

Analytical Performance of One-Step Majority Logic Decoding of Regular LDPC Codes

Rathnakumar Radhakrishnan*, Sundararajan Sankaranarayanan†, and Bane Vasić*

*Department of Electrical and Computer Engineering

University of Arizona, Tucson, AZ-85705

Email: {rathna, vasic}@ece.arizona.edu

†Seagate Technology, Pittsburgh, PA-15222

Email: ssankara@seagate.com

Abstract—In this paper, we present a combinatorial algorithm to calculate the exact bit error rate performance of regular low-density parity check codes under one-step majority logic decoding. Majority logic decoders have regained importance in nano-scale memories due to their resilience to both memory and logic gate failures. This result is an extension of the work of Rudolph on error correction capability of majority-logic decoders.

I. INTRODUCTION

With the decreasing transistor size, it is becoming increasingly possible to develop small, fast and efficient memory chips. As the demand for higher memory densities continues, a wide range of new nano-scale technologies is being actively investigated. It is widely recognized that due to their miniature size and variations in a technological process, the nano-components will be inherently unreliable. The main challenge in building stable memories in nano-scale systems is both faulty storage elements and faulty logic gates of the error-correction circuitry. It is in contrast to the state-of-the-art systems where only the memory elements are considered unreliable while error correction encoders and decoders are assumed to be made of reliable logic gates. An interesting consequence of the presence of an unreliable decoder is that complex decoding algorithms requiring complex Boolean circuits need not necessarily perform better than low complexity decoders. This has renewed interest in majority logic decoders.

In this paper we consider a Taylor-Kuznetsov low-density parity check (LDPC) coded memory system [1], [2] with a one-step majority logic decoder, and as the first step, we consider the case of perfect logic gates. This problem can be reduced to a problem of error correction capability of the one-step majority logic decoder considered by Rudolph [4], but surprisingly, his result has so far not been extended to determine the bit error rate (BER) of LDPC codes. We present a combinatorial algorithm to calculate the exact bit error rate performance of the one-step majority logic decoder on a regular, four-cycle free LDPC code over the binary symmetric channel (BSC), which is used to model failures of memory elements. We explain how the total number of error configurations that will result in a decoding error can be calculated efficiently. Using this algorithm, BER of several LDPC codes derived from finite geometry is determined. The

results presented here complete our recent work in analysis of one-step majority logic decoders constructed with faulty gates [3]. The paper is organized as follows. We start with an explanation of one-step majority logic decoder and present the generalized algorithm in Section II. We apply this algorithm to several codes in Section III and finally conclude the paper in Section IV.

II. ONE-STEP MAJORITY LOGIC DECODER

The one-step majority logic decoding algorithm is briefly explained below. Let \mathbf{H} be the parity check matrix of a (n, k) regular linear code C with column weight γ and row weight ρ . Let $\mathbf{v}=[v_1, v_2, \dots, v_{n-1}]$ be the sent codeword and $\mathbf{r} = \mathbf{v} + \mathbf{e}$ be the received vector, where $\mathbf{e}=[e_0, e_1, \dots, e_{n-1}]$ is the error vector. Every variable node or code bit is involved in γ parity check sum equations. If $\mathbf{e} \neq 0$, then some of the parity check sum equations may not be zero, i.e. $\mathbf{r} \cdot \mathbf{H}$ may not be a zero vector. Let the γ parity check equations for a variable node v_j

$$\mathbf{P} = [p_1, p_2, p_3, \dots, p_\gamma]. \quad (1)$$

If code C has no four cycles, then the only variable node common to all of these checks is v_j . These checks are said to be *orthogonal* to the variable node v_j or correspondingly, to the j^{th} error position. Among the variable nodes involved in the γ parity check equations, if there are $\lfloor \frac{\gamma}{2} \rfloor$ or fewer errors, then majority of the parity check equations in \mathbf{P} are equal to e_j , irrespective of whether there is an error or not at the j^{th} position. Similarly, there exists γ orthogonal parity check equations for all n variable nodes. Therefore, the one-step majority logic decoding is simply described as follows,

$$v_j = r_j \oplus \text{majority}(p_1, p_2, p_3, \dots, p_\gamma) \quad \forall j = 1, 2, \dots, n \quad (2)$$

where, p_i here indicates the parity of the corresponding check equation and the operation \oplus indicates modulo-2 summation.

One-step majority logic decoder can also be explained as a decoder operating on the Tanner graph of the code C . For example, consider Fig. 1, the part of the Tanner graph corresponding to the node v_j . All the variable nodes send their received value to all the corresponding checks. When decoding v_j , check p_i sends the sum (modulo-2) of all the incoming messages (except v_j) to v_j . Similarly, all other checks orthogonal to v_j send their respective messages.

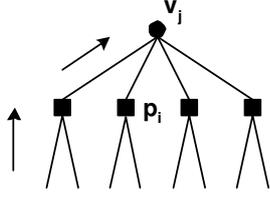


Fig. 1. One-step MLG operating on the Tanner graph

Finally, v_j is decoded as the majority of all the incoming messages. This procedure is carried out for all n variable nodes.

For such a decoder, the bit error rate of a code can be computed analytically. Though we consider only codes from BIBD in this paper, this method is generic and can be applied to any structured code with a minimum girth of at least 6. The probability of a bit being in error after decoding can be determined as follows

$$P_b = \sum_{N_e} \Pr(\text{bit decoded incorrectly} | N_e \text{ errors}) \cdot \Pr(N_e \text{ errors}) \quad (3)$$

where N_e is the number of errors in a codeword. For sake of simplicity, let b_0 and \hat{b}_0 be the channel and decoder output of the bit and let its transmitted value be 0. Then,

$$P_b = \sum_{N_e} \left[\Pr(\hat{b}_0 = 1 | \{N_e \text{ errors}, b_0 = 0\}) \cdot \Pr(b_0 = 0 | N_e \text{ errors}) + \Pr(\hat{b}_0 = 1 | \{N_e - 1 \text{ other errors}, b_0 = 1\}) \cdot \Pr(b_0 = 1 | N_e \text{ errors}) \right] \cdot \Pr(N_e \text{ errors}), \quad (4)$$

where, by $(N_e - 1)$ other errors, we mean the number of errors not counting the error in b_0 itself. Now, we describe how the expressions $\Pr(\hat{b}_0 = 1 | \{N_e - 1 \text{ other errors}, b_0 = 1\})$ and $\Pr(\hat{b}_0 = 1 | \{N_e \text{ errors}, b_0 = 0\})$ can be calculated combinatorially.

The neighboring variable nodes of b_0 is defined as the set of nodes that has at least one check in common with b_0 . We introduce the following notation:

- N_v : the number of neighboring variable nodes of b_0 ,
- \bar{N}_v : the number of non-neighboring variable nodes of b_0 ,
- N_v^e : the number of neighboring variable nodes in error,
- \bar{N}_v^e : the number of non-neighboring variable nodes in error.

Using this notation, the terms of Eqn. (4) can be further expanded as,

$$\Pr(\hat{b}_0 = 1 | \{N_e \text{ errors}, b_0 = 0\}) = \sum_{i=0}^{\bar{N}_v} \Pr(\hat{b}_0 = 1 | \{\bar{N}_v^e = i, N_v^e = N_e - i, b_0 = 0\}) \cdot \Pr(\bar{N}_v^e = i | \{N_e \text{ errors}, b_0 = 0\}). \quad (5)$$

$$\Pr(\hat{b}_0 = 1 | \{N_e - 1 \text{ other errors}, b_0 = 1\}) = \sum_{i=0}^{\bar{N}_v} \Pr(\hat{b}_0 = 1 | \{\bar{N}_v^e = i, N_v^e = N_e - 1 - i, b_0 = 1\}) \cdot \Pr(\bar{N}_v^e = i | \{N_e - 1 \text{ other errors}, b_0 = 1\}), \quad (6)$$

The errors in the received word is partitioned into two sets - errors that occur among the neighboring variable nodes and that occur among non-neighboring variable nodes. Only the errors in the neighboring nodes have an effect on the decision on b_0 . For four-cycle free codes, there can be at most one check in common between two variable nodes. Therefore, the number of neighboring variable nodes of b_0 is $\gamma(\rho - 1)$. For a given N_e , \bar{N}_v^e and b_0 , N_v^e is fixed and are distributed among these nodes and affect γ check equations. Depending on how the γ check equations are affected, some error patterns may induce incorrect decision on b_0 and some may induce correct decision on b_0 . The total number of error patterns is,

$$\binom{\gamma(\rho - 1)}{N_v^e}. \quad (7)$$

But, it is cumbersome to determine whether \hat{b}_0 is incorrect or not for each of these cases individually. Instead, we calculate the probability by considering all *valid* integer partitions of N_v^e . Let a partition be denoted as $[q_1 q_2 q_3 \dots q_{N_v^e}]$, i.e. $N_v^e = 1 \cdot q_1 + 2 \cdot q_2 + \dots + N_v^e \cdot q_{N_v^e}$. We define a partition to be valid, if

- 1) $\forall i > (\rho - 1), q_i = 0$
- 2) $q = \sum_{i=1}^{N_v^e} q_i \leq \gamma$.

Let one such valid partition be denoted as $[q_1 q_2 q_3 \dots q_{\rho-1}]$. This partition is interpreted as an error pattern with q_1 checks having 1 error, q_2 checks having 2 errors and so on. There are many error configurations with the same pattern (or partition), but all of them will result in the same \hat{b}_0 . There are q checks that have at least one error and these are among the γ checks connected to b_0 . Therefore, there are $\binom{\gamma}{q}$ ways of selecting these q checks. Given the q checks, the number of error configurations that will result in the same partition is $\binom{q!}{q_1! q_2! \dots q_{\rho-1}!}$. Given the number of errors that occur in a check, they may occur anywhere among the $\rho - 1$ nodes connected to the check (other than b_0). Also, \bar{N}_v^e errors may occur anywhere among the $n - (\gamma(\rho - 1) + 1)$ non-neighboring nodes of b_0 . Therefore, the total number of error configurations that result in the same partition is,

$$\binom{\gamma}{q} \binom{q!}{q_1! q_2! \dots q_{\rho-1}!} \prod_{j=1}^{\rho-1} \binom{\rho - 1}{j}^{q_j} \binom{n - \gamma(\rho - 1) - 1}{\bar{N}_v^e}. \quad (8)$$

The number of unsatisfied checks is the total number of checks with odd number of errors in the partition. Therefore, the decision on b_0 is determined as follows,

$$\hat{b}_0 = \begin{cases} 0 & \text{if } \sum_{i=1,3,\dots} q_i < \frac{\gamma}{2} \\ 1 & \text{otherwise} \end{cases} \quad (9)$$

If γ is even, then there is a possibility of a tie (equal number of satisfied and unsatisfied checks). For such cases, $\hat{b}_0 = b_0$.

Equation (8) can be calculated for all possible values of \bar{N}_v^e and their sum is the total number of error configurations for a given N_e . It also gives, the number of such configurations resulting in incorrect \hat{b}_0 . Thus, $\Pr(\hat{b}_0 = 1|\{N_e - 1 \text{ other errors, } b_0 = 1\})$ and $\Pr(\hat{b}_0 = 1|\{N_e \text{ errors, } b_0 = 0\})$ can be determined. Also,

$$\Pr(b_0 = 1|N_e \text{ errors}) = e = \frac{\binom{n-1}{N_e-1}}{\binom{n}{N_e}} = \frac{N_e}{n} \quad (10)$$

$$\Pr(b_0 = 0|N_e \text{ errors}) = 1 - e$$

Equation. (4) can be calculated for all values of N_e and consequently, for a given channel error rate ($\Pr(N_e \text{ errors})$), exact bit error rate for any four-cycle free code can be analytically determined. For sake of clarity the algorithm is summarized below,

Algorithm:

- 1) Calculate t , the error correcting capability of the one-step majority logic decoder for the given code. For long codes, the maximum value of N_e can be chosen based on the required accuracy. Typically, $t + 15$ is enough for practical purposes. Therefore, assume $t + 1 \leq N_e \leq t + 15$.
- 2) For every N_e , $0 \leq \bar{N}_v^e \leq \bar{N}_v$. First consider the case when the bit b_0 is not in error. For every N_e^e corresponding to \bar{N}_v^e , determine all valid partitions.
- 3) Calculate the total number of error configurations with N_e errors and the number of such configurations resulting in an incorrect decision on b_0 using Eqn. (8) and (9) for every partition. Their ratio gives $\Pr(\hat{b}_0 = 1|\{N_e \text{ errors, } b_0 = 0\})$. The same procedure is followed for the case when b_0 is in error and $\Pr(\hat{b}_0 = 1|\{N_e - 1 \text{ other errors, } b_0 = 1\})$ can be similarly determined.
- 4) Use Eqn. (10) and (4) to determine $\Pr(\hat{b}_0 = 1|N_e \text{ errors})$.
- 5) For a given channel error rate ($\Pr(N_e \text{ errors})$) use Eqn. (3) to determine the bit error rate for the given code.

III. PERFORMANCE PREDICTION OF CODES FROM FINITE GEOMETRY

LDPC codes designed from finite geometry form an important class of majority-logic decodable codes [5]. Finite geometry is a family of balanced incomplete block design (BIBD). We give a brief exposition of BIBD here. A BIBD is defined as a collection of k -subsets of a v -set P , $k < v$, such that every pair of elements of P occur together in exactly λ of the k -subsets. Each k -subset is called a *block* and each element of P is called a *point*. A BIBD is referred to as a design with parameters $2-(v, k, \lambda)$. The design is said to be balanced because each pair of points occur together in exactly λ of the blocks and is said to be incomplete because not all

possible k -subsets of points are blocks. General information on BIBDs can be found in [6] and [7].

The *incidence matrix* of a $2-(v, k, \lambda)$ design with b blocks is a $b \times v$ matrix $\mathbf{A} = (a_{ij})$ such that a_{ij} is 1 if the i^{th} block contains the j^{th} point or 0 otherwise. The parity-check matrix \mathbf{H} of a code from BIBD is the transpose of the incidence matrix \mathbf{A} . The parity-check matrix obtained from the design has uniform column and row weights. Fossorier [8] proposed the construction of LDPC codes from the incidence structures of finite geometries which are members of BIBD's.

Using the derivation given in [4] by Rudolph, it is possible to compute the error correction capability of a one-step majority logic decoder on a code derived from $2-(v, k, 1)$ designs.

Theorem 1 (Extension of Rudolph's Result): For a code from $2-(v, k, 1)$ designs with column weight γ , one-step majority logic decoder can correct up to $\frac{\gamma-1}{2}$ errors, if γ is odd, and $\frac{\gamma}{2}$ errors, if γ is even.

A. Projective Geometry (PG) Codes

The projective geometry $\text{PG}(2, q = 2^m)$ codes considered in this paper are constructed from the incidence matrix of $2-(q^2 + q + 1, q + 1, 1)$ designs, where q is a power of prime. The column weight of a $\text{PG}(2, 2^m)$ code is $q + 1$ and its minimum distance is $d_{\min} = q + 2$. Therefore, for codes from $\text{PG}(2, 2^m)$, the decoder can correct all error patterns up to weight $t = \lfloor \frac{q+1}{2} \rfloor = \lfloor \frac{d_{\min}-1}{2} \rfloor$. The row weight is same as the column weight. Therefore, the number of neighboring variable nodes of any node is

$$\gamma(\rho - 1) = q^2 + q = n - 1.$$

Therefore, for the $\text{PG}(2, 2^m)$ code, all variable nodes are neighbors of each other. To show how the probability of bit error can be calculated combinatorially, we start with a simple example.

Example: Consider the $\text{PG}(2, 2)$ code. It is a $(7, 7)$ regular code with both column and row weight equal to 3. Its \mathbf{H} matrix is as below,

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

The one-step majority logic decoder can correct one error. In order to compute the bit error rate, Eqn. (4) needs to be calculated for $1 < N_e < 8$. Let the node to be decoded be denoted as b_0 and for the purpose of this example, let $N_e = 5$. First let us consider the cases where b_0 is not in error. Therefore, all the 5 errors have to be distributed among the other 6 nodes or equivalently among the 3 checks associated to b_0 . The only way this can happen is by having one error in one of the checks and two errors each on the other two checks, which is denoted by the partition [1 2]. It is easy to

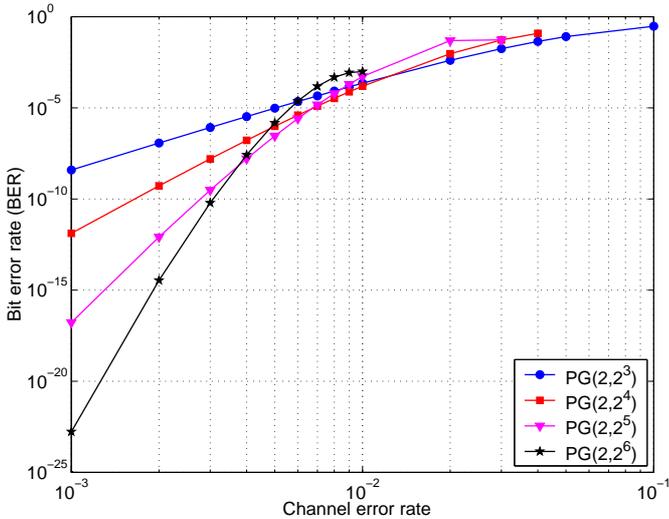


Fig. 2. Analytically calculated BER for codes from $PG(2,2^m)$ when decoded using one-step majority logic decoder

see that there are 6 ways this configuration can occur. For this specific case, the bit b_0 is decoded correctly. Therefore,

$$\Pr(\hat{b}_0 = 1 | \{5 \text{ errors}, b_0 = 0\}) = 0.$$

Similarly, consider the cases where b_0 is in error. Now, 4 errors are distributed among three checks resulting in two possible partitions [0 2] and [2 1]. In the former, b_0 will be decoded correctly and in the latter b_0 will be decoded incorrectly. There are 3 and 12 error configurations that can lead to the two partitions respectively. Therefore,

$$\Pr(\hat{b}_0 = 1 | \{4 \text{ other errors}, b_0 = 1\}) = \frac{12}{15}$$

Given the channel error rate, calculation of $\Pr(b_0 = 1 | N_e \text{ errors})$, $\Pr(b_0 = 0 | N_e \text{ errors})$ and $\Pr(N_e \text{ errors})$ are straightforward. Eqn. (4) can be calculated for all values of N_e and thus the bit error rate can be determined.

Using the above method, performance prediction of various PG codes were determined for various channel error rates of a BSC and are as shown in Fig. 2.

B. Affine Geometry (AG) Codes

The affine geometry $AG(2, q = 2^m)$ codes considered in this paper are constructed from the incidence matrix of $2-(q^2, q, 1)$ designs, where, as before, q is a power of prime. The column weight of an $AG(2, 2^m)$ is q and its minimum distance is $d_{\min} = q + 1$. Therefore, for codes from $AG(2, 2^m)$, the decoder can correct all error patterns up to weight $t = \lfloor \frac{q}{2} \rfloor = \lfloor \frac{d_{\min} - 1}{2} \rfloor$. The row weight is $q + 1$. Therefore, the number of neighboring variable nodes of any node is

$$\gamma(\rho - 1) = q^2.$$

Therefore, unlike $PG(2, 2^m)$ codes, nodes of $AG(2, 2^m)$ codes have $q - 1$ non-neighboring nodes. The method for calculating bit error rate for this code is same as before, except that the

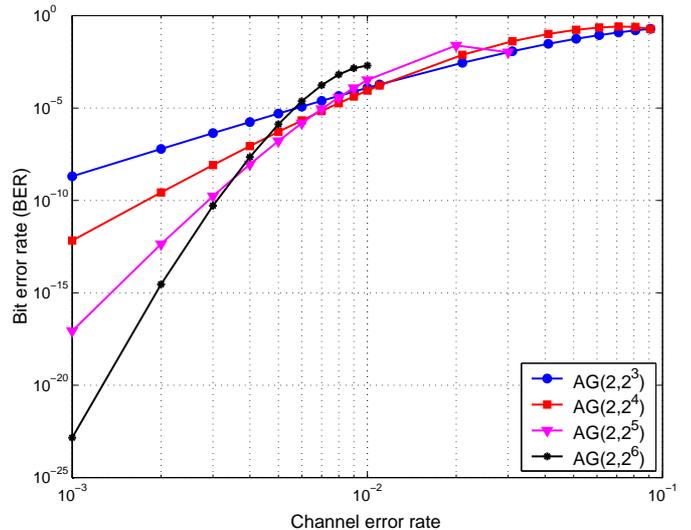


Fig. 3. Analytically calculated BER for codes from $AG(2,2^m)$ when decoded using one-step majority logic decoder

possibility of errors occurring in the non-neighboring nodes need to be taken into account. The performance prediction of various AG codes were determined for various channel error rates of a BSC and are as shown in Fig. 3.

IV. CONCLUSION

Unreliability of memory elements and logic gates, due to inherent failures of nano-components, has renewed interest in very low complexity majority logic decoders. We recently analyzed the decoding failures of one-step majority logic decoders constructed from faulty gates, which can be modeled as a BSC. In this paper, we presented a combinatorial method to compute the exact bit error rate of one-step majority logic decoder on the binary symmetric channel (BSC) for regular, four-cycle free LDPC codes. It constitutes an efficient methodology to determine the number of error patterns that result in a decoding error. This algorithm was applied to several LDPC codes designed from projective and affine geometry and their bit error rates were determined.

V. ACKNOWLEDGEMENT

This work was supported by Seagate Technology and the NSF under grant CCF-0634969.

REFERENCES

- [1] M. Taylor, "Reliable Information Storage in Memories Designed from Unreliable Components", Bell System Technical Journal, 47, pp 2299-2337, 1968.
- [2] A. Kuznetsov, "Information Storage in a Memory Assembled from Unreliable Components", Problems of Information Transmission, 9, pp. 254-264, 1973.
- [3] S. Chilappagari, M. Ivkovic, B. Vasic, "Analysis of one step majority logic decoders constructed from faulty gates", presented at *International Sym. on Information Theory*, Seattle, Jul. 2006.
- [4] L. D. Rudolf, "A class of majority logic decodable codes", *IEEE Trans. Information Theory*, Vol. 13, pp. 305-307, Apr. 1967.
- [5] R. Lucas, M.P.C. Fossorier, Y. Kou, S. Lin, "Iterative decoding of one-step majority logic deductible codes based on belief propagation", *IEEE Trans. on Communications*, Vol. 48, No. 6, pp. 931-937, Jun. 2000.

- [6] E. F. Assmus Jr. and J. D. Key, *Design and Their Codes*, Cambridge, U.K., Cambridge Univ. Press, 1992.
- [7] J. Colbourn, and J. H. Dinitz, *The handbook of combinatorial designs*, Boca, Raton, FL, CRC Press, 1996.
- [8] Y. Kou, S. Lin, M. P. C. Fossorier, "Low-density parity-check codes based on Finite geometries: A rediscovery and new results", *IEEE Trans. Information Theory*, Vol. 47, No. 7, pp. 2711-2736, Nov. 2001.