ERROR FLOOR INVESTIGATION AND GIRTH OPTIMIZATION FOR CERTAIN TYPES OF LOW-DENSITY PARITY CHECK CODES

Lingyan Sun, Hongwei Song, B.V.K Vijaya Kumar*
Carnegie Mellon University, Pittsburgh, PA, USA
*Agere System, Longmont, CO, USA

ABSTRACT

Low-density parity check (LDPC) codes with their near-Shannon capacity limit error correcting performance and iterative decoding algorithm are being evaluated for digital communications applications. For LDPC codes to be used in real systems, their error floors need to be investigated. In this paper, we evaluate the performance of disjoint difference set (DDS)-based LDPC codes (with column weights 3, 4, 5) and array code-based LDPC codes (with column weights 3, 4, 5) in additive, white Gaussian noise (AWGN) channel using a high-speed field programmable gate array (FPGA) simulation platform. The error floor regions (bit error rates down to 10^{-12}) of those codes are presented. For better performance of array codes, a girth optimization method is proposed and the FPGA evaluation results are presented.

1. INTRODUCTION

Low-density parity check (LDPC) codes [1-2] have been the focus of intensive study [3-4] because of their near-Shannon limit error correcting performance as well as the low decoding complexity. Compared to turbo codes, the main advantage of LDPC codes is in the decoding methodology. For turbo codes, the decoding complexity increases exponentially with the constraint length and the decoding is serial. For LDPC codes, the decoding complexity increases only linearly with the block length and the parallel nature of LDPC decoding makes it particularly attractive for VLSI implementation. Tradeoffs between throughput and. logic consumption can be achieved to optimize the design for different applications without the throughput bottleneck as in the case of turbo codes. Because of these benefits, LDPC codes are being investigated for digital communications applications.

Though the decoding complexity of LDPC codes is fairly low, for a random LDPC code, the chip area of the decoder is dominated by the complicated wiring [5]. Structured LDPC codes can lower the routing complexity. Compared to Turbo codes, the main disadvantage of LDPC codes is in their encoding complexity. To reduce the encoding complexity, good code structure is needed.

Another major concern with LDPC codes is their error floor. Error floor refers to the phenomenon where the rate of decrease in the bit error rate (BER) with increasing signal-to-noise ratio (SNR) slows down. An important application for low error-floor codes is in data storage systems, where high-rate codes with very low BER ($<10^{-11}$) are required. Due to the difficulty in analytically determining the code distance spectrum and the sub-optimal nature of iterative decoding, the error floor of LDPC codes remains an open question. Software simulations are too slow to achieve such very low BERs. Instead, a field programmable gate array (FPGA) platform with its high speed and reconfigurability can be used to investigate very low BERs.

In this paper, we investigate the performance of two types of LDPC codes: disjoint difference set (DDS)-based LDPC code and array code-based LDPC codes. From now on, we will refer to these codes as simply DDS codes and array codes. These codes are of interest as their encoders can be easily realized using the shift registers and their decoder benefits from the quasi-cyclic structure of the corresponding parity check matrix **H**. We evaluated the performance of these codes for column weights j=3, 4, 5 (small column weights for practical implementation) down to BERs of 10^{-12}. Girth, an important property of these codes, is the length of the shortest cycle in the bipartite graph corresponding to a code. Larger girths are preferred and for array codes, we propose a method to further improve the girth. The good performance after girth optimization is presented.

The paper is organized as follows. Section 2 is devoted to the description FPGA code evaluation platform. Section 3 describes DDS codes and array codes. The BER and block error rate (BLER) results from FPGA evaluation are shown. A girth optimization method for array code is proposed in Section 4 and the performance improvement after girth optimization is presented. Section 5 concludes the paper.

2. FPGA PLATFORM

For LDPC code investigations, a general way is through C simulation with general purpose microprocessor. The serial nature of microprocessor operation limits the simulation speed and only BERs around 1E-6 to 1E-8 are usually obtained. To get the code performance at BERs in the 1E-11 to 1E-12 region requires a high-speed simulation platform.

We have developed an FPGA platform for code evaluation. The main challenge in building the evaluation platform is to have enough flexibility in the design for reconfiguration while achieving good throughput with low logic, memory consumption to fit in a single chip. The FPGA platform for code evaluation is shown in Fig. 1. The starting and ending point of simulation is controlled by a PC through a PCI interface (PCI_IF). A noise generator generates additive white Gaussian noise (AWGN) and passes it to a log likelihood ratio (LLR) (*prob(bit=0)/prob(bit=1)*) calculator. A reconfigurable LDPC decoder (DEC) takes the LLRs and outputs

improved LLRs. An error analyzer calculates the BER and BLER for different iterations from the decoder outputs.

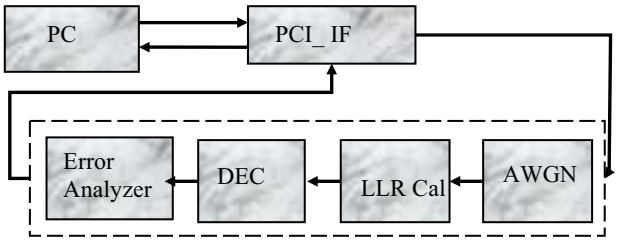

Fig. 1 FPGA Platform

This decoder can be easily reconfigured for different codes that satisfy certain constraints. For more information about the decoder constraints, architecture, throughput and logic consumptions, readers are referred to [6].

3. ERROR FLOOR INVESTIGATION

3.1 Disjoint Difference Set Based LDPC Code

Consider an arbitrary additive group Z of order *v*. A (*v*, *j*) difference set is a set $D=\{d_1,d_2,...,d_j\}$ of elements from Z_v such that no two of the $j(j-1)$ ordered differences modulo *v* are identical. By a (*v, j, t*)-disjoint difference sets (DDS) in an additive group Z_v, we mean a family $(D_i \mid i\in I, t=|I|)$ of subsets of Z_v, each of cardinality of *j*, and such that among the differences $(a-b \mid a,b\in D_i; a\neq b; i\in I)$ each nonzero element $x\in Z_v$ occurs at most once.

The parity check matrix for LDPC codes constructed based on (*v*, *j*, *t*)-DDS is shown in (1). $\mathbf{H}_i$ s are $v\times v$ binary circulant matrices defined as the incidence matrix of the (v,j,t) -DDS

$$\mathbf{H}=[\mathbf{H}_1\ \ \mathbf{H}_2\ \ \mathbf{H}_3 ...\ \mathbf{H}_t] \tag{1}$$

The code constructed has column weight *j*, row weight *k*=*jt*, block length *N*=*vt* and code rate larger than (1-1/*t*). The actual code rate is 1-*rank* (**H**)/*vt*. This code has certain properties. First, the girth of (*v, j, t*)-DDS LDPC codes is six if *j* is great than 3. Second the minimum distance of (*v, j, t*)-DDS codes is bounded by (2). For more information of DDS codes, readers are referred to [7] [8].

$$j+1\le d_{\min}\le 2j \tag{2}$$

We constructed DDS codes with column weights *j*=3, 4, 5 and rate 8/9. The evaluation platform shown in Fig. 1 is configured for different column weight. The BER and BLER after 50 iterations are shown in Fig. 2. The solid lines are for BER and dotted lines are for BLER. As we can see from the results, increasing the column weight lowers the error floor while degrading the performance at cliff BER region. Column weight *j*=3 code has error floor at BER 1E-6, *j*=4 code has error floor at 1E-7 and *j*=5 code has error floor at 1E-9.

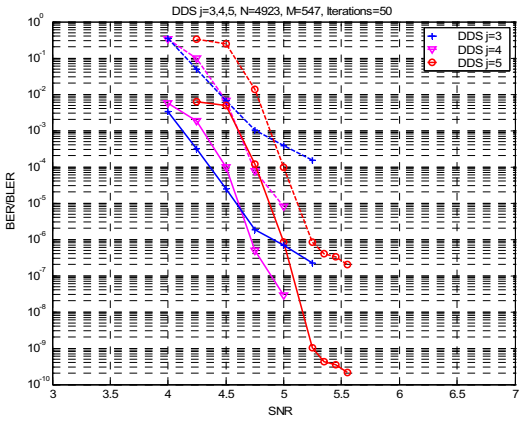

Fig. 2 DDS Code Error Floor Comparison

3.2 Array Code

Let $p\ge 5$ be a prime, and let $\boldsymbol{\sigma}$ be the $p\times p$ identity matrix cyclically left-shifted by one position. We define a parity check matrix **H** as a $j\times p$ block matrix of $p\times p$ circulant matrices

$$\mathbf{H}=\begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \\ \mathbf{I} & \boldsymbol{\sigma} & \boldsymbol{\sigma}^2 & \boldsymbol{\sigma}^3 & \cdots & \boldsymbol{\sigma}^{p-1} \\ \mathbf{I} & \boldsymbol{\sigma}^2 & \boldsymbol{\sigma}^4 & \boldsymbol{\sigma}^6 & \cdots & \boldsymbol{\sigma}^{2(p-1)} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \mathbf{I} & \boldsymbol{\sigma}^{j-1} & \boldsymbol{\sigma}^{2(j-1)} & \boldsymbol{\sigma}^{3(j-1)} & \cdots & \boldsymbol{\sigma}^{(j-1)(p-1)} \end{bmatrix} \tag{3}$$

The code specified by this matrix is a regular (*N*, *j*, *k*) LDPC code with row weight $k=p$, block length $N=p^2$. Fan [9] realized that this permutation matrix also defines a class of array codes. High-rate codes can be obtained by shortening the code generated by (3). LDPC codes based on array code are free of 4-cycles. Their minimum distance can be shown to be $d_{\min}=6$. However, these codes contain a significant number of 6-cycles.

We constructed array codes with *j*=3 *N*=4671, *j*=4 *N*=4716 and *j*=5 *N*=4635 with rate 8/9 where *j* denotes the column weight and *N* denotes the codeword length. The BER and BLER after 50 iterations are shown in Fig 3. Column weight *j*=3 code has error floor at BER 1E-8, *j*=4 code has error floor at 1E-9 and *j*=5 code has error floor at 1E-11. Array code with column weight 5 has very low error floor (BER<10E-11) and have potential for applications that require high rate and low error floor such as in data storage systems. Fig. 4 shows more details about the *j*=5 array code performance at each iteration.

In Fig. 5, we compare the BER performance of array code with *j*=3, 4, 5(solid lines) with DDS code *j*=3, 4, 5 (dot lines). From this figure, we can see that for applications requiring very low BER, array codes may be a better choice compared to DDS codes. However, when both codes are qualified for application, DDS codes are better candidates as they have lower encoding/decoding complexity compared to array codes.

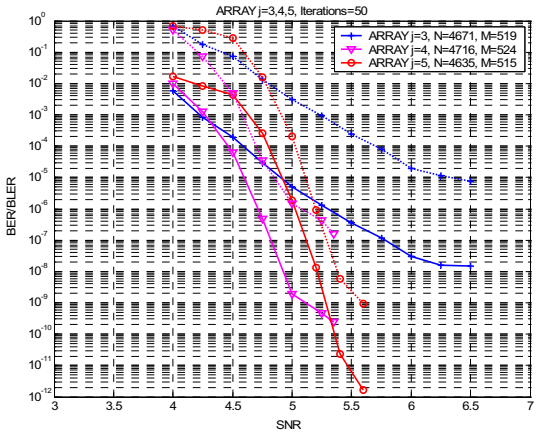

Fig. 3 Array Code Error Floor Comparison

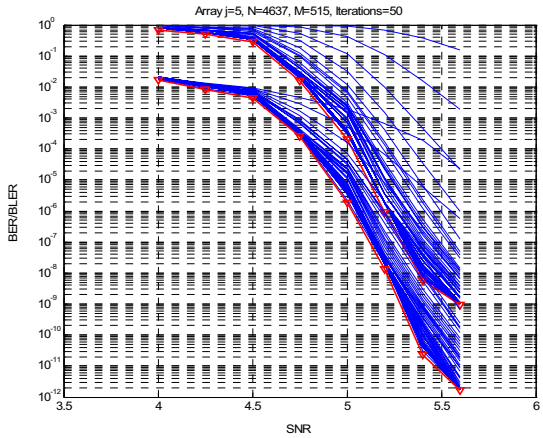

Fig. 4 J=5 Array Code Performance

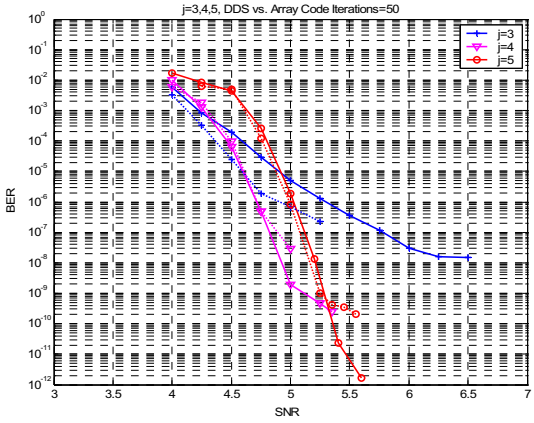

Fig. 5 Array Code vs. DDS code Performance

4. ARRAY CODE GIRTH OPTIMIZATION

Vasic [10] interpreted the parity check matrix **H** based on permutation matrices as a set of even parity equations with different slopes on a rectangular integer lattice $L = \{(x, y) : 0 \le x \le j-1, 0 \le y \le p-1\}$, with a one-to-one mapping of the lattice L to the point set V, $l : L \to V$. Fig. 6a shows an integer array with $j = 3$ and $p = 5$. Let us choose a simple linear mapping $l(x, y) = p \cdot x + y + 1$, and define a line with slope $s, 0 \le s < p$, starting at the point (x, a), containing points $\{(x, a + sx \bmod p) : 0 \le x < j\}$, where $0 \le a < p$. There are p classes of parallel lines with different slopes. We can use the lines to define a $j \cdot p^2 \times p$ matrix using the points in each line as the index of ones in each column of the matrix. Note that the lattice has the topology of a torus, so the lines are wrapped around. The blocks in **H** (3) are specified by parallel lines of increasing slope [3].

Consider the three lines {1, 7, 13}, {1, 8, 15} and {3, 8, 13} shown Fig. 6a. Notice that the points {1, 8, 13} appear twice and form a triangle in the lattice, which suggests that the three columns corresponding to the three lines in the matrix contains a 6-cycle. In general, there is one-to-one mapping between the triangles in the lattice and the 6-cycles in the graph. Therefore, we can calculate the number of 6-cycles, denoted as N_6, by counting the number of triangles in the lattice. The total number of 6-cycles is $N_6 = 2p\binom{p}{2} = p^2(p-1)$.

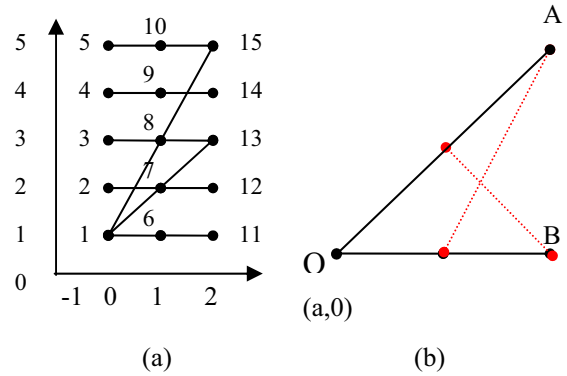

Fig. 6 Lattice and Six Cycles

It is possible to avoid or reduce the number of 6-cycles by removing lines with certain slopes to avoid triangles in the lattice. Without loss of generality, consider two lines with slopes s_a and s_b incident on a point O (a, 0) and containing points A $(a + s_a \cdot x_a, x_a)$ and B $(a + s_b \cdot x_b, x_b)$, where $0 \le s_a \ne s_b < p$, and $0 < x_a \ne x_b < m$, shown in Fig. 6b. A triangle will result if there is a line of slope s_{ab} containing points A and B. Thus, the condition to avoid 6-cycles is

$$s_b x_b - s_a x_a - s_{ab}(x_b - x_a) \ne 0 \bmod p \quad (4)$$

For j=3 case, the condition reduces to

$$2s_b - s_a - s_{ab} \ne 0 \bmod p \quad (5)$$

Using the above conditions, we can get a slope set which allows us to construct 6-cycle-free LDPC codes. For instance, the slope set $S = \{0, 1, 3, 4\}$ for p=11 satisfies the constraint in (5), and can be used to construct a regular (N=44, j=3, k=4) LDPC code having girth g=8.Table 1 shows the obtained sets for different p values when j=3.

Table1. Sets to avoid 6 cycle for j=3

p	Set
5	[0 1]
11	[0 1 3 4]
53	[0 1 3 4 9 10 12 13]
79	[0 1 3 4 9 10 12 13 27 28 30 31 36 37 39]
103	[0 1 3 4 9 10 12 13 27 28 30 31 36 37 39 40]

An 8-cycle in Tanner graph manifests itself in the lattice as a 4-cycle composed of four points and four tail-biting edges. Fig. 7 depicts two types of such 4-loops which contribute to 8-cycles in the graph code. By choosing a set of slopes properly, it is possible to avoid 4-cycle in the lattice, thus avoid 8-cycle in the graph. The mapping between the cycles and the geometry structures in the lattice can be generalized to cycles of any length. However, it is of little practical significance to go beyond 8-cycle-free codes since the code length grows fast as the girth increases, in particular for high-rate codes.

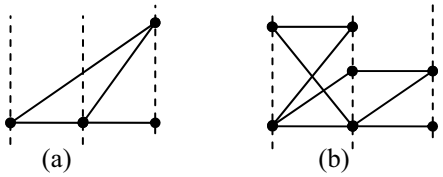

Fig. 7 (a) Triangle (b) Quadrangle in lattice

A similar equation as (5) can be derived in this case. However, a program to search on the graph can produce an easier solution. Table 2 shows the results for different p values at j=3.

Table 2. Sets to avoid 8 cycle for j=3

p	Set
5	[0 1]
11	[0 1 4]
53	[0 1 4 11 29]
79	[0 1 4 11 27 42]
103	[0 1 4 11 27 45]

To get high-rate code, sometimes it is not possible to avoid all the 6-cycles or 8-cycles. However, reducing the number of cycles can also improve the performance.

To evaluate the performance of array code after girth optimization, we take j=3 code as example. When p =173, the slop set is [0 1 3 4 9 10 12 13 27 28 30 31 36 37 39 40 81 82 84 85]. By adding seven more arbitrary sets to the slop set, we can form a rate 8/9 code with block size 4671. The performance of the code is compared to a normal array code in Fig. 8. There is about 0.5 dB gain after girth optimization.

5. CONCLUSION

For LDPC codes to be applied in real systems, error floor of LDPC code needs to be understood. Methods to further improve the performance of LDPC codes also need to be developed. In this paper, we evaluated the performance of high-rate array codes and DDS codes using an FPGA platform and observed the error floors of those codes for different column weights. For array code, a girth optimization method is proposed and the good performance of the method is demonstrated.

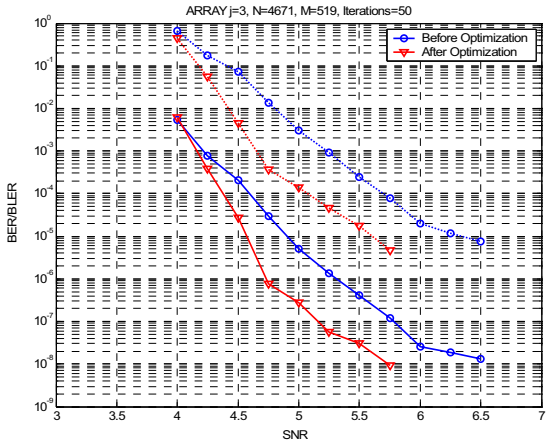

Fig. 8 Girth Optimization Effect

The authors would like to acknowledge the support of this research by Agere Systems and the Multi-Terabyte Tape System (MTS) Advanced Technology Program (ATP)

6. REFERENCES

[1] D. J. C. Mackay and R. M. Neal, "Good codes based on very sparse matrices," *Cryptography and Coding. 5th IMA Conference*, 1995.

[2]M. Sipser and D. Spielman, "Expander codes," *IEEE Trans. on Inform. Theory*, vol. 42, page 1710- 1722, November 1996.

[3] H. Song, "Iterative soft detection and decoding for data storage channels," *Ph.D dissertation*, Dept. of ECE, Carnegie Mellon University, Pittsburgh, USA, Dec. 2002.

[4] S. Y. Chung, "On the construction of some capacity-approaching coding schemes," *Ph.D. dissertation*, Massachusetts Institute Technology, September 2000.

[5] C. Howland, A. Blanksby, "690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE Journal of Solid-State Circuits*, vol. 37, page 404– 412A, 2002.

[6] L. Sun, B.V.K Vijaya Kumar "Field programmable gate array implementation of a generalized decoder for structured low-density parity check codes," accepted by *IEEEInternational Conference on Field Programmable Technology*, 2004.

[7] Y. Kou, S.Lin, M.P.C. Fossorier, "Low density parity check codes: construction based on finite geometries," *IEEE Global Telecommunications Conference*, vol. 2, page 825 – 829, 2000.

[8] S. J. Johnson, and S. R. Weller, "Construction of low density parity check codes from Kirkman triple systems," *IEEE Globecom 200*1, San Antonio, TX, Nov. 2001.

[9] J. Fan, "Array codes as Low-Density Parity Check Codes," *2nd International Symposium on Turbo Codes and Related Topics* (Brest, France), Sep. 2000.

[10] B. Vasic, "High-rate low-density parity check codes based on anti-pasch affine geometries,"*ICC'2002*, New York, May 2002.