{N}{\beta} \) and check nodes \( \mathbb{C}_j = \{c_j, c_{j+\gamma}, \ldots, c_{j+\frac{(N-M)}{\delta} - 1}\} \) with \( j = 0, \ldots, (N - M) - \frac{(N-M)}{\gamma} \) represent the equivalence classes of the graph under its automorphisms. The parameter \( \beta \) is defined in [8] and \( \gamma = \frac{\beta}{\delta} \).

Observing the regularity in the distributions of graph nodes that pertain to any equivalence class, one can conclude that the parity-check matrix \( \mathbf{H} \) of the resulting code is highly structured. Effectively, if we reorder the columns of \( \mathbf{H} \) according to the permutation \( \mathbb{P}_x = \{0, \beta, \ldots, N - \beta, 1, 1 + \beta, \ldots, 1 + N - \beta, \ldots, \beta - 1, \beta - 1 + \beta, \ldots, \beta - 1 + N - \beta, \beta - 1\} \) and the rows according to the permutation \( \mathbb{P}_y = \{0, 1, \ldots, N - M - \gamma, 1 + \gamma, \ldots, 1 + N - M - \gamma, \gamma - 1, \gamma - 1 + \gamma, \ldots, \gamma - 1 + N - M - \gamma, \gamma - 1\} \), we obtain a generalized circulant matrix. Figure 1 exemplifies this phenomenon for a regular \((3,6)\)-code with dimensions \((N = 1008, M = 504)\). This code was obtained from the permutation polynomial \( f(x) = 29x + 42x^2 \) with \( \beta = 12 \). It is easy to see that the size of the circulants is given by \( S_c = N/\beta = 84 \). Moreover, computations have shown that the girth of this code is \( g = 8 \) and the minimum distance is \( d_{\text{min}} = 44 \) [8].

IV. LDPC CONVOLUTIONAL CODES FROM PERMUTATION POLYNOMIALS

We use the QC structure of the matrices obtained through polynomial permutation to generate LDPCCCs. In general, the circulants in these matrices are in factorized form. This means that for a given code sequence \((\cdots, v_i, v_{i+1}, v_{i+2}, \cdots)\), the parity-check equations represented by the circulant-factors contain disjoint sequences of code symbols given by \((\cdots, v_i, v_{i+\delta}, v_{i+2\delta}, \cdots)\), where \( \delta \) is the distance between the symbols checked by a circulant-factor and, simultaneously, the number of circulant-factors that compose the original circulant. The LDPCCC obtained from this generalized circulant matrices code will have basically the same form of (1), but now the entries of \( \mathbf{H}(D) \) will be vectors of length \( \delta \) given by

\[
 p^{(j,k)}(D) = \left[ \sum_{i=1}^{I_{(j,k,1)}} D_1^{e(i,j,k,1)}, \ldots, \sum_{i=1}^{I_{(j,k,\delta)}} D_\delta^{e(i,j,k,\delta)} \right]^T, \tag{3}
\]

where the subindexes of \( D \) stand for the disjoint sequences of code symbols that are used by each element of \( p^{(j,k)}(D) \).

A. Encoding

A block code that has a parity-check matrix \( \mathbf{H} \) composed of circulants will have a systematic generator matrix also composed of circulants if it can be written as

\[
 \mathbf{H} = [\mathbf{A} \ \mathbf{\Gamma}]. \tag{4}
\]

In the equation above, \( \mathbf{H} \) is \((n-k) \times n\) and \( \mathbf{\Gamma} \) is an \((n-k) \times (n-k)\) invertible square matrix. The conventional method for constructing the generator matrix \( \mathbf{G} \) is to find an \((n-k) \times (n-k)\) matrix \( \Phi \) such that \( \Phi \mathbf{\Gamma} = \mathbf{I}_{(n-k)} \), where \( \mathbf{I}_{(n-k)} \) is the
Fig. 2. Generator matrix in systematic circulant form

\[(n - k) \times (n - k)\) identity matrix. Then, the generator matrix will have the following form

\[G = [I_k \ (\Phi \Lambda)^T]. \quad (5)\]

As it might be conjectured, not all circulant matrices \(H\) will have \(\Gamma\) in an invertible form. For instance, in Figure 1(b) the last \(S_c\) rows of \(\Gamma\) are zero. In this case, the generator matrix \(G\) cannot be quasi-cyclic. To circumvent this problem, we can cyclically shift \(S_c\) columns of \(H\) to the right until we obtain a \(\Gamma\) that has full rank.

Figure 2 shows the systematic generator matrix obtained from the parity-check matrix in Figure 1(b) by cyclic shifting it 3 times (in each time we shift \(S_c\) columns, i.e., a column of circulants) to the right. As we can see, the generator matrix is not low-density. However, it exhibits the desired circulant structure.

The encoder for the LDPCCC is directly obtained from the generator matrix of the QC code by translating its circulants into polynomials, as explained in Section II for the parity-check matrix. Although the generator matrix of the underlying QC code is not low-density and, consequently, the polynomial generator matrix \(G(D)\) of the LDPCCC contains lots of terms, the encoders for both codes are very practical for a hardware implementation. This feature is due to the highly structured generator matrices that allow efficient implementations based on shift-registers and XOR operations. Consequently, we can expect that high-rate implementations of these encoders are feasible with the current VLSI technology.

**B. Decoding**

Decoding of LDPCCCs can be performed using the pipeline decoder [2]. An example of this decoder for a convolutional code of memory \(m_s = 3\) and graph degrees \((\lambda = 2, \rho = 3)\) is shown in Figure 3. The symbols from the channel are continuously decoded. At each time, \(\rho = 3\) symbols enter (leave) the decoder. The \(I\) processors represent the iterations. They are identical and operate in parallel. In each processor, only the \(\rho + \lambda = 5\) highlighted graph vertices and the thick edges are active each time.

The decoders for LDPCCCs from generalized circulant matrices will have basically the same complexity and structure of Figure 3. But now, because of the parity-checks on disjoint symbol sequences, \(\delta \cdot \rho\) symbols enter (leave) the decoder each time. Thus, the LDPCCCs obtained from QC codes with generalized circulant matrices permit decoders with a new degree of parallelism \((\delta)\), which is an important feature when considering high-rate implementations.

**C. Scalability**

We can obtain shorter QC block codes from an existing mother QC code by shrinking its circulants. This operation on the mother code can be easily performed by the modulo operator. In this case, we choose a new circulant size \(S_N\) (\(S_N\) is smaller than the circulant size of the mother code) and calculate the shifts of the circulants for the new code with

\[e_N(i, j, k) = e_M(i, j, k) \mod S_N, \quad (6)\]

where \(e_N(i, j, k)\) and \(e_M(i, j, k)\) are respectively the shifts of the circulants for the new and mother codes, using the same terminology as in (1).

Naturally, the new convolutional codes associated with the new QC codes with circulant size \(S_N\) will have lower memories and are derived using the same ideas of Section II.

**D. Termination**

In order to perform optimal decoding using the pipeline decoder [2], the graphs underlying the codewords of the LDPCCCs have to be terminated. The termination is the process that brings the convolutional encoder back to the zero state. When the transmitted codeword is a terminated sequence, the soft-decoder can assume that all the symbols after the last effectively transmitted symbol are zero with infinite reliability. In the case of systematic convolutional codes, the termination consists in a sequence of zeros that is fed into the encoder. The length of this termination sequence \(L_T\) is determined by the memory \(m_s\) of the systematic convolutional code and also by the number of lines \(k\) of \(G(D)\). Actually, we can write \(L_T = k \cdot m_s\).
Because the termination sequence is not user data, it causes the so-called rate-loss. In this case, the rate of the terminated code will be
\[ R_T = \frac{L \cdot R - L_T}{L}, \]
where \( L \) is the length of the terminated codeword and \( R \) is the rate of the non-terminated (or infinite) codeword. As a matter of fact, for very large codewords \( (L \gg k \cdot m_s) \) the rate-loss is negligible.

V. SIMULATION RESULTS

Figure 4 shows the BER curve for the non-terminated LDPC convolutional code (LDPCCC) obtained from the QPP code in Figure 1. As a comparison, the BER curves for the regular QPP LDPC block code (QPP LDPCBC) code from Figure 1 and the regular \((3, 6)\) PEG-code [10] with block length \( N = 1008 \) and girth \( g = 8 \) are also shown.

As we can observe, the LDPCCC outperforms the other codes. At a BER = \( 10^{-5} \) the convolutional code has a gain of more than 1 dB. It is important to mention that this convolutional code has memory \( m_s = 83 \). Hence, it uses a decoder, which is much simpler than the decoders for block codes with \( N = 1008 \) [4].

Figure 5 shows the effect of rate-loss expressed in terms of BER for the LDPCCC with memory \( m_s = 83 \). As we can observe, for lower block lengths the terminated code performs bad. For block lengths above \( N = 4320 \), the terminated LDPCCC performs better than the QPP LDPCBC with \( N = 1008 \). Although we are comparing codes with different block sizes, the implementation complexity for the LDPCCC in question is still very low comparing to the LDPCBC with \( N = 1008 \).

In Figure 6 we study the LDPCCCs with reduced memories using the ideas of Section IV-C. As we can observe, the code with memory \( m_s = 41 \) has a performance similar to the mother code with memory \( m_s = 83 \). On the other side, the code with memory \( m_s = 20 \) presents an error-floor. We conjecture that this effect possibly occurs because the shrinking of the circulants of the parity-check matrix may result in a new code that has lots of short cycles in its graph and thus it is not appropriate for decoding using the pipeline decoder (message-passing algorithm).

Reducing the memory of the LDPCCC results in lower implementation complexity and also in lower rate-loss due to termination. Figure 7 shows a comparison that involves code memory and block length for terminated LDPCCCs. It can be observed that codes with lower memory show better results concerning the tradeoff performance vs. rate-loss for smaller block lengths, while codes with higher memory perform better with bigger block lengths. For instance, the terminated LDPCCCs with memories below \( m_s = 41 \) and block length \( N = 2160 \) are at least as good as the LDPCBC with \( N = 1008 \). From this we can conclude that for a given system with certain range of block lengths, its is an optimization problem to find the codes in a certain memory range that give the best performances considering the termination losses.
In this paper we have presented a new class of LDPC convolutional codes, which are derived from QC codes defined by permutation polynomials. These new codes can be encoded using shift-registers and, because of their highly structured graphs, they can be decoded using a slightly modified version of the pipeline decoder [2]. This modified pipeline decoder allows a new degree of parallelism that depends upon the number of factors (δ) the circulants of the parity-check matrix of the associated QC code are decomposed.

In addition, we showed that the generator matrix of the QC generator for the LDPCCC can be derived from it. Moreover, we presented a method to obtain codes with lower memories from a given mother code. The termination problem was also approached and we examined how the performance of the codes behaves with regard to the block length and code memory.

Finally, we would like to point out that further research can be done in the direction of finding optimal (performance vs. rate-loss) LDPCCCs from permutation polynomials for a given range of block lengths. Within this context, the design of irregular LDPCCCs is also an open issue.

VI. CONCLUSIONS

In this paper we have presented a new class of LDPC convolutional codes, which are derived from QC codes defined by permutation polynomials. These new codes can be encoded using shift-registers and, because of their highly structured graphs, they can be decoded using a slightly modified version of the pipeline decoder [2]. This modified pipeline decoder allows a new degree of parallelism that depends upon the number of factors (δ) the circulants of the parity-check matrix of the associated QC code are decomposed.

In addition, we showed that the generator matrix of the QC codes can be obtained in circulant form and thus a polynomial generator for the LDPCCC can be derived from it. Moreover, we presented a method to obtain codes with lower memories from a given mother code. The termination problem was also approached and we examined how the performance of the codes behaves with regard to the block length and code memory.

Finally, we would like to point out that further research can be done in the direction of finding optimal (performance vs. rate-loss) LDPCCCs from permutation polynomials for a given range of block lengths. Within this context, the design of irregular LDPCCCs is also an open issue.

REFERENCES