ON FINITE PRECISION IMPLEMENTATION OF LOW DENSITY PARITY CHECK CODES DECODER

Tong Zhang, Zhongfeng Wang and Keshab K. Parhi

Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN 55455, USA
E-mail:{tzhang, zwang, parhi}@ece.umn.edu

ABSTRACT

In this paper, we analyze the finite precision effects on the decoding performance of Gallager's low density parity check (LDPC) codes and develop optimal finite word lengths of variables as far as the tradeoffs between the performance and hardware complexity are concerned. We have found that 4 bits and 6 bits are adequate for representing the received data and extrinsic information, respectively. Simulation results indicate that the quantization scheme we have developed for the LDPC decoder is effective in approximating the infinite precision implementation.

1. INTRODUCTION

Low Density Parity Check (LDPC) codes, first introduced by Gallager [1][2], have recently received a lot of attention because of their excellent performance on the binary symmetric channel (BSC) as well as on the additive white Gaussian noise (AWGN) channel. LDPC codes are widely considered as serious competitor to turbo codes. The main advantages of LDPC codes over turbo codes are: 1) they empirically don't make undetected errors because of good distance properties; 2) there exist low complexity and highly parallelizable decoding approaches. However, the high complexity encoding process is the main deficiency of LDPC codes. Recently, several efficient encoding approaches have been proposed in [3][4].

Originally, as proposed in [1][2], LDPC codes are specified by a very sparse regular parity check matrix with a small fixed number $j > 2$ of 1's per column and a small fixed number $k > j$ of 1's per row. It has been shown that regular LDPC codes have good performance, but a little worse than turbo codes. Recently, Richardson, Shokrollahi and Urbanke [5] proposed irregular LDPC codes, which outperform, on memoryless channels, the best known turbo codes.

LDPC codes can be efficiently decoded using the iterative belief propagation (BP) algorithm [6], which approximates the maximum likelihood decoding. In the hardware implementation, the Log-BP algorithm would be employed in order to decrease the decoder complexity. As we will see later, in BP algorithm, extrinsic information need to be iteratively exchanged between two decoding steps. In LDPC decoder implementation, the hardware implementing this message exchange is called *interleaver*. With increase of block size, the interleaver will become more and more complicated and take a large portion of overall hardware resource.

This research was supported by the Army Research Office by grant number DA/DAAG55-98-1-0315.

Compared with the irregular ones, although the performance of regular LDPC codes is a bit worse, it can be efficiently constructed by using the partly "structured" approach [2], which may significantly simplify the interleaver design in the decoder. Thus, considering the tradeoff between hardware complexity and performance, the regular LDPC codes seem to be a good choice for many applications.

As far as a practical system implementation is concerned, the finite precision effects is an important issue to be considered. However, up to our best knowledge, the precision effects on the performance of the LDPC codes decoder have not been addressed in the literature. Among various variables, the word lengths of received data (soft input) and extrinsic information are especially important. In this paper, we analyze the finite precision effects on the Log-BP based regular LDPC codes decoder performance and derive the optimal word lengths considering tradeoffs between the hardware complexity and the performance. Furthermore, we propose a novel quantization scheme for extrinsic information, which can improve up to 0.1 dB over AWGN channel compared with conventional uniform quantization scheme.

2. DECODING ALGORITHM

LDPC codes can be efficiently decoded by iterative BP algorithm. But the BP algorithm [6] contains many multiplications which will result in high computation complexity if implemented directly in hardware. In order to reduce the complexity, we can convert these complicated operations into additive form by introducing some logarithmic quantities, which leads to the Log-BP algorithm [2]. In the following, we summarize the iterative decoding of LDPC codes based on the Log-BP algorithm. As to the details of BP algorithm, readers are referred to [6].

Before the brief description of the decoding algorithm, we introduce some definitions. Let H denote the parity check matrix. We define the set of bits n that participate in parity check m as $M(n) = \{n : H_{mn} = 1\}$. Similarly, we define the set of parity checks m in which bit n participates as $N(m) = \{m : H_{mn} = 1\}$. We denote the set $N(m)$ with bit n excluded by $N(m) \setminus n$, and the set $M(n)$ with parity check m excluded by $M(n) \setminus m$.

The iterative decoding algorithm based on the Log-BP approach is given as follows:

Algorithm 2.1
Input: The prior probabilities $p_n^0 = P(x_n = 0)$ *and* $p_n^1 = P(x_n = 1) = 1 - p_n^0$, $n = 1, \cdots, N$;
Output: The hard decision of codeword, $\hat{x_n}$, $n = 1, \cdots, N$;
Procedure:

1. *Initialization. For each* n*, set* $\gamma_n = \log \frac{p_n^0}{p_n^1}$ *and for each* $(m,n) \in \{(i,j)|H_{ij} = 1\}$*, compute*

$$\alpha_{mn} = sign(\gamma_n) \log \left(\frac{1 + e^{-|\gamma_n|}}{1 - e^{-|\gamma_n|}} \right).$$

2. *Iterative Decoding*

   - *Horizontal step. For each* m*,* n*, compute*

   $$\beta_{mn} = \log \left(\frac{1 + e^{-\alpha}}{1 - e^{-\alpha}} \right) \prod_{n' \in N(m) \backslash n} sign(\alpha_{mn'})$$

   where $\alpha = \sum_{n' \in N(m) \backslash n} |\alpha_{mn'}|$.

   - *Vertical step. For each* m*,* n*, update*

   $$\alpha_{mn} = sign(\gamma_{mn}) \log \left(\frac{1 + e^{-|\gamma_{mn}|}}{1 - e^{-|\gamma_{mn}|}} \right)$$

   where $\gamma_{mn} = \gamma_n + \sum_{m' \in M(n) \backslash m} \beta_{m'n}$*. For each* n*, update the "pseudo-posterior log-likelihood ratio"* λ_n *at this iteration, given by*

   $$\lambda_n = \gamma_n + \sum_{m \in M(n)} \beta_{mn}.$$

   - *Decision step.*

     (a) Do hard decision on decoded codeword $\hat{\mathbf{x}} = [\hat{x}_n]$ *such that* $\hat{x}_n = 0$ *if* $\lambda_n > 0$ *and* $\hat{x}_n = 1$ *if* $\lambda \leq 0$*;*

     (b) If $H\hat{x} = 0$ *then algorithm terminates, else go to Horizontal step. A failure will be declared if pre-set maximum number of iterations occurs without successfully decoding.* ■

The above algorithm contains a function $f(x) = \log(\frac{1+e^{-|x|}}{1-e^{-|x|}})$, which can be implemented as a Look-Up Table (LUT) in hardware. In this work, we assume that the binary (N, j, k) LDPC codes considered are modulated by BPSK and used for error control over AWGN channel with noise spectrum density N_0. So in above algorithm, the prior probabilities $P(x_n = 0) = \frac{e^{-4x_n^s E_c/N_0}}{1+e^{-4x_n^s E_c/N_0}}$ and thus the γ_n is initialized to $-4x_n^s E_c/N_0$, where x_n^s and E_c denote the soft information input and coded bit energy, respectively.

3. CODE CONSTRUCTION AND DECODER STRUCTURE

The binary regular LDPC codes $C(N, j, k)$ have block length N and parity-check matrix with exactly j $1's$ in each column and k $1's$ in each row. When constructing the regular LDPC codes, we can use the following partly "structured" approach [2] (as shown in Fig. 1): Let k be a divisor of N, and the (N, j, k) parity check matrix H consist of Nj/k rows. Matrix H is divided into j submatrices $H_1, \cdots, H_j$, each has the dimension of $(N/k, N)$ and contains a single 1 in every column. The first of these submatrices, H_1, contains all its $1's$ in descending order, such that the i^{th} row contains $1's$ in columns $(i-1)k+1$ to ik. The successive $j-1$ submatrices are formed by random permutation of the columns of the first matrix H_1, that is, $H_2 = \pi_1(H_1), \cdots, H_j = \pi_{j-1}(H_1)$, where π_i's are independent column permutations. Through Monte

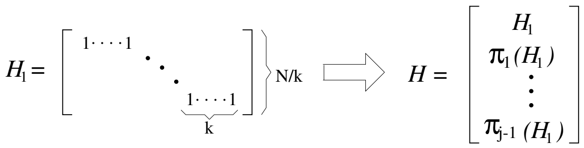

Fig. 1. Partly "Structured" Code Construction

Carlo simulation, it's shown that, empirically, there is no performance difference between this construction approach and the fully random construction ones.

Based on above construction approach, we present a decoder structure that consists of three processor arrays and three interleaver blocks. The schematic block diagram that illustrates the decoder structure is shown in Fig. 2.

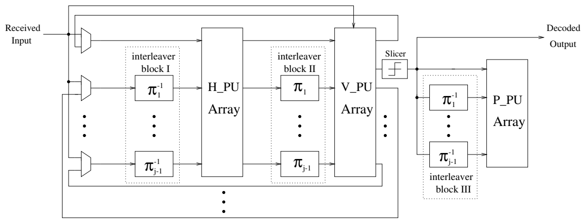

Fig. 2. Decoding Iteration Structure

In the decoder, each interleaver block consists of $j-1$ interleavers (π_i or π_i^{-1}) and each interleaver takes the incoming data block with size N and rearrange them independently according to the corresponding permutation pattern. Various interleaving (or permutation) schemes have been intensively studied for Turbo-codes and except the fully random interleaving, many other interleaving strategies were proposed for Turbo-codes, which can be very likely used for regular LDPC codes. This issue is beyond the range of this paper and interested readers are referred to [8][9][10].

H_PU array consists of a number of *Horizontal updating Processor Unit* (H_PU) which compute the extrinsic information β_{mn} in parallel. Similarly, V_PU array consists of *Vertical updating Processor Unit* (V_PU) which compute α_{mn} in parallel. These two processor units array execute the horizontal and vertical step in Log-BP algorithm, respectively. The slicer performs the hard decision on each bit x_n according to the pseudo-posterior log-likelihood ratio λ_n, and Parity check Processor Unit (P_PU) array performs the parity check on the tentative codeword, where each P_PU simply contains some XOR gates. The block diagram of H_PU and V_PU is shown in Fig. 3.

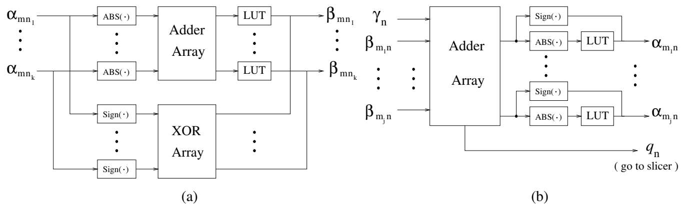

Fig. 3. (a) Horizontal updating Processor Unit (H_PU), (b) Vertical updating Processor Unit (V_PU)

4. FINITE PRECISION ANALYSIS

In this section, we analyze the finite word length effect on the performance of regular LDPC codes decoder. The possible tradeoff between hardware complexity and decoding performance is discussed. Furthermore, a novel quantization scheme for extrinsic information α_{mn} and β_{mn} is proposed. The simulation results are presented for two different code configurations, *i.e.*, (a) $N = 1020$, $j = 3$ and $k = 6$, and (b) $N = 4092$, $j = 3$ and $k = 6$. The code rate of both cases are $1/2$. In this work, LDPC codes are modulated by BPSK and transfered over AWGN channel. We note that in both code length configurations, we pick 30 permutation patterns at random and select the one leading to the best results.

4.1. Quantization of Received Data

We first consider the quantization of received data. Since receiving buffer is needed to store the received data, quantization of received data significantly affect the total decoder complexity. A large word length not only increases the hardware overhead for the buffers but also causes a large number of hardware for the iterative decoding computation. A small word length may result in very poor performance. Hence, our concerns are limited to $W_{in} = 3$, 4 and 5.

Let $q : f$ denote the quantization scheme in which totally q bits are used, of which f bits are used for the fractional part of the value. Under the same value of q, the precision that can be maintained is proportional to the value of f, but the presented dynamic range is in inverse relation to f. Various quantization schemes for the received data such as 3:1, 3:2, 4:1, 4:2, 4:3, 5:2 and 5:3 have been investigated. It's shown that for different value of q, *i.e.,* 3, 4 and 5, the quantization schemes 3:1, 4:2 and 5:3 have the best performance, respectively, and the simulation results for these schemes are shown in Fig. 4. We can see that the difference between 4:2 and 5:3 cases is quite small within a wide range of BER, but the difference between 4:2 and 3:1 is significant. Thus it turns out that using the 4:2 scheme seems to be the optimal tradeoff between hardware complexity and decoding performance.

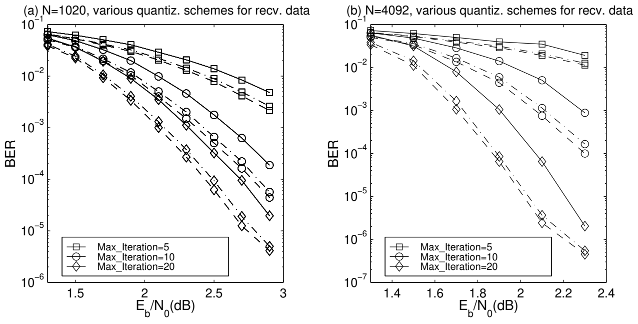

Fig. 4. Finite precision simulations with various quantization schemes of received data for (a) N=1020 and (b) N=4092, where solid lines correspond to the 3:1 scheme, dash and dash dot lines for the 5:3 and 4:2 schemes, respectively.

4.2. Quantization of α_{mn} and β_{mn}

We have known that the whole Log-BP decoding process mainly consists of iteratively exchanging and updating the extrinsic information α_{mn} and β_{mn}, performed by interleaver block I & II and updating processor unit arrays, respectively. Therefore, quantization of α_{mn} and β_{mn} is also critical for hardware implementation.

In Log-BP decoding algorithm, the magnitudes of both α_{mn} and β_{mn} are the outputs of function $f(x) = \log\left(\frac{1+e^{-|x|}}{1-e^{-|x|}}\right)$, which is implemented as a LUT in hardware as shown in Fig. 3. The curve of $f(x)$ (for $x > 0$) is shown as in Fig. 5. Since the magnitudes of input and output of function $f(x)$ may range from 0 to $+\infty$, we must saturate the input and output with the maximum value X_{max} and Y_{max}, respectively. From simulation, it shows that when both X_{max} and Y_{max} are set between 3.5 and 4, there is nearly no performance degradation and at the same time only 3 bits (including 1 sign bit) are needed for integral part of α_{mn} and β_{mn}.

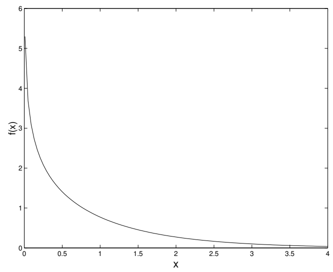

Fig. 5. Curve of function $f(x)$ ($x > 0$)

From Fig. 5, we can see that $f(x)$ is non-linear and its slope decreases with the increase of x. It is intuitive that the larger the slope, the smaller quantization step should be used for a good performance. Thus, under the same total quantization bit number q, a non-uniform quantization scheme will outperform the uniform one. Therefore, instead of using uniform quantization scheme, *i.e.*, $q : f$, for α_{mn} and β_{mn}, we propose a novel *variable precision* quantization scheme: If the magnitude of output of LUT, denoted as v, is less than 1, then v is represented by $q : q-1$ scheme and its MSB v_{q-1} is set to 0; else its MSB v_{q-1} will be set to 1 and the point lies between v_{q-3} and v_{q-4}, and the value of integral part is interpreted as

$$v_{q-1} \cdot \overline{v}_{q-2} \cdot \overline{v}_{q-3} \cdot 2^2 + v_{q-2} \cdot 2 + v_{q-3}$$

where $\overline{v}_i$ represent the bitwise complement of v_i. Combining this magnitude with its sign bit produces the final representation of α_{mn} and β_{mn}.

In this work, different values of q for α_{mn} and β_{mn}, such as 5, 6, and 7, have been examined for both uniform quantization and the above variable precision quantization schemes. It turns out that $q = 6$ is the best choice for both cases considering the optimal tradeoff between hardware complexity and performance. Simulation results for the uniform 6:3 quantization and proposed variable precision quantization schemes are shown in Fig. 6. It can be seen that compared with uniform quantization scheme, variable precision scheme improves the performance (about 0.1 dB in some range of SNR). It needs to be pointed out that using variable precision scheme will slightly increase the hardware overhead in the adder array, *i.e.*, circuit to decide the location of point, which is ignorable even only compared with the adder array.

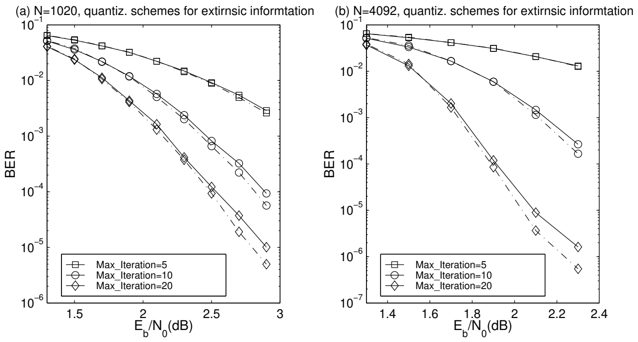

Fig. 6. Finite precision simulations with various quantization schemes of extrinsic information for (a) N=1020 and (b) N=4092, where $q = 6$, solid lines correspond to 6:3 scheme and dash dot lines for the variable precision quantization scheme.

4.3. Simulation Results Summary

Infinite precision and finite precision simulation results are shown in Fig. 7, including the BER vs. SNR and *average number of iterations* vs. SNR. The average number of iterations is an important parameter representing the decoding speed and power consumption. The quantization schemes used are 4:2 for received data and variable precision quantization for extrinsic information α_{mn} and β_{mn} with $q = 6$. Three values of maximum number of iterations, *i.e.*, 5, 10, and 20, are used. It can be seen that, for both $N = 1020$ and $N = 4092$ cases, the total quantization loss compared with the infinite precision case is no more than 0.1 dB.

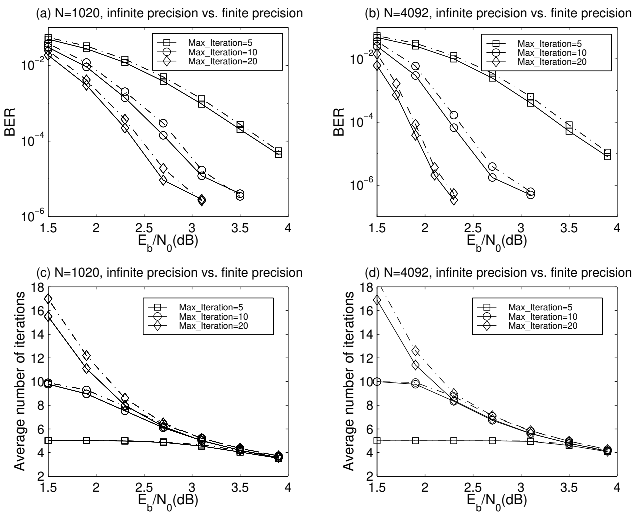

Fig. 7. Infinite precision vs. finite precision simulations for (a)(c) N=1020 and (b)(d) N=4092, where solid lines correspond to infinite case, dash dot lines for the finite case.

5. CONCLUSION

Various quantization schemes for the received data and extrinsic information for Log-BP based LDPC codes decoder were investigated and the optimal choice considering the tradeoff between the hardware complexity and the performance were addressed in this paper. A novel variable precision quantization scheme for extrinsic information was proposed to improve the performance. The overall finite precision simulations have shown that the quantization scheme we have developed for the Log-BP based decoder is effective in approximating the infinite precision implementation.

6. REFERENCES

[1] R. G. Gallager, "Low-density parity-check codes", *IRE Transactions on Information Theory*, vol. IT-8, pp. 21–28, Jan. 1962.

[2] R. G. Gallager, *Low-Density Parity-Check Codes*, M.I.T Press, 1963.

[3] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes", *submitted IEEE Transactions on Information Theory*.

[4] M. Chiani and A. Ventura, "Graph-based construction of encoders for irregular low-density parity-check codes", *submitted*.

[5] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of provably good low-density parity check codes", *submitted IEEE Transactions on Information Theory*.

[6] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices", *IEEE Transactions on Information Theory*, vol. 45, pp. 399–431, Mar. 1999.

[7] Z. Wang, H. Suzuki, and K. K. Parhi, "VLSI implementation issues of turbo decoder design for wireless applications", *in IEEE Workshop on SiGNAL PROCESSING SYSTEMS (SiPS)*, pp. 503 –512, 1999.

[8] F. Daneshgaran and M. Mondin, "Design of interleavers for turbo codes: iterative interleaver growth algorithms of polynomial complexity", *IEEE Transactions on Information Theory*, vol. 45, pp. 1845 –1859, Sept. 1999.

[9] J. Hokfelt, O. Edfors, and T. Maseng, "Interleaver structures for turbo codes with reduced storage memory requirement", *in Vehicular Technology Conference*, vol. 3, pp. 1585 –1589, 1999.

[10] S. Crozier, J. Lodge, P. Guinand, and A. Hunt, "Performance of turbo codes with relative prime and golden interleaving strategies", *in Sixth International Mobile Satellite Conference (IMSC '99)*, pp. 268–275, Ottawa, Canada, June 1999.