# Structured Bit-Interleaved LDPC Codes for MLC Flash Memory

Kathryn Haymaker and Christine A. Kelley

***Abstract*—Due to a structural feature in the programming process of MLC (two bits per cell) and TLC (three bits per cell) flash memory, the majority of errors that occur are single-bit errors. Moreover, the voltages used to store the bits typically result in different bit error probabilities for the two or three types of bits. In this work we analyze binary regular LDPC codes in the standard bit-interleaved coded modulation implementation, assuming different probabilities on the bits, to determine what ratio of each type of bit should be connected at the check nodes to improve the decoding threshold. We then propose a construction of nonbinary LDPC codes using their binary images, resulting in check node types that come close to these optimal ratios.**

***Index Terms*—Parity check codes, iterative decoding, bipartite graph**

## I. Introduction

FLASH memory is a nonvolatile storage device that is ubiquitous in modern technologies. Traditionally, the device was composed of blocks of binary cells, each of which could be injected with charge in the form of electrons. Each cell has a predetermined threshold level that dictates whether the cell will be read as a '0' or a '1'. More recently, the storage capacity of flash memory devices has increased dramatically, due in part to the development of multi-level cell (MLC) flash composed of cells that can store two bits, and triple-level cell (TLC) flash composed of cells that can store three bits [1]. The physical layout of the memory (as observed in the flash memory products used in the experiments in [2]) is as follows: the cells are organized into pages, the pages are organized into blocks, and each block contains 128 pages (resp., 384 pages) of cells in MLC (resp., TLC) flash.

In MLC flash, each cell can hold one of four symbols that may be viewed as binary 2-tuples. The left-most bit is called the most significant bit (MSB) and the right-most bit is the least significant bit (LSB). The two bits of a single cell are distributed among different pages so that pages contain only MSBs or only LSBs. Similarly, in TLC flash, each cell can hold one of eight symbols whose bits are distributed among MSB and LSB pages, as well as pages containing central significant bits (CSBs).

In [2], [3] the authors observe that in the TLC flash memory that they tested, a large majority of observed errors ($96\%$) were single-bit errors, and further that the MSB pages have a lower page error rate than the CSB and LSB pages. Similar differences in MSB and LSB bit error probabilities for MLC were observed in [4]. The extent to which the bit error probability of a MSB differs from that of a LSB or a CSB bit depends on the mappings of cell levels to binary representations in MLC and TLC flash that are used. An earlier analysis in [5] revealed further differences between the overall performance of SLC (single-level cell) and MLC flash products, including power, lifetime, and error rates.

These observations motivate the need to design codes for MLC and TLC flash memory that address this difference in bit error rates. This work analyzes how the bit assignments to MSB and LSB pages affect performance when a binary low-density parity-check (LDPC) code is used for MLC flash in a bit-interleaved coded modulation scheme. A non-binary LDPC code construction and a new implementation method based on this analysis are then presented. The resulting nonbinary codes are sensitive to the different error rates between the MSB and LSB pages, and have improved bit (and overall symbol) reliability.

Nonbinary LDPC codes have been an important area of study since the 1990s resurgence of the work of Gallager [6]. Finding efficient ways of exploiting nonbinary codes and LDPC codes in multilevel flash memories has been a goal in the last decade as flash memory became prominent [7], [8]. Recently there has been an increased focus on nonbinary LDPC codes for various applications [9], [20], [10], [11].

The contributions of the present work are twofold. First, our analysis of the connectivity of check nodes to MSB/LSB pages for binary LDPCs gives insight to the optimal check node degree distributions to MSB and LSB bits for standard MLC flash bit-interleaved coded modulation. We consider hard decision decoding for our analysis due to both its low complexity and the fact that flash applications are aimed to provide high throughputs and access speeds. Second, we use these distributions to design explicit nonbinary LDPC codes that take into account the different probabilities of error among the LSB and MSB pages. We present an implementation method for nonbinary LDPC codes on multilevel flash cells, where the decoding of the nonbinary code is done by using a binary hard-decision decoder on its binary image graph to reduce complexity.

It is worth noting that several construction and decoding strategies for binary LDPC codes have been proposed [13], [14], [15] that deal with unequal error probabilities and nonuniform channels, but these are not specific to the flash memory structure.

Manuscript received May 15, 2013; revised October 1, 2013 and December 10, 2013. This work was supported in part by the National Security Agency under Grant Number H98230-11-1-0156. The United States Government is authorized to reproduce and distribute reprints not-withstanding any copyright notation herein.

K. Haymaker and C.A. Kelley are with the Department of Mathematics, University of Nebraska-Lincoln, Lincoln, NE 68588 USA (e-mail: s-khaymak1@math.unl.edu, ckelley2@math.unl.edu).

Digital Object Identifier 10.1109/JSAC.2014.140507.

This paper is outlined as follows. Basic background and notation are presented in Section 2. In Section 3 we analyze the decoding threshold of binary regular LDPC codes for MLC flash for different check node to MSB and LSB degree distributions using the Gallager A and B decoding algorithms. In Section 4 we present an implementation method for nonbinary LDPC codes in MLC flash using their binary matrix representations. In Section 5 we give explicit choices of check degree distributions that incorporate the analysis from Section 3 and whose corresponding field elements result in structured bit-interleaved nonbinary LDPC codes. Section 6 concludes the paper.

## II. Preliminaries

One common method of storing data in MLC flash memory is to take a binary code and arbitrarily assign the bits to MSB and LSB pages, for example in an alternating fashion. In this way, the two bits that compose a symbol are encoded independently but readable from the stored symbol. This bit-interleaved coded modulation approach allows any binary code to be applicable on MLC flash. An alternative approach is to use multilevel coding in which a message is split in half and two (possibly different) codes are used on each page type. Again, this implementation uses binary codes for each page, but allows for a code with better error correction capabilities to be used on the pages with higher bit error rate.

A nonbinary low-density parity-check code $\mathcal{C}$ is the nullspace of a matrix $H(\mathcal{C})$ with entries in $\mathrm{GF}(2^m)$ that is sparse in the number of nonzero entries. The code may be represented by a sparse bipartite graph in which one vertex set (*variable nodes*) corresponds to the columns of $H(\mathcal{C})$, the other vertex set (*check nodes*) corresponds to the rows of $H(\mathcal{C})$, and an edge with label $\gamma$ connects variable node $i$ to check node $j$ if and only if the $\{j,i\}$th entry of $H(\mathcal{C})$ is $\gamma \neq 0$. A codeword is an assignment of elements from $\mathrm{GF}(2^m)$ to the variable nodes such that the linear combination of values assigned to the neighbors of each check node, with coefficients dictated by the corresponding edge labels, is the additive identity in the field.

For an LDPC code used in the bit-interleaved method for MLC flash, it is natural to ask whether the number of MSB and LSB neighbors of each check node affects the decoding performance, particularly when the voltages representing the symbols in the cells result in a greater disparity between the MSB and LSB bit error rates. In the next section we will investigate this question for binary $(j,k)$-regular LDPC codes in which each variable node has degree $j$ and each check node has degree $k$. We will say a check node has *type* $T(\alpha,\beta)$ if it has $\alpha$ MSB neighbors and $\beta$ LSB neighbors. Clearly, $\alpha+\beta$ is the degree of the check node.

## III. Bit assignments for binary regular LDPC codes

In this section, we assume a binary $(j,k)$-regular LDPC code and examine different combinations of check nodes types to determine which has the best decoding threshold based on density evolution [6], [16], [17]. For a $(j,k)$-regular "cycle-free" LDPC code with given check node types, we will analyze the probability, as a function of the decoding iteration, that a message from a variable node to a check node is in error using the Gallager A and B algorithms [6]. Let $b_1$ and $b_2$ be the initial channel probability of error of a MSB and a LSB bit, respectively, and assume that $b_1 < b_2$.

*Remark 3.1:* The case when $b_2 < b_1$ can be obtained by simply reversing the roles of the most and least significant bits. When $b_1 = b_2$ the result is the standard case where all bits have the same probability of error.

The Gallager A hard decision message passing algorithm requires all of the check node neighbors of a given variable node $v$ to agree (except the neighbor $c$ that $v$ is sending to) in order to change the value that $v$ sends to $c$ in the next iteration. When calculating the probability that the message sent from a variable node to a check node in the $(t+1)$th decoding iteration is incorrect, there are two cases: either the variable node is a MSB, denoted by $v_M$, or the variable node is a LSB, denoted by $v_L$. Let $p_M^{(t+1)}$ denote the probability that a message sent from $v_M$ to a neighboring check node in the $(t+1)$th iteration is in error, and define $p_L^{(t+1)}$ similarly. Finally, let $q_M^{(t)}$ and $q_L^{(t)}$ denote the probability that a message sent in iteration $t$ from a check node to a MSB or LSB, respectively, is in error.

Using calculations analogous to those in [6], [18] for a $(j,k)$-regular graph having girth at least $2t$, we obtain the following.

*Lemma 3.2:* If $x_1$ and $x_2$ are the number of MSB and LSB neighbors[1], respectively, involved in a message update at a check node $c$, then the probability that the message from $c$ is in error in iteration $t$ is

$$q^{(t)} = \frac{1-(1-2p_M^{(t)})^{x_1}(1-2p_L^{(t)})^{x_2}}{2}.$$

Moreover,

$$p_M^{(t+1)} = b_1\left(1-\left(1-q^{(t)}\right)^{j-1}\right)+(1-b_1)\left(q^{(t)}\right)^{j-1}, \text{ and}$$

$$p_L^{(t+1)} = b_2\left(1-\left(1-q^{(t)}\right)^{j-1}\right)+(1-b_2)\left(q^{(t)}\right)^{j-1}.$$

### A. Two check node types

Here we focus on the case where the code has two types of check nodes. For $0 \le g \le 1$, let $g$ be the fraction of check nodes having type $T(\alpha_1,\beta_1)$ and $(1-g)$ be the fraction of check nodes having type $T(\alpha_2,\beta_2)$. Our convention in this section is to consider cases where $\alpha_1 \le \alpha_2$ to avoid repetition. Let $\ell$ be the number of check nodes. Since half of the variable nodes are necessarily assigned to MSB pages and the other half to LSB pages, the following constraint holds:

$$\alpha_1 g\ell + \alpha_2(1-g)\ell = \frac{k\ell}{2} = \frac{\#\text{ edges}}{2}. \tag{1}$$

Consequently, $\alpha_2 = (\frac{k}{2}-\alpha_1 g)\frac{1}{1-g}$, and observe that $\beta_1 = k-\alpha_1$ and $\beta_2 = k-\alpha_2$.

Let $q_{1,M}^{(t)}$ denote the probability that a message from a check node of Type 1 to a MSB neighbor in iteration $t$ is in error:

$$q_{1,M}^{(t)} = \frac{(1-(1-2p_M^{(t)})^{\alpha_1-1}(1-2p_L^{(t)})^{\beta_1})}{2}.$$

[1]Here, $x_1+x_2 = k-1$ since the check node has degree $k$ and the variable node receiving the message is not included in the message update.

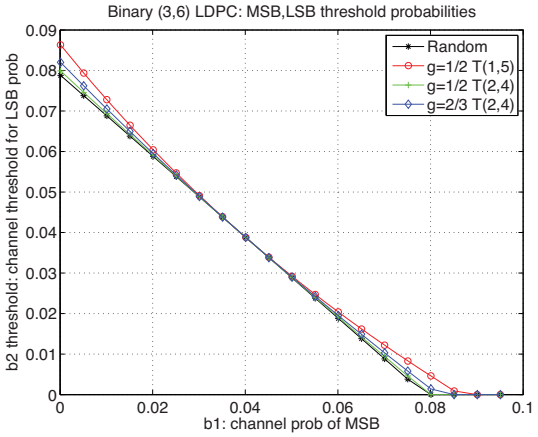


Fig. 1. Thresholds for structured bit-interleaved $(3, 6)$-regular codes and corresponding random code.

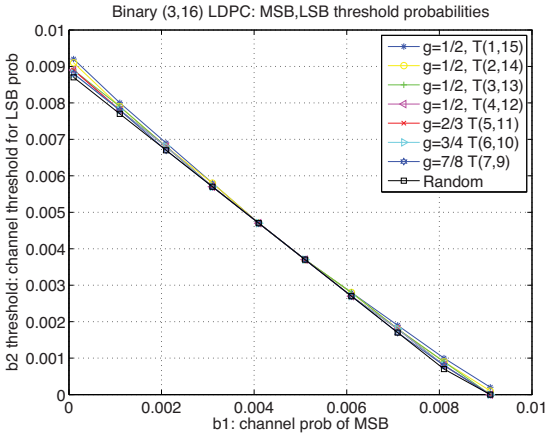


Fig. 2. Thresholds for structured bit-interleaved $(3, 16)$-regular codes, showing the best of each $\alpha_1 = 1, \ldots, 7$, and corresponding random code.

The error probabilities $q_{1,L}^{(t)}$, $q_{2,M}^{(t)}$, and $q_{2,L}^{(t)}$ are defined analogously. On average the probability that a message sent from a check node to a MSB neighbor in iteration $t$ is in error is

$$q_M^{(t)} = g(q_{1,M}^{(t)}) + (1-g)q_{2,M}^{(t)}.$$

Define $q_L^{(t)}$ in the analogous way.

The corresponding probability of error of a message from a MSB or LSB variable node in the $(t+1)$th iteration is

$$p_M^{(t+1)} = b_1(1-(1-q_M^{(t)})^{j-1}) + (1-b_1)(q_M^{(t)})^{j-1}, \text{ and}$$
$$p_L^{(t+1)} = b_2(1-(1-q_L^{(t)})^{j-1}) + (1-b_2)(q_L^{(t)})^{j-1}.$$

For fixed values of the MSB bit error probability $b_1$, we ran simulations of this algorithm in MATLAB to determine the corresponding decoding threshold for the LSB bit error probability $b_2$. We considered $b_1$ in the range of 0.0001 to 0.1 to be fixed, and ran 100 iterations for each $b_1$ tested. We tested $(3, 6)$-regular, $(3, 16)$-regular $(3, 30)$-regular, and $(4, 8)$-regular LDPC codes having different check node types, and compared their decoding thresholds for $b_2$, including that of a random code wherein each neighbor of each check node is equally likely to be a MSB or a LSB. The results are summarized in the following subsections.

*1) Results for binary $(j, k)$-regular codes using Gallager A:* Recall that $g$ is the fraction of check nodes of Type 1. We now present the results of testing the Gallager A algorithm described above for different fractions $g$ of various check node types. Once $g$ and $\alpha_1$ are fixed, the values $\alpha_2$, $\beta_1$, and $\beta_2$ are determined. Our results indicate that for a fixed probability $b_1$, the best $b_2$ threshold occurs when $g = 1/2$ and the two check types are $T(1, k-1)$ and $T(k-1, 1)$ (i.e. $\alpha_1 = 1$). This suggests that codes having highly unbalanced check nodes with respect to MSBs and LSBs will perform better than the expected result of standard bit-interleaved coded modulation, which yields on average half MSB and half LSB neighbors at each check node. It is worth noting that when the two check types are $T(0, k)$ and $T(k, 0)$, one obtains the multilevel coding situation wherein MSB and LSB pages are encoded and decoded separately.

Figure 1 includes a curve for possible combinations of $g$ and $\alpha_1$ for a binary $(3, 6)$-regular LDPC code. The corresponding type for the remaining check nodes can be found using Equation 1. The case in which on average each check node has half MSB neighbors and half LSB neighbors (denoted by "Random" in Figure 1) consistently has the lowest $b_2$ threshold, while the curve for $g = 1/2$ and Type $1 = T(1, 5)$ has the highest $b_2$ threshold for every value of $b_1$. It is therefore advantageous to design structured bit assignments for binary LDPC codes rather than using random bit assignments.

Figure 2 summarizes the results for binary $(3, 16)$-regular LDPC codes, where for each $\alpha_1 = 1, \ldots, 7$, only the best result is shown for clarity. For example, when $\alpha_1 = 7$, values of $g$ that yield a legitimate value for $\alpha_2$ include $g = 1/2, 2/3, 3/4$, and $7/8$. Figure 2 contains the best-performing curve which occurred when $g = 7/8$. The other values of $\alpha_1$ were treated analogously. As is evident from Figure 2, the gain in $b_2$ threshold that is achieved by the pair $g = 1/2, T(1, 15)$ for $(3, 16)$-regular LDPC codes is the greatest, but is more subtle than the gain seen in $(3, 6)$-regular LDPC codes. We will see that this trend continues in the case of $(3, 30)$-regular LDPC codes; the higher the code rate, the smaller the gain in $b_2$ thresholds. However, it is notable that the extremely unbalanced check node types consistently perform among the best in all cases tested.

Due to the large number of possibilities for pairs $\{g, \alpha_1\}$ when $k = 30$, and the fact that most of the curves lie close together, we ran the simulations for $g = 1/2, T(1, 29)$, and the random case to see whether a gain in $b_2$ threshold also exists in this setting. Figure 3 shows that for small values of $b_1$, the $b_2$ threshold in the $T(1, 29)$ case is higher than in the random case.

Similarly, we tested $(4, 8)$-regular LDPC codes for different check node types. The results are shown in Figure 4. Most of the cases coincided; however, all of the structured bit-interleaved LDPC codes outperform the random case for small values of $b_1$.

Observe that for some of the cases, the $b_2$ thresholds were essentially constant over certain intervals of $b_1$, suggesting that in these intervals, there is less dependence of $b_2$ on $b_1$ since the check node types do not have as many MSBs influencing LSBs.

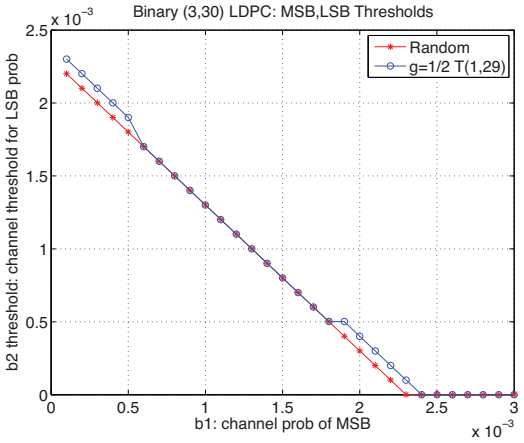


Fig. 3. Thresholds for $(3, 30)$-regular codes, showing random vs. $g = 1/2$ and $T(1, 29)$.

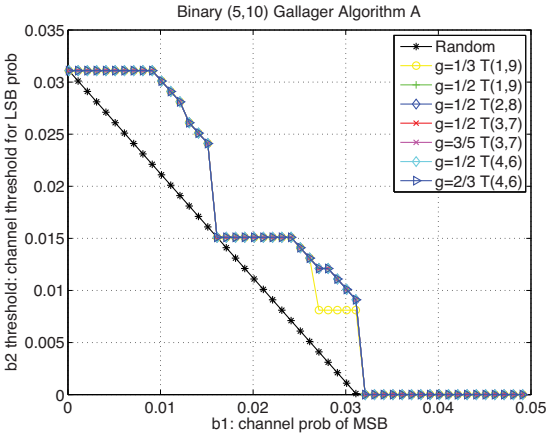


Fig. 5. Thresholds for $(5, 10)$-regular codes, Gallager A algorithm.

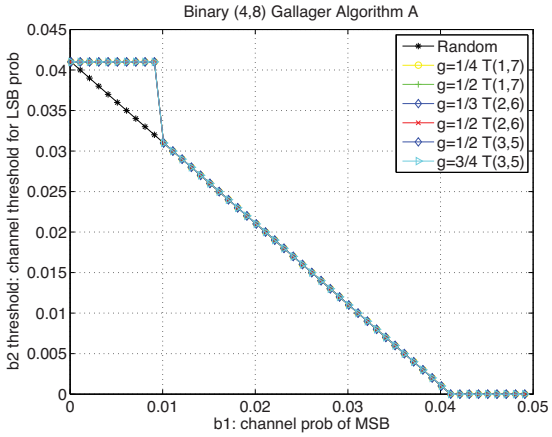


Fig. 4. Thresholds for structured bit-interleaved $(4, 8)$-regular codes and corresponding random code.

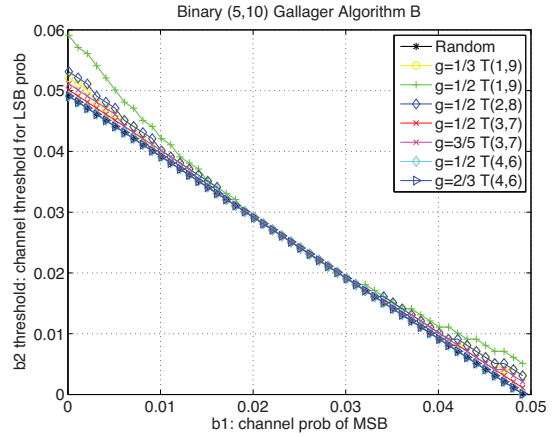


Fig. 6. Thresholds for $(5, 10)$-regular codes, Gallager B algorithm

The plots in this section used the density evolution equations presented earlier for the Gallager A algorithm applied to the two initial channel probabilities. This analysis is accurate when applied to codes with graphs of large girth. We work under this assumption since random regular bipartite graphs are known to have girth that increases logarithmically in the blocklength (see e.g., [19]).

*2) Results for binary $(j, k)$-regular codes using Gallager B:* We now consider the Gallager B algorithm from [6], which requires a majority of check messages to agree to change the variable node estimate from that of the channel. In the case of $j = 3$ and $j = 4$, the expressions for $p_M^{(t+1)}$ and $p_L^{(t+1)}$ are the same for both algorithms, so the results in the previous subsection hold for Gallager B decoding.

In Figures 5 and 6, the thresholds for structured bit-interleaved $(5, 10)$-regular LDPC codes are shown, using the Gallager A and B algorithms, respectively. The $b_2$ thresholds were better under Gallager B for fixed values of $b_1$, which was expected since Gallager B typically out-performs Gallager A for small check node degrees. Under Gallager A, most of the structured codes performed comparably and were better than the random construction. Under Gallager B, the best $b_2$ threshold was observed for the case where $g = 1/2$ of the check nodes had type $T(1, 9)$, and the remaining had type $T(9, 1)$. Furthermore, Gallager B showed more improved thresholds than Gallager A for the case of $(5, 10)$-regular LDPC graphs with these structured connections.

In Figures 7 and 8, the thresholds for $(5, 50)$-regular LDPC codes are shown using the Gallager A and B algorithms, respectively. When the check node degree is large, Gallager A is expected to perform better than Gallager B, as observed in the figures. In case A, the structured bit-interleaved codes performed noticeably better than the random bit-interleaved code.

### *B. More than two check node types*

In general, for $s$ check types, denoted by $T(\alpha_1, \beta_1), \ldots, T(\alpha_s, \beta_s)$, let $g_i$ be the fraction of check nodes of type $i$. Then $\sum_{i=1}^{s} g_i = 1$, and assuming that half of the variable nodes are MSBs and half are LSBs, the following equation holds:

$$\frac{k}{2} = g_1\alpha_1 + g_2\alpha_2 + \cdots + g_s\alpha_s.$$

For a given $k$ and $s > 2$, there are many solutions to this equation. Although restricting the values of the $g_i$'s yields a finite solution set, the problem of assigning bits to pages to

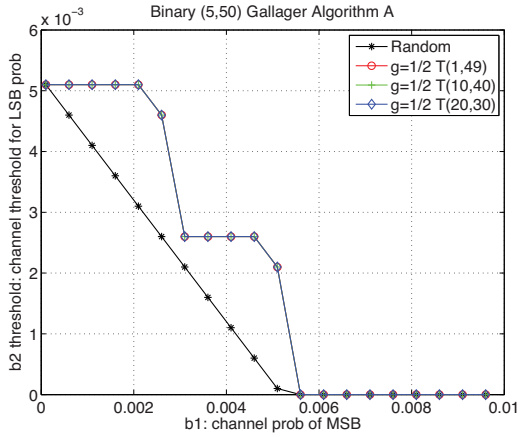


Fig. 7. Thresholds for $(5, 50)$-regular codes, Gallager A algorithm.

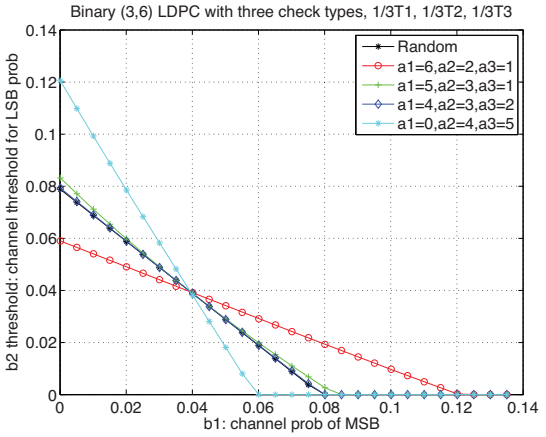


Fig. 9. Three check types, with ratios $g_1 = g_2 = g_3 = \frac{1}{3}$.

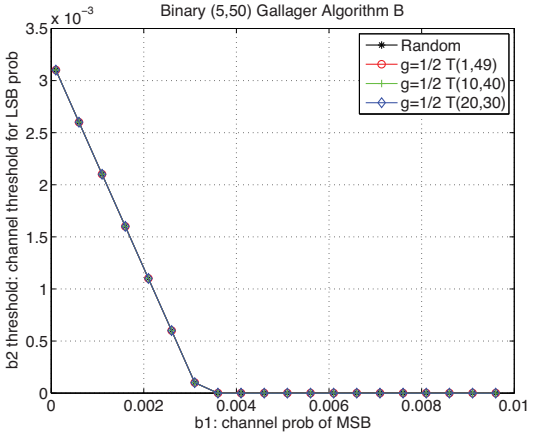


Fig. 8. Thresholds for $(5, 50)$-regular codes, Gallager B algorithm.

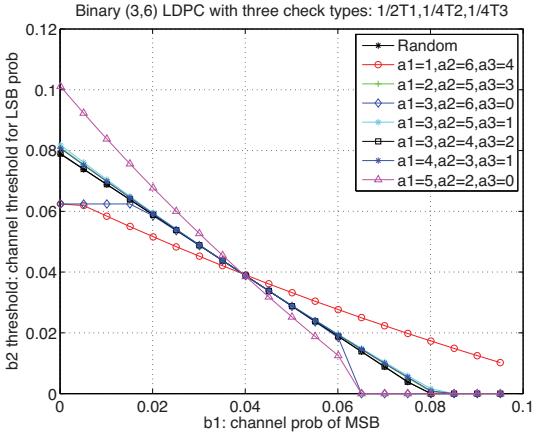


Fig. 10. Three check types, with ratios $g_1 = \frac{1}{2}$, $g_2 = g_3 = \frac{1}{4}$.

achieve the given check node types becomes more complex as $s$ increases. In Figures 9, 10, and 11 we consider three check types in a $(3, 6)$-regular LDPC code. The plots show the thresholds for possible check node types for certain fixed values of $g_1, g_2$, and $g_3$. Figure 12 contains thresholds for the case of four check types when the checks are evenly partitioned; again, the code is assumed to be $(3, 6)$-regular. Certain configurations of three check types outperform the best-performing configurations of two check types, as well as the four types tested in Figure 12.

*Observation 1:* Codes with more than half of their check nodes having a majority of MSB neighbors, i.e., $T(5, 1)$ or $T(4, 2)$, have higher thresholds than the expected BICM case.

### C. Results in terms of noise variance and SNR thresholds

Assume the binary two-tuples stored in a flash memory cell are mapped to the 4-ary PAM signal set $\{-3, -1, 1, 3\}$ using a Gray map. Further suppose that the noise is modeled as additive white Gaussian noise (AWGN) with variance $\sigma^2$. The MSB and LSB probabilities of error in terms of $\sigma$ are given by

$$b_1 \approx \frac{1}{2} Q\left(\frac{1}{\sigma}\right)$$

$$b_2 \approx Q\left(\frac{1}{\sigma}\right).$$

The decoding thresholds for $\sigma$ and corresponding SNR thresholds (in dB) demonstrate the gain over BICM that can be achieved by using structured binary interleaving in the case of this signal set. In Table I, the noise variance threshold and SNR thresholds are given for two different check node types in a $(3, 6)$-regular LDPC code. The rows are ordered by performance, best to worst. Table II shows the noise variance and SNR thresholds when three distinct check types are used. The gain over the expected BICM $\sigma$ threshold is notably greater for the best-case of the three check types than the best case of the two check types. There are also some ratios of three check types that perform worse than the expected BICM $\sigma$ threshold. The results in Table I are consistent with the observations in Figure 1. That is, when the MSB probability of bit error from the channel is smaller than that of the LSB, the check node configuration of $T(1, 5), T(5, 1)$ with $g_1 = g_2 = \frac{1}{2}$ has the best $\sigma$ and SNR threshold. Similarly, Table II shows that the same check type configurations outperform the expected BICM case as in Figures 9, 10, and 11. Moreover, we see the effect of Observation 1 here—configurations with greater than half of

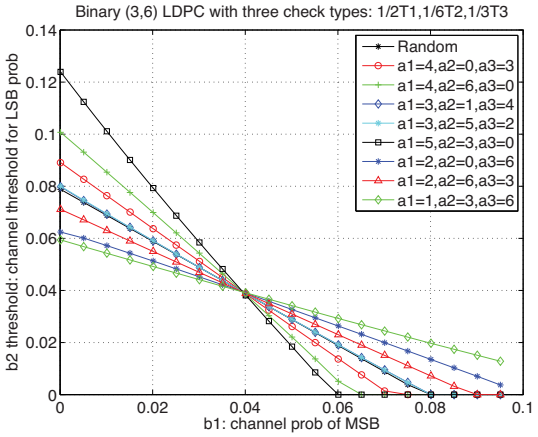


Fig. 11. Three check types, with ratios $g_1 = \frac{1}{2}, g_2 = \frac{1}{6}, g_3 = \frac{1}{3}$.

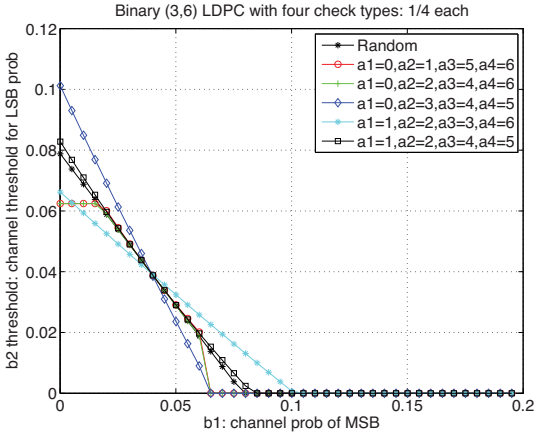


Fig. 12. Four check types, with ratios $g_1 = g_2 = g_3 = g_4 = \frac{1}{4}$.

the check nodes having type $T(5,1)$ or $T(4,2)$ (a majority of MSB neighbors) perform better than the expected BICM case.

## IV. Implementing Nonbinary LDPC Codes in MLC Flash

In this section we will demonstrate how the binary image representation of nonbinary codes facilitates their implementation in MLC flash. We use this implementation in Section 5 to determine how to add edge labels from $\mathrm{GF}(4)$ to $(j,k)$-regular graphs so that the check nodes in the resulting code have types consistent with those that performed well in Section 3.

Previous work on nonbinary LDPC codes has also exploited the binary image of the parity-check matrix. In [20], [12], the authors chose nonzero row entries of a parity-check matrix $H$ using the binary image of field elements to improve performance, assuming that the positions of the nonzero entries of $H$ were already optimized. Binary image expansion techniques were also used in [21] to iteratively decode Reed-Solomon codes. We refer the reader to [22] for details of the binary images of Galois field elements. The following example shows the binary images of elements in $\mathrm{GF}(4)$.

*Example 4.1:* Let $r$ be a root of the primitive polynomial $g(x) = x^2 + x + 1$ over $\mathrm{GF}(2)$. The binary matrix representation of $r \in \mathrm{GF}(4)$ is

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}.$$

The field $\mathrm{GF}(4)$ is composed of the elements $\{0, A, A^2, A^3\}$ (where 0 is the $2 \times 2$ all zero matrix and $A^3$ is the identity matrix), under the operations of of matrix addition and multiplication.

Note that the $\mathrm{GF}(4)$ binary matrix representation is unique, but for $m \geq 3$, the representation of elements in $\mathrm{GF}(2^m)$ depends on the choice of primitive polynomial.

If $H$ is a $l \times n$ parity check matrix over $\mathrm{GF}(2^m)$, the binary image parity-check matrix is the $ml \times mn$ matrix obtained by replacing each entry of $H$ with its $m \times m$ matrix representation. We define the *binary image graph* of a nonbinary code to be the Tanner graph obtained from its binary image parity-check matrix. Note that even if the original graph is regular, the binary image graph is usually irregular.

We now introduce a way to implement codes over $\mathrm{GF}(4)$ in MLC flash using their binary image representation. The binary image graph is treated as a binary LDPC code whose variable nodes represent the bits of the corresponding symbols and are assigned to different page types. Thus, the binary image of a code of length $n$ over $\mathrm{GF}(4)$ gives an immediate mapping of bits to MSB pages and LSB pages. Specifically, if $v_i$ is a variable node in the original graph over $\mathrm{GF}(4)$ for $i = 1, \ldots, n$, then $v_{iM}$ and $v_{iL}$ are the bit nodes in the binary expansion of the symbol represented by $v_i$, and $v_{iM}$ (resp., $v_{iL}$) is assigned to a MSB (resp., LSB) page. To obtain a simple decoder, we will use a binary decoder on the binary

TABLE I
Noise variance and SNR thresholds for $(3,6)$-regular LDPC codes with two given check types.

| Check node types | Frac. of each type | $\sigma$ thres. | SNR thres. |
|---|---|---|---|
| T(1,5), T(5,1) | $1/2, 1/2$ | 0.6189 | 11.1573 |
| T(2,4), T(5,1) | $2/3, 1/3$ | 0.6180 | 11.1699 |
| T(2,4), T(4,2) | $1/2, 1/2$ | 0.6175 | 11.1770 |
| Expected for BICM | Random | 0.6172 | 11.1812 |

TABLE II
Noise variance and SNR thresholds for $(3,6)$-regular LDPC codes with three given check types.

| Check node types | Frac. of each type | $\sigma$ thres. | SNR thres. |
|---|---|---|---|
| T(5,1), T(3,3), T(0,6) | $1/2, 1/6, 1/3$ | 0.6408 | 10.8552 |
| T(0,6), T(4,2), T(5,1) | $1/3, 1/3, 1/3$ | 0.6405 | 10.8593 |
| T(4,2), T(6,0), T(0,6) | $1/2, 1/6, 1/3$ | 0.6323 | 10.9712 |
| T(5,1), T(2,4), T(0,6) | $1/2, 1/4, 1/4$ | 0.6287 | 11.0208 |
| T(4,2), T(0,6), T(3,3) | $1/2, 1/6, 1/3$ | 0.6245 | 11.0791 |
| T(5,1), T(3,3), T(1,5) | $1/3, 1/3, 1/3$ | 0.6183 | 11.1657 |
| T(3,3), T(5,1), T(1,5) | $1/2, 1/4, 1/4$ | 0.6180 | 11.1699 |
| T(2,4), T(5,1), T(3,3) | $1/2, 1/4, 1/4$ | 0.6178 | 11.1727 |
| T(4,2), T(3,3), T(1,5) | $1/2, 1/4, 1/4$ | 0.6177 | 11.1741 |
| T(3,3), T(1,5), T(4,2) | $1/2, 1/6, 1/3$ | 0.6175 | 11.1770 |
| T(3,3), T(5,1), T(2,4) | $1/2, 1/6, 1/3$ | 0.6175 | 11.1770 |
| T(4,2), T(3,3), T(2,4) | $1/3, 1/3, 1/3$ | 0.6174 | 11.1784 |
| T(3,3), T(4,2), T(2,4) | $1/2, 1/4, 1/4$ | 0.6173 | 11.1798 |
| Expected for BICM | Random | 0.6172 | 11.1812 |
| T(3,3), T(6,0), T(0,6) | $1/2, 1/4, 1/4$ | 0.6169 | 11.1854 |
| T(2,4), T(0,6), T(6,0) | $1/2, 1/6, 1/3$ | 0.6117 | 11.2589 |
| T(2,4), T(6,0), T(3,3) | $1/2, 1/6, 1/3$ | 0.6106 | 11.2746 |
| T(1,5), T(6,0), T(4,2) | $1/2, 1/4, 1/4$ | 0.6062 | 11.3374 |
| T(1,5), T(3,3), T(6,0) | $1/2, 1/6, 1/3$ | 0.5986 | 11.4470 |
| T(6,0), T(2,4), T(1,5) | $1/3, 1/3, 1/3$ | 0.5984 | 11.4499 |

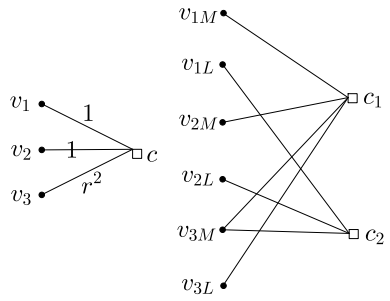


Fig. 13. The left graph has edge labels from $\mathrm{GF}(4)$.

image graph to estimate the LSB and MSB bit values of the nonbinary symbols.

*Example 4.2:* Figure 13 shows the graph of a nonbinary LDPC code over $\mathrm{GF}(4)$ on the left, and its corresponding binary image graph on the right. Each variable and check node in the graph of a nonbinary LDPC code over $GF(2^m)$ splits into $m$ copies in the binary image graph. For MLC flash implementation, we consider codes over $\mathrm{GF}(2^2)$ as shown in Figure 13 and label the copies of a variable node $v_i$ by $v_{iM}$ and $v_{iL}$ to designate the bit to be assigned to a MSB or LSB page, respectively.

The binary image graph of a code over $\mathrm{GF}(4)$ has binary check nodes $c_{i1}$ and $c_{i2}$ for each $\mathrm{GF}(4)$ check node $c_i$ in the graph of the nonbinary code (where $i = 1, \ldots, \ell$). By choosing consistent sets of labels for the edges at each check node in the original graph, all of the check nodes labeled $c_{i1}$ in the binary image graph will have the same type (as defined in Section 2), as will all of the check nodes labeled $c_{i2}$. More specifically, when choosing edge labels for a $(j,k)$-regular graph, we can fix a set of labels $\{r_1, \ldots, r_k\}$, where $r_i \in \mathrm{GF}(4)$ such that at each check node, these $k$ labels are randomly ordered and assigned to its incident edges. Figure 13 shows how the labels from $\mathrm{GF}(4)$ assigned to the edges of a check node give rise to two check node types in the binary image graph. Note that adding nonbinary edge labels to an arbitrary $(j,k)$-regular graph results in a binary image graph with left degrees from the set $\{j, j+1, \ldots, mj\}$ and right degrees from the set $\{k, k+1, \ldots, mk\}$. In particular, in Figure 13 check $c_1$ has type $T(3,1)$ and check $c_2$ has type $T(1,2)$.

*Remark 4.3:* Binary images may be used for nonbinary LDPC codes over $\mathrm{GF}(2^m)$, resulting in up to $m$ different types of check nodes in the binary image graph. Depending on the edge labels, some types may be the same. To design codes for TLC flash memory, the analysis in the previous sections can be extended to binary image graphs with three different check node types. This could then be used to choose edge labels from $GF(2^3)$ that result in the desired three check node types.

## V. Structured bit-interleaved nonbinary LDPC codes

We now examine how different assignments of nonbinary elements from $\mathrm{GF}(4)$ to the edges in an underlying regular LDPC code graph result in different binary image graphs. Using the analysis of the thresholds of $b_1$ and $b_2$ in Section 3, we identify edge label choices that yield graphs and corresponding codes that are suitable for implementation in MLC flash memory. These preferred edge labels (equivalently, assignments of nonbinary elements to nonzero positions in the parity-check matrix) result in constructions of nonbinary LDPC codes over $\mathrm{GF}(4)$ that have structured bit assignments to the MSB and LSB pages and may be decoded using simple binary LDPC decoders on their binary image graphs to estimate the MSB and LSB bits of each symbol.

TABLE III
Edge labels for (3,6)-regular graphs and corresponding check types and degree distributions.

| Edge Label Set: Type 1, Type 2 | $\Delta_M$ (MSB deg. dist.) | $\Delta_L$ (LSB deg. dist.) |
|---|---|---|
| $\{11AAA^2A^2\}$ $T(4,4)$, $T(4,4)$ | $(0,0,\frac{8}{27},\frac{4}{9},\frac{2}{9},\frac{1}{27})$ | $(0,0,\frac{8}{27},\frac{4}{9},\frac{2}{9},\frac{1}{27})$ |
| $\{111AAA^2\}$ $T(4,3)$, $T(3,5)$ | $(0,0,\frac{125}{216},\frac{25}{72},\frac{5}{72},\frac{1}{216})$ | $(0,0,\frac{8}{27},\frac{4}{9},\frac{2}{9},\frac{1}{27})$ |
| $\{111AA^2A^2\}$ $T(5,3)$, $T(3,4)$ | $(0,0,\frac{8}{27},\frac{4}{9},\frac{2}{9},\frac{1}{27})$ | $(0,0,\frac{125}{216},\frac{25}{72},\frac{5}{72},\frac{1}{216})$ |
| $\{111AAA\}$ $T(3,3)$, $T(3,6)$ | $(0,0,1,0,0,0)$ | $(0,0,\frac{1}{8},\frac{3}{8},\frac{3}{8},\frac{1}{8})$ |
| $\{111A^2A^2A^2\}$ $T(3,3)$, $T(6,3)$ | $(0,0,\frac{1}{8},\frac{3}{8},\frac{3}{8},\frac{1}{8})$ | $(0,0,1,0,0,0)$ |
| $\{1111AA^2\}$ $T(5,2)$, $T(2,5)$ | $(0,0,\frac{125}{216},\frac{25}{72},\frac{5}{72},\frac{1}{216})$ | $(0,0,\frac{125}{216},\frac{25}{72},\frac{5}{72},\frac{1}{216})$ |
| $\{1111AA\}$ $T(4,2)$, $T(2,6)$ | $(0,0,1,0,0,0)$ | $(0,0,\frac{8}{27},\frac{4}{9},\frac{2}{9},\frac{1}{27})$ |
| $\{1111A^2A^2\}$ $T(6,2)$, $T(2,4)$ | $(0,0,\frac{8}{27},\frac{4}{9},\frac{2}{9},\frac{1}{27})$ | $(0,0,1,0,0,0)$ |
| $\{11111A\}$ $T(5,1)$, $T(1,6)$ | $(0,0,1,0,0,0)$ | $(0,0,\frac{125}{216},\frac{25}{72},\frac{5}{72},\frac{1}{216})$ |
| $\{11111A^2\}$ $T(6,1)$, $T(1,5)$ | $(0,0,\frac{125}{216},\frac{25}{72},\frac{5}{72},\frac{1}{216})$ | $(0,0,1,0,0,0)$ |

We start with a parity-check matrix whose locations of nonzero entries are known (and possibly optimized), but the values have yet to be determined. For this paper we will illustrate our construction using the parity-check matrix of a random $(3,6)$-regular binary LDPC code and replace the ones in the matrix with structured choices of elements from $\mathrm{GF}(4)$. Note that better codes may be obtained if a parity-check matrix with positions optimized for a nonbinary code is used. As mentioned in Section 4, we will assume that the field elements assigned to the edges at each check node are the same, and randomly assigned to the edges at that check. Let $\Delta_M = (\delta_{M,1}, \delta_{M,2}, \delta_{M,3}, \delta_{M,4}, \delta_{M,5}, \delta_{M,6})$ denote the degree distribution of the MSBs, where $\delta_{M,i}$ is the fraction of MSBs having degree $i$, and likewise for $\Delta_L$. Table III summarizes different sets of such edge labels, and the effect that they have on the resulting MSB degree distribution, LSB degree distribution, and check node types in the corresponding binary image graph.

Using the degree distributions in Table III, we obtain iterative equations for the expected probability of error for messages sent from variable to check nodes using the Gallager A algorithm. Let $q_M^{(t)}$ denote the probability that a check node sends a message in error to a MSB in iteration $t$, as detailed in Section 3. In this setting, $g = 1/2$ and $(\alpha_1, \beta_1)$ and $(\alpha_2, \beta_2)$ are determined by the labeling of edges in the nonbinary graph (see the first column of Table III). Since the variable nodes have degree distributions given by $\Delta_M$

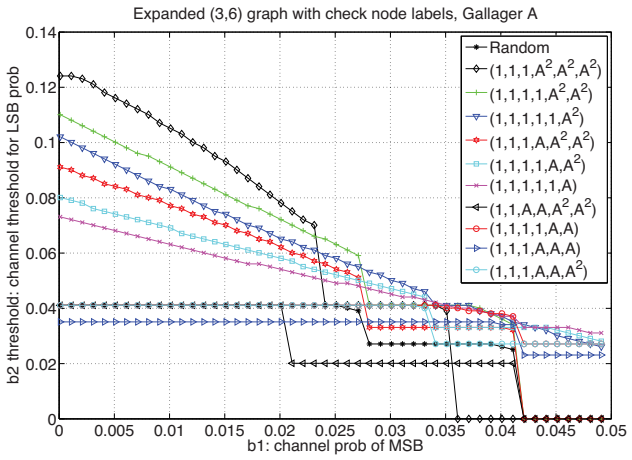


Fig. 14. Thresholds of binary image graph codes obtained from $(3, 6)$-regular graphs using edge label sets from Table III.

and $\Delta_L$, the expressions for the MSB-to-check probability of error in the $(t+1)$th iteration is a weighted sum with $\delta_{M,i}$ coefficients. First, we define the probability that a MSB node of degree $i$ sends a message in error to a neighboring check: $p_{M,i}^{(t+1)} = b_1(1-(1-q_M^{(t)})^{i-1}) + (1-b_1)(q_M^{(t)})^{i-1}$. Thus, the expected probability of error of a MSB-to-check message is given by

$$p_M^{(t+1)} = \sum_{i=1}^{6} \delta_{M,i} p_{M,i}^{(t+1)}. \tag{2}$$

We define $p_L^{(t+1)}$ in the analogous way.

Figure 14 shows the thresholds for the codes represented by the binary image graphs obtained from a $(3, 6)$-regular bipartite graph with the same given edge label set at each check (possibly permuted). For each edge label set in Table III, we ran $t = 100$ iterations of the Gallager A algorithm for fixed values of the MSB error probability $b_1$ to find the threshold for $b_2$.

It is important to note that the different edge label sets result in binary codes with different degree distributions on both the variable nodes and the check nodes. The variable node degree distributions are shown in Table III, whereas half of the check nodes have degree $\alpha_1 + \beta_1$, and the rest have degree $\alpha_2 + \beta_2$. These codes cannot be directly compared to the $(j, k)$-regular codes shown in Section 3; however, the codes shown here may be regarded as slightly irregular, with degrees varying on fixed intervals.

Recall that codes with $g = 1/2$ of check nodes of type $T(1, k-1)$ and half of type $T(k-1, 1)$ were consistently among the best for the binary regular cases tested in Section 3. Thus we expect the codes with edge label sets $\{1, 1, 1, 1, 1, A^2\}$, $\{1, 1, 1, 1, 1, A\}$, and $\{1, 1, 1, 1, A^2, A^2\}$ to be the strongest since their binary images have similar unbalanced check node types. Indeed, the codes corresponding to $\{1, 1, 1, 1, A^2, A^2\}$ and $\{1, 1, 1, 1, 1, A^2\}$ have the second and third best performance in Figure 14, for $0 < b_1 < 0.027$. However, $\{1, 1, 1, 1, 1, A\}$ did not perform as well, possibly due to the fact that the total number of LSB connections exceeds the number of MSB connections. Surprisingly the best performing code was obtained using the edge labels $\{1, 1, 1, A^2, A^2, A^2\}$. While neither check node type in this case has a large difference between $\alpha_i$ and $\beta_i$, the total number of connections to MSB neighbors exceeds the total LSB connections more than any other case tested. The edge label set closest to the random case is $\{1, 1, A, A, A^2, A^2\}$ in Table III, and yields a right-regular binary image graph in which each check node has half MSB and half LSB neighbors. Due to the higher density of edges that results from $A$ or $A^2$ labels, we chose to test check label sets with at least three ones, with the exception of the "random-like" case, $\{1, 1, A, A, A^2, A^2\}$.

The Random curve in Figure 14 refers to a $(3, 6)$-regular graph whose edges are randomly assigned nonzero elements of $\mathrm{GF}(4)$, each with equal probability. In this case, the expressions for $q_M$ and $q_L$ involve the degree distributions resulting from each check node edge label configuration weighted by the probability of the configuration occurring in the graph. Let $\xi$ denote the collection of unordered check node label sets over $\mathrm{GF}(4)\backslash\{0\}$ (Table III contains a partial list). Denote by $s \in \xi$ a multi-set of size six with elements from $\mathrm{GF}(4)\backslash\{0\}$ resulting in binary check types $T(\alpha_{s,1}, \beta_{s,1})$ and $T(\alpha_{s,2}, \beta_{s,2})$. Let $p(s)$ be the probability of edge set $s$, given that the labels are assigned randomly from $\mathrm{GF}(4) \setminus \{0\}$. The density evolution expressions for the random edge assignment case are given by the following.

*Proposition 5.1:* Let $q_{1,M}^{(t)}$ denote the expected probability of error of a message from a check node of type $T(\alpha_{s,1}, \beta_{s,1})$ to a MSB variable node in iteration $t$. Then

$$q_{1,M}^{(t)} = \sum_{s\in\xi} p(s) \left( \frac{1-(1-2p_M^{(t)})^{\alpha_{s,1}-1}(1-2p_L^{(t)})^{\beta_{s,1}}}{2} \right),$$

and likewise for $q_{1,L}^{(t)}, q_{2,M}^{(t)}, q_{2,L}^{(t)}$.

In the binary image of a $(3, 6)$-regular graph whose edges are randomly assigned nonzero elements of $\mathrm{GF}(4)$, the expected probability of error for a message from a check node to a MSB or LSB variable node, respectively, in iteration $t$ is

$$q_M^{(t)} = \frac{1}{2} q_{1,M}^{(t)} + \frac{1}{2} q_{2,M}^{(t)}$$
$$q_L^{(t)} = \frac{1}{2} q_{1,L}^{(t)} + \frac{1}{2} q_{2,L}^{(t)}.$$

Moreover, the expected probability of error for a message from a MSB (resp., LSB) variable node to a check node in iteration $(t+1)$ is given by Equation 2 with $\Delta_M = \Delta_L = (0, 0, \frac{8}{27}, \frac{12}{27}, \frac{6}{27}, \frac{1}{27})$.

Figure 15 shows the analysis for the same codes using the Gallager B algorithm, also run for $t = 100$ iterations. In this setting, the code with structured edge label set $\{1, 1, 1, 1, 1, A^2\}$ outperforms all other codes tested, including the random code with nonzero edge labels assigned with equal probability from $\mathrm{GF}(4) \setminus \{0\}$.

Although we started with a random $(3, 6)$-regular graph without small cycles, the binary expanded graph most likely does contain some small local cycles, due to the introduction of subgraphs from $A$ and $A^2$. Since density evolution assumes the graph is cycle-free, the results in Figures 14 and 15 should be regarded as estimates. These results are still meaningful because the graph may be assumed to be globally cycle-free, as in the case of random regular LDPC codes. Edge label

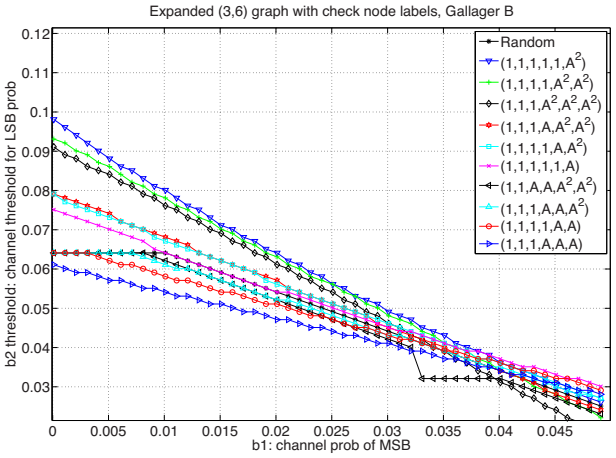


Fig. 15. Thresholds of binary image graph codes obtained from $(3, 6)$-regular graphs under Gallager B decoding.

sets dominated by ones will result in the least number of local cycles in the binary image graph. While an edge label set consisting of all ones yields a disconnected graph, the configurations we considered can be shown to be connected. Certain good edge label choices result in fewer cycles in the binary expanded graph. For example, the edge label set $\{1, 1, 1, 1, 1, A^2\}$ results in a connected Tanner graph in which each variable node has probability $5/72$ of being involved in a four-cycle in the binary expanded graph.

## VI. Conclusion

This paper focused on the design of LDPC codes for MLC flash memory using a structured approach for assigning bits to MSB and LSB pages. We first considered binary $(j, k)$-regular LDPC codes and examined different check node types to determine which had the best decoding threshold using hard decision decoding. Our results showed that a structured assignment of bits to pages can outperform the random case where each check has half MSB and half LSB neighbors on average. In particular, for codes with two check node types, the most gain was observed when each type was highly unbalanced in the number of MSB vs. LSB neighbors. For more than two check node types, we observed that when more than half of the check nodes have a majority of MSB neighbors, the thresholds are higher than in the average BICM case. Out of all cases tested, the best-performing for $(3, 6)$-regular codes was a particular instance of three check types. Using the binary images of codes over $\mathrm{GF}(4)$, we also presented a method for assigning the bits of the symbols of nonbinary LDPC codes to MSB/LSB pages. This method allows one to control the check types and corresponding degree distributions of the resulting binary LDPC graph that is then amenable to efficient binary LDPC decoding. We analyzed configurations of field elements on the edges of $(3, 6)$-regular codes to determine which gave the best decoding thresholds when used in this way, resulting in structured bit-interleaved nonbinary LDPC codes.

The work presented here may be extended in several ways. Our approach for three check types may be applied to design structured bit-interleaved codes for TLC flash, taking into account the effect of MSB, CSB and LSB pages. Another interesting problem stemming from this work is the question of how to assign bits to pages in existing algebraic binary LDPC codes (such as those in [23]) to obtain desired check node types. We are currently analyzing the performance of the codes designed in Section 5 under nonbinary decoding. Finally, we note that the results presented here also apply to any coding problem involving different initial channel bit error probabilities.

## VII. Appendix

*Proof: (Lemma 3.2)* Assume that $c$ has $\alpha$ MSB neighbors and $\beta$ LSB neighbors. If $c$ is sending a message to a MSB neighbor $v_M$, then $x_1 = \alpha - 1$ and $x_2 = \beta$. Likewise, if the message is being sent from $c$ to $v_L$, then $x_1 = \alpha$ and $x_2 = \beta - 1$. Denote by $p_{Y_i}^{(t)}$ the probability that the $i$th neighbor, $v_{Y_i}$, of $c$ sends an incorrect message on the $t$th iteration, where $Y_i \in \{M, L\}$ for $i = 1, \ldots, k-1$. Note that $c$ sends an incorrect message exactly when an odd number of neighboring variable nodes are incorrect. Let $g(x)$ be the generating function in which the coefficient of $x^l$ records the probability that exactly $l$ neighbors of $c$ are incorrect on iteration $t$:

$$g(x) = \prod_{i=1}^{k-1} ((1 - p_{Y_i}^{(t)}) + p_{Y_i}^{(t)} x).$$

The function $\frac{g(x)+g(-x)}{2}$ yields precisely the even powers of $x$. Substituting $x = 1$ into this expression gives the probability that an even number of neighbors of $c$ send incorrect messages. Thus, the probability that $c$ is incorrect on the $t$th iteration is

$$\begin{aligned} 1 - \frac{g(1) + g(-1)}{2} &= 1 - \frac{1 + \prod_{i=1}^{k-1}(1 - 2p_{Y_i}^{(t)})}{2} \\ &= \frac{1 - \prod_{i=1}^{k-1}(1 - 2p_{Y_i}^{(t)})}{2} \\ &= \frac{1 - (1 - 2p_M^{(t)})^{x_1}(1 - 2p_L^{(t)})^{x_2}}{2}. \end{aligned}$$

The second part of the Lemma deals with the probability of an erroneous message being sent from a variable node to a check node under the Gallager A algorithm on iteration $(t + 1)$. A variable node sends an incorrect message if either (1) the channel information is incorrect and at least one of the incoming check node messages is incorrect, or (2) the channel information is correct and all incoming check messages agree and are incorrect. The expressions $p_M^{(t+1)}$ and $p_L^{(t+1)}$ give exactly these probabilities for $v_M$ and $v_L$, respectively. ■

*Proof: (Proposition 5.1)* The expression $q_M^{(t)} = \frac{1}{2} q_{1,M}^{(t)} + \frac{1}{2} q_{2,M}^{(t)}$ requires justification. Given a random edge label from $\mathrm{GF}(4) \setminus \{0\}$ on an edge $\{v, c\}$ in the nonbinary $(3, 6)$-regular graph, the binary image graph contains one of the corresponding subgraphs shown in Figure 16. Since each of the labels $1$, $A$, and $A^2$ has equal probability of being assigned to $\{v, c\}$, the probability that $v_M$ is adjacent to $c_1$ in the binary image graph is the same as the probability that $v_M$ is adjacent $c_2$ (both are $2/3$, since these events are not independent). Similarly, $v_L$ is equally likely to have $c_1$ as a neighbor as

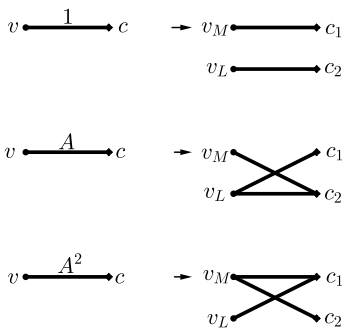


Fig. 16. Nonzero $\mathrm{GF}(4)$ edge labels and the corresponding subgraphs.

$c_2$. More precisely, we have that the probability of a random check node sending an incorrect message to a MSB node is

$$q_M^{(t)} = \frac{1}{3}q_{1,M}^{(t)} + \frac{1}{3}q_{2,M}^{(t)} + \frac{1}{3}\left(\frac{1}{2}q_{1,M}^{(t)} + \frac{1}{2}q_{2,M}^{(t)}\right)$$
$$= \frac{1}{2}q_{1,M}^{(t)} + \frac{1}{2}q_{2,M}^{(t)}.$$

The rest of the proof is straightforward. ■

## REFERENCES


[1] H. Weingarten, "New strategies to overcome 3bpc challenges", *Flash Memory Summit*, Santa Clara, August 2010.

[2] E. Yaakobi et al., "Characterization and error-correcting codes for TLC flash memories", IEEE International Conference on Computing, Networking and Communications, Maui, HI, Jan.-Feb. 2012.

[3] R. Gabrys, E. Yaakobi, and L. Dolecek, "Graded bit-error-correcting codes with applications to flash memories", *IEEE Trans. Inf. Theory*, April 2013.

[4] E. Yaakobi et al., "Error characterization and coding schemes for flash memories", IEEE GLOBECOM Workshops, Dec 2010.

[5] L. Grupp et al., "Characterizing flash memory: anomalies, observations, and applications", *MICRO-42*, pp. 24-33, Dec. 2009.

[6] R. G. Gallager, "Low density parity check codes," *IRE Trans. Information Theory*, Jan. 1962.

[7] Y. Maeda and H. Kaneko, "Error control coding for multilevel cell flash memories using nonbinary low-density parity-check codes", *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, Oct. 2009.

[8] F. Zhang, H. D. Pfister, A. Jiang, "LDPC codes for rank modulation in flash memories," *Proceedings of the IEEE International Symposium on Information Theory*, Austin, TX, June 2010.

[9] M. C. Davey and D. J. C. MacKay, "Low-density parity-check codes over $GF(q)$", *IEEE Communications Letters*, vol. 2, pp. 165-167, June 1998.

[10] C. A. Kelley, D. Sridhara, and J. Rosenthal, "Pseudocodeword weights for nonbinary LDPC codes", *Proceedings of the IEEE International Symposium on Information Theory*, Seattle, July 9-14, 2006.

[11] D. Divsalar and L. Dolecek, "Graph Cover Ensembles of Non-binary Protograph LDPC Codes," in *Proceedings of the IEEE International Symposium on Information Theory*, Boston, MA, July 2012.

[12] C. Poulliat, M. Fossorier, D. Declercq, "Design of regular $(2, d_c)$-LDPC codes over $\mathrm{GF}(q)$ using their binary images," *IEEE Trans. Inf. Theory*, vol. 58, no. 10, October 2008.

[13] H. Pishro-Nik, N. Rahnavard, F. Fekri, "Nonuniform error correction using low-density parity-check codes," *IEEE Trans. Inf. Theory*, July 2005.

[14] V. Kumar and O. Milenkovic, "On unequal error protection LDPC codes based on plotkin-type constructions," *IEEE Trans. Inf. Theory*, June 2006.

[15] C. Poulliat, D. Declercq, I. Fijalkow, "Enhancement of unequal error protection properties of LDPC codes," *EURASIP Journal on Wireless Communications and Networking*, Volume 2007.

[16] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity approaching low-density parity-check codes", *IEEE Trans. Inf. Theory*, vol. 47, pp. 619-637, Feb, 2001.

[17] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding", *IEEE Trans. Inf. Theory*, vol. 47, pp. 599-618, Feb, 2001.

[18] M.G. Luby et al., "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, Feb. 2001.

[19] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, Sept. 1981.

[20] C. Poulliat, M. Fossorier, D. Declercq, "Design of nonbinary LDPC codes using their binary image: algebraic properties," *Proc. IEEE International Symposium on Information Theory*, July 2006.

[21] J. Jiang and K. R. Narayanan, "Iterative Soft Input Soft Output Decoding of Reed-Solomon Codes", *IEEE Trans. Inf. Theory*, pp. 3746-3756, Vol. 52, No.8, August 2006

[22] F. J. MacWilliams, N. J. A. Sloane, *The theory of error- correcting codes.* North-Holland, Amsterdam, New York, Oxford, 1977.

[23] Y. Kou, S. Lin, M.P.C. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovering and new results," *IEEE Trans. Inf. Theory*, Nov. 2001.


**Kathryn Haymaker** is a Ph.D. candidate in the Department of Mathematics at the University of Nebraska-Lincoln. She received the B.A. degree in Mathematics from Bryn Mawr College in 2007, and the M.S. degree in Mathematics from the University of Nebraska-Lincoln in 2010. Her research interests include combinatorial and algebraic coding theory and applied discrete mathematics.

**Christine Kelley** received the M.S. and Ph.D. degrees in Mathematics at the University of Notre Dame in 2003 and 2006, respectively. She was a Postdoctoral Fellow at the Fields Institute in Toronto and the Department of Mathematics at the Ohio State University. Since August 2008 she has been in the Department of Mathematics at the University of Nebraska-Lincoln, where she is now an Associate Professor. Her main areas of research interest are graph-based and algebraic coding theory, and applied discrete mathematics. In Spring 2010, Dr. Kelley received the University of Nebraska's Harold and Esther Edgerton Junior Faculty Award for "creative research, extraordinary teaching abilities, and academic promise".