International Conference on Innovative Trends in Electronics Communication and Applications 2015 [ICIECA 2015]

ISBN	978-81-929742-6-2		**VOL**	01
Website	icieca.in		**eMail**	icieca@asdf.res.in
Received	02 – April - 2015		**Accepted**	15 - November - 2015
Article ID	ICIECA001		**eAID**	ICIECA.2015.001

LDPC Decoder LLR Stopping Criterion

Janak Sodha[1]

[1]Department of Physics
University of the West Indies
Cave Hill Campus, Barbados

Abstract: *The log-likelihood ratio test on a single check node within the LDPC decoder is monitored to develop a stopping criterion for the decoder that is better than previous stopping criteria, without sacrificing the BER performance. Simulation results are presented for the transmission of the rate* $\frac{1}{2}$ *(288, 576) WiMAX 802.16e LDPC code digits using binary phase shift keying (BPSK) over an AWGN channel.*

Keywords: LDPC codes, Stopping Criterion, Decoding Failure, Iterative Decoding

INTRODUCTION

The decoder for a *Low density parity check* (LDPC) code or *Galleger* code [1] will iterate until either a valid codeword has been found or the predefined maximum number of iterations $L_{\max}$ has been reached before stopping. For latency critical applications, the disadvantage is that an LDPC decoder requires many more iterations than a turbo decoder [2], [3], [4], [5]. At low *signal-to-noise ratios* (SNRs), $L_{\max}$ is much larger than the *average number of iterations* $\overline{L}$ required establishing a valid code word under high SNRs. If it could be established that further decoder iterations are unlikely to yield a valid code word, then the decoding latency can be reduced under low SNRs by stopping the decoder early, before $L_{\max}$ iterations. In this paper, a new stopping criterion is presented that out performs the well-known stopping criteria [6], [7], [8] with the added advantage of a much lower implementation complexity.

BACKGROUND

We shall review the iterative log-likelihood decoding algorithm for binary LDPC codes to establish the notation that will simplify the explanation of the stopping algorithm. Let $\mathbf{H}$ represent the LDPC parity check matrix of size $(M \times N)$, which can be viewed as a Tanner graph [1] with N *bit nodes* and M *check nodes*. Let

$$\mathbf{c}^T = \left[c_0, c_1, \cdots, c_n, \cdots, c_{N-1}\right]$$

of $(N \times 1)$ size represent the transmitted code word and let

Cite this article as: Janak Sodha. "LDPC Decoder LLR Stopping Criterion". *International Conference on Innovative Trends in Electronics Communication and Applications (2015)*: 01-08. Print.

$$\mathbf{r}^T = [r_0, r_1, \cdots, r_n, \cdots, r_{N-1}]$$

represent the corresponding received noisy word, where N is the length of the code word. If after l iterations, the LDPC decoder outputs a valid code word $\widehat{\mathbf{c}}^{(l)}$, then the syndrome $\mathbf{s}$ is given by

$$\mathbf{s} = \mathbf{H}\widehat{\mathbf{c}}^{(l)} = \mathbf{0}$$

where the M syndrome bits within $\mathbf{s}$ correspond to the M check nodes. Using *binary phase shift keying* (BPSK) to transmit the LDPC code digits over an *additive white gaussian noise* (AWGN) channel with a single-sided noise power spectral density N_o W/Hz, the BPSK signal points will have the amplitudes $r_n = \pm\sqrt{E_b R} + n_n$, where E_b is the energy per binary digit, $R = 1 - \frac{M}{N}$ is the code rate and n_n is a zero mean Gaussian random variable with variance $\sigma = \sqrt{\frac{N_o}{2}}$. Let N_m for $m = 0,1,\ldots,M-1$ represent the set of non-zero binary digits on the m^{th} row of $\mathbf{H}$. Furthermore, let $\mathrm{N}_{m,n}$ for $n = 0,1,\ldots,N-1$ represent the set of non-zero binary digits on the m^{th} row of $\mathbf{H}$ excluding n^{th} column or equivalently, the bit nodes connected to the m^{th} check node, except the n^{th} bit node on a Tanner graph. Finally, let Z_n for $n = 0,1,\ldots,N-1$ represent the set of non-zero binary digits on the n^{th} column of $\mathbf{H}$ or equivalently, the check nodes corrected to n^{th} bit node. Let $\lambda_n^{[l]}$ denote the *log-likelihood ratio* (LLR)

$$\lambda(c_n \mid \mathbf{r}) = \log_e \frac{P(c_n = 1 \mid \mathbf{r})}{P(c_n = 0 \mid \mathbf{r})}$$

after the l^{th} iteration, where $l = 1, 2, \cdots, L_{\max}$, so that

$$\lambda_n^{[l]} = L_c r_n + \sum_{m \in Z_n} \eta_{m,n}^{[l]}$$

where $L_c r_n$ is the *intrinsic* information [9] in which the *channel reliability*

$$L_c = 2\frac{\sqrt{E_b R}}{\sigma^2}$$

and $\eta_{m,n}^{[l]}$ is the extrinsic information on c_n according to the m^{th} check node on the l^{th} iteration of the decoder given by

$$\eta_{m,n}^{[l]} = -2\tanh^{-1}\left[\prod_{j \in \mathrm{N}_{m,n}} \tanh\left(-\frac{\left(\lambda_j^{[l-1]} - \eta_{m,j}^{[l-1]}\right)}{2}\right)\right].$$

The decoding algorithm steps are as follows:

S0 Initialization: Set $\eta_{m,n}^{[0]} = 0$ for all (m,n) with $H(m,n) = 1$

S1 Set $\lambda_n^{[1]} = L_c r_n$

S2 Set the maximum number of decoder iterations $L_{\max}$

S3 For each (m,n) with $H(m,n) = 1$, compute $\eta_{m,n}^{[l]}$

Cite this article as: Janak Sodha. "LDPC Decoder LLR Stopping Criterion". *International Conference on Innovative Trends in Electronics Communication and Applications (2015)*: 01-08. Print.

S4 For $n = 0,1,\cdots N-1,$ compute $\lambda_n^{[l]} = L_c r_n + \sum_{m \in Z_n} \eta_{m,n}^{[l]}$

S5 If $\lambda_n^{[l]} > 0,$ set $\widehat{c_n} = 1,$ otherwise set $\widehat{c_n} = 0$

S6 If $\mathbf{H}\widehat{\mathbf{c}}^{(l)} = \mathbf{0},$ stop decoding, otherwise if $l < L_{\max},$ go to S3, otherwise stop decoding.

The stopping criterion to be presented in the next section will replace step S6 and add a few more steps.

STOPPING CRITERIA FOR LDPC CODES

Typically, l will reach $L_{\max}$ iterations because the extrinsic information $\sum_{m \in M_n} \eta_{m,n}^{[l]}$ oscillates after an initial increase due to a few $\lambda_n^{[l]}$ values distributed throughout $\widehat{c_n}$ that provide overwhelming incorrect evidence in favor of either $\widehat{c_n} = 1$ or $0.$ As in [8], we shall categorize the decoding behavior in terms of $\lambda_n^{[l]}$ as either *convergence*, *stuck* or *oscillation*. In convergence, the average magnitude of the LLRs $\left|\overline{\lambda^{[l]}}\right| = \frac{1}{N}\sum_{n=0}^{N-1}\left|\lambda_n^{[l]}\right|$ increases with each iteration and a valid code word is eventually found. In the stuck case, $\left|\overline{\lambda^{[l]}}\right|$ is stuck on a particular value after a certain number of iterations and a valid code word is not found. Finally in oscillation, $\left|\overline{\lambda^{[l]}}\right|$ oscillates after an initial increase. Although rare, the decoding behavior can change from oscillation to convergence. This behavior is referred to as *slow convergence* [8]. In [6], the stopping criterion is based on monitoring the *variable node reliability* $VNR^{(l)}$ defined by $VNR^{(l)} = \sum_{n=0}^{N-1}\left|\lambda_n^{[l]}\right|,$ which is simply $\left|\overline{\lambda^{[l]}}\right| N.$ At each iteration, the decoder monitors the variation of $VNR^{(l)}$ in relation to a threshold $VNR_{off} = 4N\left(\frac{E_b}{N_o}\right)_{WR},$ where $\left(\frac{E_b}{N_o}\right)_{WR}$ is the *signal-to-noise ratio* (SNR) point near the waterfall region in the bit-error rate (BER) curve for the LDPC code, using the following steps: S1 If $VNR^{(l)} \le VNR^{(l-1)}$ for $l > 1,$ stop decoding; S2 If $VNR^{(l)} > VNR_{off},$ S1 is switched off and further iterations allowed. Thus, if $VNR^{(l)}$ does not change or is less than the previous value, then further iterations are stopped because the stuck or oscillation conditions are assumed to be true. Slow convergence is assumed if $VNR^{(l)} \ge VNR_{off}$ at which point the decoder is allowed to iterate until a valid code word is found or $l = L_{\max}.$

A similar method was adopted by Li *et. al*. [7] in which the average LLR magnitude $\left|\overline{\lambda^{[l]}}\right| = \frac{VNR^{(l)}}{N} = \frac{1}{N}\sum_{n=0}^{N-1}\left|\lambda_n^{[l]}\right|$ is computed at the end of each iteration and utilized as follows: S1 Initialize a counter to zero, set thresholds J and $P.$ Note the symbol λ in [7] has been replaced here by J to avoid confusing it with the LLR symbol $\lambda_n^{[l]}$; S2 If $\left|\frac{VNR^{(l)}}{N} - \frac{VNR^{(l-1)}}{N}\right| < J\left|\frac{VNR^{(l-1)}}{N}\right|$, increase the counter by one. Otherwise reset the counter to zero; S3 If counter reaches P or $l = L_{\max},$ stop decoding. Otherwise proceed to the next iteration. Notice the slight clever modification in comparison to [6], which monitors a factor J increase over the previous value $\left|\overline{\lambda^{[l-1]}}\right|$ over P iterations. It turns out that the optimum value for P is 2 for any LDPC code [7].

Shin *et. al*. [8] proposed a stopping criterion which outperforms the method in [6], but unfortunately, there was no mention of the Li *et. al.* method [7]. The algorithm is based on the *number of satisfied parity-check constraints* $N_{spc}^{(l)}$ given by $N_{spc}^{(l)} = M - 1^T\mathbf{H}\widehat{\mathbf{c}}^{(l)} = M - \sum_{i=0}^{M-1} s_i,$ where 1^T is the all-one column vector of length $M.$ If a valid code word is found, then the syndrome $\mathbf{s} = \mathbf{H}\widehat{\mathbf{c}}^{(l)} = \mathbf{0},$ so that $\sum_{i=0}^{M-1} s_i = 0,$ and $N_{spc}^{(l)} = M.$ Thus, $N_{spc}^{(l)}$ is simply algebraically adding up all the non-zero syndrome bits and taking the total away from the syndrome length $M.$ The stopping criterion employed is to monitor the oscillation of the variable $N_{spc}^{(l)}$ using three thresholds $\theta_d,$ $\theta_{\max}$ and θ_{spc} and a counter c_d as follows: S0 If $l = 1,$ initialize

$c_d = 0;$ S1 Wait for the test $\mathbf{H}\widehat{\mathbf{c}}^{(l)} = \mathbf{s};$ S2 Compute $N_{spc}^{(l)} = M - \sum_{i=0}^{M-1} s_i;$ S3 If $l > 1,$ compute $d_{spc}^{(l)} = N_{spc}^{(l)} - N_{spc}^{(l-1)}.$ Otherwise go to S1; S4 If $d_{spc}^{(l)} \le \theta_d,$ increase c_d by 1 $(c_d \leftarrow c_d + 1).$ Otherwise reset $c_d = 0$ and go to S1; S5 If $c_d < \theta_{\max}$, go to S1; S6 If $N_{spc}^{(l)} \le \theta_{spc},$ stop decoding. Otherwise reset $c_d = 0$ and go to S1, where c_d counts how long the small increment successively persist. If $c_d \ \ge \ \theta_{\max}$, further decoder iterations are stopped if $N_{spc}^{(l)} \le \theta_{spc}.$ If $N_{spc}^{(l)} > \theta_{spc},$ slow convergence is assumed and further iterations are allowed. Clearly this algorithm is monitoring the level of oscillation of $N_{spc}^{(l)}.$ If the oscillations are small and prolonged, further decoder iterations are stopped if the current $N_{spc}^{(l)}$ is not sufficiently large enough ($N_{spc}^{(l)} \le \theta_{spc}$) to indicate the possibility of slow convergence. The disadvantage is that three thresholds $\theta_d,$ $\theta_{\max}$ and θ_{spc} have to be optimized for a given LDPC code. More recently in [10], based once again on $N_{spc}^{(l)},$ a counter was used to only accumulate the evidence in favor of iterating the received noisy word towards a valid code word to slightly outperform the Shin *et. al*. criterion.

PROPOSED STOPPING CRITERION

Gallager [1] proved that for a sequence of K independent binary digits $a_i,$ with a probability p_i for $a_i = 1,$ the probability that the whole sequence contains an even number of binary digits 1's is given by $\left[\frac{1}{2} + \frac{1}{2}\prod_{i=0}^{K-1}(1 - 2p_i)\right].$ Thus, if we let $\mathrm{N}_{m,n}^{*}$ represent the bit nodes connected to the m^{th} check node, *including* the n^{th} bit node, then to a good approximation the probability $P(s_m = 0 \mid \mathbf{r})$ for the m^{th} check node syndrome is given by

$$P(s_m = 0 \mid \mathbf{r}) = \frac{1}{2} + \frac{1}{2}\prod_{n \in \mathrm{N}_{m,n}^{*}} \left(1 - 2P\left(c_n = 1 \mid \mathbf{r}\right)\right)$$

where

$$P\left(c_n = 1 \mid \mathbf{r}\right) = \frac{e^{\lambda_n^{[l]}}}{1 + e^{\lambda_n^{[l]}}}$$

because the $N_{m,n}^{*}$ bit nodes are sparsely separated. For a given SNR, if the decoder iterations will eventually lead to a valid code word, then $P(s_m = 0 \mid \mathbf{r})$ will gradually increase towards the value 1. To monitor this feature, we shall use the LLR ratio

$$\Lambda = \log_e\left(\frac{P\left(s_m = 0 \mid \mathbf{r}\right)}{1 - P\left(s_m = 0 \mid \mathbf{r}\right)}\right)$$

which will increase to a large value as $P(s_m = 0 \mid \mathbf{r})$ increases towards 1. If the decoder is unable to establish a valid code word, then Λ will be close to zero. To minimize complexity, we shall use only the single M^{th} check node to develop a stopping criterion as follows. After the first $L_{\min}$ iterations, if $\Lambda = \ \log_e\left(\frac{P(s_{M-1}=0|\mathbf{r})}{1-P(s_{M-1}=0|\mathbf{r})}\right)$ has reduced in comparison to its previous value, then on the next iteration, stop the decoder if the magnitude of the change is less than a step-threshold $T.$ If this change is larger than $T,$ whether it be positive or negative, this would be a good indication that the iteration process is still beneficial and the decoder should be allowed to continue iterating. Further decoder iterations are stopped if the maximum number $L_{\max}$ or until a valid code word has been found, whichever comes first. Let $\Lambda^{[l]}$ represent the current value of $\log_e\left(\frac{P(s_{M-1}=0|\mathbf{r})}{1-P(s_{M-1}=0|\mathbf{r})}\right)$ and $\Lambda^{[l-1]}$ represent its previous value. Let $L_{\min}$ represent the minimum of the decoder iterations before the stopping criterion is activated. In addition to the initialization step S0 of the decoding algorithm, an alert-flag is set to 0 and the following steps are inserted to replace step S6 of the LDPC decoding algorithm presented in section II:

Cite this article as: Janak Sodha. “LDPC Decoder LLR Stopping Criterion”. *International Conference on Innovative Trends in Electronics Communication and Applications (2015)*: 01-08. Print.

S6 If $\mathbf{H}\widehat{\mathbf{c}}^{(l)} = \mathbf{0}$ stop decoding, otherwise continue to the next step
S7 If alert-flag = 1, then execute S8, otherwise skip to S9
S8 If $\left|\Lambda^{[l]} - \Lambda^{[l-1]}\right| < T,$ then stop decoding, otherwise set alert-flag = 0
S9 If alert-flag = 0 **and** $\left(\Lambda^{[l]} < \Lambda^{[l-1]}\right)$ **and** $\left(l > L_{\min}\right)$ then set alert-flag = 1
S10 **If** $\left(l < L_{\max}\right)$, **go to S3, otherwise stop decoding.**

WIMAX 802.16E LDPC CODE

The WiMAX 802.16e LDPC code is formed from the expansion of a model matrix H_{bm} of size $\left(m_b \times n_b\right)$, where $n_b = 24$ and $m_b = (1-R)24,$ where R is the code rate. The size of the parity check $\mathbf{H}$ depends on the expansion factor q, with the codeword length $N = qn_b$ and the number of parity checks $M = qm_b$. The values of q range from 24 to 96 in increments of 4 and therefore, the smallest code is of length 576 bits and the largest is 2304 bits. The first $\left(n_b - m_b\right)$ columns represent the systematic bits, with the remaining m_b columns representing the parity bits. Each entry $p(i, j)$ of the base matrix H_{bm} is either a $(q \times q)$ all zero matrix or a $(q \times q)$ permuted identity matrix. If the entry is blank or less than zero, it is expanded into the all zero matrix. Otherwise the value represents the circular right shift size of the identity matrix. The base model matrix for rate $\frac{1}{2}$ codes in the WiMAX 802.16e standard is shown in Fig. 1. The shift sizes listed are for the largest code length (N = 2304). For shorter length rate $\frac{1}{2}$ codes, the shift size $s(f, i, j)$ is scaled depending on the expansion factor q_f as follows

$$s(f,i,j) = \left\lfloor \frac{p(i,j) \times q_f}{q_{\max}} \right\rfloor$$

where $\lfloor x \rfloor$ denotes the flooring function applied to x which gives the nearest integer to $-\infty$, $q_{\max}$ is the maximum expansion value of 96 and q_f is one of the 19 expansion values ranging from 24 to 96. For example, the shift size of entry (4,1), which is equal to 61, using an expansion factor of 24 gives $\left\lfloor \frac{61 \times 24}{96} \right\rfloor$, equates to 15. The resulting permutation matrix is shown in Fig. 2. When circularly shifting the identity matrix to the right, the 1 that reaches the last column is brought back to the first column of the same row. This process continues for the total number of shifts. As the bottom row of the identity matrix has a 1 in its last column, shifting it 15 times would result in this 1 being placed at column 15 of that row as shown in Fig. 2. The process of determining the shift size and applying it to the identity matrix is repeated for all nonnegative entries in Fig 1.

-1	94	73	-1	-1	-1	-1	-1	55	83	-1	-1	7	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	27	-1	-1	-1	22	79	9	-1	-1	-1	12	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	24	22	81	-1	33	-1	-1	-1	0	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1
61	-1	47	-1	-1	-1	-1	-1	65	25	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1
-1	-1	39	-1	-1	-1	84	-1	-1	41	72	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	46	40	-1	82	-1	-1	-1	79	0	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1
-1	-1	95	53	-1	-1	-1	-1	-1	14	18	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1
-1	11	73	-1	-1	-1	2	-1	-1	47	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1
12	-1	-1	-1	83	24	-1	43	-1	-1	-1	51	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1
-1	-1	-1	-1	-1	94	-1	59	-1	-1	70	72	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1
-1	-1	7	65	-1	-1	-1	-1	39	49	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0
43	-1	-1	-1	-1	66	-1	41	-1	-1	-1	26	7	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

Fig. 1 The WiMAX 802.16e base model matrix

$$\begin{matrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0
\end{matrix}$$

Fig. 2 Permutation matrix formed by circularly right shifting the identity matrix by 15

SIMULATION RESULTS

Using BPSK over an AWGN to transfer the WiMAX [10], 802.16e [IEEE] rate $\frac{1}{2}$ (288, 576) LDPC code digits, simulation results are presented in Figs. 3 and 4 showing the dependence of the *probability of an information binary digit error* P_e and $\overline{L}$ on the step-threshold T over a range of channel SNRs (dB).

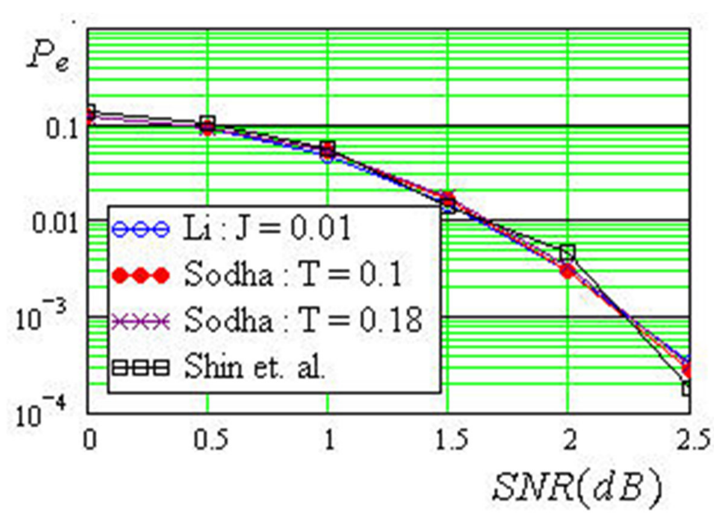

Fig. 3 LDPC decoder performance

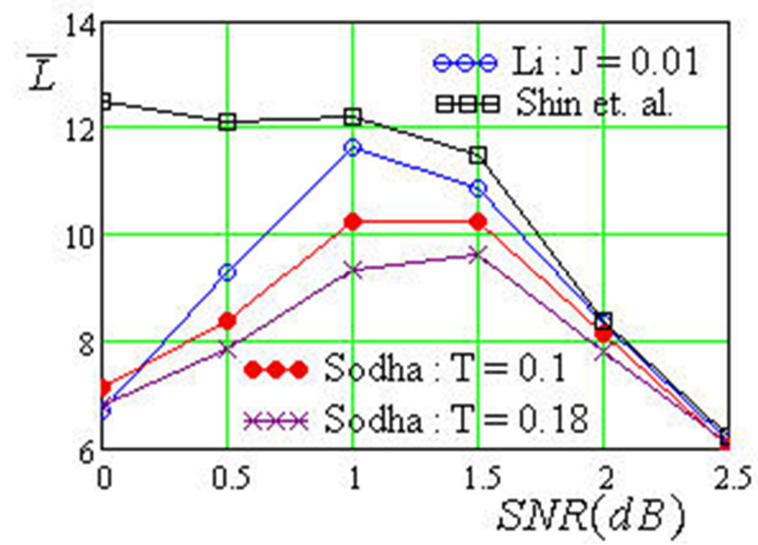

Fig. 4 Average number of iterations

The proposed criterion is compared with previous criteria in these figures ensuring in each case that a given stopping criterion should not significantly impact P_e for a given SNR. The Shin-curve corresponds to θ_d = 8, $\theta_{\max}$ = 6 and θ_{spc} = 260 and the Li-curve corresponds to $P=2$ and $J=0.01$. In [7], $J=0.001$ was recommended in general, but the curve for $J=0.01$ was selected because the corresponding $\bar{L}$ performance is better. Also, extensive simulations were undertaken to verify as stated in [7] that $P=2$ is the optimum threshold for any LDPC code. These results have not been shown for brevity. As expected, the stopping criteria curves merge with the $L_{\max}=15$ (no stopping criteria) curve at high SNRs because of fast convergence. Notice how the proposed stopping criterion outperforms Li and Shin *et. al.*'s algorithm at low SNRs using only a single check node syndrome probability that is calculated using only those bit nodes connected to M^{th} check node. For T larger than 0.18, there is a noticeable increase in P_e that is accompanied with a further reduction in $\bar{L}$. Taking a closer look at the level of complexity involved, specifically for the WiMAX 802.16e rate $\frac{1}{2}$ ($M=288, N=576$) LDPC code, the index n of the bit nodes ranges from $n=0,1,\cdots,575$ and m ranges from $m=0,1,\cdots,287$. The set of bit nodes connected to the 288^{th} check node are

$$\mathbf{N}^*_{287,n}=\{9,135,177,269,288,575\}$$

Given the separation of these bits nodes, the assumption that the probabilities $P(c_n=1\,|\,\mathbf{r})$ at these six nodes are independent is a good approximation to determine

$$P(s_{287}=0\,|\,\mathbf{r})=\frac{1}{2}+\frac{1}{2}\prod_{n\in\mathbf{N}^*_{287,n}}\left(1-2\frac{e^{\lambda_n^{[l]}}}{1+e^{\lambda_n^{[l]}}}\right)$$

which in turn is used to calculate

$$\Lambda^{[l]}=\log_e\left(\frac{P(s_{287}=0\,|\,\mathbf{r})}{1-P(s_{287}=0\,|\,\mathbf{r})}\right)$$

at the l^{th} iteration.

Cite this article as: Janak Sodha. "LDPC Decoder LLR Stopping Criterion". *International Conference on Innovative Trends in Electronics Communication and Applications (2015)*: 01-08. Print.

CONCLUSIONS

Using the short-length WiMax 802.16e rate $\frac{1}{2}$ (288, 576) LDPC code, it was shown that the stopping criterion proposed outperforms all the previous algorithms. Specifically, the average number of decoder iterations $\overline{L}$ can be reduced below 10 at the low SNR of 0 dB, instead of the standard 15 required over low SNRs. This can be further improved by increasing the value of the step-threshold T for the penalty of a slight increase in P_e. The stopping criterion proposed reduces the time taken to decode, with a lower complexity than previous methods, to create a latency efficient LDPC decoder.

REFERENCES

[1] R. G. Gallager, Low density parity check codes, *IRE Trans. Information Theory* (8) (1962) 21-28.

[2] C. Berrou and A. Glavieux, Near Optimum Error Correcting Coding and Decoding: Turbo-codes, *IEEE Trans. Commun.* 44 (10) (1996) 1261-1271.

[3] D. J. MacKay and R. M. Neal, Near-Shannon-limit performance of low-density parity-check codes, *Electron. Lett.* 33 (6) (1997) 457-458.

[4] T. J. Richardson, A. Shokrollahi and R. Urbanke, Design of capacity-approaching low-density parity-check codes, *IEEE Trans. Inform. Theory* 47 (2001) 619-637.

[5] J. Sodha and A. Als, Shape nature of error-control codes, *Elsevier Signal Processing Journal* 83 (2003) 1457-1465.

[6] F. Kienle and N. Wehn, Low complexity stopping criterion for LDPC code decoders, Proc. IEEE VTC 2005-Spring, Stockholm, Sweden, June 2005, pp. 606--609.

[7] J. Li, X. You and J. Li (Tiffany), Early Stopping for LDPC Decoding: Convergence of Mean Magnitude (CMM), *IEEE Commun. Letters*, 10 (9) (2006) 667-669.

[8] D. Shin, K. Heo, S. Oh and J. Ha, A Stopping Criterion for Low-Density Parity-Check Codes, *Proc. IEEE VTC2007-Spring*, Dublin, Ireland, April 2007, pp. 1529--1533.

[9] T. K. Moon, Error Correction Coding: Mathematical Methods and Algorithms, John Wiley, NJ, 2005, Chapter 15, pp. 649-652.

[10] IEEE 802.16e. Air interface for fixed and mobile broadband wireless access systems. IEEE Std 802.16e-2005.

Cite this article as: Janak Sodha. "LDPC Decoder LLR Stopping Criterion". *International Conference on Innovative Trends in Electronics Communication and Applications (2015)*: 01-08. Print.