Low Density Parity Check Codes: Bandwidth Efficient Channel Coding

Wai Fong[1], Shu Lin[2], Gary Maki[3] and Pen-Shu Yeh[1]
[1]Code 567, Goddard Space Flight Center, Greenbelt, MD 20771
[2]University of California at Davis, Davis, CA 95616
[3]Center for Advanced Microelectronics and Biological Research, U. of Idaho, Post Falls, ID 83854

***Abstract*-Low Density Parity Check (LDPC) Codes provide near-Shannon Capacity performance for NASA Missions. These codes have high coding rates R=0.82 and 0.875 with moderate code lengths, n=4096 and 8176. Their decoders have inherently parallel structures which allows for high-speed implementation. Two codes based on Euclidean Geometry (EG) were selected for flight ASIC implementation. These codes are cyclic and quasi-cyclic in nature and therefore have a simple encoder structure. This results in power and size benefits. These codes also have a large minimum distance as much as $d_{min} = 65$ giving them powerful error correcting capabilities and error floors less than 10^{-10} BER. This paper will present development of the LDPC flight encoder and decoder, its applications and status.**

I. Introduction

Largely neglected for 35 years, LDPC Codes has recently been undergoing a rediscovery. Invented by Gallager in 1965 [1], it's near-Shannon Capacity performance was first documented by Mackay in 1996 [2]. These are large block codes with a very sparse (small density of "ones") parity check matrix. Current research follows on two paths: computer generated codes (generally unstructured and semi-random) and geometric codes (highly structured).

II. Computer Generated LDPC Codes

Computer generated LDPC codes can be divided into two categories: Regular and Irregular. Regular LDPC codes have the property every column have a constant number of "ones" and every row has a constant number of "ones", though the number of column "ones" are generally not same as the number of row "ones". Irregular LDPC codes can have varying number of columns "ones" and row "ones".

A. Regular LDPC Codes

The properties of regular LDPC codes can be summarized as follows: the number of "ones" of each row and column are constant (because of this property Gallager codes are now classified as Regular LDPC codes) but small compared to the entire matrix and the number of "ones" in common between any two columns or two rows are small, preferably one. This last property allows for iterative decoding to converge to or close to the Shannon limit and provides for good minimum distance. Code synthesis is achieved through computer generation as each new column is randomly synthesized and tested to ensure compliance with the properties outlined above. With soft-in and soft-out (SISO) iterative decoding Gallager LDPC codes perform close to the Shannon capacity limit [2].

B. Irregular LDPC Codes

Irregular LDPC codes were introduced by Luby *et al.* [3] These are similar to the Gallager codes except that there is variance in the number of "ones" of each row and column. Like Gallager codes, irregular codes are computer generated and are decoded using the same iterative technique. They do outperform Regular codes in terms of distance to Shannon capacity at low signal to noise ratios. However, due to their small to moderate minimum distances both regular and irregular computer synthesized codes can have error floors around 10^{-9} BER or larger.

As a result of the semi-random nature of computer generated LDPC codes, there is the issue of the encoder complexity (on the order of n^2, where n is the length of the codewords in bits) since there are no algebraic or geometric relationships within the generator matrix. And as with any random-like code, a very large memory must be used to generate the code. The implementation of which can be resource consuming and can impact the space, power and weight for a spacecraft.

III. Geometry-Based LDPC Codes

Geometry based LDPC codes were developed by Kou, Lin and Fossorier [4]. These are a class of Regular codes based on parallel lines of finite geometry. Due to their geometric structure some of these codes are cyclic and quasi-cyclic in nature, have large minimum distances, require less iterations to achieve close to Shannon limit decoding performance and have no low error floor $>10^{-10}$ BER. They also have many decoding options: iterative decoding with belief propagation (IDBP), bit-flipping (BF), weighted BF, majority-logic (ML), and weighted ML decoding. Their encoding complexity are on the order of n and their encoding designs consist of a simple chain of shift registers.

One possible drawback to Geometry codes is that the decoders are could be larger due to fact that they be more dense (larger number of "ones") than Irregular codes. Thus Geometry codes require more processing elements in SISO decoding. This, however, is mitigated by the fact that the processing elements all have the same internal and external structure which allows for easy routing and layout of an integrated circuit (IC). This also cuts down on design time since once one element is designed then effectively all of the elements are designed. Computer generated codes are more difficult to route due to their randomness which impacts IC size and require more design time since the processing elements are not identical. Ultimately, there may be very little difference in decoder size between Geometry and computer generated codes but in design time, Geometry codes have a clear advantage. And since Geometry codes require much less decoding iterations, they can outperform computer generated codes in decoding speed.

IV. Code Selection

For near-Earth space missions, Geometry-based LDPC codes are the obvious choice. The decision is based primarily on encoding complexity, BER performance, and decoding speed. The rationale, analysis and code details were presented in a CCSDS Panel 1B White Paper [5] as a proposal for channel coding standardization. Two codes based on EG construction: (4095, 3367) and (8176, 7156) was selected. (The notation used here is (n, k) where n is the length of the codeword and k is the information size, both in bits.)

There are a number of good texts explaining, in general, finite Euclidean Geometry and it's application in coding theory [6, 7]. The reader may wish to consult these texts as an aid to understanding what's to follow.

V. Eg (4095, 3367)

This code, denoted C_1, is a (4095, 3367) cyclic LDPC code with rate $R = 0.822$ and minimum distance $d_{min} = 65$. Since it is cyclic, it is uniquely specified by a generator polynomial $g(X)$ of degree 728 [4]. The degree of g(X) also defines the number parity-check bits of the code. Its encoding circuit can be easily implemented with a feedback shift-register using 728 flip-flops and no more than 728 X-OR gates.

C_1 can be shortened, by deleting 7 information bits from each codeword, to a (4088, 3360) shortened cyclic code C_1^* with minimum distance of at least 65 and a code rate at about the same rate as the original code. For this shortened code, both data and block sizes are multiples of 8 to better suit the requirements of spacecraft and ground station processing elements. The encoding and decoding circuits for the original code can be used for the shortened code without any changes.

A. Code Construction

C_1 is constructed based on the lines and points of the 2-dimensional Euclidean geometry, EG(2, 2^6), over the Galois field GF(2^6). The geometry EG(2, 2^6) consists of 4096 points and 4160 lines. One of the points is the origin of the geometry. Each line consists of 64 points. Let L be a line not passing through the origin of the geometry. The incidence vector of line L is defined as a 4095-tuple over GF(2),

$$\mathbf{v}_L = (v_1, v_2, . . . , v_n). \qquad (1)$$

where $v_i = 1$ if and only if the ith non-origin point of EG(2, 2^6) is on L, otherwise $v_i = 0$. Then the parity-check matrix H_1 of C_1 is a 4095 x 4095 square circulant (a cyclic matrix) with column and row weights of 64. The rows of H_1 are simply the incidence vectors of the 4095 lines in EG(2, 2^6) not passing through the origin. This parity-check matrix can be easily generated by simply cyclically shifting the incidence vector of any line L not passing through the origin 4095 times.

B. Decoding Methods

C_1 can be decoded with various methods, ranging from low to high complexity and reasonably good to very good error performance. These decoding methods are: one-step ML, BF, weighted BF, weighted ML, and IDBP. In general, the higher the decoder complexity, the better the performance. (A more comprehensive description of these decoders can be found in [4,6,7].)

These decoders can be classified as either hard decision or soft decision and iterative or one-shot. ML and BF decoders are hard-decision decoders which can be simply implemented with logic gates and require only logic operations. But ML is one-shot, while BF is an iterative approach.

The IDBP decoder is a soft decision (SISO), iterative decoding which requires real-number computations. The weighted BF decoder is an iterative reliability-based decoding method which also requires some real number computations so it can be viewed as a cross between a soft and hard design decoder. ML decoding is the simplest decoding but should give the least coding gain over the uncoded system compared with the other decoding methods. However it can be implemented to operate at a very high speed, for instance 10-40 Gbps. There is also a weighted ML decoder that should give an improvement over the straight ML approach whose approach is similar to the approach used for weighted BF. This is another hard decision decoder that has elements of soft decision only this is a one-shot approach. BF and ML are similar in complexity and so are weighted BF and weighted ML. However, BF approaches have a longer decoding delay than the ML approaches due to their iterative nature. The complexity of the weighted decoders are higher than their non-weighted counterparts but they should provide better error performance.

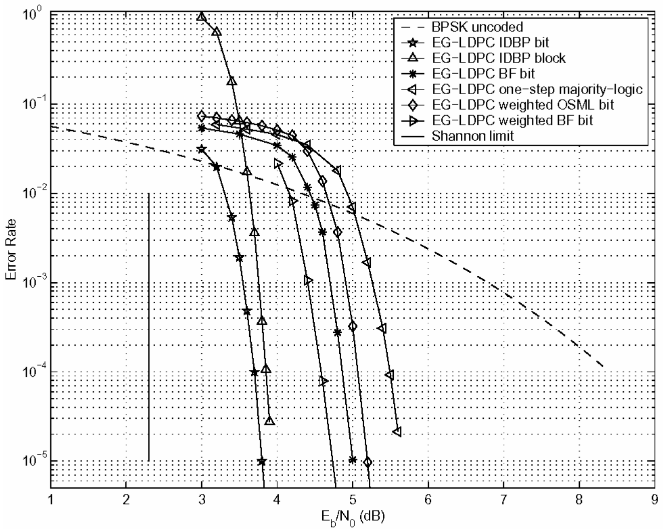

Fig. 1. EG-LDPC (4095, 3367) BER Performance with Various Decoding Options

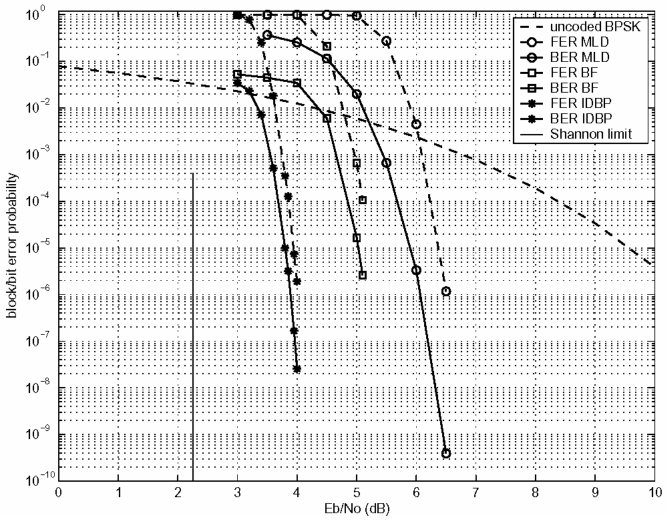

Fig. 2. EG-LDPC (4095, 3367) BER and FER Performance

C. Simulated Bit Error Rate (BER) Performance

Figs. 1, 2 and 3 illustrate the simulated code C_1 performance based on BPSK over an additive white Gaussian noise (AWGN) channel.

Fig. 1 shows the error performance of bit and frame (block) error rates above 10^{-6}. Fig. 2 shows the error performance of C_1 above BER of 10^{-10}. At the BER of 10^{-8}, the code with ML decoding achieves a 5.5 dB coding gain over the uncoded BPSK, while with IDBP, it achieves almost 8 dB coding gain over the uncoded BPSK. ML, weighted ML, BF, weighted BF and IDBP performance range from worst to best in that order. No error floor occurs above 10^{-8} BER for the iterative decoding. Note that the IDBP decoder, as expected, provides the best BER performance with the largest decoding complexity. However, it requires a longer decoding time due to the iterative decoding process. To achieve high-speed decoding, a parallel/pipeline decoding process is needed.

The error floor of an LDPC code depends very much on its minimum distance. Code C_1 has a very large minimum distance = 65. For a code with a large minimum distance, either there is no error floor or the error floor occurs at a very low BER. Therefore, there should be no error floor at all or an error floor below 10^{-10} BER. If there is no error floor above BER of 10^{-10}, C_1 with IDBP would achieve at least a 9 dB coding gain over the uncoded BPSK.

Figs. 1 and 2 also show that the code with IDBP also has good frame error performance. Frame error performance also depends very much on the minimum distance of the code. Computer generated LDPC codes, in general, do not have large minimum distance, and hence their frame error performance in general has error floor at a high frame error rate, for instance 10^{-4}.

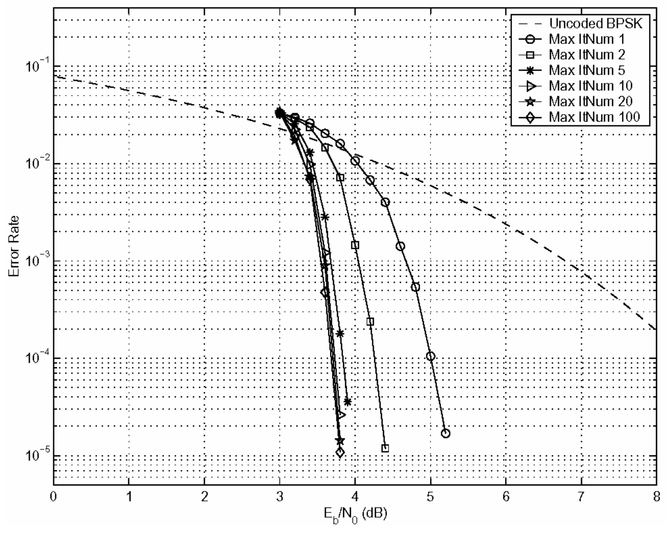

Fig. 3. EG-LDPC (4095, 3367) Iterative Convergence

Though not simulated, it is possible to build a two-stage IDBP/ML decoder where two IDBP soft decision iterations are followed by an ML decoder to shorten the decoding time. This hybrid decoder not only shortens the decoding delay but also reduces the computational complexity with a small performance degradation compared to pure IDBP decoding.

D. Simulated Iterative Performance

Another special feature of this code is that the IDBP decoder converges very fast as shown in Fig. 3. The performance gap between 2 iterations and 100 iterations is within 0.5 dB, while the gap between 5 iterations and 100 iterations is less than 0.2 dB. Therefore, the decoding can be terminated after 5 iterations with only a 0.2 dB loss. This reduces the

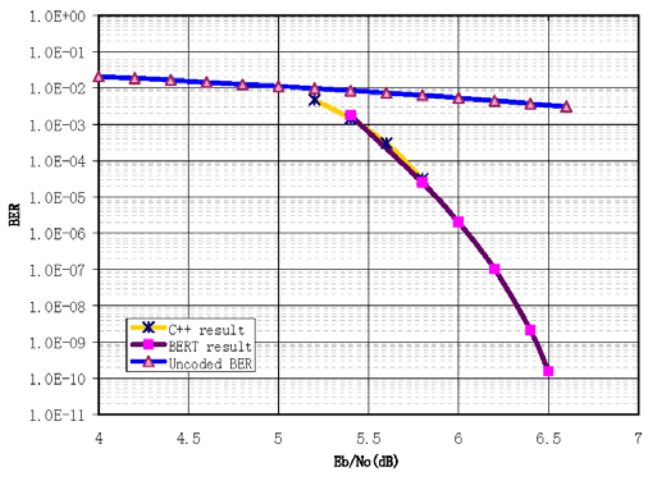

Fig. 4. Majority Logic Decoding Testing at 400 Mbps

decoding computational complexity and speeds up the decoding process (or shorten the decoding delay) significantly.

Alternatively, as mentioned before, the decoding can be terminated after 2 iterations followed with ML decoding (two-stage iterative/ML decoding).

It is worthwhile to note that Computer generated LDPC codes, in general, converge at a much slower rate and take at least 20 iterations to be close to the performance of 100 iterations.

E. Measured Performance

Fig. 4 shows the results of the Field Programmable Gate Array (FPGA) implementation of the EG (4095, 3367) encoder and an FPGA ML decoder operating at 400 Mbps. These are Xilinx FPGA implementations which were tested with a BPSK modulation and AWGN channel. The graph verifies the simulated performance found in Fig. 1. It is also shows that there is no error floor down to 10^{-10} BER. These results give confidence that the IDBP decoder will perform very close to simulations.

VI. EG (8176, 7156)

This code, denoted C_2, is a (8176,7156) Euclidean geometry LDPC code with rate $R = 0.875$. It is not cyclic but quasi-cyclic [6,7]. Every cyclic shift of 4 bits of a codeword is also a codeword. Its encoding can also be implemented with shift-registers.

A. Code Construction

C_2 is constructed based on the 3-dimensional Euclidean geometry EG($3,2^3$) over the field GF(2^3). This geometry consists of 512 points and 4672 lines. Each line consists of 8 points. For each point, there are 73 lines intersecting at it. The incidence vectors of the 4599 lines, not passing through the origin of the geometry, can be partitioned into 9 cyclic classes, $Q_1, Q_2, \ldots, Q_9$, each class consists of 511 incidence vectors. Each cyclic class Q_i can be obtained by cyclically shifting any vector in Q_i 511 times. For each cyclic Class Q_i, a 511 x 511 square circulant matrix A_i is formed whose rows are simply the incidence vectors of Q_i. Both the column and row weights of A_i are 8. Splitting each column of Q_i into 4 columns of the same length with its weight uniformly distributed into the 4 new columns as suggested by Kou, Lin and Fossorier [4], Q_i can be partitioned into four 511 x 511 square circulant matrices, $A_i^{(1)}$, $A_i^{(2)}$, $A_i^{(3)}$, $A_i^{(4)}$. Each circulant $A_i^{(j)}$ has column and row weights of 2. Using these 4 circulants, form a 511x2044 matrix,

$$G_i = [A_i^{(1)}, A_i^{(2)}, A_i^{(3)}, A_i^{(4)}] \tag{2}$$

The column and row weights of G_i are 2 and 8, respectively. Then the parity check matrix H_2 of C_2 is given below:

$$H_2 = \begin{bmatrix} G_1 & G_2 & G_3 & G_4 \\ G_5 & G_6 & G_7 & G_8 \end{bmatrix} \tag{3}$$

The column and row weights of H_2 are 4 and 32, respectively. C_2 is simply the null space of H_2.

B. Simulated Performance

The bit and frame error performance of C_2 with IDBP is shown in Fig. 5. Both the BER and FER curves have a sharp waterfall characteristic. The BER performs very well all the way down to the BER of $4\text{x}10^{-10}$ without any error floor while the FER exhibits no error floor down to 10^{-6}. At the 10^{-9}, the BER achieves more than 9 dB coding gain over the uncoded BPSK and performs only 1.2 dB from the Shannon limit. If there is no error floor above BER of 10^{-10} then C_2 with iterative decoding should provide 10 dB coding gain over the uncoded BPSK. Decoding of C_2 also converges very fast as shown in Fig. 6. At the BER of 10^{-8}, the performance gap between 10 iterations and 100 iterations is about 0.2 dB, while the performance gap between 20 iterations and 100 iterations is within 0.05 dB. To shorten the decoding delay and reduce the computational complexity, the maximum number of iterations can be set to 10 with only a 0.2 dB performance loss.

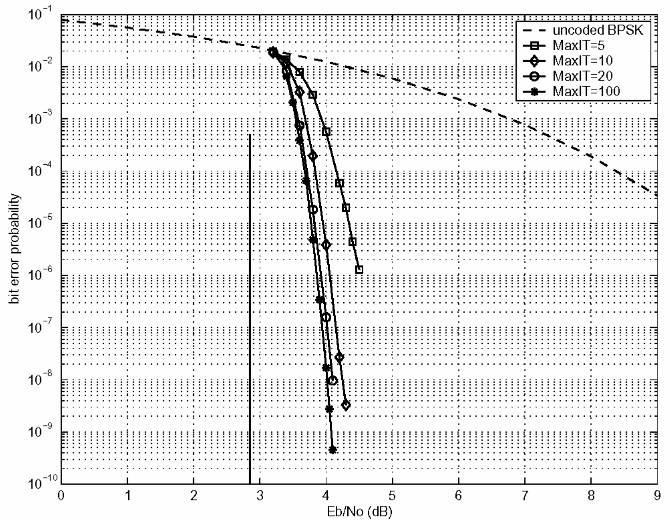

Fig. 5. EG-LDPC (8176, 7156) BER and FER Performance.

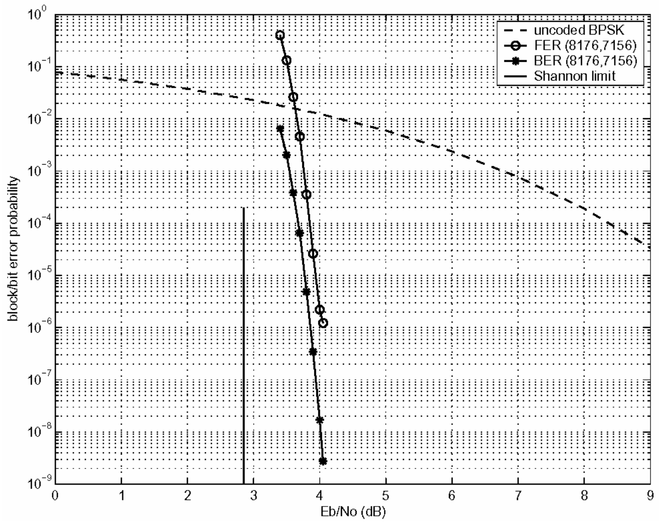

Fig. 6. EG-LDPC (8176, 7156) Iterative Convergence

The exact minimum distance of C_2 is unknown. But since there is no error floor down to the BER of almost 10^{-10}, its minimum distance should be relatively large. Experiments indicate that the minimum distance of C_2 is at least 7.

VII. Performance Comparison With Current Standards

Most near-Earth missions currently follow the CCSDS recommendation of concatenating a (1/2, 7) convolutional code (CC) with a (255, 233) Reed-Solomon (RS) code. Although this coding technique provides good coding gain, a heavy penalty is placed on the bandwidth efficiency. With a code rate of 0.44, an overhead of 0.56 or 56% of the bandwidth is consumed by the parity check bits. Bandwidth constrained missions will sometimes chose to use RS coding alone to reduce the parity check overhead to 12%. However this comes with a large cost in additional power at the transmitter.

Fig. 7 compares the BER performance of EG LDPC codes C_1 and C_2 with RS and CC/RS. Note that at 10^{-5} BER both C_1 and C_2 outperform straight RS by 2.5 dB. This is a substantial improvement. C_1 and C_2 also comes within about 1 dB of CC/RS at 10^{-5} BER. This signifies that these EG codes can replace CC/RS with only a 1 dB penalty in E_b/N_o. And thus the complexity of the communication system is greatly simplified by the replacement of two codes (CC/RS) with one (EG) and thereby doubling the bandwidth efficiency at the same time. Notice the closeness of the C_1 and C_2 performance curves with their corresponding capacity limit. C_1 is within 1.5 dB away while C_2 is within 1 dB away.

VIII. Schedule And Status

There are two parallel development tracks: 1. Flight ASIC encoder along with the ASIC decoder and 2. the FPGA encoders and decoders. As of the beginning of June 2003, C_1 has been designed for the flight ASIC and the FPGA. The flight encoder design has simulated to greater than 1 Gbps operation. It's fabrication is scheduled to be completed by the first quarter of 2004. As mentioned before, the FPGA encoder as well as the FPGA MGL decoder has been tested at 400 Mbps. Currently, an FPGA IDBP decoder has been designed and is being optimized for operating speed. It's testing will be completed by the end of August 2003. The FPGA effort for C_2 will begin immediately following this. An FPGA encoder and decoder of C_2 will probably be completed by end of 2003. Also the ASIC decoders for C_1 and C_2 are

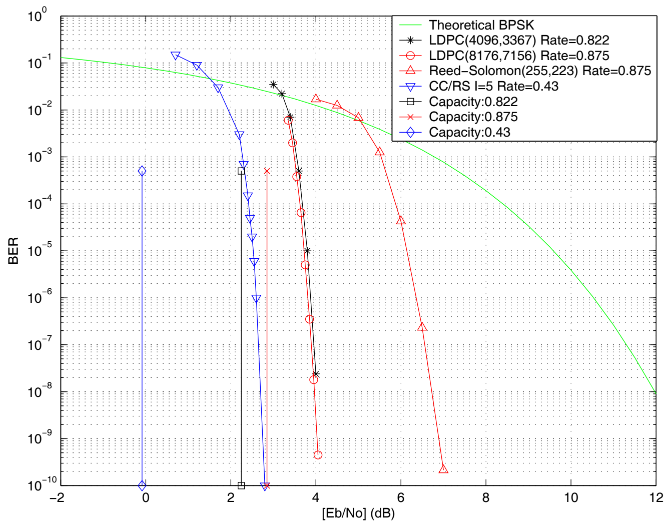

Fig 7. Comparison of LDPC EG codes with RS and CC/RS codes

undergoing an architectural study. Their fabrication won't be completed until first quarter 2005.

Acknowledgements

The authors wish to thank Zongwang Li for his efforts in the FPGA design and for the production of Fig. 4.

References

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

[2] D. J. C. Mackay, "Near Shannon limit performance of low density parity check codes," *IEEE Electronic Letters*, vol. 32, pp. 1645-1646, Aug. 1996.

[3] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Speilman, and V. Stemann, "Practical loss-resilient codes," in *Proc. 29th Annu. ACM Symp. Theory of Computing*, 1997, pp. 150-159.

[4] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Trans. Information Theory*, vol. 47, pp. 2711-2736, Nov. 2001.

[5] W. Fong, "White Paper for Low Density Parity Check (LDPC) Codes for CCSDS Channel Coding Blue Book," CCSDS Panel 1B Meeting Paper, Sept. 2002.

[6] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications.* Englewood Cliffs, NJ: Prentice-Hall, 1983.

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