An Efficient BF LDPC Decoding Algorithm Based on A Syndrome Vote Scheme

Jui-Hui Hung[1] and Sau-Gee Chen[2]
Institute of Electronics & Department of Electronics Engineering
National Chiao Tung University
Hsinchu, Taiwan
[1]paholisi.nctu@gmail.com [2]sgchen@cc.nctu.edu.tw

***Abstract*—This work presents a high-performance bit-flipping (BF) algorithm utilizing a proposed syndrome vote technique of a defined culprit bit set, named CVBF algorithm for LDPC decoding. It can achieve significant decoding performance improvements by about 0.9dB over the most efficient BF algorithm, owing to the introduced an additional syndrome vote procedure after updating the flipping reliabilities of all the received bits in each iteration. Moreover, its performance is comparable to the min-sum algorithm (MSA) under the condition of same iteration number, but with much lower complexity per iteration. The proposed syndrome vote scheme only costs little overhead in hardware realization. Besides, an early termination strategy suited for the proposed algorithm is also devised to further reduce the iteration number.**

***Index Terms*—Channel coding, LDPC code, bit flipping algorithm.**

I. INTRODUCTION

Low-density parity check (LDPC) code was introduced in 1962 [1], which can achieve performance close to Shannon bound. As such, LDPC has been adopted by many state-of-the-arts communication systems, such as the recent DVB-S2, DMB-TH and 802.16e systems. It is a kind of binary linear block code whose parity check matrix is sparse which has much fewer 1s than a common matrix. A sparse parity check matrix facilitates simple decoding algorithms and low-complexity decoder designs.

Check matrix of a LDPC code is often represented by a bipartite graph, called Tanner graph [2], which is composed of n variable nodes and m check nodes. Those variable nodes and check nodes are connected by edges defined by nonzero entries of matrix H. Fig. 1 shows an example with 4 check nodes and 8 variable nodes. The number of 1s in each column of H determines the number of edges for each variable node connected to check nodes, and the number of 1s in each row of H determines the connections from each check node to variable nodes. Tanner graph shows a clear picture of all the information exchange links in a decoding process.

Many LDPC decoding algorithms [3-8], have been proposed. The sum-product algorithm (SPA) [3] has very high decoding performances closed to the Shannon limit. Due to its high complexity, a popular min-sum algorithm (MSA) [4] is proposed to approximate SPA with much lower complexity, at the cost of minor loss in BER performance. However, the complexities of these two algorithms are still too high and difficult to implement. On the other hand, there is a low-complexity bit-flipping (BF) [1] decoding algorithm. The basic idea of BF algorithm is simply to flip those potential culprit error bits one by one, until the check matrix equation is equal to zero. As such, BF algorithm has much lower complexities than SPA and MSA. However, the disadvantages of the BF algorithm are its poor performances and high iteration counts.

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

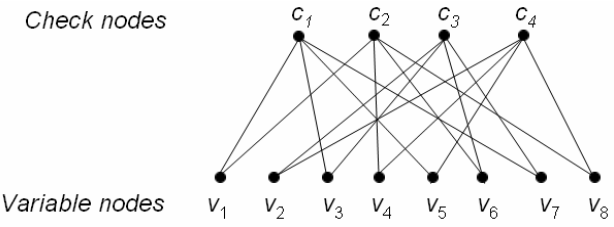

Fig. 1. Tanner graph of a parity check matrix.

The weighted bit-flipping (WBF) algorithm [5], the modified WBF (MWBF) algorithm [6], the improved MWBF (IMWBF) algorithm [7] and the WBF (RRWBF) [8] algorithm based on a flipping reliability ratio function are improved versions of the original BF algorithm [1], with better performances and lower iteration counts. However, all of them are still significantly inferior to SPA and MSA in terms of both performances and iteration counts. To remedy these problems, this work proposes an efficient BF algorithm by devising a new voting scheme on those syndromes associated with a chosen potential culprit bit with high flipping reliability, and then decide if the culprit bit should be flipped or not in each iteration. As such, the iteration numbers can be reduced and the performances can be noticeably increased simultaneously. Additionally, a simplified early termination strategy is applied to the proposed algorithm for further performance improvement. The overhead introduced by the novel syndrome vote scheme and the early

termination method is small as will be detailed below.

II. DECODING ALGORITHMS FOR LDPC CODES

Consider a LDPC codes which has $m \times n$ parity check matrix H (whose element at the i-th column and j-th row is h_{ij}). Let $\vec{b} = [b_1, b_2, b_3, ..., b_n]$ be the binary hard decision sequence obtained from a receiver as follows:

$$b_i = \begin{cases} 1 \text{ if } y_i \le 0 \\ 0 \text{ if } y_i > 0 \end{cases}, \; i = 1 \text{ to } n \qquad (1)$$

where y_i is the received channel value of the i-th bit. Let $\vec{s}$ be the syndrome of $\vec{b}$, i.e.,

$$\vec{s} = [s_1, s_2, s_3, ..., s_m] = \vec{b} \bullet H^T \qquad (2)$$

where

$$s_j = \sum_{i=1}^{n} b_i h_{ij}, \; i=1 \text{ to } n \text{ and } j=1 \text{ to } m \qquad (3)$$

The received vector $\vec{b}$ is a codeword if and only if $\vec{s} = 0$. If $\vec{s} \neq 0$, errors are detected, and any nonzero syndrome s_j indicates a parity failure.

A. Modified Weighted Bit-Flipping Algorithm [6]

The MWBF algorithm first computes the following partial flipping reliabilities,

$$| y |_{\min - j} \triangleq \min_{i: b_k \in B(c_j)} |y_i|, \text{ for } j=1 \text{ to } m \qquad (4)$$

where $B(c_j)$ denotes the set of bit nodes that participate in the j-th parity-check equation, i.e., the bit nodes associated with the positions of 1s in the j-th row of the parity-check matrix. As shown, $| y |_{\min - j}$ is a measure of the flipping reliability of the j-th check node. Next, MWBF decoding operations are carried out as follows:

STEP 1) Initialization: Set N_{itr} as the maximum iteration number. Set the initial iteration index value p=1. For row index value j=1 to m, compute the syndrome $s_j = \sum_{i=1}^{n} b_i h_{ij}$.

STEP 2) Computation of Flipping reliabilities: For column index value i=1 to n, compute the total flipping reliability of each bit node:

$$E_i = \sum_{j: c_j \in C(b_i)} (2s_j - 1) \cdot | y |_{\min - j} - \alpha \cdot |y_i| \qquad (5)$$

where $C(b_i)$ denotes the set of check nodes that is associated with the i-th column of the parity-check matrix, and weighting factor α is a positive real number. The optimal value of α is determined by simulation results

STEP 3) Bit Flipping: Flip the bit which has the largest total flipping reliability from all bit nodes.

STEP 4) Syndrome update and check of iteration number: Update the syndrome vector and set $p=p+1$. If syndrome vector is zero or $p>N_{itr}$, go to STEP 5; otherwise, go to STEP 2.

STEP 5) End of the algorithm: Output the decoding result.

Due to poor decoding performances and large iteration numbers of MWBF algorithm, an improved version of MWBF is proposed by [7] as discussed below.

B. Improved Modified Weighted Bit-Flipping Algorithm [7]

In the WBF or MWBF algorithms, the partial flipping reliability (4) of a particular check node is equal to the minimum absolute value of the received channel values associated with the check node. Since this partial flipping reliability will be passed to all the bit nodes connected to this particular check node, it should be excluded in the flipping reliability computation of a bit node if it is from the bit node itself. Therefore, the work in [7] proposes an improved MWBF algorithm (IMWBF). Specifically, IMWBF algorithm modifies the partial flipping reliability function (4) as:

$$| y |_{\min - i, j} = \min_{k: b_k \in B(c_j) \setminus b_i} |y_k|, \; b_i \in B(c_j) \qquad (6)$$

where $B(c_m) \setminus b_l$ represents the set $B(c_m)$ excluding the l-th bit node.

In the iterative decoding process of the improved algorithm, the advantages of low complexity and high speed of WBF and MWBF algorithms are preserved. However, the performance of the IMWBF algorithm is still not good enough, compared to SPA. By observing the flipped bit in each iteration of the both IMWBF and WBF algorithms, one can find a major weakness: when a bit is un-correctly flipped, it may induce chain reaction in wrongly flipping all the related message bits. Those errors bits cost additional decoding iterations and sometimes end up with infinite iteration loop. To solve the bit-error propagation problem, we proposes to add an extra flipping reliability check scheme and decide if a bit should be flipped or not, in each iteration, as introduced next.

III. THE PROPOSED BF LDPC DECODING ALGORITHM

A. Idea of the Proposed Algorithm

Conventionally, the existing BF-based algorithms flip the bit with the largest error probability in each iteration. However, it doesn't mean that this bit really has the largest error probability, based on the observed error probabilities. Additionally, if the bit is wrongly flipped, then there is high possibility that this bit error will propagate to the following iterations and cause error chain reaction. To alleviate this problem, a good BF strategy is to flip the bit that can attain a syndrome vector closer to the zero vector than before. Doing so, the probability of wrongly flipping a bit will be significant decreased, and there will be much less bit error propagation problem. Next, we will show how to achieve the design goal.

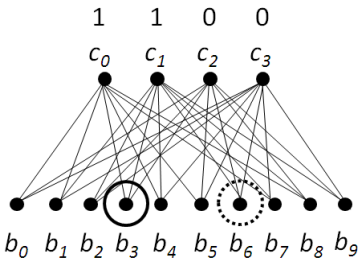

Fig. 2. A Tanner graph of a (10,6) LDPC code.

Since BF-based algorithms only flip one bit in each iteration, only the associated syndrome of the flipped bit will be inverted (i.e., 0 becomes 1 or 1 becomes 0). Take Fig. 2 as an example, if one flips the message bit circled by the dotted line (i.e., b_6), the syndrome vector will become (1,0,1,1). In this case, the total number of 1s of the syndrome vector is increased from two to three. The error may induce some chain reaction in wrongly flipping all the related message bits as mentioned before. Alternatively, if we flip the message bit b_3, the syndrome vector will become (0,0,1,0). The total number of 1s of the syndrome vector is reduced from two to one. It means that this flipped bit can make the syndrome vector closer to the zero vector than the original one.

In summary, if one flips the bit which has a larger number of 1s in the associated syndrome vector than the number of 0s, the syndrome vector will get closer to the zero vector. This tactic can be simply achieved by applying the majority vote scheme to the associated syndrome vector. Before doing that, the proposed algorithm first defines a set, called the culprit set, which includes all the bits considered to have high probabilities of incorrectly decoded. Take IMWBF algorithm as an example, those bits with higher flipping reliabilities than the others in each iteration can be included in the culprit set. For the ensuing discussion, let the number of elements in the culprit set be denoted as N_c.

To find the culprit set, in this case, one can intuitively include those N_c bits which have the largest flipping reliability values among all bits. After the culprit set is constructed, the bit has the largest flipping reliability in the culprit set should be identified first. As such, its connected check nodes and the associated syndromes can be obtained as well. As mentioned before, we can choose the associated syndrome as the participants of majority vote. By the result of majority vote, the proposed algorithm can decide if this bit should be flipped or not. If the result is 1 (i.e., the number of 1s is larger than number of 0s in the associated syndrome vector), the bit will be flipped and then the same process will be repeated in the next iterations. Otherwise, this bit will be seen as an innocent one and the bit with the next highest flipping reliability will be chosen and tried for the majority vote and so on.

Notice that if there is no any bit in the culprit set can be flipped according to their syndrome majority votes, the bit with the largest flipping reliability in the culprit set will still be flipped to keep the iterative decoding process running. In what follows, the detailed steps of the proposed algorithm will be shown.

B. Detailed Flows of the Proposed CVBF Algorithm

The steps of the proposed CVBF algorithm are:

STEP 1) Initialization: Set N_{itr} and N_c as the maximum iteration number and the culprit set number, respectively. Set the initial culprit index value k=1 and the initial iteration index value j=1.

STEP 2) Formation of culprit set: Using any proper reliability function of exiting BF algorithms (such as IMWBF algorithm) to build the culprit set and arrange the N_c member bit nodes in descending order based on their flipping reliability values. As a result, the k-th member bit node has the k-th highest bit flipping reliability in the set.

STEP 3) Majority vote: If $N_{k,1} > N_{k,0}$ (where $N_{k,1}$ and $N_{k,0}$ are defined as the number of 1s and 0s of the syndrome vector which is associated with the k-th member bit node of the culprit set, respectively), flip the bit and go to STEP 5; otherwise go to STEP 4.

STEP 4) Test of early exit: Set k=k+1 and go back to STEP 3 for checking the next element of culprit set should be flipped or not. Notice that if culprit index value k is larger than the culprit set number N_c, flip the first candidate bit and go to STEP 5.

STEP 5) Check of iteration number: Set j=j+1. If j>N_{itr}, go to STEP 6; otherwise update the syndrome vector by the connectivity of flipped bit and go to STEP 2.

STEP 6) End of the algorithm: Stop the decoding procedure.

Notice that CVBF algorithm can be applied to any other BF algorithms. CVBF algorithm is only responsible for finding a culprit set and using majority vote to decide the bit to be flipped in the set. To further reduce the iteration number, an early termination strategy is introduced next.

C. Early termination strategy

From the simulation results, we find that when all the culprit bits do not pass the majority syndrome vote within a certain number of iterations, the following iterations always incorrectly flip the culprit bits and induce the chain error reaction in all the related message bits. Hence, the proposed algorithm incorporates a low-complexity early termination strategy which simply exits the decoding iteration loop when the culprit number is reached. The flow chart of the proposed algorithm with the early termination strategy is shown in Fig. 3.

IV. SIMULATION RESULTS

Before conducting simulations of CVBF algorithm, we need to select one known good BF algorithm to work with. Here, we chooses IMWBF algorithm, because it has a good tradeoff between decoding performance and computational complexity. Some other simulation conditions are defined below:

1) Gallager's method is applied to randomly generate the coding matrix, which has regular column and row weights. In the following simulation, we set code length to 816, code rate to 1/2 and column weight to 5.

2) AWGN channels and BPSK modulation are assumed.

The performance of the proposed algorithm without early termination strategy is shown in Fig. 4 at iteration number of 50 and various N_c values. From this simulation result, we can find

that the proposed algorithm can improve about 0.9dB when the value of N_c equal to n-1, where n is the code length. Notice that the performance of the proposed algorithm saturates when the value of N_c is larger than 50. This means that the decoding performance of the proposed algorithm is still limited by the BF algorithm it combines with, namely, the IMWBF algorithm, in this case.

Fig. 5 shows the proposed algorithm's performance with early termination strategy in the same simulation condition. Note that the decoding performance of the proposed algorithm is lower than the IMWBF algorithm when the value of N_c is equal to 1 and 2. Intuitively, this is because that if the value of N_c is too small, the iterative decoding process will exit very early and will not be able to make good error correction results.

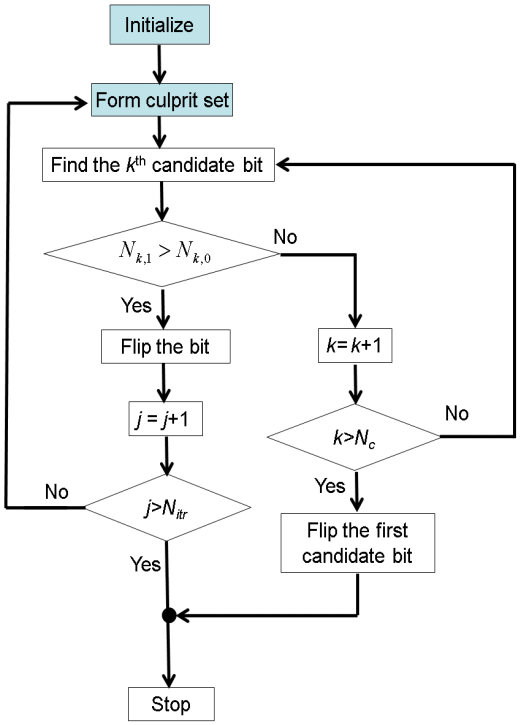

Fig. 3. Flow chart of the proposed LCMBF algorithm.

Fig. 6 shows the performances with and without the early termination scheme. One can observe that the performance with early termination is equal to the performance without early termination when N_c=50. Thus, it suggests that the proposed algorithm combine the early termination strategy due to its low computational complexity, low iteration number and competitive decoding performance.

Fig. 7 shows the comparison results of the proposed algorithm (with N_c=50), RRWBF algorithm and MSA. As shown, the decoding performance of CVBF algorithm is better than the RRWBF algorithm by about 0.2dB. It is also very close to the decoding performance of MSA even at the same iteration number in this case.

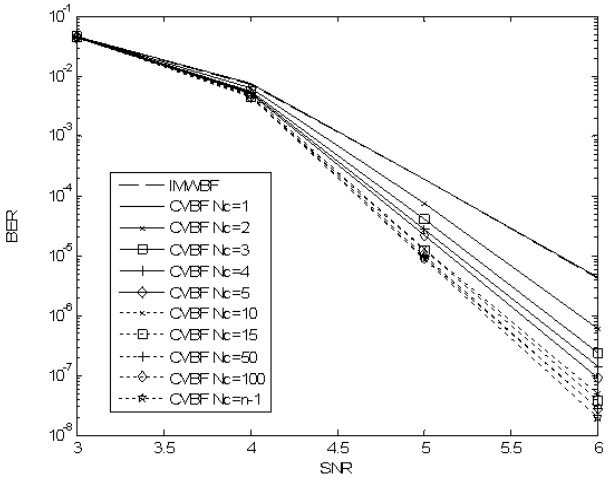

Fig. 4. Performance comparisons of the proposed CVBF algorithm without early termination strategy at code length=816, code rate=1/2, column weight=5 and iteration number=50 and various N_c values.

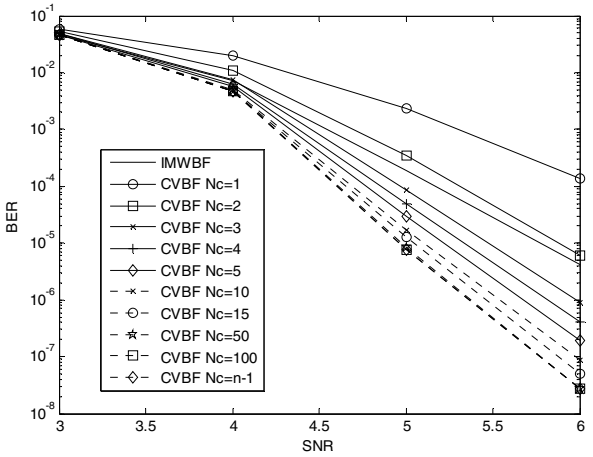

Fig. 5. Performance comparisons of the proposed CVBF algorithm with early termination strategy at code length=816, code rate=1/2, column weight=5 and iteration number=50 and various N_c values.

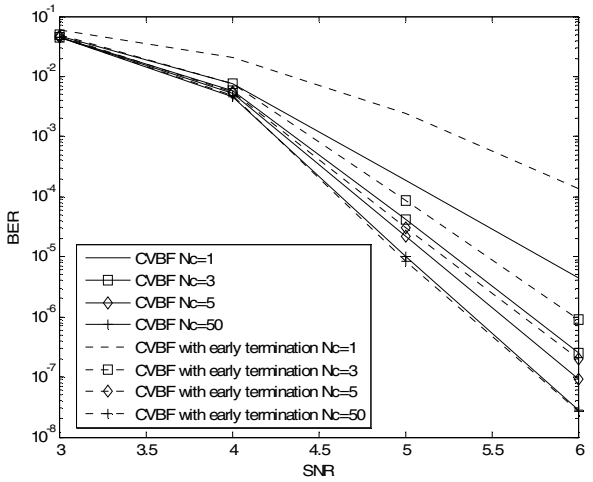

Fig. 6.Comparison results of the proposed CVBF algorithm with early termination strategy and without that at code length=816, code rate=1/2, column weight=5 and iteration number=50 and various N_c values.

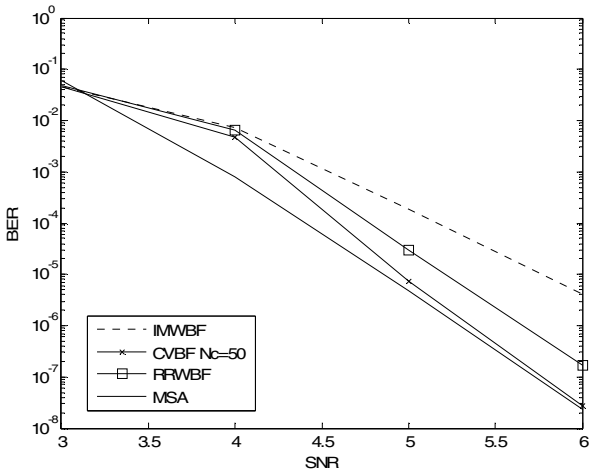

Fig. 7.Comparison results of the proposed CVBF algorithm with early termination strategy and other decoding algorithms that at code length=816, code rate=1/2, column weight=5 and iteration number=50.

V. CONCLUSION

The proposed algorithm can achieve excellent performances under the condition when the culprit number is properly set. It improves the decoding performance by about 0.9dB compared with IMWBF algorithm, for a (816,408) regular LDPC code generated by Gallager's method. Its decoding performance is comparable to the MSA even in the condition of same iteration number. As a result, the proposed algorithm has the high potential to become a useful LDPC decoding algorithm due to its much lower computational complexity than the popular MSA algorithm. By combining the introduced early termination scheme, the proposed technique can further reduce the computational complexity with little performance loss.

ACKNOWLEDGMENT

This work is supported in part by the grants NSC 95-2219-E-009 -004 and MOEA 95-EC-17-A-01-S1-048, Taiwan.

REFERENCES

[1] R. G. Gallager, *Low-Density Parity-Check Codes*, MA: MIT Press, 1963.
[2] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, Vol. 27, pp. 533-547, Sept. 1981.
[3] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
[4] X. Y. Hu, E. Eleftheriou, D. M. Arnold, and A. Dholakia, "Efficient implementation of the sum-product algorithm for decoding LDPC codes," *IEEE GLOBECOM' 01*, Vol. 02, pp. 1036-1036E, Nov. 2001.
[5] Y. Kou, S. Lin, and, M.P.C. Fossorier, "Low-Density parity check codes based on finite geometries: a rediscovery and more," *IEEE Trans. Inform. Theory*, pp. 2711-2736, 2001.
[6] J. Zhang, M.P.C. Fossorier, "A Modified Weighted Bit-Flipping Decoding of Low-Density Parity-Check Codes," *IEEE Commun. Lett.*, vol. 8, pp. 165-167,March 2004.
[7] M. Jiang, C. Zhao, Z.Shi, and Y. Chen, "An Improvement on the Modified Weighted Bit Flipping Decoding algorithm for LDPC Codes," *IEEE Commum.* , Vol. 9, No.9, pp.814-816, Sept. 2005.
[8] F. Guo, and L. Hanzo, "Reliability ratio based weighted bit-flipping decoding for low-density parity-check codes", *Electron. Lett.*, pp. 1356-1358, 2004.