EFFICIENT HARDWARE IMPLEMENTATION OF MIN-SUM WITH CONDITIONAL CORRECTION FOR LDPC DECODING

Kiran Gunnam, Gwan Choi
Dept. of Electrical and Computer Engineering, Texas A&M University, College Station, TX-77840

ABSTRACT
The LDPC decoder complexity is largely influenced by interconnect and storage requirements. Recent research shows that Min-Sum decoding algorithm with conditional correction achieves performance that of ideal Sum of Products algorithm within 0.2 dB with 4 bits of uniform quantization is applied on check node and bit node messages. This paper presents a transformation for the Min-Sum with conditional correction to achieve efficient hardware implementation.

1. INTRODUCTION

Low-Density Parity Check (LDPC) codes and Turbo codes are among the best known near Shannon limit codes that can achieve record-breaking performance for low SNR applications [1,9]. When compared to the decoding algorithm of Turbo codes, LDPC decoding algorithm has more parallelization, low implementation complexity, low decoding latency, as well as no error-floors at high signal-to-noise ratios (SNRs). LDPC decoders require simpler computational processing. While initial LDPC decoder designs suffered from complex interconnect issues, structured LDPC codes simplify the interconnect complexity. LDPC codes are adopted/being adopted in next generation digital video broadcasting (DVB-S2), MIMO-WLAN 802.11n,, 802.12, 802.20, Gigabit Ethernet 802.3, magnetic channels(storage/recording systems),and long-haul optical communication systems .

The rest of the paper is organized as follows. Section 2 presents the Min-Sum algorithm and its variations. Section 3 presents Algorithm transformation for conditional Min-Sum suitable for hardware implementation. Section 4 presents performance comparison. Section 5 concludes with a summary.

2 MIN-SUM AND ITS VARIATIONS

LDPC codes can be decoded by the Gallager's iterative belief-propagation (BP) or sum of products algorithm (SP) [1]. Min Sum decoding algorithm is an approximation for Sum of Products algorithm to decode LDPC codes and does not have the complexity associated with non-linear functions used in BP[2].. The recently proposed correction methods for Min Sum[3-6] makes the Min-Sum approach a superior candidate for decoding as these methods give the performance very close to that of ideal sum of product algorithm. While reduced complexity SP algorithms[7-9] are available, they need higher precision of bits to avoid error floors while Min Sum with conditional correction can operate with just 4 bits[5].

We assume BPSK (Binary Phase Shift Keying) modulation (a 1 is mapped to -1 and a 0 is mapped to -1) over an AWGN (Additive White Gaussian channel). The received values y_n are Gaussian with mean $x_n = \pm 1$ and variance $\sigma^2 = N_0/2$.

The reliability messages used in BP based Min Sum algorithm can be computed in two phases viz. check node processing (1) and bit node processing (2) and this is repeated iteratively till the decoding criterion is satisfied.

The message passing equations for each iteration t are given by

$$R_{j,i}^{(t)} = \delta_{j,i}^{(t)} \kappa_{j,i}^{(t)} \tag{1}$$

$$\kappa_{j,i}^{(t)} = \min_{i' \in Row[j][i] \setminus i} \left| Q_{i',j}^{(t-1)} \right|. \tag{2}$$

$$Q_{i,j}^{(t)} = \left(\sum_{j'=Col[i][1]}^{Col[i][r]} R_{j',i}^{(t)} \right) - R_{j,i}^{t} + \wedge_i \tag{3}$$

equivalently

$$Q_{i,j}^{(t)} = \left(\sum_{j'=Col[i][1] \setminus j}^{Col[i][r]} R_{j',i}^{(t)} \right) + \wedge_i \tag{4}$$

where

$R_{j',i}^{(t)}$ is the message from check j to bit i computed in iteration t ,

$Q_{i',j}^{(t)}$ is the message from bit i to check j computed in iteration t ,

$\delta_{j,i}^{(t)}$ is ± 1 and is given by

$$\delta_{j,i}^{(t)} = \left(\operatorname{sgn}\left(Q_{i,j}^{(t)}\right) \cdot \prod_{i' \in Row[j]} \operatorname{sgn}\left(Q_{i,j}^{(t)}\right) \right). \tag{5}$$

$\wedge_i$ is the intrinsic reliability metric of bit i .

$Row[j][1,2,...,c]$ gives the locations of bits connected to the check node j

$Col[i][1,2,...,r]$ gives the locations of checks connected to the bit node i

Check node processing in (2) would overestimate the reliability metric $R_{j,i}^{(t)}$ and can be corrected by a scaling factor λ which depends on the code parameters and the iteration number [19]. This is referred as Normalized Min-Sum:

$$R_{j,i}^{(t)} = \lambda \delta_{j,i}^{(t)} \kappa_{j,i}^{(t)} \tag{6}$$

Overestimation in $R_{j,i}^{(t)}$ and can also be corrected by subtracting a positive constant β from the magnitude of the $R_{j,i}^{(t)}$ in the following way:[19] This is referred as Offset Min-Sum:

$$R_{j,i}^{(t)} = \delta_{mn}^{(i)} \max\left(\kappa_{j,i}^{(t)} - \beta, 0\right) \tag{7}$$

The authors in [5] ,independent of work presented in [4], proposed method same as in (7)-but they would apply the correction factor conditionally. This method and the above presented methods perform same when the number of quantization bits is 5 or more. However, with the conditional correction method, a four-bit uniform quantizer and four-bit messages provide 0.2 dB performance loss when compared to ideal Sum of products algorithm. The unconditional correction methods have 0.8 dB performance loss when compared to ideal Sum of products algorithm. When number of quantization bits are 5 or 6 , all the correction methods perform within 0.1dB from the ideal sum of products algorithm[5].

It should be noted that quantization parameters dictate the decoder complexity of computational unit as well as interconnects- we see 25% increase in hardware complexity if the number of bits to represent the messages is increased from 4 to 5.

However the conditional correction algorithm as presented in [5] is not suitable for simple hardware implementation and increases the complexity. So we present a simple algorithm transformation method to perform the conditional correction. The implementation of the same algorithm but in a different transformation form resulted in having additional comparator unit in Check node processing unit(CNU)

There are two conditional correction methods, "Correction A" and "Correction B" presented in [5]. Both conditional correction methods achieve the same result and performance while the latter one is simpler of the two. The conditional correction algorithm "Correction B" presented in [5] is as follows with a notation according to the one followed in this paper..

Let M_{ij} denote the set of magnitude of extrinsic incoming messages for the i edge of check node j .

$$M_{ij} = \left\{ \left| Q_{i',j}^{(t-1)} \right| : i' \in Row[j][i] \setminus i \right\} \tag{8}$$

Set a flag $\eta_{j,i}^{(t)}$ to 1or 0 based on existence or non-existence of a message q in

$$M_{ij} \setminus \left\{ \kappa_{j,i}^{(t)} \right\} \text{such that } q \in \left[\kappa_{j,i}^{(t)}, \kappa_{j,i}^{(t)} + d_c \right] \tag{9}$$

$$\begin{aligned} R_{j,i}^{(t)} &= \delta_{mn}^{(i)} \max\left(\kappa_{j,i}^{(t)} - y_c, 0\right) \text{ if } \eta_{j,i}^{(t)} = 1 \\ &= \delta_{mn}^{(i)} \kappa_{j,i}^{(t)}, \text{ if } \eta_{j,i}^{(t)} = 0 \end{aligned} \tag{10}$$

Here d_c and y_c are the parameters determined from simulations and depend on the quantization word length and code parameters [5]

3. Algorithm Transformation for Min-Sum with condition correction

We present an algorithm transformation such that step (9) is transformed into a concurrent operation rather than a post processing operation of the regular Min-Sum algorithm. *The key intuition behind this algorithm is to transform a post processing operation of (9) into concurrent operation of finding the 3rd minimum in the set along with the already required set of the least and 2nd least minimums.* If (9) is implemented in its original form, it requires a maximum of $c * 2c$ additional comparisons for condition evaluations. The proposed algorithm transformation needs only c maximum number of additional comparisons to find the 3rd least number in the set in case of a serial Check Node Processor implementation. In case of a binary tree implementation method, trace back with survivors method [6] can find 3 minimums with some additional complexity when compared to that of finding 2 minimums.

The following transformation of Min-Sum with conditional correction needs to find the least 3 minimum values of the set of all the bit node messages arriving at the check node j Since now we need to find 3 minimum values, we need at most $3c-1$ comparisons in serial search, or with at most $c+\log 2c-2$ comparisons with the help of a binary tree After finding 3 minimum values, (2) will be evaluated using (14) and (9) will be evaluated by finding 3 difference terms as in (15) and 3 more comparison operations as in (16).

Define

$$\kappa 1_j^{(t)} = \min_{i' \in Row[j][i]} \left| Q_{i,j}^{(t-1)} \right|. \tag{11a}$$

$$\kappa 1\_index_j^{(t)} = i, \text{if } \kappa 1_j^{(t)} = \left| Q_{i,j}^{(t-1)} \right| \tag{11b}$$

$$\kappa 2_j^{(t)} = 2nd \min_{i' \in Row[j][i]} \left| Q_{i,j}^{(t-1)} \right|. \tag{12a}$$

$$\kappa 2\_index_j^{(t)} = i, \text{if } \kappa 2_j^{(t)} = \left| Q_{i,j}^{(t-1)} \right| \tag{12b}$$

$$\kappa 3_j^{(t)} = 3rd \min_{i' \in Row[j][i]} \left| Q_{i,j}^{(t-1)} \right|. \tag{13}$$

$$\begin{aligned}\kappa_{j,i}^{(i)} &= \kappa 1_j^{(i)},\ \forall i' \in Row[j][i] \setminus \kappa 1\_index_j^{(t)} \\ &= \kappa 2_j^{(i)}, i = \kappa 1\_index_j^{(t)}\end{aligned} \tag{14}$$

Evaluate 3 difference terms as follows:

$$d_{12} = \kappa 1_j^{(t)} - \kappa 2_j^{(t)} \tag{15a}$$

$$d_{13} = \kappa 1_j^{(t)} - \kappa 3_j^{(t)} \tag{15b}$$

$$d_{23} = \kappa 2_j^{(t)} - \kappa 3_j^{(t)} \tag{15c}$$

Evaluate 3 more conditions as follows:

$$\begin{aligned}\vartheta_{12} &= 1, \text{ if } d_{12} \le d_c \\ &= 0, \text{ else}\end{aligned} \tag{16a}$$

$$\begin{aligned}\vartheta_{13} &= 1, \text{ if } d_{13} \le d_c \\ &= 0, \text{ else}\end{aligned} \tag{16b}$$

$$\begin{aligned}\vartheta_{23} &= 1, \text{ if } d_{23} \le d_c \\ &= 0, \text{ else}\end{aligned} \tag{16c}$$

Now evaluate (9) in this manner

$$\begin{aligned}\eta_{j,i}^{(t)} &= \vartheta_{23} \text{ if } i = \kappa 1\_index_j^{(t)} \\ &= \vartheta_{13} \text{ if } i = \kappa 2\_index_j^{(t)} \\ &= \vartheta_{12} \text{ ,else}\end{aligned} \tag{17}$$

4. Performance comparison

The complexity comparison is presented in the following table for evaluating c check node edge messages $R_{j,i}^{(t)}$ for a check node j and $i \in Row[j][i]$

	Min-Sum un conditional offset correction	Min-Sum – Proposed Hardware implementation for conditional offset correction
Maximum Number of real additions Serial Processing (Parallel Processing)	$2c-1$ ($c+\log 2c-2$) For finding the 2 of minimums out of the set	$3c-1$ ($c+\log 2c-2$) For finding the 3 minimums out of the set
Number of real additions for correction	4	4
Number of real additions for condition evaluations	0	6
Number of real additions for signed magnitude to 2's complement	1 to 2 depending on (5).	1 to 3 depending on (5)
Maximum Number of unique outputs	3	4
Performance Loss with 4 bit quantization against ideal BP[5]	0.8 dB	0.2 dB

Table 1: Complexity Comparison for Implementation of Min-Sum conditional and un-conditional correction methods.

5. SUMMARY

This paper presents a low complexity algorithm transformation for the LDPC decoding using Min-Sum with conditional correction. This would facilitate the low complexity hardware implementation in terms of Check node processor implementation. The implementation of this conditional Min-Sum algorithm would reduce total decoder complexity upto 25% as only 4 bits of uniformly quantized message computation and passing is needed to achieve with in 0.2 dB performance loss when compared to ideal Sum of products algorithm and 0.1 dB performance loss when compared to 5 bit version of Min-Sum with un conditional correction respectively.

REFERENCES

[1] R. G. Gallager, Low-Density Parity-Check Codes, M.I.T Press, 1963. Available at http://justice.mit.edu/people/gallager.html

[2] J. Chen and M. Fossorier, "Near-optimum universal belief-propagation-based decoding of low-density parity-check odes," *IEEE Trans Commun.*, vol. 50, pp. 406–414, Mar. 2002.

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

[4] - "Density evolution for BP-based decoding algorithms of LDPC codes and their quantized versions," in *Proc. IEEE Globecom*, Nov. 2002, pp. 1378–1382.

[5] J. Zhao, F. Zarkeshvari and A. H. Banihashemi, ``On implementation of min-sum algorithm and its modifications for decoding LDPC codes," IEEE Trans. Comm., vol. 53, no. 4, pp. 549-554, April 2005.

[6] J. Heo. Analysis of Scaling Soft Information on Low Density Parity Check Codes. Elect. Letters, 39(2):219–221, Jan 2003.

[7] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, "Efficient implementation of the sum-product algorithm for decoding LDPC codes," in *Proc. IEEE Globecom*, San Antonio, TX, Nov. 2001, pp. 1036–1036E.

[8] Abhiram Prabhakar and Krishna Narayanan, "A memory efficient Serial LDPC decoder architecture", ICASSP 2005, Philadelphia, Pages V-41 – V-44.

[9] S. Chung, T. Richardson, and R. Urbanke. Analysis of Sum-Product Decoding of Low-Density Parity-Check Codes Using a Gaussian Approximation. IEEE Trans. on Inform. Theory, 47(2):657–670, Feb 2001.

[10] D. MacKay and R. Neal. Near Shannon Limit Performace of Low Density Parity Check codes. In Elec. Letters, volume 32, pages 1645–6, Aug 1996

Clock Cyle	SUB3	SUB2	SUB1
0-31	Compare with **Min3** to find Min 3 En3 = signbit(Input – Min3)	[Conditional]Compare with **Min2** to find Min2 and **Min2_ index** En2 = signbit(Input – Min2)	[Conditional]Compare with **Min1** to find Min1 and **Min1_index** En1 = signbit(Input – Min1)
32	D13=Min1-Min3 (15b)	**Min2_neg** =0-Min2	D12_bar=Min2-Min1 Slight transformation of (15a) to ease the implementation further
33	D23=Min2-Min3 (15c)	**Min2_O** =Min2-offset	**Min1_neg** =0-Min1
34	(16b)	if sign bit of (Min2-offset) =1 Min2_O=0 Min2_O_neg=0 Else **Min2_O_neg=** [0-(Min2-offset)]	**Min1_O** =Min1-offset
35	(16c)	(16c)	if sign bit of (Min1-offset) =1 Min1_O=0 Min1_O_neg=0 Else **Min1_O_neg=** [0-(Min1-offset)]
0-31(i.e. 36-68)	*		

*Results are available at the start of 36 clock cycle

Table 1: Scheduling Table for Conditional Min Sum Correction

According to (10), Now we just have to set the output to a)Min 1 or b) –Min 1 or c)Min1_O or 3) –Min1_O based on the condition flags and delayed sign bits.

When the index is equal to Min1_index, you have to set the output to a)Min 2 or b) –Min 2 or c)Min2_O or 3) -Min2_O based on the condition flags and delayed sign bits.