

Encoders for Block-Circulant LDPC Codes

Kenneth Andrews, Sam Dolinar, and Jeremy Thorpe
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, USA
Email: {andrews, sam, jeremy}@shannon.jpl.nasa.gov

Abstract—In this paper, we present two encoding methods for block-circulant LDPC codes. The first is an iterative encoding method based on the erasure decoding algorithm, and the computations required are well organized due to the block-circulant structure of the parity check matrix. The second method uses block-circulant generator matrices, and the encoders are very similar to those for recursive convolutional codes. Some encoders of the second type have been implemented in a small Field Programmable Gate Array (FPGA) and operate at 100 Msymbols/second.

I. INTRODUCTION

Recently, block-circulant LDPC codes have been found that provide both excellent error correction performance and well structured decoder architectures. Constructions have been presented by Lin *et al* [1], [2], Tanner *et al* [3], [4], Milenkovic *et al* [5], the authors [6], and others. In this paper, we explore some encoder designs for these codes, and discuss a hardware encoder implementation.

We define a circulant as a square binary matrix where each row is constructed from the previous row by a single right cyclic shift; we do not require that each row has Hamming weight 1. An $rT \times nT$ parity check matrix H can be constructed by concatenating $r \times n$ sparse circulants of size $T \times T$. The density of each circulant matrix is indicated by the corresponding value in an $r \times n$ base matrix H_{base} . The Tanner graph corresponding to this matrix is called a protograph [7]. Entries greater than 1 in the base matrix correspond to multiple edges in the protograph. Base matrices can be expanded into block-circulant LDPC codes by replacing each entry in H_{base} with a circulant containing rows of the specified Hamming weight; the resulting codes are quasicyclic. Alternatively, they can be expanded into less structured codes by replacing each entry with a sum of arbitrary permutation matrices.

Protographs for our AR3A and AR4A codes of rate 1/2 are shown in Figures 1 and 2, and we use these as examples throughout the paper. Squares are parity check nodes and circles are variable nodes, where the solid circles represent transmitted symbols and the open ones are punctured. These designs were derived from a three step encoding procedure: accumulate, repeat-by-3 (or 4), and accumulate [8]; hence their names. Each protograph describes a 3×5 block-circulant parity

check matrix, and the number of parallel edges shows the degree of the corresponding circulant.

In practice, these protographs cannot be directly expanded into block-circulant codes without introducing low weight codewords, regardless of the choice of circulants. A practical solution is to expand the protographs twice, first with small permutation matrices, such as of size 4×4 or 8×8 , and then with circulants to build the full code. The result is a parity check matrix such as the one shown in Figure 3 for a very small AR4A code, where each nonzero entry in the matrix is represented by a dot. This code was constructed by putting the AR4A protograph variable nodes in the order (4, 2, 1, 5, 3) and check nodes in order (A, B, C) as demarcated by the solid lines, expanding with 4×4 permutations, and then expanding with 16×16 circulants. The resulting 12×20 block-circulant structure is emphasized by dotted lines.

In Section II, we examine iterative encoders, similar to those of Richardson and Urbanke in [9], that also take advantage of the block-circulant structure of the parity check matrix. In Section III, we examine the existence and construction of systematic block-circulant generator matrices. In Section IV, we develop simple hardware circuits for encoders using block-circulant generator matrices, and describe an FPGA implementation. Concluding remarks are in Section V.

II. ITERATIVE ENCODERS

An encoder for any (N, K) LDPC code can be built from an erasure correcting decoder. A set of K linearly independent variable nodes are selected as the systematic symbols, and these are initialized with the K information bits to be encoded. If there are no stopping sets, then the remaining $N - K$ parity symbols are computed iteratively with the standard erasure

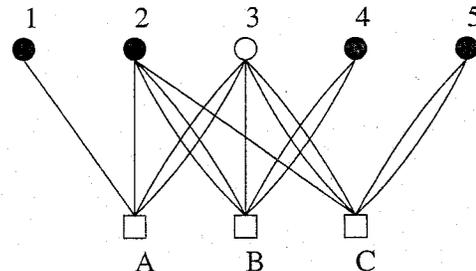


Fig. 1. The AR3A protograph

¹This work was funded by the IND Technology Program and performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

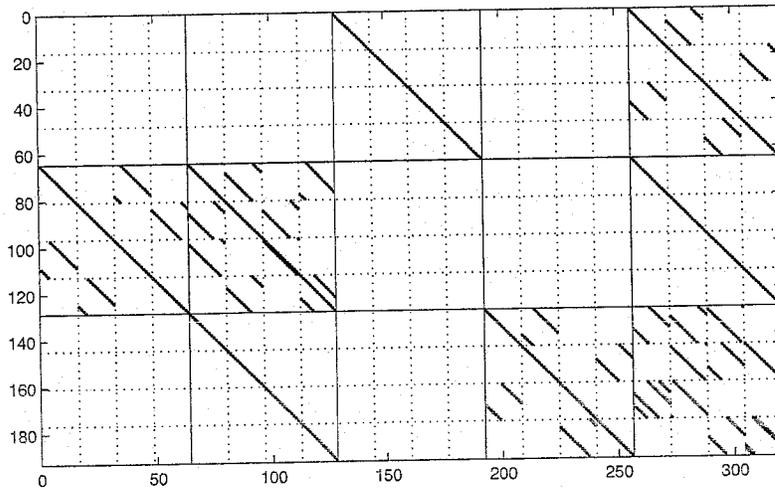


Fig. 3. A Block-Circulant Parity Check Matrix Built From the AR4A Protograph

circulants in S . It is not hard to show that S is invertible if and only if $\det(s)$ is not a zero divisor. Moreover, when S is invertible, there exists an $r \times r$ polynomial matrix w such that $ws = I_r$, the $r \times r$ identity (polynomial) matrix.

Our particular interest is in LDPC codes defined by a block matrix H composed of circulants. Let H have size $rT \times nT$, where $r < n$. A quasicyclic code is one for which a "quasicyclic shift" of a codeword is also a codeword. That is, if we partition any codeword c into binary strings of length T , and circularly shift each string by the same amount, the resulting vector is also a codeword. It is immediate that any LDPC code defined by a block-circulant H matrix is quasicyclic.

In some cases, such a code has a systematic generator matrix G of size $(n-r)T \times nT$ that is entirely composed of circulants. To show this, we partition H so that $H = [Q \ S]$, where S is square. If S is invertible, the traditional method for constructing G is to find a matrix W such that $WS = I_{rT}$, the $rT \times rT$ identity matrix. Then a systematic generator matrix is $G = [I_{(n-r)T} \ (WQ)^T]$. Algebraically, we can perform the same construction in the isomorphic polynomial ring. After finding the $r \times r$ matrix w such that $ws = I_r$, it is a simple matter to compute $g = [I_{n-r} \ q(x^{-1})w(x^{-1})]$. Replacing the elements of this array with the corresponding circulants, we have a block-circulant generator matrix G for the original code.

Not all block-circulant LDPC codes have block-circulant generator matrices. As a particularly small example, suppose H is described by the single circulant $1 + x + x^3$ with $T = 7$. As noted above, this only has rank 4. One codeword is $[1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0]$, and because the code is quasi-cyclic (in fact cyclic, because H consists of a single circulant), all cyclic shifts of this codeword are also codewords. However, the circulant corresponding to $1 + x + x^2 + x^4$ only has rank 3, and so cannot be used in its entirety as a generator matrix.

In general, if S is not full rank (or equivalently, $\det(s)$ is a zero divisor), then G cannot be quasicyclic.

In the remainder of this section, we return to the AR3A and AR4A codes introduced earlier as practical examples.

The $3T \times 5T$ parity check matrix for AR3A is full rank, and so a generator matrix for this code will have dimension $2T$. We partition H into $[Q \ S]$, where Q contains the columns we wish to make systematic, and S is the square matrix that must be invertible. If we choose Q to include the circulants corresponding to variable nodes 4 and 2 in the protograph, as we did for the iterative encoder, we find that S has rank $rT - 1$, deficient by 1. This misfortune occurs because of the closed loop of degree-2 variable nodes created by protograph nodes 5 and C.

Alternatively, we can choose to make protograph variable nodes 4 and 5 systematic. In this case, S has full rank, and a systematic block-circulant G can be calculated exactly as described. An encoder that performs matrix multiplication by G is particularly suitable for hardware implementation as described in the next section.

As a second example, we look at the AR4A code. For this code, there is no set of R columns that can be selected from H to form an invertible square matrix S , because H itself is rank deficient by 1. Perhaps remarkably, these two defects cancel and the method for constructing G can proceed with minor modifications. We select variable nodes 4 and 2 to be systematic, and when H is arranged to put these on the left, it appears as shown in Figure 3. The left two fifths of H is the matrix Q , and the remaining square portion on the right is S . We solve to find codewords of the form $c_4 = [1 \ 0 \ p_1(x) \ p_5(x) \ p_3(x)]$, and of the form $c_2 = [0 \ 1 \ p_1(x) \ p_5(x) \ p_3(x)]$. By expanding these solutions into circulants, we can form a block-circulant

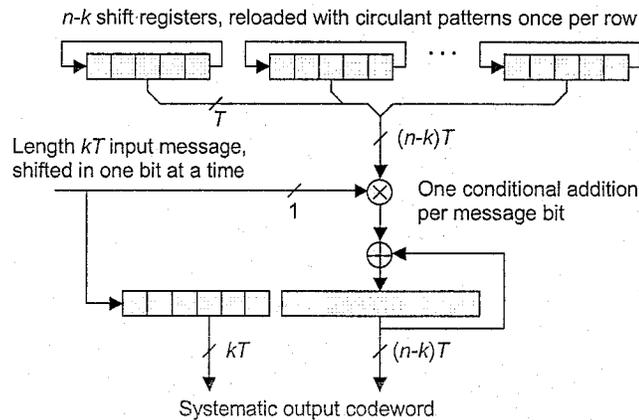


Fig. 5. The Direct Implementation of a Quasicyclic Encoder

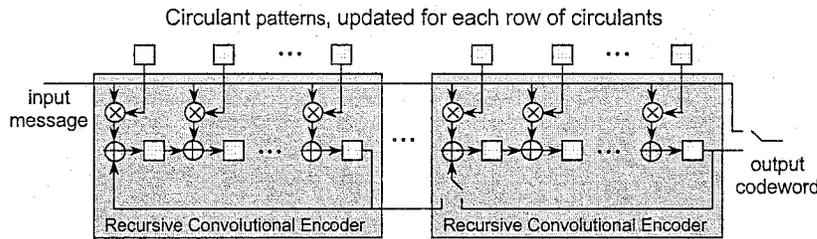


Fig. 6. A Quasicyclic Encoder Using Feedback Shift Registers

these LDPC codes often possess. Such an encoder requires remarkably little hardware, and provides a fast, simple, bit-serial architecture. We have implemented these encoders in a small FPGA operating at 100 Msymbols/second.

REFERENCES

- [1] Y. Kou, H. Tang, S. Lin, and K. Abdel-Ghaffar, "On Circulant Low Density Parity Check Codes," in *IEEE International Symposium on Information Theory*, p. 200, June 2002.
- [2] S. Lin, "Quasi-Cyclic LDPC Codes." CCSDS working group white paper, Oct. 2003.
- [3] R. M. Tanner, "On Graph Constructions for LDPC Codes by Quasi-Cyclic Extension," in *Information, Coding and Mathematics* (M. Blaum, P. Farrell, and H. van Tilborg, eds.), pp. 209–220, Kluwer, June 2002.
- [4] A. Sridharan, D. Costello, and R. M. Tanner, "A Construction for Low Density Parity Check Convolutional Codes Based on Quasi-Cyclic Block Codes," in *IEEE International Symposium on Information Theory*, p. 481, June 2002.
- [5] O. Milenkovic, I. Djordjevic, and B. Vasic, "Block-Circulant Low-Density Parity-Check Codes for Optical Communication Systems," *IEEE Journal of Selected Topics in Quantum Electronics*, pp. 294–299, Mar. 2004.
- [6] J. Thorpe, K. Andrews, and S. Dolinar, "Methodologies for Designing LDPC Codes Using Protographs and Circulants," in *IEEE International Symposium on Information Theory*, p. 238, June 2004.
- [7] J. Thorpe, "Low-Density Parity-Check (LDPC) Codes Constructed from Protographs," IPN Progress Report 42-154, JPL, Aug. 2003.
- [8] A. Abbasfar, D. Divsalar, and K. Yao, "Accumulate Repeat Accumulate Codes," in *IEEE International Symposium on Information Theory*, (Chicago, Illinois), June 2004.
- [9] T. Richardson and R. Urbanke, "Efficient Encoding of Low-Density Parity-Check Codes," *IEEE Transactions on Information Theory*, pp. 638–656, Feb. 2001.