On a Different Perspective and Approach to Implement Adaptive Normalized BP-based Decoding for LDPC Codes

Panupat Poocharoen and Mario E. Magaña

***Abstract*— In this paper, we propose an improved version of the min-sum algorithm for low density parity check (LDPC) code decoding, which we call "adaptive normalized BP-based" algorithm. Our decoder provides a compromise solution between the belief propagation and the min-sum algorithms by adding an *exponent offset* to each variable node's intrinsic information in the check node update equation. The extrinsic information from the min-sum decoder is then adjusted by applying a negative power of two scale factor, which can be easily implemented by right shifting the min-sum extrinsic information. The difference between our approach and other adaptive normalized min-sum decoders is that we select the normalization scale factor using a clear analytical approach based on underlying principles. Simulation results show that the proposed decoder outperforms the min-sum decoder and performs very close to the BP decoder, but with lower complexity.**

***Index Terms*—**LDPC codes, iterative decoding, belief propagation, sum product, min-sum, modified min-sum

I. INTRODUCTION

LDPC codes have become very popular and are currently used in many wireless applications standards such as the second generation digital video broadcasting - satellite (DVB-S2) and IEEE 802.16e-2005. In decoding LDPC codes, the belief propagation (BP) algorithm performs very close to the Shannon limit [1], at the expense of high computational complexity due to the check node update equation. This complexity can be reduced using the min-sum algorithm to approximate the BP algorithm by [2]

$$L_{e,j,l}^{(i)} = 2\text{arctanh}\left(\prod_{l'\in B(j)\setminus l}\tanh(L_{j,l'}^{(i-1)}/2)\right) \approx \prod_{l'\in B(j)\setminus l} sign(L_{j,l'}^{(i-1)})\cdot \min_{l'\in B(j)\setminus l}\left\{\left|L_{j,l'}^{(i-1)}\right|\right\}, \quad (1)$$

where $L_{e,j,l}^{(i)}$ is the log-likelihood ratio (LLR) of check node j at variable node l. The superscript i represents the i^{th} iteration and the subscript e denotes the extrinsic information. $B(j)$ is the set of variable nodes connected to the j^{th}

check node. The min-sum algorithm can be viewed as an approximation of BP at high SNR, i.e. $L_{j,l'}^{(i-1)} >> 1$. From (1), the magnitude of $L_{e,j,l}^{(i)}$ computed by the BP algorithm, however, is always less than that of the min-sum algorithm and equality holds only when all variable nodes in $B(j) \setminus l$, but the minimum value, have their LLR magnitude very large, ideally ∞. On the other hand, when the LLRs of the variable nodes are small or the check weight $\rho = |B(j)|$ becomes large, $\prod_{l' \in B(j) \setminus l} \tanh(|L_{j,l'}^{(i-1)}|/2)$ becomes much smaller than $\min_{l' \in B(j) \setminus l} \{|L_{j,l'}^{(i-1)}|\}$, causing differences in calculations between the min-sum and BP algorithms [3], [9] and hence performance loss.

To mitigate this problem, several approaches which introduce a correction term to the min-sum algorithm have been proposed. In [3] and [4], normalized BP-based and offset BP-based algorithms are proposed. The former introduces a constant correction factor α to scale the LLR of each check node's extrinsic information, i.e.

$$L_{e,j,l}^{(i)} \approx \prod_{l' \in B(j) \setminus l} sign(L_{j,l'}^{(i-1)}) \cdot \min_{l' \in B(j) \setminus l} \{|L_{j,l'}^{(i-1)}|\} / \alpha , \tag{2}$$

whereas the latter uses a constant offset β to correct the LLR, i.e.

$$L_{e,j,l}^{(i)} \approx \prod_{l' \in B(j) \setminus l} sign(L_{j,l'}^{(i-1)}) \cdot \max\left(\min_{l' \in B(j) \setminus l} \{|L_{j,l'}^{(i-1)}|\} + \beta, 0 \right). \tag{3}$$

Although these algorithms perform relatively well, the normalized BP-based algorithm does not work well when the LDPC code is long [5]. Also, most of the offset BP-based algorithm studies only consider low-weight LDPC codes such as 1/2 code rate (3,6) LDPC codes, and do not guarantee good performance when applied to larger check weights. As noted in [3] and [6], some algorithms may not perform well when the check weights become large. To address this, the density evolution technique [1], [5] is applied to pre-design optimal scale factor and offset values. However, these values are code-rate specific. Several designed values and their performances for different code rates are found in [7].

While the density evolution technique provides a very good pre-design single constant, it requires analysis for each code rate and the calculation of the estimated probability density function of the check node's LLR [7] involves fast Fourier transforms.

Variants of the offset BP-based algorithm are proposed in [4], [8], [9], where attempts are made to endow decoders with flexibility by adaptively changing the offset value to achieve good decoding without using density evolution.

These decoders mostly use either piecewise constant or linear approximations [22] to represent the offset, which is a non-linear function of the intrinsic information, and are based on Jacobian logarithm [13]. Since (3) performs pair-wise comparisons of the LLR of the variable nodes in $B(j)\setminus l$, it can be simplified to

$$L\left(l_1^{'} \oplus l_2^{'}\right) = sign(L_1)sign(L_2)\cdot \min\left\{|L_1|,|L_2|\right\} + \beta, \tag{4}$$

where L_1 and L_2 are the LLRs of the variable node pair $l_1^{'}, l_2^{'} \in B(j)\setminus l$, respectively, and β is given by

$$\beta = \log\left(\frac{1+e^{-|L_1+L_2|}}{1+e^{-|L_1-L_2|}}\right). \tag{5}$$

Most modified BP-based algorithms focus on the offset BP-based algorithm and propose different approaches to approximate the offset value [4], [8], [22], in order to lower the overconfidence of the LLR in the min-sum algorithm. One of the reasons may be that the normalized BP-based algorithm involves a scale factor and a multiplication operation. Nevertheless, the normalized BP-based algorithm can still be very attractive. In fact, by properly selecting α one can show that it can even outperform the offset BP-based algorithm [7].

In general, BP algorithm approximations may be implemented using a look up table (LUT) to compute the non-linear functions. However, the goal of a modified BP–based decoder is to avoid, as much as possible, using time consuming complex operations such as non-linear functions, multiplications, and searching LUTs, especially if implemented outside the decoder processor [10].

Finally, very few researchers have considered adaptive versions of the normalized BP-based algorithm. To the best of our knowledge, only the work of [21] was published during the same time period we were conducting our research.

Motivated by these observations, we propose a decoder which is a modified version of the normalized algorithm. We call it “adaptive normalized BP-based algorithm”. It adaptively changes the scale factor depending on the LLR values of the variable nodes, which is different than the approach proposed in [21], where the scale factor changes adaptively depending on the syndrome result. The novelty of our approach lies on how the intrinsic information is approximated and used to estimate the scale factor. Moreover, the algorithm’s scale factor is provided in the form of a negative power of 2, which can be implemented easily using a shifter instead of a multiplication/division as in [7]. We show herein that our decoder attains a performance close to that of the BP decoder, especially with large check

weights while maintaining a complexity comparable to that of the min-sum decoder. The only additional hardware components for its implementation are summations and barrel-shifters to handle the extrinsic information.

Before proceeding with our algorithm, it is worth mentioning analog LDPC decoders and stochastic decoders. Analog decoders have recently been applied to iterative decoders such as those used in turbo and LDPC codes [11]-[15]. This is due to their infinite precision and seamless iterations which provide fast convergence when large numbers of iterations and low power consumption are required [11]. They require careful design in order to prevent mismatches due to component variations and noise [11]. Nevertheless, as mentioned in [12], when comparing analog and digital decoders, the debate is not over fundamentals, but how to exploit the advantages of each. Thus, digital decoders are still very attractive and innovative methods can be used to avoid high complexity, large numbers of iterations and still achieve excellent performance. Digital implementation of stochastic computations used in artificial neural network have been introduced for decoding LDPC codes [23] known as stochastic decoders. The approach approximates the sum-product algorithm and allows low complexity implementations of parallel LDPC decoders using low-cost FPGAs.

The paper is organized as follows: In section II, details of our adaptive normalized BP-based algorithm are provided. Section III presents and discusses the simulation results. Complexity analysis and conclusions are provided in sections IV and V, respectively.

II. ADAPTIVE NORMALIZED BP-BASED ALGORITHM

We now derive our adaptive normalized BP-based (ANBP) algorithm which allows α to change adaptively and makes the modified BP-based decoder applicable to a wide range of codes with different check weights without pre-designing α.

A. Algorithm derivation

The ANBP algorithm is based on the following observations on (1):

1) $0 \le \tanh(\left|L_{j,l'}^{(i-1)}\right| / 2) \le 1$.

2) $\prod_{l' \in B(j) \setminus l, l_{\min}} \tanh(\left|L_{j,l'}^{(i-1)}\right| / 2) << 1$ when the check weight ρ is large or $\tanh(\left|L_{j,l'}^{(i-1)}\right| / 2) << 1$.

3) The hyperbolic tangent can be linearly approximated over a large range of LLR values.

Let $a \equiv \prod_{l' \in B(j)\setminus l} \tanh(L_{j,l'}^{(i-1)}/2)$. When the row weight becomes large or when $L_{j,l'}^{(i-1)}$ is small, the value of a becomes even smaller than 1. The extrinsic information can then be expressed as

$$L_{e,j,l}^{(i)}(ANBP) \triangleq L_{e,j,l}^{(i)} = 2\text{arctanh}(a) = \log\left(\frac{1+a}{1-a}\right) = 2\left(a + \frac{a^3}{3} + \frac{a^5}{5} + \ldots\right) \approx 2a = 2\prod_{l' \in B(j)\setminus l} \tanh(L_{j,l'}^{(i-1)}/2). \quad (6)$$

Furthermore, we can rewrite (6) as

$$L_{e,j,l}^{(i)}(ANBP) \approx \prod_{l' \in B(j)\setminus l} sign(L_{j,l'}^{(i-1)}) \cdot 2 \min_{l' \in B(j)\setminus l} \left\{ \tanh\left(\frac{\left|L_{j,l'}^{(i-1)}\right|}{2} \right) \right\} \times \prod_{l' \in B(j)\setminus l, l_{\min}} \tanh\left(\left|L_{j,l'}^{(i-1)}\right|/2\right), \quad (7)$$

where

$$l_{\min} = \arg\min_{l' \in B(j)\setminus l} \left\{ \tanh\left(\frac{\left|L_{j,l'}^{(i-1)}\right|}{2} \right) \right\}. \quad (8)$$

For small $L_{j,l'}^{(i-1)}$, say, $L_{j,l'}^{(i-1)} < 1$, $\tanh\left(\frac{L_{j,l'}^{(i-1)}}{2}\right) \approx \frac{L_{j,l'}^{(i-1)}}{2}$, with only a small error. Hence,

$$L_{e,j,l}^{(i)}(ANBP) \approx \prod_{l' \in B(j)\setminus l} sign(L_{j,l'}^{(i-1)}) \cdot \min_{l' \in B(j)\setminus l} \left\{ \left|L_{j,l'}^{(i-1)}\right| \right\} \times \prod_{l' \in B(j)\setminus l, l_{\min}} \tanh\left(\left|L_{j,l'}^{(i-1)}\right|/2\right). \quad (9)$$

We can see that the extrinsic information computed in (9) is clearly less than that of the min-sum algorithm, i.e.,

$$L_{e,j,l}^{(i)}(ANBP) \le \prod_{l' \in B(j)\setminus l} sign(L_{j,l'}^{(i-1)}) \cdot \min_{l' \in B(j)\setminus l} \left\{ \left|L_{j,l'}^{(i-1)}\right| \right\}. \quad (10)$$

The extrinsic information obtained from this approach not only depends on the value of $\min_{l' \in B(j)\setminus l} \left\{ \left|L_{j,l'}^{(i-1)}\right| \right\}$ but also on the LLR values of other variable nodes. Therefore, it provides a relatively good alternative approximation of (1) and reduces the overconfidence in the min-sum algorithm. However, the approach is less accurate when $\min_{l' \in B(j)\setminus l} \left\{ \left|L_{j,l'}^{(i-1)}\right| \right\}$ becomes large and the check (row) weight is small, due to the nonlinearity of the hyperbolic tangent function.

We further reduce the complexity of (9) by approximating $\tanh\left(\left|L_{j,l'}^{(i-1)}\right|/2\right)$ with negative powers of two. Because $\tanh\left(\left|L_{j,l'}^{(i-1)}\right|/2\right) \in [0,1]$, we subdivide this interval into $2P$-1 subintervals whose centers are $c_1, c_2, \ldots, c_{2P-1}$, respectively, where

$$c_p = \begin{cases} 2^{-(|P-p|+1)}, & 1 \le p \le P \\ 1 - c_{2P-p}, & P+1 \le p \le 2P-1 \end{cases}. \tag{11}$$

The values $c_1, c_2, ..., c_P$ are intentionally designed to be negative powers of two, while the $c_{P+1}, c_{P+2}, ..., c_{2P-1}$ values are selected as the mirror images of $c_1, c_2, ..., c_P$ around 0.5. Thus, only *P* values of $c_1, c_2, ..., c_P$ actually need to be determined. The selection of $c_{P+1}, c_{P+2}, ..., c_{2P-1}$ is not intended to be optimized but rather to reduce storage requirements. The upper and lower limits of each interval are denoted as c_p^u and c_p^l, respectively, and are given by

$$c_p^u = \begin{cases} (c_p + 1)/2, & p = 2P-1 \\ (c_p + c_{p+1})/2, & 1 \le p < 2P-1 \end{cases}, \tag{12}$$

$$c_p^l = \begin{cases} c_{p-1}^u, & 1 < p \le 2P-1 \\ 0, & p = 1 \end{cases}. \tag{13}$$

The lowest and highest limits are 0 and c_{2P-1}^u, respectively. Each intrinsic information $\left|L_{j,l'}^{(i-1)}\right|$ whose hyperbolic tangent falls in the interval

$$c_p^l < \tanh\left(\left|L_{j,l'}^{(i-1)}\right|/2\right) \le c_p^u, \tag{14}$$

is approximated by $(1/2)^{d_p}$, where d_p is described by

$$d_p = \begin{cases} -\log_2(c_p), & c_p^l < \tanh\left(\left|L_{j,l'}^{(i-1)}\right|/2\right) \le c_p^u \\ 0, & otherwise \end{cases}. \tag{15}$$

Let $f_1, f_2, ..., f_{2P-1}$ be the number of the LLR values that lie in each interval, then (9) is approximated by

$$L_{e,j,l}^{(i)}(ANBP) \approx \prod_{l' \in B(j) \setminus l} sign(L_{j,l'}^{(i-1)}) \cdot \min_{l' \in B(j) \setminus l} \left\{ \left| L_{j,l'}^{(i-1)} \right| \right\} \cdot (c_1)^{f_1} \cdot ... \cdot (c_{2P-1})^{f_{2P-1}}$$

$$= \prod_{l' \in B(j) \setminus l} sign\left(L_{j,l'}^{(i-1)}\right) \cdot \min_{l' \in B(j) \setminus l} \left\{ \left| L_{j,l'}^{(i-1)} \right| \right\} \cdot \left(\frac{1}{2}\right)^{d_1 f_1 + ... + d_{2P-1} f_{2P-1}}.$$

Therefore, the check node update equation can be simplified to

$$L_{e,j,l}^{(i)}(ANBP) \approx \prod_{l' \in B(j) \setminus l} sign(L_{j,l'}^{(i-1)}) \cdot \min_{l' \in B(j) \setminus l} \left\{ \left| L_{j,l'}^{(i-1)} \right| \right\} \cdot \left(\frac{1}{2}\right)^{D}, \tag{16}$$

where

$$D = \left\lceil \frac{(\underbrace{d_1 + \ldots + d_1}_{f_1} + \ldots . + \underbrace{d_{2P-1} + \ldots + d_{2P-1}}_{f_{2P-1}})}{\text{total } \rho - 2 \text{ terms}} \right\rceil . \tag{17}$$

Eq. (16) is obtained by comparing each variable node pair-wise and taking the maximum LLR among the two, in order to compare it with the 2*P-1* thresholds and generate offset *D* by adding the d_p's. Doing so precludes using multiplications between each variable node's LLR in (2). Moreover, since the scale factor is a negative power of 2, it can be easily implemented by right shifting all bits at once (see Fig. 1).

B. Procedure outline

Check node update step:

Initialization: Set $D = 0$.

Step 1: Conduct a minimum search similar to the min-sum decoder. Perform pair-wise comparisons on the variable nodes' LLR. The minimum value between the two is kept and the larger value $\left| L_{j,l'}^{(i-1)} \right|$ is then compared with the designed threshold

$$T_p = \begin{cases} 2\text{arctanh}\left| c_p^u \right|, & 1 \le p \le 2P-1 \\ 0 \quad , & p = 0 \end{cases} . \tag{18}$$

Once found, the interval p is such that $T_{p-1} \le \left| L_{j,l'}^{(i-1)} \right| \le T_p$. The corresponding offset d_p is given by (15). Otherwise, if $T_{2P-1} < \left| L_{j,l'}^{(i-1)} \right|$, then $d_p = 0$.

Step 2: $D = D + d_p$.

Step 3: Repeat steps 1 and 2 for all variable nodes $l' \in B(j) \setminus l$.

Step 4: Right-shift the minimum value $\left| L_{j,l'}^{(i-1)} \right|$ from step 1 by *D* to obtain the check node LLR as (16).

Variable node update step: For variable node l with the set of check nodes $A(l)$ connected, The LLR can be computed by

$$L_{j,l}^{(i)} = L_{j',l}^{(0)} + \sum_{j' \in A(l) \setminus j} L_{e,j',l}^{(i)} . \tag{19}$$

Assuming BPSK modulation and transmission over an AWGN channel, the received signal is given by $y = x + w$, with *x* the signal component and *w* the zero-mean Gaussian random noise with variance $N_0 / 2$, and $L_{e,j',l}^{(0)} = \frac{4y_l}{N_0}$. The soft decision value is given by $r_l = L_{j',l}^{(0)} + \sum_{j' \in A(l)} L_{e,j',l}^{(i)}$. The decoder repeats the steps until either the syndrome is satisfied or the maximum number of iterations is reached.

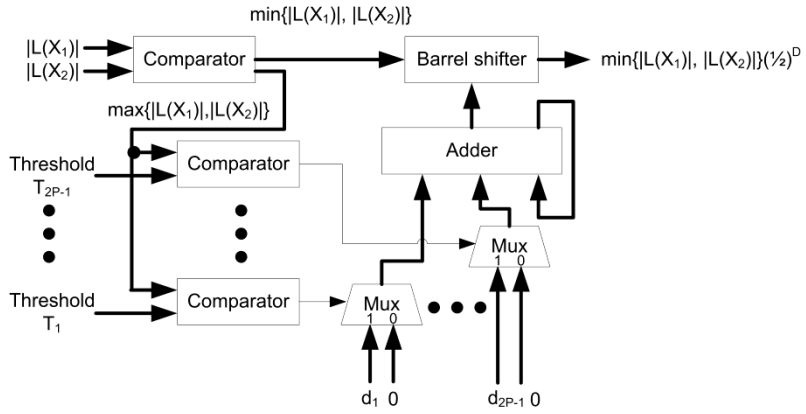

Fig. 1. Proposed decoder block diagram.

C. Comments on the decoder and d_p

Our decoder is similar to the one in [7] (see Fig. 1). The additional components in the proposed algorithm are the set of *2P-1* thresholds and the *2P-1* exponent offset values. Table I shows the values for T_p and d_p when *P* = 1, 2, 3 and 4. Notice that only 7 values for T_p and 7 values for d_p is needed to be known for *P* = 4, which is fairly small and very manageable when stored as constants within the decoder. It will be shown in the following section that the decoder that uses $P = 4$ performs very well.

TABLE I. VALUES FOR T_p AND d_p FOR P =1, 2, 3 AND 4

p	P=1			p	P=2		
	C	T_p	d_p		C	T_p	d_p
1	0.50	1.9459	1	3	0.75	2.7081	0.4150
0	-	0	-	2	0.50	1.4663	1
				1	0.25	0.7885	2
				0	-	0	-

p	P=3			p	P=4		
	C	T_p	d_p		C	T_p	d_p
5	0.875	3.4340	0.1926	7	0.9375	4.1431	0.0931
4	0.75	2.2687	0.4150	6	0.875	3.0123	0.1926
3	0.50	1.4663	1	5	0.75	2.2687	0.4150
2	0.25	0.7885	2	4	0.50	1.4663	1
1	0.125	0.3795	3	3	0.25	0.7885	2
0	-	0	-	2	0.125	0.3795	3
				1	0.0625	0.1881	4
				0	-	0	-

D. ANBP v.s. AN-MS

The adaptive normalized min-sum (AN-MS) algorithm which was recently published in [21] differs from ours in that it requires two normalization factors η and μ, $0 \le \eta, \mu \le 1$, in the check node update equation, i.e.

$$L_{e,j,l}^{(i)} \approx \eta \cdot \mu \prod_{l' \in B(j) \setminus l} sign(L_{j,l'}^{(i-1)}) \cdot \min_{l' \in B(j) \setminus l} \left\{ \left| L_{j,l'}^{(i-1)} \right| \right\}. \tag{20}$$

μ is the regular scale factor applied every time the check node is updated and η is multiplied when the syndrome of the parity check is not equal to zero. This approach has the drawback that η and μ cannot be selected analytically, but through simulation. Moreover, the selection method through simulation for a given code rate is unclear, whereas our proposed ANBP design is based on analytic principles.

IV. NUMERICAL RESULTS

A. Bit error rate performance

The proposed ANBP algorithm is tested via Monte-Carlo simulation using BPSK modulation over an AWGN channel. Two (n, k) Euclidean geometry LDPC codes [14] with different codeword length n and message sequence length k are used to evaluate the BER for different P values. These LDPC codes are: (1023,781) type-I EG-LDPC and (4095,3367) type-I EG-LDPC codes. They are regular structured LDPC codes with code rates 0.76 and 0.82. They have 32 and 64 row and column weights, respectively. We also evaluate performance over a low check weight

1/2 rate (3,6) LDPC code with n = 4000, k = 2000. Our algorithm is compared with BP, min-sum, Normalized (DE), Offset (DE), offset BP using criteria from [4] (Offset), i.e.

$$\beta \approx \begin{cases} 0.5 \ , |L_1 + L_2| \le 1, |L_1 - L_2| > 1 \\ -0.5, |L_1 + L_2| > 1, |L_1 - L_2| \le 1. \\ 0 \ , \qquad else \end{cases} \tag{21}$$

For normalized (DE) and Offset (DE), we adopt the pre-designed parameters from [5] with scale factor α =1.25 and offset value β = -0.15. Note that all decoders are set to have a maximum number of iterations equal to 50. The simulations of the AN-MS algorithm are not included due to the unclear selection procedure of the scale factors.

Let us consider the performance of the ANBP decoder with different P values. Fig. 2, where a (1023, 781) type-I EG-LDPC code is used, shows that with P = 4, the decoder outperforms the BP decoder. It is possible that other types of decoders could perform better than the BP algorithm. The BP algorithm performs as well as the optimal decoder, i.e. the maximum likelihood decoder, only when decoding cycle-free LDPC codes. That is, the graph representing the parity check of the code must be acyclic [23]. The BP decoder's performance is suboptimal when decoding codes with short length cycles. LDPC codes with short length cycle cause high correlation between successive iterations and reduce the decoding performance [14]. The authors in [7, Fig.2] and [8, Fig. 4(a)] have also observed that their proposed decoders slightly outperform the BP decode for some LDPC codes and high signal to noise ratio.

When the check weights of the LDPC code become high, as in the (4095, 3367) type-I EG-LDPC code, ANBP performs very close to the BP decoder (see Fig. 3). This is because the approximation in (6) becomes more accurate and closer to the check node update result from BP when the number of check weights increases.

ANBP decoder BER performance over LDPC codes with long length and small number of check weights is shown in Fig. 4 using a (3,6) LDPC code. Though the ANBP decoder loses some accuracy and causes a higher BER in the low SNR region, as the SNR increases, it performs better. It is observed that ANBP needs 0.1dB or less to achieve the same performance as the BP decoder at BER 10^{-6} or less. When comparing the performance of the ANBP decoder with other decoders, we can see that the normalized (DE) decoder performs the best. It even slightly outperforms the BP decoder with maximum iteration of 50. The ANBP decoder performs slightly worse than the normalized (DE), but outperforms every other decoder at high SNR.

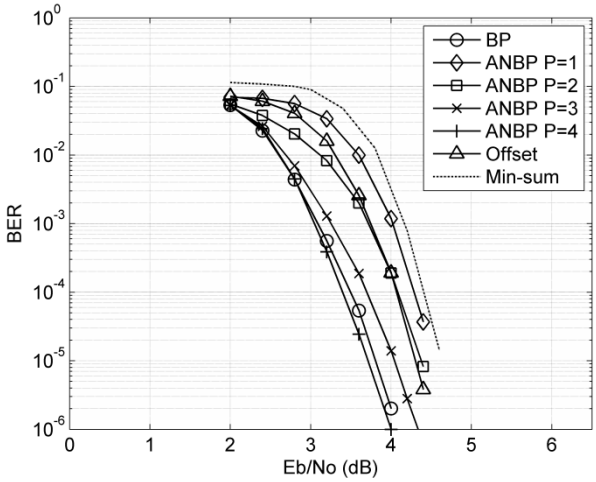

Fig. 2. BER performance of min-sum, offset, ANBP and BP with (1023,781) type-I EG-LDPC code ($\rho = 32$).

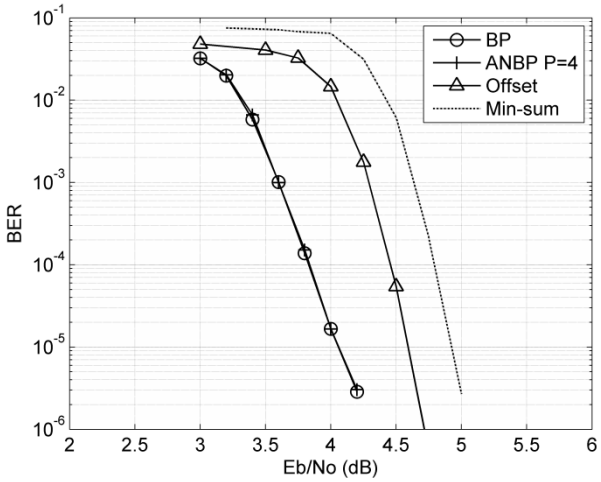

Fig. 3. BER performance of min-sum, offset, ANBP and BP with (4095, 3367) type-I EG-LDPC code ($\rho = 64$).

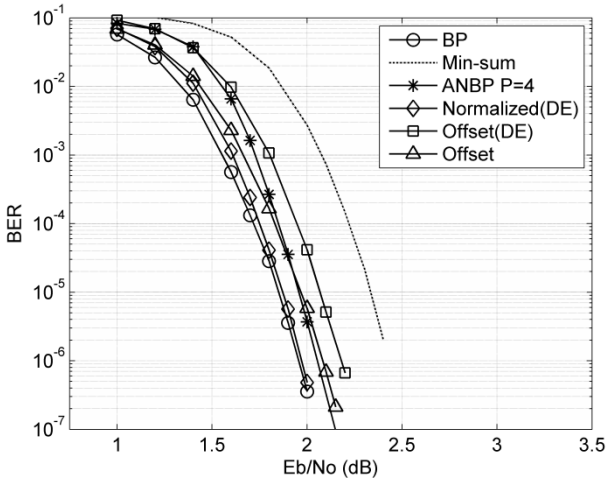

Fig. 4. BER performance of ANBP and various decoders with (3,6) LDPC code (n = 4000, k = 2000).

Notice that the offset BP-based decoder based on (21) performs very well for (3, 6) LDPC codes. Its BER performance is similar to that of ANBP at high SNR and even better at low SNR. Unfortunately, its performance degrades when applied to EG-LDPC codes with larger check weight as shown in Figs. 2 and 3.

We can see that the piece-wise approximation method in (21) becomes less accurate when the number of check weights is high. This is due to the large number of comparisons, which in turn cause more errors to accumulate and error propagation to occur because of the LLR comparisons at each variable node pair. Moreover, the criteria given in (21) may not fit well when the minimum value is less than 1. To explain this, let us substitute $\delta = |L_2| - |L_1|$ into (5). Then the offset becomes

$$\beta = \log\left(\frac{1+e^{-(2|L_1|+\delta)}}{1+e^{-\delta}}\right). \tag{22}$$

If we assume that $|L_1| < |L_2|$, then $\delta = |L_2| - |L_1| > 0$. The offset β in (22) depends on both the magnitude of the minimum value, $|L_1|$, and the difference between the LLRs magnitudes of both variable nodes, δ. We plot the relationship between β and δ for a range of values of $|L_1|$ in Fig. 5. Clearly, when $|L_1| \geq 1$ the offset values of β are pretty much the same curve. However, when $|L_1| < 1$, the variations of β may be significant. This can be verified by computing the ratio between the required offset magnitude $|\beta|$ and $|L_1|$ as a function of δ.

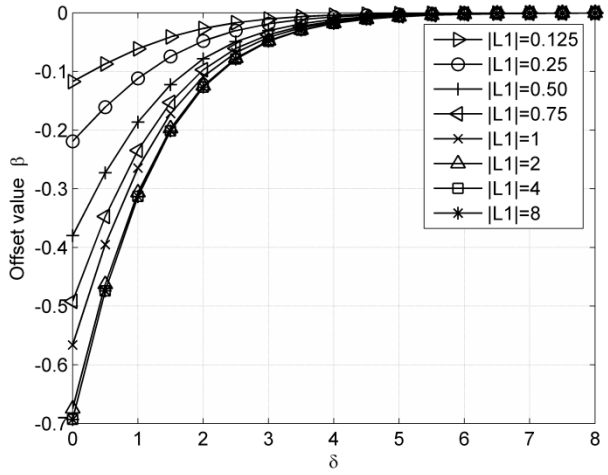

Fig. 5. Offset β plot for different values of δ and $|L_1|$.

Thus, to approximate β for LDPC codes with a large number of check weights, a more precise method is required. Equivalently, β can be expressed by $\beta = \log(1+e^{-|L_1+L_2|}) - \log(1+e^{-|L_1-L_2|})$. The $\log(1+e^{-|\cdot x|})$ function is then approximated using a small look up table with a few more entries with the requirement of more addition operations and accessing the LUT twice to obtain the value of β (see more details on BP algorithm using Jacobian logarithm in the following complexity analysis). More discussion of the approach can be found in [22].

Our ANBP decoder does not use the Jacobian logarithm, which uses the difference between the minimum of the LLR pair, but rather compares the LLRs of individual variable nodes against the thresholds T_p. This makes our decoder more attractive especially when the row weight is large.

When *P* is large enough to better approximate the LLR value, the algorithm has very similar number of iterations as that of the BP decoder. This can be observed in Figs. 6 and 7, especially for high check weights LDPC codes. However, when the decoder is applied to a (3, 6) LDPC code the number of iterations increases significantly at low SNR (see Fig. 8). This is expected since the decoder does not perform well at very low SNR. Nevertheless, due to its steeper waterfall region, the number of required iterations gap reduces dramatically to within 0.5 dB as the SNR increases from 1.4 dB to 1.9 dB.

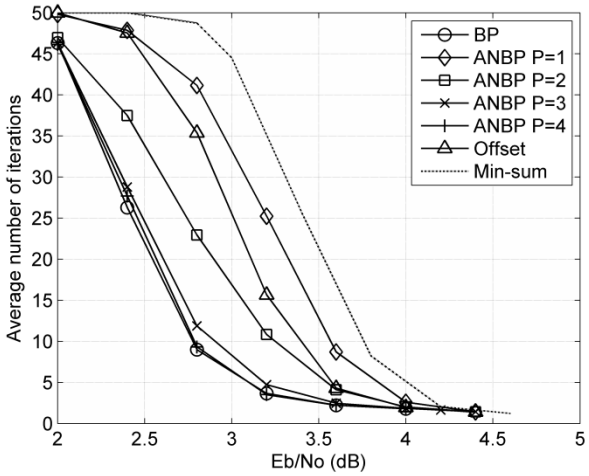

Fig. 6. Average number of iterations for various decoders to decode a (1023, 781) type-I EG-LDPC

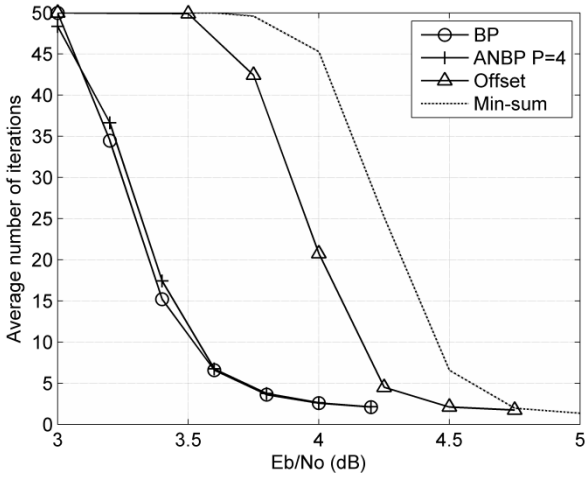

Fig. 7. Average number of iterations for various decoders to decode a (4495, 3367) type-I EG-LDPC

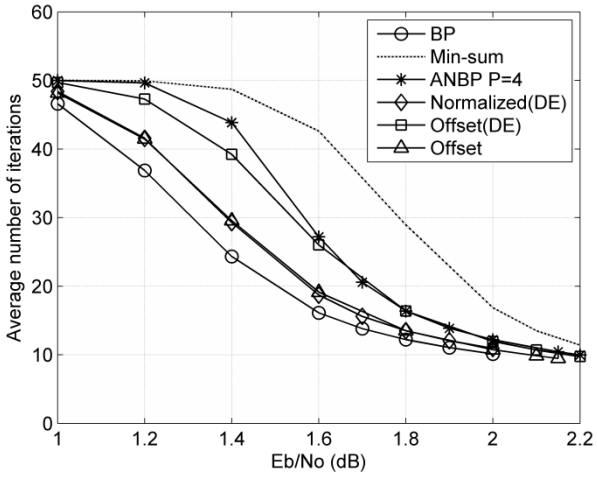

Fig. 8. Average number of iterations for various decoders to decode a (3, 6) LDPC code with $n = 4000$, $k = 2000$.

B. Mean square error of the approximation method used in ANBP

In the normalized BP-based algorithm, D may be considered as the exact value of the exponent of the normalized factor in each iteration in order to normalize $L_{e,j,l}^{(i)}$ in the min-sum algorithm to have the exact same result as in the BP-algorithm. Let us consider the BP algorithm in (1) and the normalized BP-based algorithm in (2). By equating both (1) and (2), we get

$$\frac{L_{e,j,l}^{(i)}(\min-\text{sum})}{L_{e,j,l}^{(i)}(\text{BP})} = \frac{\prod_{l'\in B(j)\setminus l} sign(L_{j,l'}^{(i-1)}) \cdot \min_{l'\in B(j)\setminus l}\left\{\left|L_{j,l'}^{(i-1)}\right|\right\}}{2\text{arctanh}\left(\prod_{l'\in B(j)\setminus l}\tanh(L_{j,l'}^{(i-1)}/2)\right)} = \alpha = 2^{D}. \tag{23}$$

In this section the approximation method used in (16) and (17) of ANBP algorithm is verified. The accuracy of using the ceil function in (17) to approximate D (denoted as D_{ceil}) is compared with the floor and round functions which are defined by

$$D_{floor} = \left\lfloor \underbrace{(\underbrace{d_1 + \ldots + d_1}_{f_1} + \ldots + \underbrace{d_{2P-1} + \ldots + d_{2P-1}}_{f_{2P-1}})}_{\text{total } \rho-2 \text{ terms}} \right\rfloor \tag{24}$$

and

$$D_{round} = \left\lfloor \underbrace{(\underbrace{d_1 + \ldots + d_1}_{f_1} + \ldots + \underbrace{d_{2P-1} + \ldots + d_{2P-1}}_{f_{2P-1}})}_{\text{total } \rho-2 \text{ terms}} + 0.5 \right\rfloor, \tag{25}$$

repectively. The extrinsic information in (16) is then computed using (17), (23) and (24) and compare their mean-square error (MSE) with the actual extrinsic information from the BP-algorithm's check node update equation. The results are shown in Table II. The actual value of D is computed and compared with $D_{ceil}, D_{floor}, D_{round}$. Since D depends on the intrinsic information $L_{j,l'}^{(i-1)}$, it depends on the received signal-to-noise ratio E_b / N_o, code rate, check weight d_c and the current decoding iteration. Hence, in our verification, we realize 1/2 code rate LDPC codes with d_c = 6, 10, 16 and 32, each 1000 codewords in BIAWGN channel. The MSE of the extrinsic information is obtained from the first iteration of the check node update equation only.

The value of D in the ANBP algorithm is typically adopted when there exist a difference in the extrinsic information compared with the BP-algorithm (i.e., $D > 0$) and require the normalization factor. According to Table II, D is highly preferred especially when E_b / N_o is low and the amount of check weights of the LDPC code become large. For instance, the expected value E[D] is 21.377 for d_c = 32 and E_b / N_o = 0 dB while E[D] is 0.407 for d_c = 6 and E_b / N_o = 4 dB. As mentioned earlier, the approximation of D with D_{ceil}, D_{floor}, and D_{round} becomes more accurate as the amount of check weight becomes higher. It is also confirmed that the difference between $D - D_{ceil}$ on average is mostly negative as we designed. That is, the approximation of D using ceiling function is most likely to be slightly larger than the actual D value. This is quite desirable since any approximation error that may occur in

approximating D will slightly decrease the likelihood confidence, i.e., the extrinsic information. However, $D - D_{floor}$ and $D - D_{round}$ are, on average, mostly positive. This can be observed by the results of $E[L^{(i)}_{e,j,l_{(BP)}} - L^{(i)}_{e,j,l_{(ANBP)}}]$ which are mostly positive for the ceiling function. While the approximation using the floor and round functions tend to increase the value of extrinsic information as $E[L^{(i)}_{e,j,l_{(BP)}} - L^{(i)}_{e,j,l_{(ANBP)}}]$ are mostly negative values. Finally, the MSE $E\left[\left(L^{(i)}_{e,j,l_{(BP)}} - L^{(i)}_{e,j,l_{(ANBP)}}\right)^2\right]$ of using ceil function tends to provide the least approximation error in the check sum update equation. When E_b / N_o becomes high, the difference between the approximated LLR and the actual results from BP-algorithm become less and then literally tend to increase again. This is quite expected due to the non-linearity of $\tanh()$ function and the larger rounding error corresponding to the interval range of $(c_p^l, c_p^u]$ when the value of $\left|L^{(i-1)}_{j,l'}\right|$ is high.

TABLE II. MSE OF COMPUTED EXTRINSIC INFORMATION BETWEEN BP-ALGORITHM AND ITS APPROXIMATED VALUE USING CEIL(), FLOOR() AND ROUND() TO COMPUTED D

			Check weight = 6			Check weight = 10			Check weight = 16			Check weight = 32		
		Eb/No(dB)	0	2	4	0	2	4	0	2	4	0	2	4
E[D]	D	P	1.940	1.012	0.407	4.407	2.396	1.057	8.947	4.883	2.132	21.377	12.263	5.507
$E[D_{ANBP}]$	ceil()	1	1.408	0.632	0.193	3.145	1.671	0.615	6.018	3.361	1.370	13.456	8.064	3.651
	floor()	1	1.408	0.632	0.193	3.145	1.671	0.615	6.018	3.361	1.370	13.456	8.064	3.651
	round()	1	1.408	0.632	0.193	3.145	1.671	0.615	6.018	3.361	1.370	13.456	8.064	3.651
	ceil()	2	2.117	1.081	0.394	4.507	2.489	1.086	8.581	4.868	2.150	19.015	11.349	5.288
	floor()	2	1.344	0.518	0.131	3.560	1.637	0.528	7.589	3.899	1.341	18.015	10.349	4.316
	round()	2	1.629	0.682	0.171	3.983	1.995	0.691	8.023	4.339	1.671	18.516	10.779	4.729
	ceil()	3	2.356	1.348	0.568	4.816	2.738	1.353	9.307	5.200	2.412	21.237	12.435	5.691
	floor()	3	1.431	0.561	0.142	3.824	1.775	0.575	8.307	4.202	1.467	20.237	11.435	4.691
	round()	3	1.826	0.818	0.209	4.326	2.247	0.840	8.786	4.707	1.935	20.697	11.908	5.218
	ceil()	4	2.403	1.467	0.747	4.902	2.810	1.485	9.515	5.314	2.488	21.950	12.744	5.859
	floor()	4	1.433	0.561	0.140	3.904	1.816	0.579	8.515	4.314	1.501	20.950	11.744	4.859
	round()	4	1.891	0.864	0.228	4.428	2.310	0.893	9.020	4.815	1.993	21.469	12.265	5.391
	ceil()	5	2.418	1.526	0.903	4.929	2.832	1.546	9.553	5.349	2.529	22.151	12.868	5.922
	floor()	5	1.432	0.559	0.139	3.929	1.833	0.581	8.553	4.349	1.531	21.151	11.868	4.922
	round()	5	1.888	0.865	0.224	4.447	2.339	0.902	9.064	4.856	2.036	21.665	12.380	5.470
$E[D-D_{ANBP}]$	ceil()	1	5.32E-01	3.80E-01	2.13E-01	1.26E+00	7.25E-01	4.42E-01	2.93E+00	1.52E+00	7.62E-01	7.92E+00	4.20E+00	1.86E+00
	ceil()	2	-1.77E-01	-6.89E-02	1.31E-02	-9.95E-02	-9.36E-02	-2.85E-02	3.65E-01	1.49E-02	-1.85E-02	2.36E+00	9.14E-01	2.19E-01
	ceil()	3	-4.16E-01	-3.36E-01	-1.62E-01	-4.08E-01	-3.42E-01	-2.96E-01	-3.60E-01	-3.18E-01	-2.81E-01	1.39E-01	-1.72E-01	-1.84E-01
	ceil()	4	-4.63E-01	-4.55E-01	-3.40E-01	-4.95E-01	-4.14E-01	-4.28E-01	-5.68E-01	-4.32E-01	-3.57E-01	-5.73E-01	-4.81E-01	-3.52E-01
	ceil()	5	-4.78E-01	-5.14E-01	-4.96E-01	-5.21E-01	-4.37E-01	-4.89E-01	-6.06E-01	-4.66E-01	-3.98E-01	-7.75E-01	-6.05E-01	-4.15E-01
	floor()	1	5.32E-01	3.80E-01	2.13E-01	1.26E+00	7.25E-01	4.42E-01	2.93E+00	1.52E+00	7.62E-01	7.92E+00	4.20E+00	1.86E+00
	floor()	2	5.96E-01	4.94E-01	2.76E-01	8.48E-01	7.59E-01	5.29E-01	1.36E+00	9.84E-01	7.91E-01	3.36E+00	1.91E+00	1.19E+00
	floor()	3	5.09E-01	4.51E-01	2.65E-01	5.84E-01	6.21E-01	4.82E-01	6.39E-01	6.80E-01	6.64E-01	1.14E+00	8.28E-01	8.16E-01
	floor()	4	5.07E-01	4.51E-01	2.67E-01	5.03E-01	5.80E-01	4.79E-01	4.32E-01	5.68E-01	6.30E-01	4.27E-01	5.19E-01	6.48E-01
	floor()	5	5.08E-01	4.53E-01	2.67E-01	4.78E-01	5.63E-01	4.76E-01	3.94E-01	5.34E-01	6.01E-01	2.25E-01	3.95E-01	5.85E-01
	round()	1	5.32E-01	3.80E-01	2.13E-01	1.26E+00	7.25E-01	4.42E-01	2.93E+00	1.52E+00	7.62E-01	7.92E+00	4.20E+00	1.86E+00
	round()	2	3.11E-01	3.30E-01	2.35E-01	4.24E-01	4.01E-01	3.66E-01	9.24E-01	5.43E-01	4.61E-01	2.86E+00	1.48E+00	7.78E-01
	round()	3	1.14E-01	1.93E-01	1.98E-01	8.09E-02	1.49E-01	2.17E-01	1.61E-01	1.76E-01	1.97E-01	6.80E-01	3.55E-01	2.89E-01
	round()	4	4.89E-02	1.48E-01	1.79E-01	-2.11E-02	8.62E-02	1.64E-01	-7.34E-02	6.79E-02	1.38E-01	-9.28E-02	-2.27E-03	1.16E-01
	round()	5	5.19E-02	1.47E-01	1.83E-01	-3.95E-02	5.65E-02	1.55E-01	-1.17E-01	2.66E-02	9.58E-02	-2.88E-01	-1.17E-01	3.74E-02
Difference in error of the LLR $E[(L_{BP}-L_{ext,ANBP})]$	ceil()	1	-1.16E-01	-2.20E-01	-2.59E-01	-4.42E-02	-1.41E-01	-2.64E-01	-6.98E-03	-4.90E-02	-2.15E-01	-8.10E-05	-4.38E-03	-7.12E-02
	ceil()	2	9.37E-03	-6.16E-02	-1.54E-01	4.85E-03	-1.49E-02	-1.06E-01	4.55E-04	-1.73E-03	-5.15E-02	-3.03E-06	-3.49E-04	-1.35E-02
	ceil()	3	5.59E-02	7.76E-02	-2.85E-02	1.11E-02	2.93E-02	3.77E-02	1.04E-03	5.23E-03	2.60E-02	1.16E-06	7.93E-05	-2.20E-03
	ceil()	4	6.45E-02	1.61E-01	1.62E-01	1.31E-02	4.28E-02	1.41E-01	1.09E-03	7.05E-03	5.39E-02	1.79E-06	1.73E-04	2.57E-03
	ceil()	5	6.55E-02	2.07E-01	3.29E-01	1.42E-02	4.72E-02	1.99E-01	1.06E-03	8.77E-03	6.44E-02	2.08E-06	2.08E-04	4.74E-03
	floor()	1	-1.16E-01	-2.20E-01	-2.59E-01	-4.42E-02	-1.41E-01	-2.64E-01	-6.98E-03	-4.90E-02	-2.15E-01	-8.10E-05	-4.38E-03	-7.12E-02
	floor()	2	-1.75E-01	-2.87E-01	-2.89E-01	-5.61E-02	-1.91E-01	-3.17E-01	-4.82E-03	-5.22E-02	-2.59E-01	-1.53E-05	-2.19E-03	-6.76E-02
	floor()	3	-1.70E-01	-2.82E-01	-2.87E-01	-4.80E-02	-1.81E-01	-3.13E-01	-3.66E-03	-4.48E-02	-2.44E-01	-6.94E-06	-1.33E-03	-5.71E-02
	floor()	4	-1.74E-01	-2.84E-01	-2.89E-01	-4.41E-02	-1.75E-01	-3.15E-01	-3.56E-03	-4.14E-02	-2.39E-01	-5.68E-06	-1.14E-03	-4.80E-02
	floor()	5	-1.76E-01	-2.85E-01	-2.89E-01	-4.20E-02	-1.73E-01	-3.15E-01	-3.63E-03	-3.80E-02	-2.32E-01	-5.09E-06	-1.07E-03	-4.36E-02
	round()	1	-1.16E-01	-2.20E-01	-2.59E-01	-4.42E-02	-1.41E-01	-2.64E-01	-6.98E-03	-4.90E-02	-2.15E-01	-8.10E-05	-4.38E-03	-7.12E-02
	round()	2	-9.46E-02	-2.22E-01	-2.68E-01	-2.36E-02	-1.10E-01	-2.56E-01	-2.15E-03	-2.50E-02	-1.75E-01	-7.76E-06	-1.24E-03	-3.99E-02
	round()	3	-5.54E-02	-1.76E-01	-2.57E-01	-1.33E-02	-6.31E-02	-2.05E-01	-1.06E-03	-1.33E-02	-1.13E-01	-2.19E-06	-5.41E-04	-2.24E-02
	round()	4	-5.00E-02	-1.64E-01	-2.49E-01	-1.03E-02	-5.81E-02	-1.98E-01	-9.40E-04	-1.05E-02	-1.06E-01	-9.38E-07	-4.17E-04	-1.56E-02
	round()	5	-4.72E-02	-1.62E-01	-2.50E-01	-1.01E-02	-5.25E-02	-1.90E-01	-8.20E-04	-8.64E-03	-8.84E-02	-1.16E-06	-3.19E-04	-1.18E-02
$E[(L_{BP}-L_{ext,ANBP})^2]$	ceil()	1	7.58E-02	1.41E-01	1.45E-01	2.18E-02	1.00E-01	1.84E-01	6.30E-04	2.38E-02	1.66E-01	1.78E-07	1.21E-03	4.80E-02
	ceil()	2	2.51E-02	8.77E-02	1.23E-01	2.74E-03	3.13E-02	1.18E-01	4.25E-05	4.27E-03	6.41E-02	5.02E-09	1.30E-05	1.00E-02
	ceil()	3	1.59E-02	7.66E-02	1.37E-01	1.42E-03	1.33E-02	9.36E-02	2.84E-05	8.63E-04	2.50E-02	7.60E-10	5.16E-06	1.81E-03
	ceil()	4	1.60E-02	9.54E-02	2.17E-01	1.53E-03	1.18E-02	1.04E-01	2.69E-05	8.26E-04	1.90E-02	1.28E-09	4.52E-06	9.69E-04
	ceil()	5	1.64E-02	1.16E-01	3.34E-01	1.60E-03	1.19E-02	1.24E-01	2.34E-05	8.99E-04	2.12E-02	1.44E-09	4.25E-06	1.09E-03
	floor()	1	7.58E-02	1.41E-01	1.45E-01	2.18E-02	1.00E-01	1.84E-01	6.30E-04	2.38E-02	1.66E-01	1.78E-07	1.21E-03	4.80E-02
	floor()	2	9.60E-02	1.71E-01	1.56E-01	2.48E-02	1.20E-01	2.07E-01	4.71E-04	2.15E-02	1.86E-01	4.07E-08	1.53E-04	3.99E-02
	floor()	3	9.35E-02	1.68E-01	1.54E-01	1.81E-02	1.14E-01	2.04E-01	2.35E-04	1.63E-02	1.71E-01	8.74E-09	7.02E-05	3.17E-02
	floor()	4	9.63E-02	1.69E-01	1.56E-01	1.50E-02	1.08E-01	2.06E-01	2.28E-04	1.54E-02	1.68E-01	5.93E-09	5.63E-05	2.28E-02
	floor()	5	9.77E-02	1.70E-01	1.56E-01	1.33E-02	1.06E-01	2.05E-01	2.60E-04	1.22E-02	1.63E-01	5.22E-09	5.27E-05	2.16E-02
	round()	1	7.58E-02	1.41E-01	1.45E-01	2.18E-02	1.00E-01	1.84E-01	6.30E-04	2.38E-02	1.66E-01	1.78E-07	1.21E-03	4.80E-02
	round()	2	5.11E-02	1.33E-01	1.43E-01	9.37E-03	6.77E-02	1.69E-01	1.60E-04	9.16E-03	1.22E-01	1.22E-08	4.98E-05	2.10E-02
	round()	3	3.54E-02	1.13E-01	1.39E-01	3.44E-03	3.77E-02	1.41E-01	7.71E-05	2.92E-03	7.73E-02	2.90E-09	2.05E-05	9.88E-03
	round()	4	3.65E-02	1.11E-01	1.37E-01	2.41E-03	3.54E-02	1.44E-01	6.78E-05	2.29E-03	7.60E-02	1.57E-09	2.15E-05	4.37E-03
	round()	5	3.49E-02	1.10E-01	1.37E-01	2.15E-03	3.32E-02	1.38E-01	6.15E-05	2.05E-03	6.46E-02	1.73E-09	1.56E-05	2.46E-03

V. COMPLEXITY ANALYSIS

The complexity analysis of various algorithms is presented in Table III. Our comparison uses an approach similar to the ones used in [7] and [9]. Our analysis is based on the number of different operations used to update $L_{e,j,l}^{(i)}$ of check j for all $l \in B(j)$ in the parity check matrix.

The check node j is connected to ρ variable nodes. The signs of the check nodes' LLRs are computed using $2\rho - 1$ XOR operations. This is the same for all algorithms except the algorithms based on (4). The BP algorithm in (1) requires 2 look up tables, LUT1 and LUT2, to obtain the values of the hyperbolic tangent and its inverse functions, respectively. By assuming each LUT to have the same amount of entries equal to M, it requires at least $\lceil \log_2(M) \rceil$ comparisons to search through each LUT. However, in practice, the time and number of operations spent on each attempt to access the LUT may be even more dependent on the hardware implementation. It also requires in total 2ρ multiplications for multiplying the factor 1/2, 2 and all results after accessing LUT1 together. For the BP algorithm using the Jacobian logarithm, it requires $3(\rho - 2)$ pair wise operations of (4) and (5) [7], where each pair-wise operation in (4) requires 1 XOR, 5 additions and 2 accessing the LUT of $\log(1 + e^{-|x|})$ to implement (5).

For BP-based algorithms, the value of $\min_{l' \in B(j) \setminus l} \left\{ \left| L_{j,l'}^{(i-1)} \right| \right\}$ is either the first or second smallest among ρ LLR values. It is shown in [16] that $\rho - 1$ and $\lceil \log_2(\rho) \rceil - 1$ comparisons are required for searching, respectively. The min-sum algorithm requires the least amount of operations among BP-based algorithm. Offset (DE) and normalized (DE) require 2 additions and 2 multiplications, respectively, for the first and second smallest LLR values. The Offset algorithm based on (4) and (21) requires 5 comparisons (i.e. addition operations) for each pair off LLR values to complete conditions checking. The decoder also requires a total of $3(\rho - 2)$ comparisons in (4) to complete the computation for check node j [7].

In the ANBP algorithm, since the number of values of T_p and d_p are small, they may be stored either inside the decoder as constant terms or as a LUT in memory, e.g. if $P = 4$, it requires only 14 values to be stored, 7 comparisons and 7 constant values. If the values are stored in a LUT, the adaptive normalized BP-based decoder

requires $\rho - 1$ access times, since $\min_{l' \in B(j) \setminus l} \left\{ \left| L_{j,l'}^{(i-1)} \right| \right\}$ has already been computed. On the other hand, if the values are stored inside the decoder as constants, a binary search which requires $\lceil \log_2(2P-1) \rceil$ comparisons for each LLR value can be applied [16].

We may further reduce the number of operations to find the first and second smallest LLR. If the first and second smallest LLR values are computed after finding d_p for each $\left| L_{j,l'}^{(i-1)} \right|$, the number of additions may be reduced by eliminating some variable node candidates to a subset of $\tilde{B}(j) \subset B(j)$ where $\tilde{B}(j)$ is the set of variable nodes in which the LLR falls in the lowest interval c_p described in (13). The minimum required comparisons (additions) can then be found when the first and second minimum values each falls in a different interval and is also the only value that lies in the interval. In this case, no comparisons are required. On the other hand, the maximum number of comparisons occurs when all LLR values lie in the same interval. This requires $\rho + \lceil \log_2(\rho) \rceil - 2$ additions. The D value can be obtained using $2\rho - 4$ additions and ρ right shifts.

Table IV presents the estimated number of operations required for updating each check node's LLR for a (3, 6) LDPC code. Clearly, the BP algorithm in (1) requires a large number of multiplications and accessing LUT. Since these operations take much longer time than addition and right shifting. Also, multiplications consist of several additions and shifting operations, we expect the adaptive normalized BP-Based algorithm to perform faster than the BP algorithm. When the BP algorithm uses the Jacobian logarithm, the number of additions and accessing the LUT increases significantly, since the algorithm requires $3(\rho - 2)$ pair-wise operations of (4).

The min-sum algorithm has the least complexity followed by the offset (DE) and the normalized (DE) decoder. Our ANBP decoder which uses LUT is quite attractive. Even though the number of additions is about twice as many as that used by the offset (DE) decoder, it trades off with the improvement in BER. The LUT here is small and can be accessed quickly. Our ANBP without LUT requires more additions. However, it requires about half the number as that of the off-set decoder due to the comparison used for its conditions in (21).

Finally, since we use a *D*-bit shifting operation, a fast implementation can be realized using a barrel shifter [15], since it can shift *D* bits in a single clock cycle.

VI. CONCLUSION

We presented a new adaptive normalized BP-based algorithm which reduces the complexity of the BP decoder. It allows the decoder scale factor to change adaptively. We showed that our proposed decoder can be used with LDPC codes with many code rates with different check weights. For low check weights codes, though a slight loss of the proposed decoder's accuracy is observed, the decoder performance is comparable to that achieved by other decoders at higher SNR values. For example, with as low as 0.1dB extra, it can achieve the same performance as the BP decoder at a BER of 10^{-6}. The ANBP performs very well at large check weights and outperforms other modified min-sum algorithms, achieving a BER very close to that of the BP decoder with similar amount of iterations. The algorithm eliminates the use of multiplications and reduces the number of memory accesses, number of entries, and access time when using an LUT. We have also shown that the complexity of the decoder is low and comparable to other decoders of the same class. The benefits of the adaptive normalized BP-based decoder are that the decoder is not code rate/code weight specific and no pre-design parameter is required. Thus, our decoder can be used as a multi–code rate or reconfigurable decoder. This is highly desirable for future mobile devices in order to handle different applications and standards [20].

TABLE III. ESTIMATED NUMBER OF OPERATIONS USED FOR A CHECK UPDATE j OF ALL $l \in B(j)$

Operation	XOR	Addition	Multipli-cation	LUT	Right shift
BP (tanh() (1)) [7],[9] $L_{e,j,l}^{(i)} = 2\text{arctanh}\left(\prod_{l' \in B(j)\setminus l} \tanh(L_{j,l'}^{(i-1)}/2)\right)$	$2\rho - 1$	-	2ρ	2ρ (LUTs of tanh and arctanh)	-
BP (Jacobian logarithm (4),(5)) [7] $L(l_1' \oplus l_2') = sign(L_1)sign(L_2)\cdot\min\{\lvert L_1\rvert,\lvert L_2\rvert\} + \log\left(1+e^{-\lvert L_1+L_2\rvert}\right) - \log\left(1+e^{-\lvert L_1-L_2\rvert}\right)$	$3(\rho-2)$	$3(\rho-2)\times 5$	-	$2\times 3(\rho-2)$ (LUT of $\log(1+e^{-\lvert x\rvert})$)	-
Min-sum (1) $L_{e,j,l}^{(i)} \approx \prod_{l' \in B(j)\setminus l} sign(L_{j,l'}^{(i-1)}) \cdot \min_{l' \in B(j)\setminus l}\left\{\left\lvert L_{j,l'}^{(i-1)}\right\rvert\right\}$	$2\rho - 1$	$\rho + \lceil \log_2(\rho) \rceil - 2$	-	-	-
Normalized (DE) (2), [7] $L_{e,j,l}^{(i)} \approx \prod_{l' \in B(j)\setminus l} sign(L_{j,l'}^{(i-1)}) \cdot \min_{l' \in B(j)\setminus l}\left\{\left\lvert L_{j,l'}^{(i-1)}\right\rvert\right\} \Big/ \alpha$	$2\rho - 1$	$\rho + \lceil \log_2(\rho) \rceil - 2$	2	-	-
Offset (DE) (3), [7] $L_{e,j,l}^{(i)} \approx \prod_{l' \in B(j)\setminus l} sign(L_{j,l'}^{(i-1)}) \cdot \max\left(\min_{l' \in B(j)\setminus l}\left\{\left\lvert L_{j,l'}^{(i-1)}\right\rvert\right\} + \beta, 0\right)$	$2\rho - 1$	$\rho + \lceil \log_2(\rho) \rceil - 2 + 2$	-	-	-

Offset (4), (21), [4] $L\left(l_1' \oplus l_2'\right) = sign(L_1)sign(L_2)\cdot\min\left\{\left\|L_1\right\|,\left\|L_2\right\|\right\}+\beta,$ $\beta \approx \begin{cases} 0.5, & \left\|L_1+L_2\right\|\le 1, \left\|L_1-L_2\right\|>1 \\ -0.5, & \left\|L_1+L_2\right\|>1, \left\|L_1-L_2\right\|\le 1 \\ 0, & else \end{cases}$	$3\rho-2$	$3(\rho-2)\times 5$	-	-	-
ANBP using LUT (16) $L_{e,j,l}^{(i)}(ANBP) \approx \prod_{l'\in B(j)\setminus l} sign(L_{j,l'}^{(i-1)})\cdot \min_{l'\in B(j)\setminus l}\left\{\left\|L_{j,l'}^{(i-1)}\right\|\right\}\cdot\left(\frac{1}{2}\right)^D$	$2\rho-1$	$\rho+\lceil\log_2(\rho)\rceil-2$ $+(2\rho-4)$	-	$\rho-1$	ρ
ANBP without using LUT (max) (16)	$2\rho-1$	$\rho+\lceil\log_2(\rho)\rceil-2$ $+\rho\times\lceil\log_2(2P-1)\rceil$ $+(2\rho-4)$	-	-	ρ
ANBP without using LUT (min) (16)	$2\rho-1$	$\rho\times\lceil\log_2(2P-1)\rceil$ $+(2\rho-4)$	-	-	ρ

TABLE IV. ESTIMATED NUMBER OF OPERATIONS FOR A CHECK UPDATE j OF ALL $l \in B(j)$ FOR $(3,6)$ LDPC CODE

Operation	XOR	Addition	Multiplication	LUT	Right shift
BP (tanh()) M=8	11	-	12	12	-
BP (Jacobian) M=8	12	60	-	24	
Min-sum	11	7	-	-	-
Normalized (DE)	11	7	2	-	-
Offset (DE)	11	9	-	-	-
Offset	12	60	-	-	-
ANBP (P=4) using LUT	11	15	-	5	6
ANBP (P=4) without using LUT (max)	11	33	-	-	6
ANBP (P=4) without using LUT (min)	11	26	-	-	6

ACKNOWLEDGMENT

We would like to thank Prof. B. Lee, Y. R. Park, J. He, and M. Sinky for their valuable comments and discussions regarding the proposed decoding algorithm.

REFERENCES

[1] Chung S.Y., Forney Jr. G.D., Richardson T.J., Urbanke R.: On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. IEEE Commun. Lett. 2001 Feb, 5, pp. 58-60.

[2] Hagenauer J., Offer E., Papke L.: Iterative decoding of binary block and convolutional codes. IEEE Trans. Inf. Theory. 1996 Mar, 42, pp. 429-445.

[3] Chen J., Fossorier M.: Near Optimum Universal Belief Propagation Based Decoding of Low-Density Parity Check Codes. IEEE Trans. Commun. 2002 Mar, 50, pp. 406–413.

[4] Anastasopoulos A.: A comparison between the sum-product and the min-sum iterative detection algorithms based on density evolution. Proc. IEEE Globecom. 2001 Nov, San Antonio, TX, pp. 1021–1025.

[5] Chen J., Fossorier M.: Density evolution for two improved BP-based decoding algorithms of LDPC codes, IEEE Commun. Lett. 2002 May, 6, pp.208–210.

[6] Fossorier M., Mihaljević M., Imai H.: Reduced complexity iterative decoding of low density parity check codes based on belief propagation. IEEE Trans. Commun. 1999 May, 47, pp.673-680.

[7] Chen J., Dholakia A., Eleftheriou E., Fossorier M.P.C., Hu X.Y.: Reduced-Complexity Decoding of LDPC Codes. IEEE Trans. Commun. 2005 Aug, 53, pp. 1288–1299.

[8] Zhao J., Zarkeshvari F., Banihashemi A.H.: On implementation of min-sum algorithm and its modifications for decoding low-density Parity-check (LDPC) codes. IEEE Trans. Commun. 2005 Apr, 53, pp. 549- 554.

[9] Jones C., Valles E., Smith M., Villasenor J.: Approximate-MIN constraint node updating for LDPC code decoding. IEEE Military Commun. Conf. Oct 2003, 1, pp. 157-162.

[10] Hennessy J.L., Patterson D.A.: Computer Architecture, System. New York: Wiley; 1999, pp.269-289.

[11] Winstead C.: Analog Iterative Error Control Decoders [dissertation]. Alberta, Canada: Univ. of Alberta; 2005.

[12] Loeliger H.A., Tarkoy F., Lustenberger F., Helfenstein M.: Decoding in analog VLSI. IEEE Commun. Mag. 1999 Apr, 37, pp. 99-101.

[13] Erfanian J., Pasupathy S., Gulak G., Reduced-complexity symbol detectors with parallel structures for IS1 channels. IEEE Trans. Commun. 1994 Feb-Apr, 42, pp. 1661-1671.

[14] Kou Y., Lin S., Fossorier M.: Low-density parity-check codes based on finite geometries: A Rediscovery and new results. IEEE Trans. Inf. Theory. 2001 Nov, 47, pp. 2711-2736.

[15] Ercegovac M., Lang T., Moreno J.H.: Introduction to Digital System. New York: Wiley; 1999, pp. 268, 286.

[16] Knuth D.E.: The Art of Computer Programming, In: Sorting and Searching. 2nd ed. Mass.: Addison-Wesley, 1998, pp. 207-217, 409.

[17] Ardakani M., Kschischang F.R.: Gear-shift decoding. IEEE Trans. Commun. 2006 Jul., 54, pp. 1235–1242.

[18] Darabiha A., Carusone A.C., Kschischang F.R.: A bit-serial approximate min-sum LDPC decoder and FPGA implementation. Proc. IEEE Int. Symp. Circuits and Syst., 2006 May 21-24, Island of Kos, Greece, pp. 149-152.

[19] Planjery S.K., Chilappagari S.K., Vasic B., Declercq D., Danjean L.: Iterative decoding beyond belief propagation. Inf. Theory and App. Workshop, 2010 Jan. 31-Feb. 5, San Diego, CA, pp.1-10.

[20] Mohsenin T., Baas B.: Trends and challenges in LDPC hardware decoders. Conf. Rec. 43rd. Asilomar Conf. on Signals, Syst. and Comput, 2009 Nov.1-4, Pacific Grove, CA, pp.1273-1277.

[21] Wu X., Song Y., Jiang M., Zhao C.: Adaptive-Normalized/Offset Min-Sum Algorithm. IEEE Commun. Lett. 2010 Jul., 14, pp. 667-669.

[22] Xiao-Yu H., Eleftheriou E., Arnold D.-M., Dholakia A.: Efficient implementations of the sum-product algorithm for decoding LDPC codes. IEEE GLOBECOM, 2001, 2, pp.1036-1036E.

[23] Schlegel B. C., Pérez C. L.: Trellis and Turbo Coding. Wiley-IEEE Press; 2004 , pp.231, 234.

[24] Sharifi Tehrani S., Mannor S. Gross W.J.: Fully Parallel Stochastic LDPC Decoders. IEEE Trans. Signal Processing , Nov. 2008, 56, pp.5692-5703.