ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 4, Issue 5, May 2015

DESIGN AND PERFORMANCE EVALUATION OF WIRELESS COMMUNICATION SYSTEM USING LDPC CODES WITH MIMO SYSTEM

Kritika Puri, Madhwendra Tripathi

Abstract-Near-capacity performance and parallelizable decoding algorithms have made Low- Density Parity Check (LDPC) codes a powerful competitor to previous generations of codes, such as Turbo and Reed Solomon codes, for reliable high-speed digital communications. As a result, they have been adopted in several emerging standards. In this paper, first explain the design and implementation of both regular and irregular LDPC codes. The design of a good irregular LDPC code needs a pair of distributions to be specified. Then, we present the idea of adapting the scaling factor of the Min-Sum decoder with iterations through a simple approximation. Sum- Product decoding algorithm of LDPC codes is an iterative decoding algorithm with excellent performance. Min-Sum decoding algorithm is a kind of modified Sum-Product decoding algorithm with a reduced implementation complexity. VLSI implementation complexities of soft-input soft-output (SISO) decoders are discussed. These decoders are used in iterative algorithms based on Turbo codes or Low Density Parity Check (LDPC) codes.

1. INTRODUCTION

Low-density parity check (LDPC) codes were first presented by Gallager in the early 1960s. It has been shown that these codes have remarkable performance that is very close to Shannon limit when using iterative decoding.

***Kritika Puri**, ECE, Hindu college of engineering, Sonipat, India*

***MadhwendraTripathi**, ECE, Hindu college of engineering, India*

They become strong competitors to turbo codes for error control for many digital communication systems [3]. It has been known recently that LDPC codes can outperform the best known Turbo codes if designed properly. Usage of LDPC codes in video broadcasting standards assures better protection against errors which decrease the video quality. It also allows more data to be transported over a given channel [3]. The used decoding technique of LDPC code is an important parameter in its performance and its implementation complexity. There are many types of decoding algorithms that can be used to decode LDPC codes. The soft-decision decoding algorithms are widely employed because of their superior performance over hard-decision algorithms. The log likelihood ratio sum-product algorithms (LLR-SPA), developed by we, are proven to achieve excellent capacity performance, by approaching to Shannon bound. However, one drawback for the LLR-SPA is the high complexity that implies large decoding delay that may be critical for some delay sensitive applications such as DVB. So, many modified approximations of LLR-SPA are developed to reduce its high complexity. One of the most important algorithms that satisfy this goal is the Min-Sum algorithm, Min-Sum is introduced in [3] as a simplification of LLR-SPA by using minimum operation instead of complex implemented tanh and $\tanh^{-1}$ functions. Many modified versions of Min-Sum algorithm were proposed to increase its performance with acceptable increasing in decoding complexity. One of the most

important modifications is Scaled Min-Sum. It is a modification of Min-Sum algorithm, where a scaling factor is used to decrease the error introduced by using the minimum operation. Scaled Min-Sum has a very good performance in regular LDPC codes. On the other hand, irregular LDPC codes require different scaling factor strategy. The scaling factor is calculated by approximating a nonlinear post-processing function to linear function. The non-linear function is highly affected by SNR and requires updating per iteration. In other words, irregular LDPC codes require different scaling factor per-iteration to achieve the optimum scaling scenario. Although changing the scaling factor with iterations gives a very good performance (low BER and avoiding error floor), it requires complex calculations (in design phase) and extra storage to store the scaling factor sequence. The two-dimension normalization was proposed where different scaling factor is used for each variable and check node degree. So two scaling vectors (α, **β**) are required for both check nodes' output and variable nodes' output respectively. Scaling factor vectors (**α, β**) calculation requires multi-dimension optimization (in design phase). In addition to design complexity, 2-D scaling factor implementation requires extra storage to store the scaling factors and extra complexity in scaling stage to choose different scaling factor for each degree. Another modification of min-sum algorithm is selective max-min algorithm. It uses maximum operation instead of summation in variable nodes processors. So it has lower complexity than scaled min-sum, however it has lower performance than scaled min-sum. In other words, selective max-min algorithm has an intermediate complexity and performance between scaled min-sum and min-sum algorithms.

In this paper, we propose a Simplified Variable Scaling (SVS) Min-Sum algorithm. SVS Min-Sum algorithm is based on using logical heuristic equation to calculate an easy implemented scaling factor sequence. This heuristic equation comes from observing the behavior of scaling factor sequence, where the scaling factors increase exponentially with iterations and its final value equals 1. So we get the advantage of adaptive scaling factor with iteration but with lower complexity in both design and implementation phases. In addition to lower implementation complexity, we avoid using two scaling stages for both variable and check nodes' output [3].

We [2] have shown that construction of good LDPC code requires an irregular distribution of check node and variable node degrees of the parity check matrix. Further, a good irregular degree distribution can be obtained with density evolution. It then remains to explicitly specify a method of constructing LDPC codes with the specified number of ones in each row and each column. We should also take care to eliminate cycles of length four in the LDPC code. We implemented a randomized construction of such a parity check matrix using the bit filling approach.

A basic communication system is composed of three parts: a transmitter, channel, and receiver. Transmitted information becomes altered due to noise corruption and channel distortion. To account for these errors, redundancy is intentionally introduced, and the receiver uses a decoder to make corrections. In error-correcting control (ECC), it is pertinent to have a high code rate while maintaining low complexity. Soft-input-soft-output (SISO) decoders differ in that they have as inputs and outputs probabilistic information, instead of sequences of bits. SISO decoders use digital signal processing and therefore are more robust, but result in a high complexity [4]. In addition, SISO decoders ease in passing information in an iterative fashion. Iterative decoding is a new

and powerful technique for error correction in communication systems; it leads to a significant improvement in bit error rate (BER) over conventional decoders.

Low Density Parity Check (LDPC) coding is implemented as a SISO iterative decoder that represents a fundamentally new approach to error correction in wired, wireless, and optical communications. A large, dense parity-check matrix defines the LDPC message passing algorithm, and is essential to a successful implementation. LDPC coding has many advantages including large coding gains, and less computational complexity compared to other coding schemes like the BCJR, Viterbi, and Turbo codes. This type of coding also has a more effective BER performance than other schemes, like the Reed Solomon codes. Another advantage is the ability to pipeline the decoder for high-speed implementations in order to reduce delay at the cost of registers and latency. Yet another advantage is the smaller number of required iterations, and therefore lower complexity, due to being able to determine if a codeword is reached based on the parity-check matrix. Disadvantages of the LDPC decoder include the complicated memory structure for serial architectures and the complexity with interconnect routing for parallel architectures. Serial architectures are slower, resulting in reduced cost. Parallel architectures have higher throughput and lower power dissipation [4].

The studies about good LDPC codes are mainly focused on two aspects: one is the study about irregular binary LDPC codes, whose performance is very close to Shannon Limit when the codeword length is large (N > 10000) [5]; The other is the study about non binary LDPC codes with moderate or small codeword length (N < 5000). The results of show that the performance of non binary LDPC codes is better than binary LDPC codes, and gives out the conclusion that the best LDPC codes are the irregular non binary LDPC codes. The decoding algorithm of non binary LDPC codes is the sum-product algorithm (SPA) over non binary fields (GF (q), q > 2), and this algorithm ensures the good performance of LDPC codes. But this algorithm has a large computation complexity scaled as q^2. Many literatures have focused on the simplified decoding algorithms: reference proposes a fast algorithm of non binary LDPC codes, which replaces the convolution operation on the process of check nodes iteration with fast Fourier transform over GF(q) ; reference studies the log-domain decoding algorithm (Log- SPA) for non binary LDPC codes, and the computation complexity is scaled as $(q-1)^2$; reference gives out the extended min-sum decoding algorithm (EMS) for non binary LDPC codes, which has even lower complexity than Log-SPA. But the multiplication operation and signal-to-noise power rate (SNR) estimation are still needed for these algorithms, which increase the complexity in engineering implementation.

In this paper, we propose a novel simplified decoding algorithm for non binary LDPC codes. This algorithm only contains addition operation, and the operation complexity is very small with little performance loss. The SNR estimation is also not needed [5].

2. RELATED WORK

This paper presents a review of bit-flipping algorithms and other methods related to the new NGDBF algorithms described. The original bit-flipping algorithm (BFA) was introduced by Gallager in his seminal paper on LDPC codes [1]. Gallager's BFA is a hard-decision algorithm for decoding on the binary symmetric channel (BSC), in which only hard channel bits are available to the decoder. To correct errors, the BFA computes a sum over the adjacent parity-check equations for each bit in the code. If, for any bit, the number of adjacent parity violations exceeds a specified threshold,

then the bit is flipped. This process is repeated until all parity checks are satisfied, or until a maximum iteration limit is reached. The BFA has very low complexity since it only requires, in iteration, a summation over binary parity-check values for each symbol; however the BFA provides weak decoding performance. We considered a probabilistic BFA (PBFA) which adds randomness to the bit-flip decision, resulting in improved performance. In PBFA, when a bit's parity-check sum crosses the flip threshold, it is flipped with probability p. The parameter p is optimized empirically and is adapted toward 1 during successive iterations. We introduced the Weighted Bit-Flipping (WBF) algorithm which improves performance over the BFA by incorporating soft channel information, making it better suited for use on the Additive White Gaussian Noise (AWGN) channel and other soft-information channels. In the WBF algorithm, all parity-check results are weighted by a magnitude that indicates reliability. For each parity-check, the weight value is obtained by finding the lowest magnitude in the set of adjacent channel samples. During each iteration, a summation E_k is computed over the adjacent weighted parity check results for each symbol position k. The symbol with the maximum E_k (or minimum, depending on convention) is flipped. The weights are only calculated once, at the start of decoding, however the WBF algorithm requires, at iteration, a summation over several weights for each symbol - a substantial increase in complexity compared to the original BFA. In addition to the increased arithmetic complexity, WBF has two major drawbacks: first, a potentially large number of iterations are required because only one bit may be flipped in iteration. Second, the algorithm must perform a global search to find the maximum E_k out of all symbols, resulting in a large latency per iteration that increases with codeword length, thereby hindering a high-throughput implementation. Researchers introduced several improvements to the WBF. We introduced the Modified WBF (MWBF) algorithm, which obtained improved performance with a slight increase in complexity. We described another Improved MWBF (IMWBF) algorithm which offered further improvement by using the parity-check procedure from the MS algorithm to determine the parity-check weights — another substantial increase in complexity. Both of these methods inherit the two key drawbacks associated with single-bit flipping in the WBF algorithm. Recently, we introduced a Parallel WBF (PWBF) algorithm, which reduces the drawbacks associated with single bit flipping in the other WBF varieties [1]. In the PWBF algorithm, the maximum (or minimum) E_i metric is found within the subset of symbols associated with each parity-check. In the PWBF algorithm, it is still necessary to find the maximum E_i from a set of values, which costs delay, but the set size is significantly reduced compared to the other WBF methods, and it is independent of codeword length. In spite of these improvements, PWBF retains the complex arithmetic associated with IMWBF. To reduce the arithmetic complexity of bit-flipping algorithms, we devised the GDBF algorithm as a gradient-descent optimization model for the ML decoding problem. Based on this model, the wes of obtained single-bit and multi-bit flipping algorithms that require mainly binary operations, similar to the original BFA. The GDBF methods require summation of binary parity-check values, which is less complex than the WBF algorithms that require summation over independently weighted syndrome values. The single-bit version of the GDBF algorithm (S-GDBF) requires a global search to discover the least reliable bit at

iteration. The multi-bit GDBF algorithm (M-GDBF) uses local threshold operations instead of a global search, hence achieving a faster initial convergence. In practice, the M-GDBF algorithm did not always provide stable convergence to the final solution. To improve convergence, the wes adopted a mode switching strategy in which M-GDBF decoding is always followed by a phase of S-GDBF decoding, leveraging high speed in the first phase and accurate convergence in the second. Although the mode-switching strategy provided a significant benefit, the algorithm was still subject to spurious local maxima. We obtained further improvements by introducing a "hybrid" GDBF algorithm (H-GDBF) with an escape process to evade local maxima. The H-GDBF algorithm obtains performance comparable to MS, but the escape process requires evaluating a global objective function across all symbols. When the objective function crosses a specified threshold during the S-GDBF phase, the decoder switches back to M-GDBF mode, then back to S-GDBF mode, and so on until a valid result is reached. To date, H-GDBF is the best performing GDBF variant, but requires a maximum of 300 iterations to obtain its best performance, compared to 100 for M-GDBF and S-GDBF. The major disadvantages of this algorithm are its use of multiple decoding modes, the need to optimize dual thresholds for mode switching and bit flipping, the global search operation and the global objective function used for mode switching. These global operations require an arithmetic operation to be computed over the entire code length, and would be expensive to implement for practical LDPC codes with large codeword length. Several researchers proposed alternative GDBF algorithms in order to obtain fully parallel bit-flipping and improved performance.

3. Code Construction

The objective is to design a parity check matrix of size M ×N with number of ones in rows and columns distributed according to the specified degree distribution pair. In this article, we specify the check node degree distribution as dc and dv which are 2×K arrays. The first row contains number of ones for a fraction of rows (or columns) and the second row contains the fraction of rows (or columns). The algorithm that we used is described next. We start with an empty matrix, and for each column, select a random row and assign ones to that (row, column) till the number of ones in that column equals the required number for that column. Before assigning a one, the following checks are made:

• Check if the degree constraint for the corresponding row is violated.

• Check if any cycles of length four will be formed.

If any of the above two conditions are violated, select and another random row and check again. Continue till a row is found which together with that column satisfies the above two constraints. Note that for the algorithm to be able to distribute ones properly the degree distribution equation should be satisfied.

$$M = N \frac{\Sigma dv}{\Sigma dc}$$

There are various methods of representing the bipartite graph required for decoding. One way is to look at the bipartite graph as an "inter leaver". The decoder structure exactly resembles the decoder of a Repeat Accumulate code, only without the accumulator [2]. The received vector is passed on to the Variable Node Decoder (VND) which in the first round simply repeats the elements of the received vector. The output of the variable node decoder is then passed into the inter leaver the output of which is fed in to the Check Node Decoder (CND). The CND's output is then de interleaved and passed to the VND. It

only remains now to explicitly specify the inter leaver, given the parity check matrix. There are two ways. One is to use a block inter leaver, like the classical approach. Write into the parity check matrix row wise and read from it column wise & vice versa for the de inter leaver. However, there is another better way [2].

3.1 Sum-Product algorithm (SPA)

The tanh-based SPA can be described in the following steps.

1) Initialization step

The initial values of the LLR can be obtained from the QAM demodulator output y_n. These initial values are used as $q_{n\to m}$, the first iteration's input message to the check node update step (Horizontal step).

2) Horizontal step

The horizontal step at a check node m is dedicated to process the messages which are coming from the variable nodes $q_{n\to m}$ to calculate the reply messages $r_{m\to n}$ for all $n \in \jmath\{m\}$. So for each check node m

$$r_{m\to n} = \left[\prod_{n \in N(m)\backslash n} * \text{sign}(q_{n\to m}) \right] \times 2\tanh^{-1}\left[\prod_{n \in N(m)\backslash n} \times \tanh\left(\frac{|q_{n\to m}|}{2} \right) \right]$$

3) Vertical step

The vertical step at a variable node n is dedicated to process the messages which are coming from the check nodes $r_{m\to n}$ to calculate the reply messages for $q_{n\to m}$ all $m \in N\{n\}$. So for each variable node n compute:

$$q_{n\to m} = y_n + \sum_{m \in M(n)\backslash n} r_{m\to n}(x_n)$$

4) Decision step:

For each variable node, the LLR values are updated according to:

$$L(x_n) = y_n + \sum_{m \in M(n)} r_{m\to n}(x_n)$$

The LLR values are applied to hard decision to decide on the possible value of x_n to be 1 if $L(x_n) < 0$ and zero otherwise. The syndrome is then calculated and checked to decide successful decoding if the syndrome is zero or to proceed to the next iteration if the syndrome condition is not satisfied. This process continues till either the code word is successfully decoded or the maximum iterations are exhausted. Despite the optimum performance of the tanh-based SPA algorithm, it is difficult to implement due to the need to calculate $\tanh^{-1}(.)$ and tanh(.) functions which requires a series computation or saving in look up tables. The tanh rule can be alternatively approximated using the Jacobi rule. This approximation yields the Min-Sum algorithm [3] which is more implementation friendly.

3.2 Min-Sum algorithm

The Min-Sum algorithm follows the same steps as the tanh rule SPA. It is composed of the same steps with only single change in the calculation of the horizontal step which can be manipulated to be:

$$r_{m\to n} = \left[\prod_{n \in \jmath(m)\backslash n} * \text{sign}(q_{n\to m}) \right] \times \min_{n \in \jmath(m)\backslash n} (|q_{n\to m}|)$$

The above algorithm is easier to implement as it gets rid of the tanh calculation. However, the approximation to the exponential calculations to the min (.) leads to some loss of performance compared to the tanh-based SPA algorithm. This loss of performance is partially recovered by Scaled Min-Sum algorithm [3].

3.3 Scaled Min-Sum algorithm

In order to improve the performance of the Min-Sum algorithm, and make it closer to the performance of the tanh based SPA algorithm, a constant scaling factor ($\alpha < 1$) can be applied to the check node update equation (Horizontal step) in all iterations. In other words, converts the Horizontal step to:

$$r_{m\to n} = \alpha \times \left[\prod_{n \in \jmath(m)\backslash n} * \text{sign}(q_{n\to m}) \right] \times \min_{n \in \jmath(m)\backslash n} (|q_{n\to m}|)$$

This scaling factor can be calculated by either density evolution [3] or EXIT chart to maximize the performance of Scaled Min-Sum algorithm.

rule. The scaling factor should increase exponentially with iterations and its final value is 1. So we approximates the scaling factor graph to a stair graph with constant horizontal step S, and the scaling factor takes values which is exponential and at the same time easy to implement. The variable scaling factor can be calculated as:

$$\alpha = 1 - 2^{[i/s]}$$

Where i / s is the first integer larger than i / S. i is the iteration index which take values {1, 2, 3 …}. So the scaling factor sequence is {0.5, 0.75, 0.875, 0.9375….}. And this sequence is:-

1) Easy to design, because it is based only on parameter S.

2) Does not need to store a specific scaling sequence for each code rate. The scaling sequence of every code rate requires storing only the step size S and number of required shifts.

3) Easy to implement, because it only requires shifting right by i / s then subtraction. Number of required shifts can be stored in a register and increased by 1 every S iterations [3].

4. SIMULATION AND RESULTS

Comparisons between various decoding algorithms are displayed in fig.1and fig.2.

3.4 Low complexity variable scaled min-sum algorithm

The Simplified Variable Scaled (SVS) Min-Sum algorithm addresses the particular point of simplified per-iteration updated scaling

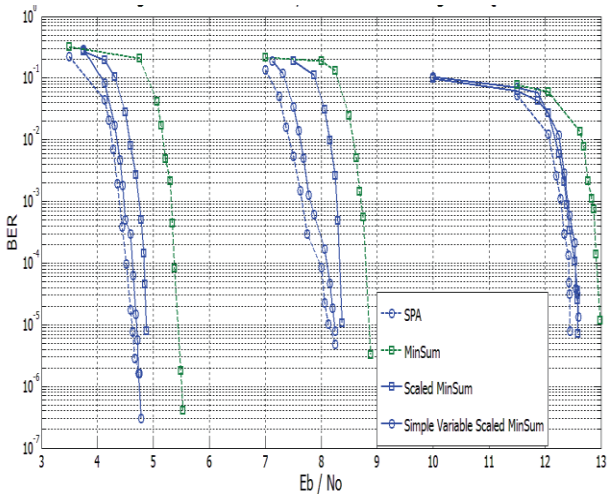

Fig.1. Short LDPC codes with rates = {0.25, 0.5, and 0.75}.

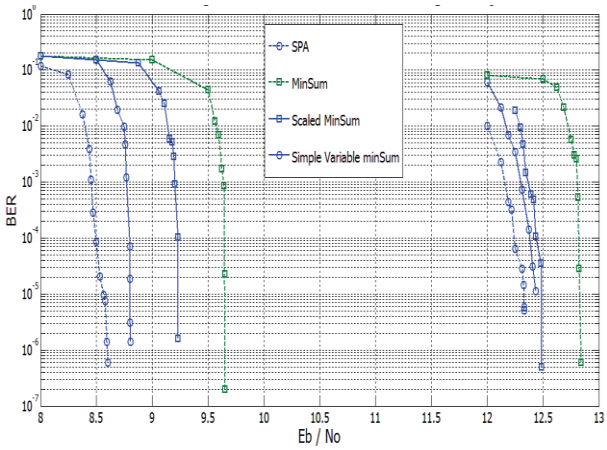

Fig.2. Normal length LDPC code with rates = {0.5, 0.75}.

As shown in fig 1 and 2, SVS Min-Sum algorithm has superior performance than constant Scaled Min-Sum algorithm with optimum α. SVS Min-Sum performance is very similar to SPA performance but with much lower complexity.

5. CONCLUSION AND FUTURE WORK

The algorithm for constructing LDPC codes using the Extended Bit-Filling algorithm is certainly attractive. A performance comparison of all known constructions of LDPC codes is the target. Such a performance comparison will be very useful in giving insights into the trade-offs between different parameters that influence the performance of LDPC codes.

We suggest one problem in particular. The concentration bounds we use apply to the asymptotic behavior of low-density parity-check codes, but they do not adequately explain the behavior of small codes, say with only thousands of bits. For such small codes, the corresponding bipartite graphs necessarily have small cycles, which is a complication our asymptotic analysis cannot adequately handle. More understanding of small codes could be extremely useful for designing low-density parity-check codes and making them the code of choice in practice. Simulation results indicated the validity of the idea of the heuristic easy to implement update of the scaling factor and its superior performance compared to both the regular Min-Sum and the fixed scaling factor Scaled Min-Sum algorithms. As for future work, the results can be extended to other irregular LDPC code.

REFERENCES

1] Gopalakrishnan Sundararajan, Winstead and Emmanuel Boutillon,"Noisy Gradient Descent Bit-Flip Decoding for LDPC Codes", http://dx.doi.org/10.1109/TCOMM.2014.2356458.

[2] Prasanna Sethuraman,"Design and Implementation of Low Density Parity Check Codes", prasanna@mytum.de.

[3] Ahmed A. Emran and Maha Elsabrouty, "Simplified Variable-Scaled Min Sum LDPC decoder for irregular LDPC Codes,"

[4] Jean Nguyen, Dr. Borivoje Nikolic and Engling Yeo, "Design of a Low Density Parity Check Iterative Decoder,".

[5] Ce-lun Liu, Jian-ping An, Xiang-yuan Bu, "A Simplified Decoding Algorithm for Non binary LDPC Codes,"