A Low-Complexity High-Performance Decoding Algorithm for Fixed-Point LDPC Decoders

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*—The bit-error-rate (BER) performance of an LDPC decoding algorithm will be seriously degraded when the algorithm is realized with fixed-point implementation in practical applications, due to the introduced quantization errors and rounding errors. To remedy the problem, this paper proposes an improved min-sum algorithm (MSA), called CMVP algorithm. It achieves better performances than the popular min-sum algorithm (MSA), under the condition of the same fixed-point precision. Besides, the hardware overhead of the new algorithm over conventional MSA is small. On the other hand, under the condition of comparable performances, MSA algorithm needs a higher fixed-point precision and hardware costs than the new CMVP algorithm, according to the simulations and hardware synthesis results. The new algorithm also works well in a wide range of code rates and code lengths.**

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 1's 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 "1" in each column of *H* determines the number of edges for each variable node connected to check nodes, and the number of "1" 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. Existing decoding algorithms based on Tanner graph generally composed of check-node steps and bit-node steps, which are realized by CNU (Check Node Unit) and VNU (Variable Node Unit), respectively, as detailed below.

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

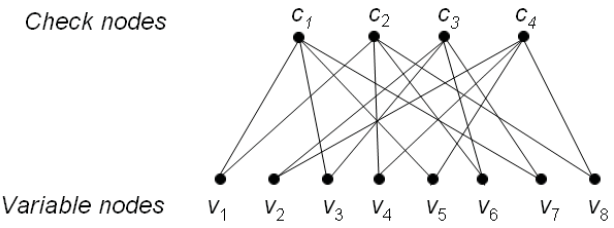

Fig. 1. Tanner graph of a parity check matrix.

A. Decoding Algorithms for LDPC Codes

Decoding of a LDPC code is based on Sum-Product algorithm [3], which is represented in the log-likelihood ratio (LLR) of $L(x) = \log(P\{x=0\}/P\{x=1\})$. As mentioned previously, this algorithm is divided into two types of steps. In a bit-node step, message $L_{v_i \to c_j}$ sent by variable node v_i to check node c_j is

$$L_{v_i \to c_j} = channel(v_i) + \sum_{c_k \in C \setminus c_j} L_{c_k \to v_i} \qquad (1)$$

where $C \setminus c_j$ is a set containing all the check nodes connected to variable node v_i (excluding check node c_j), and $channel(v_i)$ is the channel value of variable node v_i. This equation is implemented in a VNU. In a check-node step, message $L_{c_j \to v_i}$ passed from check node c_j to variable node v_i can be expressed by:

978-1-4244-4242-3/08/$25.00 ©2008 IEEE

$$L_{c_j \to v_i} = 2\tanh^{-1}\left(\prod_{v_k \in B\backslash v_i} \tanh\left(\frac{L_{v_k \to c_j}}{2}\right)\right) \quad (2)$$

where $B\backslash v_i$ is a set containing all the variable nodes connected to check node c_j excluding variable node v_i. Equation (2) is hard to implement. It will cost considerable hardware area. Min-sum algorithm (MSA) [4] is a popular, accurate and low-complexity approximation to (2), as shown below.

$$L_{c_j \to v_i} \approx \left(\prod_{v_k \in B\backslash v_i} sign(L_{v_k \to c_j})\right) \min_{v_k \in B\backslash v_i} (|L_{v_k \to c_j}|) \quad (3)$$

MSA reduces decoder hardware complexities significantly, at the cost of a little performance loss. After a combined check-node and bit-node step, the updated bit message described by (4) will be sent to a hard decision circuit for obtaining the decoded bit.

$$L_{v_i} = channel(v_i) + \sum_{c_k \in C} L_{c_k \to v_i} \quad (4)$$

The above process is repeated and the decoded bits can be output when the iteration number exceed a defined maximum iteration number N_{itr}, or when the decoded information bits satisfy the check matrix constraint: $Hx^T = 0$.

In practical realization, for the consideration of low power and low complexity designs, fixed-point implementation instead of floating-point implementation is adopted. However, fixed-point realization always causes quantization errors and rounding errors. As such, decoding performances will be significantly degraded. To alleviate the problem, based on the MSA, we propose a post-processing algorithm by defining additional information of confidence, majority vote and persistency (CMVP for short). By utilizing the extra information, we proposed the improved fixed-point CMVP algorithm over the popular MSA algorithm. The proposed algorithm's performance approaches to that of the floating-point MSA algorithm, at the cost of small hardware overhead.

II. The Proposed CMVP Algorithm

A. Concept of the proposed algorithm

The proposed algorithm can be divided into two parts, one is reliability check part and the other is majority vote part as introduced below. In the reliability check part, we propose two design control factors for the decision of error bits, called factors *C* and *P*.

***Factor C (Confidence)*:** In a decoding process, the LLR form is to be decide whether a message bit is originally 0 or 1. Hence, intuitively, we expect that a decoded bit will be very likely wrong if its absolute LLR value is close to zero. Therefore, the proposed CMVP algorithm defines an confidence threshold *C* to assist the judgment of the correctness of a decoded bit, i.e., if the absolute LLR value of a decoded bit exceeds the confidence threshold, the proposed algorithm will treat this bit as an error bit.

***Factor P (Persistency)*:** By observation, the later iteration results are more credible than pervious iteration results. Hence, the other way to decide the reliability of the decoded bit is to save the decoding results of the latest *P* iterations and check if they are all the same (i.e., unchanged). Actually, we can ignore one of these two factors to reduce the hardware complexity or use both of them to double check the reliability.

In majority vote part, we propose a design factor for the correction of the error bits, called factor *MV*.

***Factor MV (number of Majority Vote participants)*:** After the reliability check, the proposed algorithm adopts the majority vote process to further confirm and correct the error bits. It saves the last *MV* iterative decoding results and does the majority vote process. In other words, if a target bit is unreliable and its decoded results have more 1's than 0's, then set this bit to 1, and vice versa. Description of the proposed algorithm is detailed below.

B. Detailed flows of the proposed algorithm

The proposed algorithm is divided into the following six steps. Notice that this algorithm is executed after each iterative decoding process and needs to store the last *MV* decoding results.

Step 1) Initialize: initialize proper values of *C*, *MV* and *P*. Set the value of bit count variable N_b=1. Define L_{N_b} as the LLR value of the N_bth decoded bit in (4).

Step 2) Check the reliability (part 1): If $\left|L_{N_b}\right| \le C$, go to Step3, otherwise, go to Step 5.

Step 3) Check the reliability (part 2): Check if the last *P* decoding results of the N_bth bit are all the same, go to Step 5 if yes; otherwise go to Step 4. To make sense, *P* should be smaller than half of *MV* and larger than 1.

Step 4) Execute majority vote: Execute the majority vote based on the last *MV* decoding results, if it has more 1 than 0, set this bit to 1, and vice versa. Make sure that $N_{itr} \geqq MV \geqq 3$ and *MV* should be odd.

Step 5) Check termination condition: N_b=N_b+1. If N_b is larger than the code length, go to Step 6; otherwise go back to Step 2.

Step 6) Stop the algorithm: Stop the procedure.

Fig.2 shows the flow chart of the proposed algorithm. As mentioned before, the proposed algorithm can be specified for different designs considering all factors (i.e., *C*, *MV* and *P)* or two factors (i.e., *C* and *MV* or *P* and *MV)* or only factor *MV*.

III. SIMULATION RESULTS

Here, we choose the block LDPC codes for 802.16e standard to conduct the simulations. In the beginning, we need to decide the proper iteration number with good decoding performances. Fig. 3 shows the BER simulation results vs. SNR of the modified MSA [5] assuming a normalization factor of 0.75 in AWGN channel, in different iteration numbers, at code rate of 1/2 and length of 576, BPSK modulation. Obviously, the performance improvement tends to be insignificant after 10 iterations, which is about 0.2 dB. As this result, 10 iterations is a good choice for practical implementation.

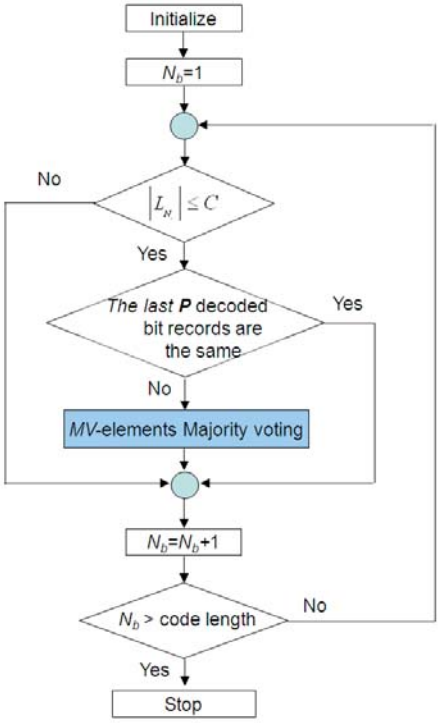

Fig. 2. The flow chart of the proposed CMVP algorithm.

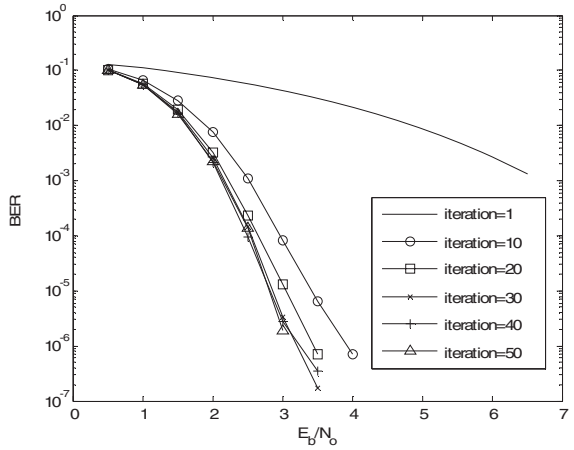

Fig. 3. Decoding performance at different iteration number.

Since practical applications are based on fixed-point implementation, we need to determine suitable quantization configurations. Let [*t*:*f*] denote the quantization scheme with total *t* bits where *f* bits are used for the fractional part of a value. Performances of several quantization configurations are shown in Fig. 4 with the same iteration number. We find [7:5] is a good cost-performance choice considering hardware complexity and decoding performance. Notice that the proposed algorithm can also be used in floating-point implementations. Unfortunately, according to our extensive simulations, little performance improvements are obtained. As such, we omit the case here.

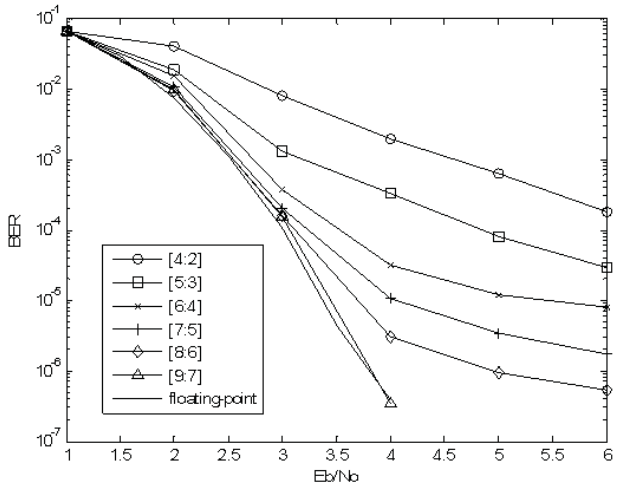

Fig. 4 Fixed-point BER simulation of modified MSA, iteration number =10.

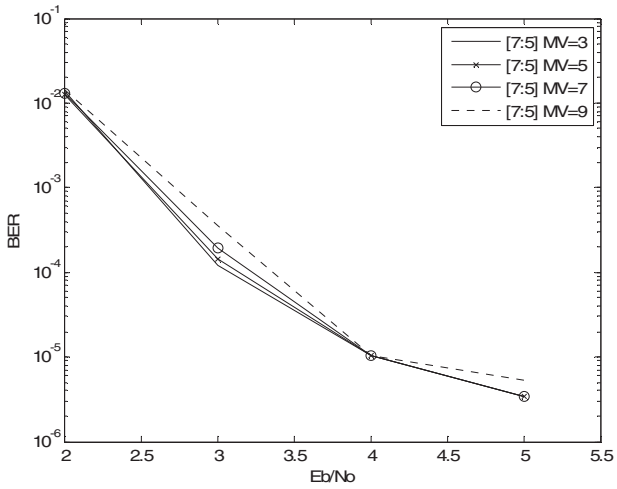

Fig. 5 Performance of [7:5] fixed-point simulation of the proposed CMVP algorithm combined with the modified MSA in different *MV* values, iteration number=10.

Next, we need to decide good values for *C*, *MV* and *P*. Intuitively, we should find *MV* first because the majority vote is the key part of the proposed algorithm and can work alone without *C* and *P*. According to simulation results, choosing *MV*=3 can get the best performance of all as shown in Fig. 5. Next, we should decide *C* and *P*. In this case, we don't need

to consider P, because any P value does not satisfy the constraint (i.e., P should be smaller than half of MV and larger than 1). To test the effect of C and for the consideration of low hardware complexity, we assume various C values in the form of 2^x as shown in Fig. 6, where x is an integer. As shown, the best result is obtained when C=1. Note that the case of MV=3 correspond to C=2-2^{-5} for the given word length. Figure 7 compares the performances of the proposed method (assuming MV=3 and C=1) combined with the modified MSA and the modified MSA alone. As shown, the proposed algorithm achieves better performances than the modified MSA by more than 0.2dB.

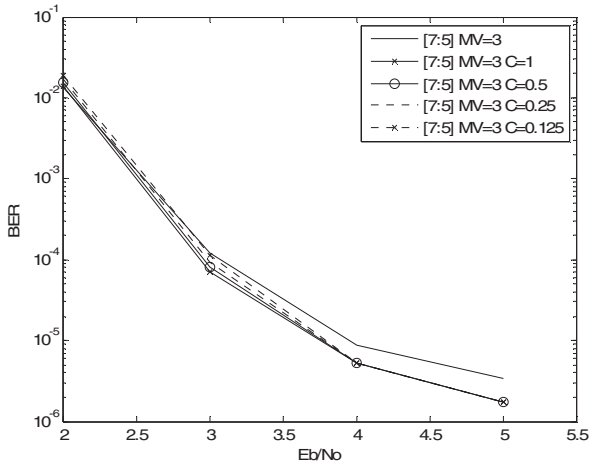

Fig. 6 Performance of [7:5] fixed-point simulation of the proposed CMVP algorithm combined with the modified MSA in different C values, iteration number=10 and MV=3.

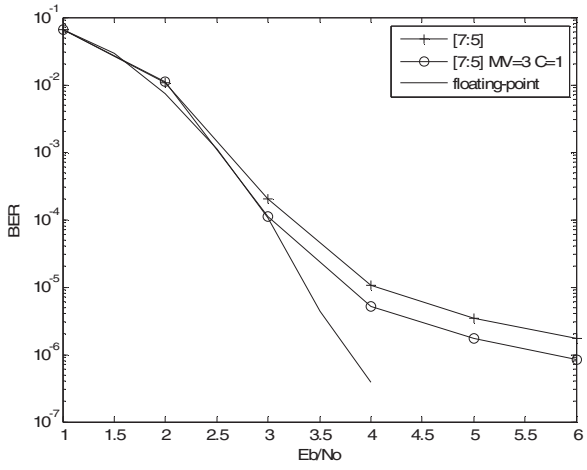

Fig. 7 Fixed-point [7:5] simulations of the proposed CMVP algorithm combined with the modified MSA, iteration number=10.

Fig. 8 to Fig. 10 shows the simulation results correspond to the [5:3] quantization scheme. By the same procedures, we decide the MV value first, fix the value (i.e., MV=3), and then find out the best C value as shown in Fig. 8 and Fig. 9. Similar to the [7:5] case, Fig. 10 shows that the proposed algorithm with [5:3] scheme also improves performance by about 0.2 dB.

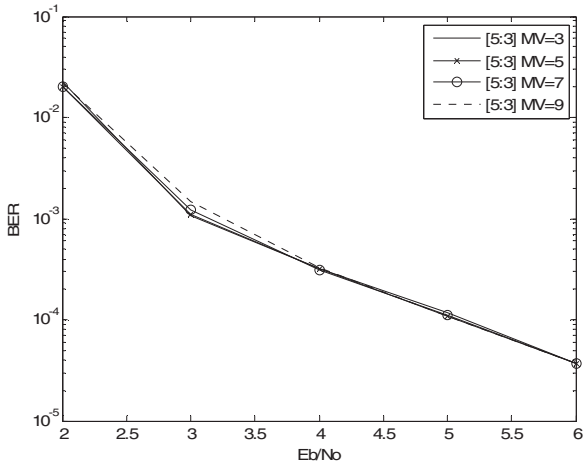

Fig. 8 Performance of [5:3] fixed-point simulation of the proposed CMVP algorithm combined with the modified MSA in different MV values, iteration number=10.

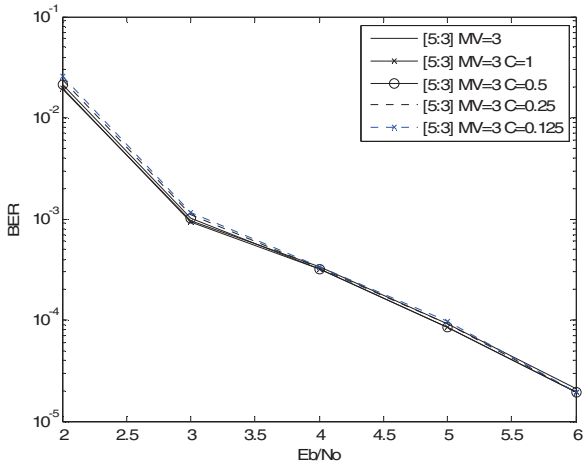

Fig. 9 Performance of [5:3] fixed-point simulation of the proposed CMVP algorithm combined with the modified MSA in different C values, iteration number=10 and MV=3.

To verify the performances of CMVP algorithm versus code length and code rate, we take the LDPC code of 802.16e standard with code length of 2304 and code rate of 5/6 as an example. Notice that different code lengths and code rates have different normalization factors which have been optimized by simulations. Fig. 11 shows the performances in various MV values, where MV of 5 achieves the best performance. Fig. 12 shows that C=1 and P=2 has the best performance and Fig. 13 shows that the proposed algorithm still exhibits very good performances for code rate of 5/6.

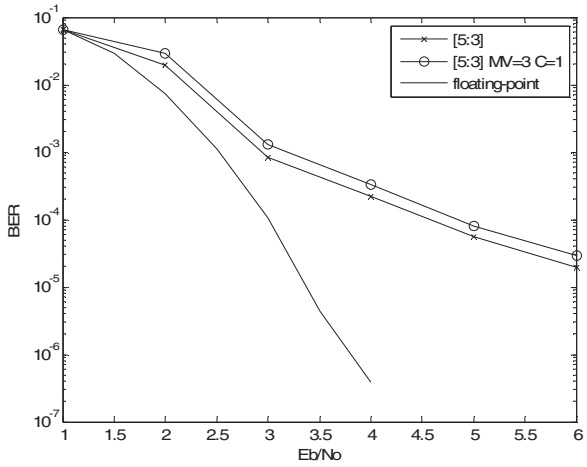

Fig. 10 Fixed-point [5:3] simulations of the proposed CMVP algorithm combined with the modified MSA, iteration number=10.

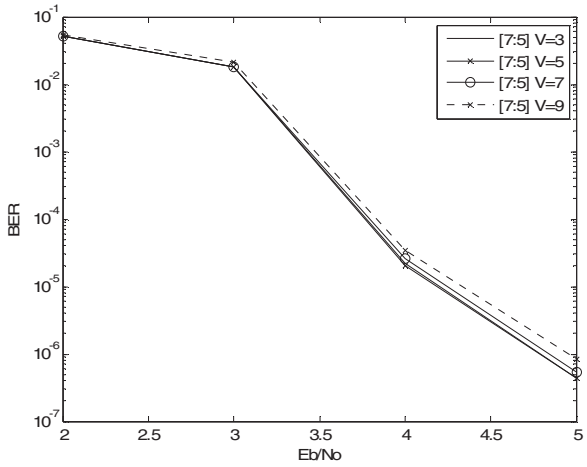

Fig. 11 Performance of [7:5] fixed-point simulation of the proposed CMVP algorithm combined with modified MSA in different *MV* values at code length=2304, code rate=5/6 and iteration number=10.

IV. Conclusion

The fixed-point realization of an LDPC decoder will make a huge performance loss. The proposed algorithm can remedy the loss effectively and improves the performance significantly, by introducing the control parameters of confidence and persistency. Doing so, it tends to find the most potential error bits for correction. Further, in order to compensate additional possible errors, we employ the majority vote scheme to correct those possible error bits. Overall, the incurred hardware cost of the proposed technique is small, because it can be easily realized with some simple majority gates, logic gates and wire connections. By applying the new method to the popular MSA method, BER performance can be improved by about 0.2 dB in various code rates and code lengths.

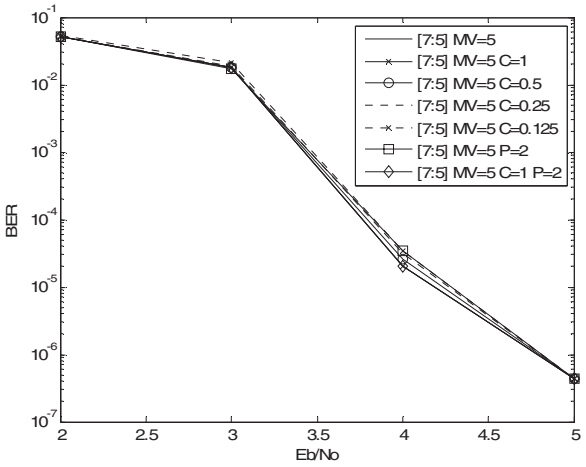

Fig. 12 Performance of [7:5] fixed-point simulation of the proposed CMVP algorithm combined with modified MSA in different *C* and *P* values, code length=2304, code rate= 5/6 and iteration number=10 and *MV*=5.

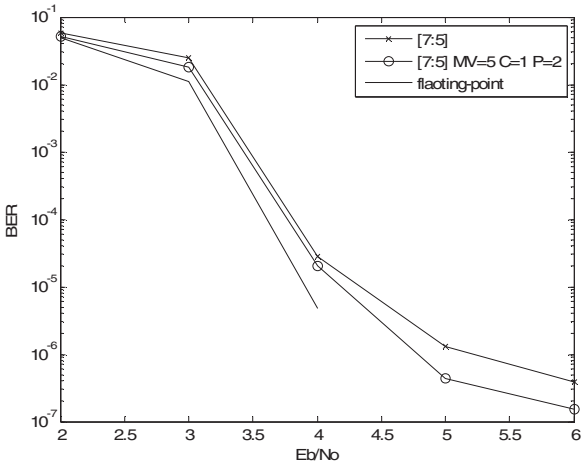

Fig. 13 Performance of [7:5] fixed-point simulation of the proposed CMVP algorithm combined with the modified MSA at code length =2304, code rate=5/6 and iteration number=10.

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] J. Chen and M.P.C. Fossorier, "Near Optimum Universal Belief Propagation Based Decoding of Low-Density Parity Check Codes," IEEE Trans. Commun., Vol. 50, pp. 583-587, NO.3 Mar. 2002.