Analog Iterative LDPC Decoder Based on Margin Propagation

Chhay Kong, *Student Member, IEEE*, and Shantanu Chakrabartty, *Member, IEEE*

***Abstract*—A margin propagation (MP) algorithm that can be used for implementing analog decoders for low-density parity check (LDPC) codes is presented. Unlike conventional sum–product analog decoders that rely on translinear operation of transistors, MP decoders use addition, subtraction and threshold operations, and therefore can be mapped onto different analog circuit topologies (current-mode, charge-mode, or nonelectronic circuits). This brief describes salient properties of the MP decoding algorithm and compares its performance to the sum–product and the min–sum (MS) decoding algorithms. Simulation results demonstrate that MP based LDPC decoders achieve nearly an identical bit error rate performance as their sum–product counterparts and achieve superior performance as compared to the MS decoding algorithm. Results presented in this brief also demonstrate that, when messages in LDPC decoding are corrupted by additive noise, the MP decoding algorithm delivers superior performance compared to its sum–product and MS counterparts.**

***Index Terms*—Analog decoding, error-control coding, low-density parity check (LDPC), margin propagation, reverse water filling.**

I. Introduction

ONE of the key factors underlying widespread acceptance of low-density parity check (LDPC) codes [1], [2] in modern communication standards has been their efficient and hardware realizable decoding algorithm. Conventional LDPC decoding algorithms are either based on the sum–product formulation [3], [4] or are based on suboptimal approximation techniques such as the min–sum (MS) formulation [5]. Both of these approaches are soft-decision algorithms that operate either in probability or log–likelihood ratio (LLR) domain [5] and have been successfully implemented using analog and digital hardware [6]–[11], [14]. In particular, analog decoders are attractive as compared to their digital counterparts because they exploit computational primitives inherent in device physics to achieve high energy efficiency [16]. For example, CMOS based analog LDPC decoders use translinear response of MOS transistor (biased in weak-inversion) to implement the sum–product algorithms [10], [16], [12], [13]. Strong dependence on device physics, however, constrains the applicability of analog sum–product decoding to a specific class of devices (bipolar or MOS transistors) and their operational regime (weak-inversion for MOS transistors). Approximation techniques like the MS algorithm have been proposed to extend the scope of analog decoding beyond translinear MOS circuits [14]. For instance, in [15] the MS algorithm has been mapped onto optical devices. However, when compared to an equivalent sum–product decoder, MS approximation incurs a performance penalty in terms of its bit-error rate (BER).

In this brief we present an alternative analog decoding algorithm that is based on margin propagation (MP) principles [18]. At the core of MP is a reverse water-filling procedure that can be implemented using basic conservation laws. This property enables the MP decoding algorithm to be scalable across different computing devices that operate on diverse set of physical parameters (charge, current, mass, etc.). We had previously reported a current mode network that implemented a MP analog decoder for convolutional codes [18]. The network used only Kirchhoff's current conservation law for implementing the MP decoding algorithm. The operation of the decoder was shown to be independent of the MOS transistor biasing (weak, moderate and strong inversion) and was shown to operate at higher decoding rates than an equivalent sum–product decoder. Additional features that make the MP decoding attractive include: 1) insensitivity to variations in ambient conditions, as the algorithm relies solely on universal conservation laws; 2) improved dynamic range and faster convergence, as the MP decoding operates using log–likelihood scores instead of probability measures.

This brief expands on our recently reported results on MP [19], with a particular emphasis on LDPC decoding. The organization of the brief is as follows. Section II briefly introduces the sum–product LDPC decoding algorithm and describes the computational core that will be implemented using MP. Section III introduces MP and presents some examples of its implementation using analog circuits. Section IV describes simulation results, comparing the performance of the MP decoding algorithm to alternative approaches. Section V provides conclusions with some final remarks.

II. LDPC Decoding and Log–Sum Formulation

LDPC codes are a class of binary linear block codes whose parity check matrix contains only a few 1's in comparison to the number of 0's [1], [20]. An example of a parity check matrix H, and its tanner or factor graph [16], [4] is shown in Fig. 1(a) and (b). The factor graph shown in Fig. 1(b) consists of variable nodes $v_k, k = 1, \ldots, N_c$ which are connected to check nodes $c_i, i = 1, \ldots, M$ using *edges*. For the description that follows, the number of edges associated with each node

Manuscript received April 22, 2006,; revised October 18, 2006; December 25, 2006, and May 13, 2007. This paper was recommended by Associate Editor F. M. Maggio.

C. Kong is with the Department of Electrical and Computer Engineering at University of Illinois, Urbana-Champaign, IL 61801 USA.

S. Chakrabartty is with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824 USA (e-mail: shantanu@msu.edu).

Digital Object Identifier 10.1109/TCSII.2007.906190

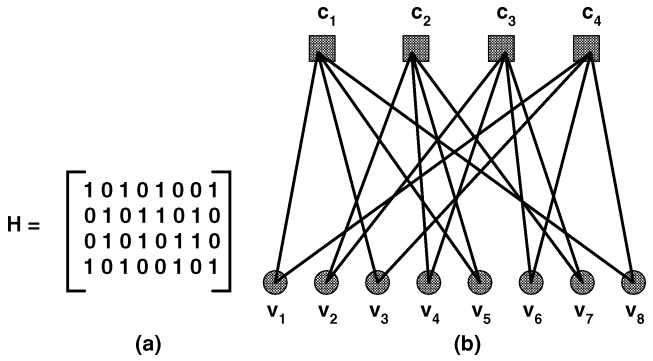

Fig. 1. (a) A $H(2,4)$ parity check matrix. (b) Its corresponding factor graph.

(also known as degree of the node) will be denoted by d_c and d_v, each corresponding to the check and variable nodes respectively. The parity check matrix for LDPC code will be represented by the symbol $H(d_v, d_c)$. Let $\mathcal{V}_i$ denote a set of check nodes connected to the variable node $\mathbf{v}_i$ and $\mathcal{V}_{i\sim j}$ denote the set of check nodes other than $\mathbf{c}_j$ that are connected to variable node $\mathbf{v}_i$. Similarly, let $\mathcal{C}_j$ denote a set of variable nodes connected to the check node $\mathbf{c}_j$ and $\mathcal{C}_{j\sim i}$ represent the set of variable nodes other than the node $\mathbf{v}_i$ connected to $\mathbf{c}_j$.

In a sum–product LDPC decoding algorithm [5], each check node $\mathbf{c}_j$ receives messages from its set of neighbors $\mathcal{C}_j$ (denoted by $L(\mathbf{v}_i \rightarrow \mathbf{c}_j)$) and computes messages to be sent to the variable nodes $\mathbf{v}_i \in \mathcal{C}_j$ (denoted by $L(\mathbf{c}_j \rightarrow \mathbf{v}_i)$) according to

$$L(\mathbf{c}_j \rightarrow \mathbf{v}_i) = 2\tanh^{-1}\left[\prod_{\mathbf{v}_k \in \mathcal{C}_{j\sim i}} \tanh\left(\frac{L(\mathbf{v}_k \rightarrow \mathbf{c}_j)}{2}\right)\right] \quad (1)$$

where $\tanh(x) = (\exp(x) - \exp(-x))/(\exp(x) + \exp(-x))$. In subsequent iterations, each variable node $\mathbf{v}_i$ receives messages from its neighboring check nodes $\mathbf{c}_j \in \mathcal{V}_i$ and re-computes messages to be sent to the check node $\mathbf{c}_j$ (denoted by $L(\mathbf{v}_i \rightarrow \mathbf{c}_j)$) according to

$$L(\mathbf{v}_i \rightarrow \mathbf{c}_j) = L(\mathbf{v}_i) + \sum_{\mathbf{c}_k \in \mathcal{V}_{i\sim j}} L(\mathbf{c}_k \rightarrow \mathbf{v}_i). \quad (2)$$

Messages are propagated back and forth between the variable and check nodes for a pre-determined number of iterations before a decision on the received bits is made [5].

Based on the approach described in [5], (1) can be computed recursively according to the following algorithm.

1) Given the messages $L(\mathbf{v}_k \rightarrow \mathbf{c}_j), \forall \mathbf{v}_k \in \mathcal{C}_{j\sim i}$, initialize $L(\mathbf{c}_j \rightarrow \mathbf{v}_i) = -\infty$.
2) Repeat the following recursion for each element $\mathbf{v}_k \in \mathcal{C}_{j\sim i}$:

$$L(\mathbf{c}_j \rightarrow \mathbf{v}_i) \leftarrow \log\left[\frac{1 + e^{L(\mathbf{v}_k \rightarrow \mathbf{c}_j) + L(\mathbf{c}_j \rightarrow \mathbf{v}_i)}}{e^{L(\mathbf{v}_k \rightarrow \mathbf{c}_j)} + e^{L(\mathbf{c}_j \rightarrow \mathbf{v}_i)}}\right]. \quad (3)$$

Equation (3) is fundamental to the implementation of LLR based LDPC decoders. The operation (3) can be compactly represented as $\mathbf{x} \leftarrow \log(1 + e^{x+y}) - \log(e^{\mathbf{x}} + e^{\mathbf{y}})$ where the arbitrary variables $\mathbf{x}$, $\mathbf{y}$ represent likelihood scores. The variables $\mathbf{x}$ and $\mathbf{y}$ can be decomposed into their differential forms as $\mathbf{x} = \mathbf{x}^+ - \mathbf{x}^-$ and $\mathbf{y} = \mathbf{y}^+ - \mathbf{y}^-$, with $\mathbf{x}^+, \mathbf{x}^-, \mathbf{y}^+, \mathbf{y}^- \geq 0$. In particular, because of their robustness to common-mode interference, differential representations are useful for implementation on analog circuits. Equation (3) can be written in a differential form as

$$\mathbf{x}^+ \leftarrow \log[e^{\mathbf{x}^+ + \mathbf{y}^+} + e^{\mathbf{x}^- + \mathbf{y}^-}] \quad (4)$$

$$\mathbf{x}^- \leftarrow \log[e^{\mathbf{x}^+ + \mathbf{y}^-} + e^{\mathbf{x}^- + \mathbf{y}^+}]. \quad (5)$$

Each of the updates (4) and (5) constitute a log–sum factor for which several approximations have been reported [5], [14]. In the next section, we will use MP to approximate (4) and (5).

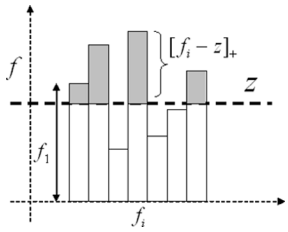

Fig. 2. Illustration of margin normalization procedure.

III. MP as an Approximation to Log–Sum Factor

In its general form, a log–sum factor is computed over a set of scores $f_i \in \mathcal{R}, i = 1, \ldots, N$ according to

$$z_{\log} = \log \sum_i^N e^{f_i}. \quad (6)$$

MP algorithm computes an approximation $z \approx z_{\log}$ according to a reverse water-filling constraint as

$$\sum_i^N [f_i - z]_+ = \gamma. \quad (7)$$

$[.]_+ = \max(., 0)$ in (7) denotes a threshold operation and $\gamma \geq 0$ represents a parameter of the algorithm. The solution to (7) is illustrated in Fig. 2, where the cumulative score beyond the normalization factor z (shown by the shaded area) equals γ. In the limit $\gamma \rightarrow 0$, the normalization factor $z \rightarrow \max_k f_k$. Thus, as $\gamma \rightarrow 0$, the normalization factor z tracks the largest of the scores. Also, when the number of scores increases, the value of γ decreases for the fixed value of z. An algorithm for calculating z using (7) requires nested routines and involves sorting and binary search, making its digital implementation complex and cumbersome. However, (7) can be implemented on several analog computing devices using basic conservation laws (charge, current, mass etc). Some of the examples of possible implementations are described in Section III-B.

Fig. 3 compares margin normalization factor computed according to (7), with a log–sum factor computed using (6). In this comparison, one of the scores (f_1) was varied, and the rest of the scores ($f_2, \ldots, f_N$) were fixed at monotonic increments.

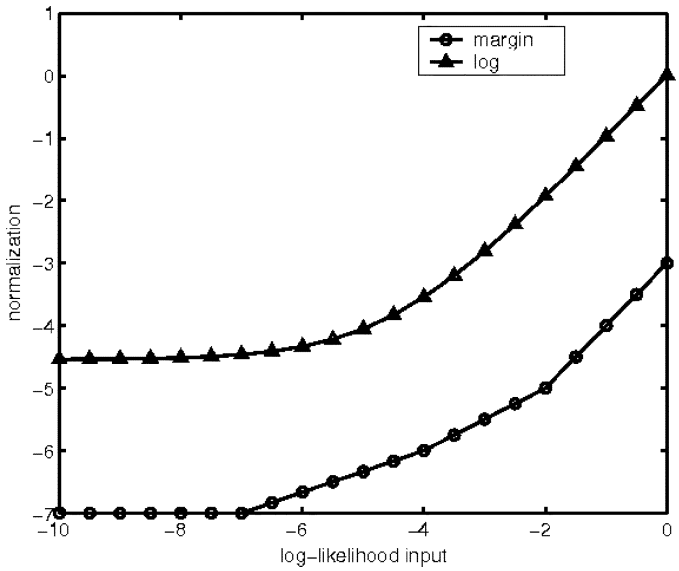

Fig. 3. Comparison of normalization scores computed according to margin normalization and by log–MAP normalization.

Fig. 3 shows that margin normalization factor is a piece-wise linear approximation of the log–sum factor, differing only by a constant (proportional to γ). The effect of the constant can be eliminated when differential representations are used in (4) and (5).

A. LDPC Decoders and MP

MP based approximation, when applied to recursions (4) and (5), leads to

$$L(\mathbf{c}_j \rightarrow \mathbf{v}_i) \leftarrow (z_1 - z_2) \tag{8}$$

where the normalization factors z_1 and z_2 are computed according to the following reverse water-filling constraints

$$[\mathbf{x}^+ + \mathbf{y}^+ - z_1]_+ + [\mathbf{x}^- + \mathbf{y}^- - z_1]_+ = \gamma \tag{9}$$

and

$$[\mathbf{x}^- + \mathbf{y}^+ - z_2]_+ + [\mathbf{x}^+ + \mathbf{y}^- - z_2]_+ = \gamma. \tag{10}$$

$\mathbf{x}^+$, $\mathbf{x}^-$, $\mathbf{y}^+$ and $\mathbf{y}^-$ are differential representations of likelihoods as $\mathbf{x}^+ = L^+(\mathbf{v}_i \rightarrow \mathbf{c}_j)$, $\mathbf{x}^- = L^-(\mathbf{v}_i \rightarrow \mathbf{c}_j)$, $\mathbf{y}^+ = L^+(\mathbf{c}_j \rightarrow \mathbf{v}_i)$ and $\mathbf{y}^- = L^-(\mathbf{c}_j \rightarrow \mathbf{v}_i)$. The choice of parameter γ depends on the size and degree of the nodes in the parity check matrix, as will be shown using simulation results in Section IV.

B. Examples of Analog Margin Normalization Circuits

Several computing devices can naturally implement the reverse water-filling (7) using only conservation laws. For instance, Fig. 4(a)–(c) illustrates an implementation using a charged coupled device (CCD). A CCD consists of potential wells whose depths can be adjusted by the application of an external voltage to CCD electrodes. As shown in Fig. 4(a), the initial charge in the potential wells is set to γ and the depths of the wells are set to be equal. The depth of the wells are then monotonically adjusted (adiabatically) in parallel, such that the device is maintained close to equilibrium. This is illustrated in Fig. 4(b) where the charge re-distributes amongst the potential wells. The procedure terminates when all the potential wells have reached a level proportional to scores $f_i, i = 1, \ldots, N$. The final configuration is shown in Fig. 4(c), where charge conservation naturally satisfies the reverse water-filling criterion given by (7). The normalization parameter z is then proportional to the substrate potential of the CCD and the residual charge in the ith potential well is proportional to the value $[f_i - z]_+$. Even though the above description uses charge as a physical parameter for analog computation, other parameters can also be used. For instance, the potential wells in Fig. 4 could be replaced by micro-fluidic chambers in a MEMS device, where the computation could be performed using a noncompressive fluid. Reverse water-filling functionality could also be emulated on CMOS devices using currents as has been reported in [18].

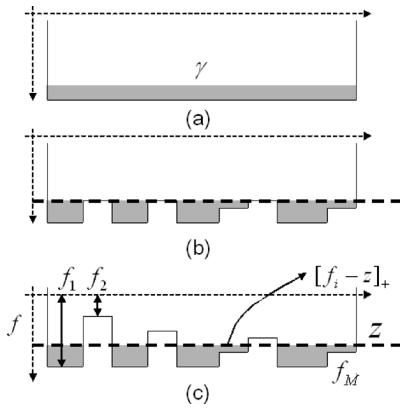

Fig. 4. Implementation of reverse water-filling using conservation principles.

A charge mode CMOS circuit that emulates the CCD procedure described above is shown in Fig. 5. The circuit consists of reset switches that regulate the flow of charge. During the reset phase ($R = 1$), the charge stored on the node z is proportional to the differential parameter $\gamma^+ - \gamma^-$. When the reset switch is off ($R = 0$), charge distributes along the node z, subject to the constraints imposed by the pMOS transistors. Thus, each capacitor stores charge proportional to the value $[f_i - z]_+$. At equilibrium the voltage at node z settles down to satisfy the reverse water-filling criterion. Once the differential normalization factor z has been calculated, additions in update (9) and (10) can be performed using switched-capacitor techniques.

IV. Results

Computer simulations were used to compare the performance of the margin based LDPC decoding algorithm to the sum–product and the MS based LDPC decoding algorithms. In our experiments, a rate 1/2 $H(4,8)$ LDPC code with a codeword length of $N_c = 256$ and $N_c = 1024$ was chosen. An additive white Gaussian noise (AWGN) channel with noise variance $N_0/2$ was used for simulations. An all reference zero codeword was transmitted using binary phase shift keying

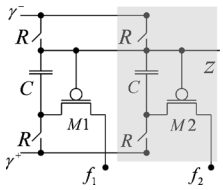

Fig. 5. Charge mode implementation of the reverse waterfilling algorithm.

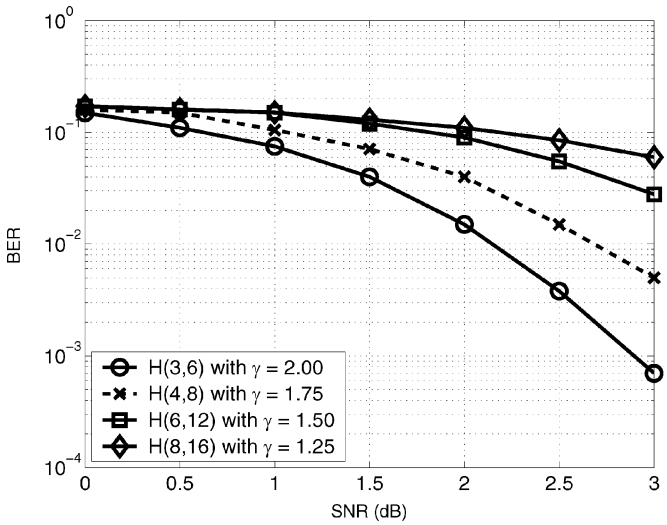

Fig. 6. BER curves obtained using optimal values of margin parameter γ corresponding to different LDPC parity check matrices.

(BPSK) modulation. The BER was evaluated for different levels of signal-to-noise ratio (SNR). This was computed as $\mathrm{SNR} = 10\log_{10}(E_b/N_0)$ with E_b representing energy per bit. 150 error events were used for calculating BER and 20 message passing iterations were used for LDPC decoding.

In the first set of experiments, BER curves were obtained for parity check matrices with different node degree. First, an optimal value of γ (value that produced the lowest BER) was determined for a given node degree and BER curves were obtained, as shown in Fig. 6. All simulations were performed using a rate 1/2 parity check matrix with code length fixed to $N_c = 256$. Fig. 6 shows a monotonic relationship between the node degree and the corresponding BER rate (for optimal γ). This behavior is consistent with the results reported in [20] that used the sum–product decoding algorithm. The Fig. 6 also shows that superior BER performance can be achieved for parity check matrix with lower node degree. Also, the optimal value of γ increases with the node degree. This behavior is attributed to an increase in density of 1's in an LDPC parity matrix (or number of interconnections in the LDPC factor graph), leading to a decrease in the relative margin (distance between scores) and, hence, decrease in optimal value of γ.

The next set of experiments compared the performance of the MP decoding algorithm to the sum–product and the MS algorithms. A rate 1/2 LDPC code with length $N_c = 256$ was chosen for this experiment. The results, summarized in Fig. 7, show that the BER performance of the margin based decoder is near identical to that of the sum–product decoder, but is superior to the MS decoder by 0.3 dB. The performance penalty

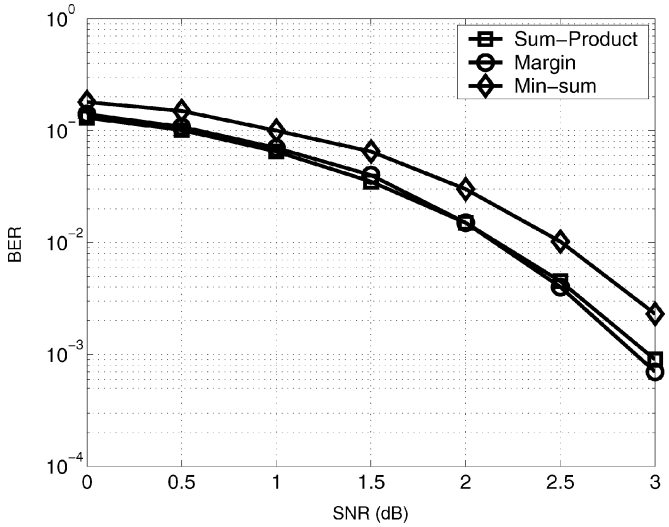

Fig. 7. Comparison of BER curves for iterative decoders implemented using sum–product, MS and MP algorithm.

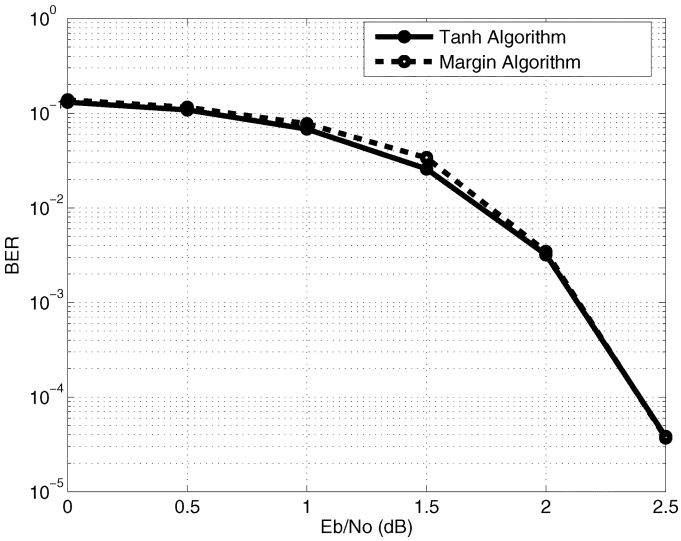

Fig. 8. Comparison of BER curves obtained for a 2048 length LDPC code decoded using sum–product ($\tanh$) and MP algorithm.

for the MS decoder has already been reported in [5] and is attributed to the approximation error of the log–sum factors. As shown in Fig. 8, the similarity in BER performance between the MP and the sum–product LDPC decoder has also been verified for LDPC codes with larger code length ($N_c = 2048$).

The advantage of an MP decoder over a sum–product decoder has been demonstrated using a condition where the communication channel used for propagating the messages between variable and check nodes is nonideal. Such a scenario arises in mixed-signal implementation where the messages over analog lines can be corrupted by substrate noise or capacitive coupling. To simulate such a condition, two sources of additive noise were considered: 1) primary noise that corrupts the BPSK signal at the receiver; 2) secondary noise that corrupts the messages propagated on communication channels between nodes of the code graph. For this experiment, a rate 1/2 LDPC code of length $N_c = 256$ was chosen. The SNR for the channel

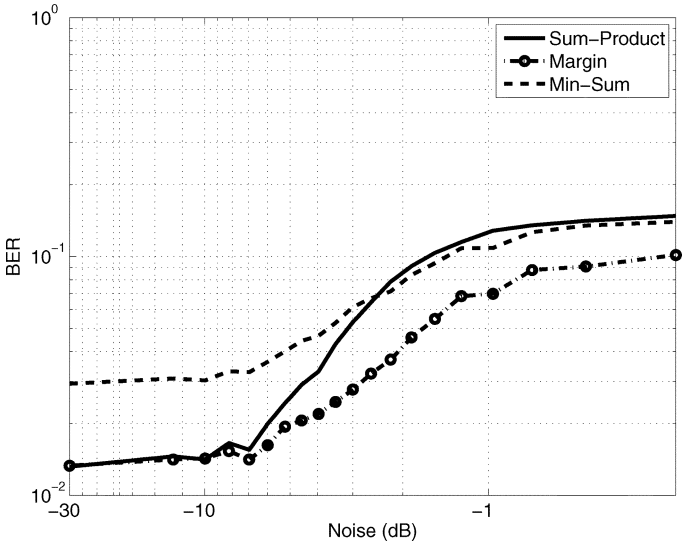

Fig. 9. Comparison of BER for a $N_c = 256$ LDPC code at 2 dB primary SNR obtained for different secondary noise levels.

corrupted by primary noise was fixed to 2 dB. The SNR for the channel corrupted by secondary noise was computed relative to the signal level in the primary channel. Fig. 9 shows a comparison of BER for LDPC decoders implemented using the sum–product, the MS and the MP algorithm where the SNR due to secondary noise was varied. The results indicate a clear advantage of the MP decoder over both sum–product and MS decoders for higher secondary noise levels (greater than -10 dB). The figure also shows that the MS decoder is more robust than the sum–product (tanh) decoder when the secondary noise level is increased beyond a certain threshold. This observation is consistent with previously reported results that have shown that the MS decoders are more robust to message quantization than sum–product based decoders [21].

V. Conclusion

In this brief, we described an algorithm for implementing analog LDPC decoders based on a MP principle. At the core of MP is a reverse water-filling procedure that can be implemented using universal conservation laws (charge, current, energy, mass, etc). It was shown that MP can be used for approximating the log–sum factors in a conventional sum–product based LDPC decoding algorithm. Simulation results have indicated that the BER performance of margin-based LDPC decoders is near-identical to the sum–product decoders and is superior to the MS LDPC decoders. It has also been shown that when messages in LDPC decoding algorithm are corrupted by additive noise, MP decoders demonstrate a superior performance compared to both the sum–product and the MS decoders.

References

[1] R. G. Gallager, "Low-density parity check codes," *IRE Trans. Inf. Theory*, vol. IT-8, no. 1, pp. 21–28, 1962.
[2] D. J. C. MacKay and M. C. Davey, "Evaluation of Gallager codes for short block length and high rate applications," in *Codes, Systems and Graphical Models*, B. Marcus and J. Rosenthal, Eds. : , 2000, vol. 123, IMA Volumes in Mathematics and Its Applications, pp. 113–130.
[3] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.
[4] F. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum–product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.
[5] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of block and convolutional codes," *IEEE Trans. Inf. Theory*, vol. IT-42, no. 3, pp. 429–445, Mar. 1996.
[6] M. Moerz, T. Gabara, R. Yan, and J. Hagenauer, "An analog 0.25-μm BICMOS tailbiting MAP decoder," in *Proc. Int. Solid-State Circuits Conf. (ISSCC'00)*, Feb. 2000, pp. 356–357.
[7] A. J. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 404–412, Mar. 2002.
[8] M. M. Mansour and N. R. Shanbhag, "A 640 Mbps 2048-bit programmable LDPC decoder chip," *IEEE J. Solid-State Circuits*, vol. 41, pp. 684–698, Mar. 2006.
[9] V. Gaudet and G. Gulak, "A 13.3-Mb/s 0.35-μm CMOS analog turbo decoder IC with a configurable interleaver," *IEEE J. Solid-State Circuits*, vol. 38, no. 11, pp. 2010–2015, Nov. 2003.
[10] C. Winstead, J. Dai, S. Yu, C. Myers, R. Harrison, and C. Schlegel, "CMOS analog MAP decoder for (8,4) hamming code," *IEEE J. Solid-State Circuits*, vol. 39, no. 1, pp. 122–131, Jan. 2004.
[11] D. Vogrig, A. Gerosa, A. Neviani, A. G. Amat, G. Montorsi, and S. Benedetto, "A 0.35-μm CMOS analog turbo decoder for the 40-bit rate 1/3 UMTS channel code," *IEEE J. Solid-State Circuits*, vol. 40, no. 3, pp. 753–762, Mar. 2005.
[12] J. P. Chamorro, C. Lahuec, F. Seguin, and M. Jezequel, "Design rules for subthreshold MOS circuits," presented at the Analog Decoding Workshop (ADW'06), Turin, Italy, Jun. 2006.
[13] C. Winstead, N. Nguyen, V. Gaudet, and C. Schlegel, "Low-voltage CMOS circuits for analog iterative decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 4, pp. 829–841, Apr. 2006.
[14] S. Hemati, A. H. Banihashemi, and C. Plett, "An 80-Mb/s 0.18-μm CMOS analog min–sum iterative decoder for a (32,8,10) LDPC code," *IEEE J. Solid-State Circuits*, vol. 41, no. 11, pp. 2531–2540, Nov. 2006.
[15] A. H. Banihashemi and S. Hemati, "Decoding in optics," in *Proc. Int. Symp. Inf. Theory*, 2002, p. 231.
[16] H.-A. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarkoy, "Probability propagation and decoding in analog VLSI," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 837–843, Feb. 2001.
[17] S. Chakrabartty and G. Cauwenberghs, "Margin propagation and forward decoding in analog VLSI," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS'04)*, Vancouver, Canada, May 23–26, 2004.
[18] S. Chakrabartty, "CMOS analog iterative decoders using margin propagation circuits," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS'06)* , Kos, Greece, 2006.
[19] C. Kong and S. Chakrabartty, "Analog iterative LDPC decoders based on margin propagation," in *Proc. Analog Decoding Workshop*, Turin, Italy, Jun. 5–6, 2006.
[20] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.
[21] L. Ping and W. K. Leung, "Decoding low-density parity check codes with finite quantization bits," *IEEE Commun. Lett.*, vol. 4, no. 2, pp. 62–64, Feb. 2000.