# 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 one of our recent works [1], we considered a Taylor-Kuznetsov low-density parity check (LDPC) coded memory system [2], [3] with a one-step majority logic decoder constructed with faulty gates. For 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 regular, four-cycle free LDPC codes 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 the work in [1]. There are many papers on the performance of LDPC codes under iterative decoding, most notably the work by Richardson and Urbanke [5]. But, the analysis in general is for average performance of an ensemble of codes. For a given code, its performance under iterative decoding cannot yet be determined analytically. For the case of low complexity hard decision decoders, there has been some work that determines undecodable error patterns for a specific code. For example, in [6], for a special class of Euclidean geometry codes, the authors determine such patterns for the bit-flipping algorithm.

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 $\gamma$ and row weight $\rho$. Let $\mathbf{v}$=$[v_0, v_1, v_2, .., v_{n-1}]$ be the sent codeword and $\mathbf{r}$ = $\mathbf{v}$ + $\mathbf{e}$ be the received vector, where $\mathbf{e}$=$[e_0, e_1, ..., e_{n-1}]$ is the error vector. Every variable node or code bit is involved in $\gamma$ 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 $\gamma$ parity check equations for a variable node $v_j$ be

$$\mathbf{P} = \left[p_1^{(j)}, p_2^{(j)}, p_3^{(j)}, ..., p_\gamma^{(j)}\right]. \tag{1}$$

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. Similarly, there exists $\gamma$ orthogonal parity check equations for all $n$ variable nodes. Therefore, the one-step majority logic decoding is simply described as follows,

$$\hat{v}_j = r_j \oplus \text{majority}\left(p_1^{(j)}, p_2^{(j)}, p_3^{(j)}, ..., p_\gamma^{(j)}\right) \ \forall j = 1, 2, \ldots, n \tag{2}$$

where, $p_i^{(j)}$ indicates the $i^{th}$ check node parity of the $j^{th}$ bit node and the operation $\oplus$ indicates modulo-2 summation.

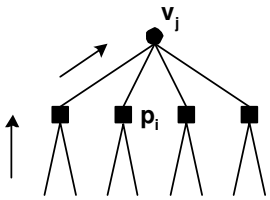


Fig. 1. One-step MLG operating on the Tanner graph

One-step majority logic decoder can also be explained as a decoder operating on the Tanner graph of the code. 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 $p_i^{(j)}$ sends the sum (modulo-2) of all the incoming messages (except from $v_j$) to $v_j$. Similarly, all other checks orthogonal to $v_j$ send their respective messages. 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. This method is generic and can be applied to any code with a minimum girth of at least 6. In this paper, we apply this algorithm to several LDPC codes designed from projective and affine geometries.

The probability of a bit being in error after decoding can be determined as follows,

$$P_{\mathrm{b}} = \sum_{N_e} \Pr(\text{bit decoded incorrectly} | N_e \text{ errors}) \cdot \Pr(N_e \text{ errors}) \tag{3}$$

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,

$$\begin{aligned} P_{\mathrm{b}} = \sum_{N_e} \Big[ & \Pr(\hat{b}_0 = 1 | \{N_e \text{ errors}, b_0 = 0\}) \cdot \\ & \Pr(b_0 = 0 | N_e \text{ errors}) + \\ & \Pr(\hat{b}_0 = 1 | \{N_e - 1 \text{ other errors}, b_0 = 1\}) \cdot \\ & \Pr(b_0 = 1 | N_e \text{ errors}) \Big] \cdot \Pr(N_e \text{ errors}), \end{aligned} \tag{4}$$

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:

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

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

$$\begin{aligned} & \Pr\left(\hat{b}_0 = 1 \middle| \{N_e \text{ errors}, b_0 = 0\}\right) = \\ & \sum_{i=0}^{\bar{N}_v} \Pr\left(\hat{b}_0 = 1 \middle| \{\bar{N}_v^e = i, N_v^e = N_e - i, b_0 = 0\}\right) \cdot \\ & \qquad \Pr\left(\bar{N}_v^e = i \middle| \{N_e \text{ errors}, b_0 = 0\}\right). \end{aligned} \tag{5}$$

$$\begin{aligned} & \Pr\left(\hat{b}_0 = 1 \middle| \{N_e - 1 \text{ other errors}, b_0 = 1\}\right) = \\ & \sum_{i=0}^{\bar{N}_v} \Pr\left(\hat{b}_0 = 1 \middle| \{\bar{N}_v^e = i, N_v^e = N_e - i - 1, b_0 = 1\}\right) \cdot \\ & \qquad \Pr\left(\bar{N}_v^e = i \middle| \{N_e - 1 \text{ other errors}, b_0 = 1\}\right), \end{aligned} \tag{6}$$

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 of $b_0$. 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 $N_v = \gamma(\rho - 1)$. For a given $N_e, \bar{N}_v^e$ and $b_0$, $N_v^e$ is fixed and are distributed among the $N_v$ neighboring nodes of $b_0$ and affect $\gamma$ check equations. Depending on how the $\gamma$ 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,

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

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 first considering all *valid integer partitions* of $N_v^e$. Let a partition be denoted as $[q_1\ q_2\ q_3 \cdots\ q_{N_v^e}]$, i.e. $N_v^e = 1 \cdot q_1 + 2 \cdot q_2 + \cdots + N_v^e \cdot q_{N_v^e}$. We define a partition to be valid, if

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

A valid 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 $\gamma$ 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 $\left(\frac{q!}{q_1! \cdot q_2! \cdots q_{\rho-1}!}\right)$. 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,

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

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,

$$\hat{b}_0 = \begin{cases} 0 \text{ if } \sum\limits_{i=1,3,\ldots} q_i < \dfrac{\gamma}{2} \\ 1 \text{ otherwise} \end{cases} \tag{9}$$

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

Eqn. (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($\hat{b}_0 = 1|\{N_e - 1$ other errors, $b_0 = 1\}$) and Pr($\hat{b}_0 = 1|\{N_e$ errors, $b_0 = 0\}$) can be determined. Also,

$$\begin{aligned} \Pr\left(b_0 = 1|N_e \text{ errors}\right) = e &= \frac{\binom{n-1}{N_e-1}}{\binom{n}{N_e}} = \frac{N_e}{n} \\ \Pr\left(b_0 = 0|N_e \text{ errors}\right) &= 1 - e \end{aligned} \tag{10}$$

Eqn. (4) can be calculated for all values of $N_e$ and consequently, for a given channel error rate (Pr($N_e$ 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. Minimum value of $N_e$ is $t + 1$. The maximum value of $N_e$ can be chosen based on the required accuracy. For the codes considered in the next section, we chose $t + 1 \leq N_e \leq t + 15$.
2) For every $N_e$, $0 \leq N_v^e \leq N_v$. For every $N_v^e$, determine all valid partitions and the corresponding number of configurations that gives rise to the same partition using Eqn. (8).
3) Using Eqn. (9), determine if each of the error patterns represented by the partitions will result in a decoding error. The total number of possible error patterns for a given $N_v^e$ is given in Eqn. (7). Thus, considering independently, the cases when $b_0$ is in error and not in error at the channel output, Pr($\hat{b}_0 = 1|\{N_e$ errors, $b_0 = 0\}$) and Pr($\hat{b}_0 = 1|\{N_e - 1$ other errors, $b_0 = 1\}$) can be calculated using Eqn. (5) and Eqn. (6) respectively.
4) Use Eqn. (10) and Eqn. (4) to determine Pr($\hat{b}_0 = 1|N_e$ errors).
5) For a given channel error rate, use Eqn. (3) to determine the bit error rate for the given code.

The algorithm explained above facilitates the calculation of bit error rate for a particular code bit. But, for regular and structured LDPC codes, as ones considered in the next section, the probability of bit error calculation for every bit will be the same. So, this algorithm needs to be executed only once. The most complex task of the algorithm is to determine all valid partitions for all possible values of $N_v^e$. However, in the process of evaluating the performance of a code, the run-time of this task will be greatly reduced, if the valid partitions for a given $N_v^e$ is stored and re-used for different values of $N_e$ and also for the evaluation of other codes.

## III. Performance Prediction of Codes from Finite Geometry

LDPC codes designed from finite geometry form an important class of majority-logic decodable codes [7]. 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 $\lambda$ 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 $\lambda$ 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 [8] and [9].

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 [10] 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 can be determined that a code from $2-(v, k, 1)$ design, with column weight $\gamma$, can correct up to $\frac{\gamma-1}{2}$ errors, if $\gamma$ is odd, and $\frac{\gamma}{2}$ errors, if $\gamma$ is even under one-step majority logic decoding.

### A. Projective Geometry (PG) Codes

The projective geometry 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 PG(2,$2^m$) code is $q + 1$ and its minimum distance is $d_{\min} = q + 2$. Therefore, for codes from 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

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

Therefore, for the 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 PG(2,2) code. It is a (7,7) regular code with both column and row weight equal to 3. Its **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 see that there are 6 ways this configuration can occur. For this specific case, the bit $b_0$ is decoded correctly. Therefore,

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

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,

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

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.

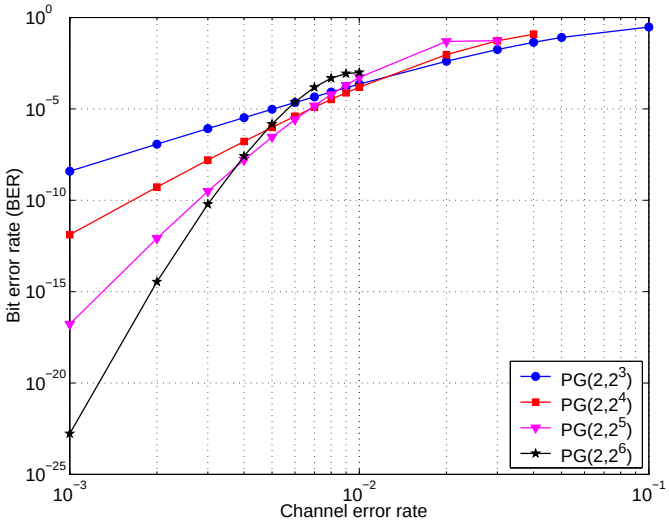


Fig. 2. Analytically calculated BER for codes from PG(2,$2^m$) when decoded using one-step majority logic decoder

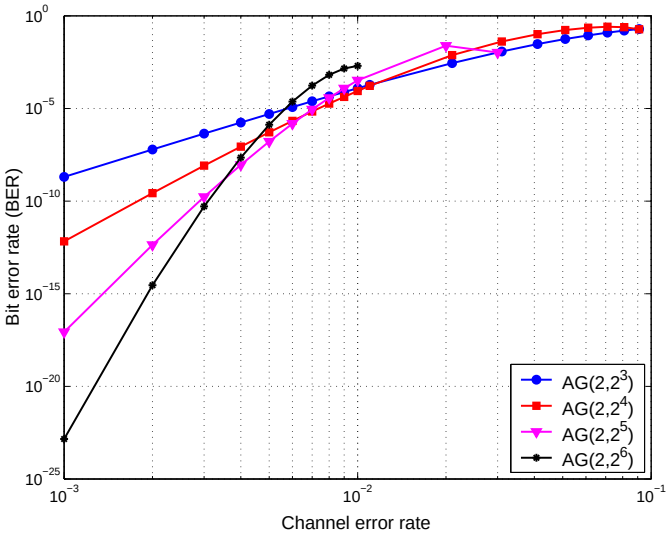


Fig. 3. Analytically calculated BER for codes from AG(2,$2^m$) when decoded using one-step majority logic decoder

### *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 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] 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.

[2] M. Taylor, ”Reliable information storage in memories designed from unreliable components”, *Bell System Technical Journal*, 47, pp 2299-2337, 1968.

[3] A. Kuznetsov, ”Information storage in a memory assembled from unreliable components”, *Problems of Information Transmission*, 9, pp. 254-264, 1973.

[4] L. D. Rudolf, “A class of majority logic decodable codes”, *IEEE Trans. Information Theory*, Vol. 13, pp. 305-307, Apr. 1967.

[5] T. J. Richardson, R. L. Urbanke, “The capacity of Low-Density Parity Check codes under message passing decoding”, *IEEE Trans. on Information Theory*, Vol. 47, No. 2, Feb. 2001.

[6] T. Hoholdt, J. Justesen and B. Jonsson, “Euclidean geometry codes, minimum weight words and decodable error-patterns using bit-flipping”, *Proc. International Symposium on Information Theory*, pp. 2171-2173, Sept. 2005.

[7] 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.

[8] E. F. Assmus Jr. and J. D. Key, *Design and their codes*, Cambridge, U.K., Cambridge Univ. Press, 1992.

[9] J. Colbourn, and J. H. Dinitz, *The handbook of combinatorial designs*, Boca, Raton, FL, CRC Press, 1996.

[10] 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.