

# Threshold Multi Split-Row algorithm for decoding irregular LDPC codes

Chakir Aqil<sup>\*1</sup>, Rachid El Alami<sup>2</sup>, Ismail Akharraz<sup>1</sup>, Abdelaziz Ahaitouf<sup>1</sup>

<sup>1</sup>Sidi Mohamed Ben Abdellah University, Mathematics Physics and Computer Science, LSI, FP, Taza, Morocco

<sup>2</sup>Sidi Mohamed Ben Abdellah University, LESSI Laboratory, FSDM, Fez, Morocco

## ARTICLE INFO

Article history:

Received: 10 June, 2017

Accepted: 15 July, 2017

Online: 26 December, 2017

Keywords:

LDPC codes

Threshold Multi Split-Row algorithm

Multi Split-Row algorithm

## ABSTRACT

In this work, we propose a new threshold multi split-row algorithm in order to improve the multi split-row algorithm for LDPC irregular codes decoding. We give a complete description of our algorithm as well as its advantages for the LDPC codes. The simulation results over an additive white gaussian channel show that an improvement in code error performance between 0.4 dB and 0.6 dB compared to the multi split-row algorithm.

## 1 Introduction

Low-density parity check (LDPC) codes are a class of linear block codes, which were first, introduced by Gallager in 1963 [1]. As soon as they were re-invented in 1996 by Mackay [2], LDPC codes have received a lot of attention because their error performance is very close to the Shannon limit when decoded using iterative methods [3]. They have emerged as a viable option for forward error correction (FEC) systems and have been adopted by many advanced standards, such as 10 gigabit ethernet (10GBASET) [4], [5] and digital video broadcasting (DVB-S2) [6], [7]. In addition, the generations of WiFi and WiMAX are considering LDPC codes as part of their error correction systems [8], [9].

In the present paper, we propose the threshold multi split-row method for decoding irregular low-density parity-check (LDPC) codes [10], to have better performance in terms of bit error rate. We will apply the thresholding algorithm in [11] for multi split row of irregular LDPC codes [10].

The proposed split-row decoder [10],[11] splits the row processing into two or multiple nearly independent partitions. Each block is simultaneously processed using minimal information from an adjacent partition. The key idea of split-row is to reduce communication between row and column processors which has a major role in the interconnect complex-

ity of existing LDPC decoding algorithms such as sum product (SPA) [3] and minsum (MS) [12].

This paper introduces threshold multi split-row decoding which significantly reduces wire interconnect complexity and considerably improves the error performance compared to non-threshold multi-split decoding [10]. The paper is organized as follows: Section II reviews the sum product and minsum, split-row and split-row threshold decoding algorithms. Threshold multi-split-row and its error performance result is presented in section III. Finally, the conclusion is in section IV.

## 2 Previous algorithms

### 2.1 Sum Product Decoding (SPD)

The SPD assumes a binary code word  $(x_1, x_2, \dots, x_N)$  transmitted using a binary phase-shift keying (BPSK) modulation. The sequence is transmitted over an additive white gaussian noise (AWGN) channel and the received symbol is  $(y_1, y_2, \dots, y_N)$ .

We define:

$V_{(i)} = \{j : H_{ij} = 1\}$  as the set of variable nodes which participate in the check equation  $i$ .

$C_{(j)} = \{i : H_{ij} = 1\}$  as the set of check nodes which participate in the variable node  $j$  update.

Also

$V_{(i)} \setminus j$  denote all variable nodes in  $V_{(i)}$  except node  $j$ .

\*Corresponding Author: Chakir Aqil, FP, Taza, Morocco & aqil.chakir@hotmail.com

$C_{(j)} \setminus i$  denote all check nodes in  $C_{(j)}$  except node  $i$ . Moreover, we define the following variables used in this paper.

$\lambda_j$  : is defined as the information derived from the log-likelihood ratio of received symbol  $y_j$ .

$$\lambda_j = \log(P(x_i = 0|y_j)/P(x_i = 1|y_j)) = (2y_j)/\sigma^2 \quad (1)$$

where  $\sigma^2$  is the noise variance.  $\alpha_{ij}$  is the message from check node  $i$  to variable node  $j$ . This is the row processing output.  $\beta_{ij}$  is the message from variable node  $j$  to check node  $i$ . This is the column processing output.

The sum product algorithm decoding as described by mohsenin and Baas [13], can be summarized in the following four steps:

### 2.1.1 Initialization

For each  $i$  and  $j$ , initialize  $\beta_{ij}$  to the value of the log-likelihood ratio of the received symbol  $y_j$ , which is  $\lambda_j$ . During each iteration,  $\alpha\beta$  messages are computed and exchanged between variables and check nodes through the graph edges according to the following steps numbered 2-4.

#### 2. Row processing or check node update

Compute  $\alpha_{ij}$  messages using  $\beta$  messages coming from all other variable nodes connected to check node  $C_i$ , excluding the  $\beta$  information from  $V_j$ :

$$\alpha_{ij} = \prod_{j' \in V(i) \setminus j} \text{sign}(\beta_{ij'}) \times \phi(\sum_{j' \in V(i) \setminus j} \phi(|\beta_{ij'}|)) \quad (2)$$

where the non-linear function

$$\phi(x) = -\log(\tanh(|x|/2)) \quad (3)$$

The first product term in the equation that update the parameter, is called the parity (sign) update and the second product term is the reliability (magnitude) update.

#### 3. Column processing or variable node update

Compute  $\beta_{ij}$  messages using channel information  $\lambda_j$  and incoming, messages  $\alpha_i$  from all other check nodes connected to variable node  $V_j$ , excluding check node  $C_i$ .

$$\beta_{ij} = \lambda_j + \sum_{i' \in C(j) \setminus i} \alpha_{i'j} \quad (4)$$

#### 4. Syndrome check and early termination

When the column processing is finished, every bit in column  $j$  is updated by adding the channel information  $\lambda_j$  and  $\alpha$ , messages from neighboring check nodes.

$$z_j = \lambda_j + \sum_{i' \in C(j)} \alpha_{i'j} \quad (5)$$

From the updated vector, an estimated code vector  $\widehat{X} = \{\widehat{x}_1, \widehat{x}_2, \dots, \widehat{x}_N\}$  is calculated by:

$$\widehat{x}_i = \begin{cases} 1 & \text{if } z_i \leq 0 \\ 0 & \text{if } z_i > 0 \end{cases} \quad (6)$$

If  $H \times \widehat{x}^t = 0$  then  $\widehat{X}$  is a valid codeword and therefore the iterative process has converged and decoding stops. Otherwise, the decoding repeats from step 2 until a valid codeword is obtained or the number of iterations reaches a maximum number, which terminates the decoding process.

## 2.2 MinSum Decoding

The check node or row processing stage of SP decoding can be simplified by approximating the magnitude computation in Eq. 2 with a minimum function. The algorithm using this approximation is called min-sum (MS):

$$\alpha_{ij} = \prod_{j' \in V(i) \setminus j} \text{sign}(\beta_{ij'}) \times \text{Min}_{j' \in V(i) \setminus j} (|\beta_{ij'}|) \quad (7)$$

In MS decoding, the column operation is the same as in SP decoding. The error performance loss of MS decoding can be improved by scaling the check ( $\alpha$ ) values in Eq. 7 with a scale factor  $S \leq 1$  which normalizes the approximations [14], [15].

$$\alpha_{ij} = S \times \prod_{j' \in V(i) \setminus j} \text{sign}(\beta_{ij'}) \times \text{Min}_{j' \in V(i) \setminus j} (|\beta_{ij'}|) \quad (8)$$

## 2.3 Split Row and Split Row Threshold Decoding

Recall that a standard message passing two-phase algorithm consists of a check node update followed by a variable node update as shown in Fig. 1 (a). The basic idea in split-row [10], [16] and split-row threshold [11], is to divide the check node processing into two or multiple nearly-independent partitions. Each check node processor, simultaneously, computes a new message while using minimal information from its adjacent partitions. Split-row is illustrated in Fig. 1 (b) showing how the check node processing is partitioned into two blocks. A single bit of information (Sign) for each check node processor must be sent between partitions to improve the error performance.

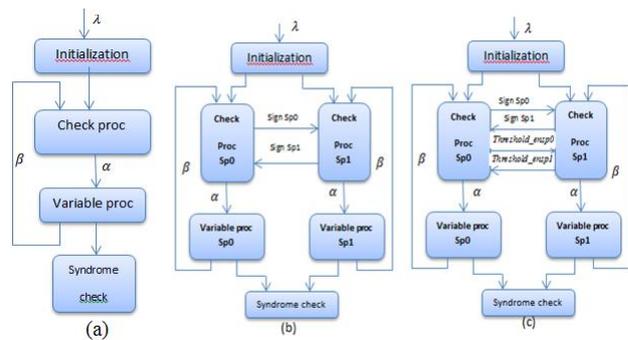


Figure 1. Block diagram of (a) standard two-phase decoding (b) split-row (c) split-row threshold block diagram.

The loss error performance is 0.5-0.7 dB of split-row that is the major drawback, when compared to minsum normalized and SPA decoders. This performance degradation is dependent on the number of check node partitions. For minsum split-row each partition has no information of the minimum value of the other partition, and a Sign signal is sent to minimize the error due to incorrect sign information of the true check node output  $\alpha$ .

The split-row threshold algorithm mitigates the error caused by incorrect magnitude by providing a signal,  $threshold\_en$ , which indicates whether a partition has a minimum less than a given threshold (T). This causes all check nodes to take the min of their own local minimum or T. Thus, any large deviations from the true minimum because of the partitioning are reduced, which then makes multi-split implementations feasible.

Figure 1 (c) shows the additional single bit signal ( $threshold\_en$ ) added to the original split-row architecture by the split-row threshold algorithm. This signal allows the split-row to remain essentially unchanged while adding some extra logic and minimal wiring to improve error significantly [11].

Therefore, the split-row architectures reduce the communication between check node and variable node processors, which is the major cause of the interconnect complexity found in existing LDPC decoder implementations. In addition, the area of each check node processor will be reduced. However, note that the variable node operation in the SPA, minsum, split-row and split-row threshold algorithms are all identical. Thus, the logic of the variable node processor is left unchanged.

### 3 Threshold Mutli Split Row Decoding

#### 3.1 Threshold Mutli Split Row Algorithm

In order to increase the parallelism and reduce the complexity of the decoder, the multi-split-row

method divides the lines of the systematic submatrix  $H_s$  into  $S_{pn}$  blocks called split- $S_{pn}$ . This approximation requires of the wire to correctly process the sign bits between the blocks. For the threshold multi split-row method, Each partition sends the status of its local minimum against a threshold to the next partition with a single wire, called  $Threshold\_ensp$ .

The block diagram of threshold multi split-row decoding with  $S_{pn}$  partitions, highlighting the sign and  $Threshold\_ensp$  passing signals, is shown in Fig. 2. These are the only wires passing between the partitions [17]. In each partition, local minimums are generated and compared with a threshold T simultaneously. If the local minimum is smaller than T then the  $Threshold\_ensp$  signal is asserted high. The magnitudes of the check node outputs are finally computed using local minimums and the  $Threshold\_ensp$  signal from neighboring partitions.

If a local partition's minimums are larger than T, and at least one of the  $Threshold\_ensp$  signals is high, then T is used to update its check node outputs. Otherwise, local minimums are used to update check node outputs.

in the threshold multi split-row algorithm, as shown in Eq. (10) and (11), the first and second Mins are compared with a predefined threshold, and a single-bit threshold-enable ( $Threshold\_en\_out$ ) global signal is sent to indicate the presence of a potential global minimum to other partitions.

#### 3.1.1 Initialization

The first and second minimums are determined for each partition according to the following relationships:

$$Min_{j' \in V(i) \setminus j}(|\beta_{ij'}|) = \begin{cases} Min1_i; & \text{if } j \neq argmin(Min1_i) \\ Min2_i; & \text{if } j = argmin(Min1_i) \end{cases} \quad (9)$$

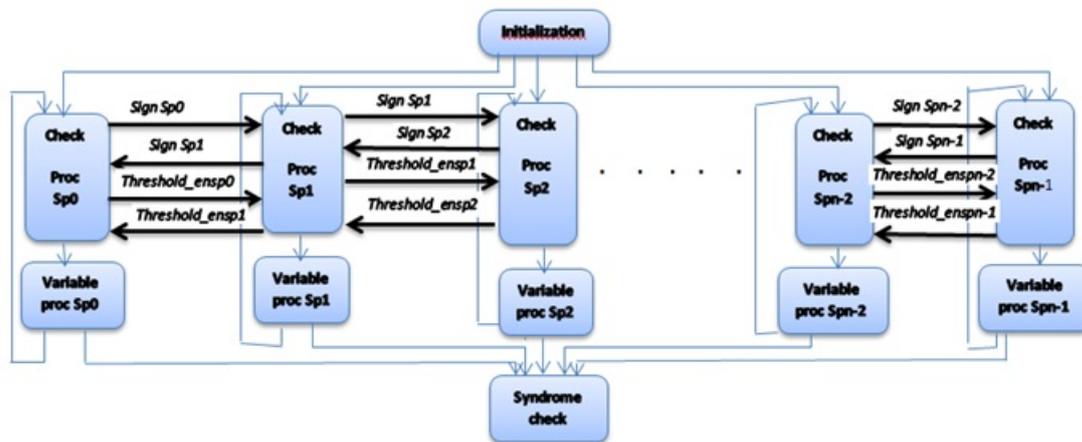


Figure 2. Block diagram of threshold multi split row decoding with  $S_{pn}$  partitions.

$$Min1_i = \min_{j \in V(i)}(|\beta_{ij}'|) \quad (10)$$

$$Min2_i = \min_{j'' \in V(i) \setminus \text{argmin}(min1_i)}(|\beta_{ij}''|) \quad (11)$$

to execute the algorithm one needs the number of partitions and the threshold value T. Finds  $Threshold\_ensp(i)_{out(i \mp 1)}$  for the ith partition of a spn-decoder.

**Algorithm 1** Threshold Multi Split-Row Algorithm

1-if  $Min1_i \leq T$  and  $Min2_i \leq T$  then  
 $Threshold\_ensp(i)_{out(i \mp 1)} = 1$

$$Min_{j' \in V(i) \setminus j}(|\beta_{ij}'|) = \begin{cases} Min1_i & \text{if } j \neq \text{argmin}(Min1_i) \\ Min2_i & \text{if } j = \text{argmin}(Min1_i) \end{cases} \quad (12)$$

2- if  $Min1_i \leq T$  and  $Min2_i \geq T$  then  
 $Threshold\_ensp(i)_{out(i \mp 1)} = 1$   
 $Threshold\_ensp(i+1) == 1$  or  
 $Threshold\_ensp(i-1) == 1$  then

$$Min_{j' \in V(i) \setminus j}(|\beta_{ij}'|) = \begin{cases} Min1_i & \text{if } j \neq \text{argmin}(Min1_i) \\ T & \text{if } j = \text{argmin}(Min1_i) \end{cases} \quad (13)$$

else

$$Min_{j' \in V(i) \setminus j}(|\beta_{ij}'|) = \begin{cases} Min1_i & \text{if } j \neq \text{argmin}(Min1_i) \\ Min2_i & \text{if } j = \text{argmin}(Min1_i) \end{cases}$$

end if

3- else if  $Min1_i \geq T$  and  $(Threshold\_ensp(i+1) == 1$  or  
 $Threshold\_ensp(i-1) == 1)$  then  
 $Threshold\_ensp(i)_{out(i \mp 1)} = 0$

$$Min_{j' \in V(i) \setminus j}(|\beta_{ij}'|) = T \quad (14)$$

4- else

$Threshold\_ensp(i)_{ou(i \mp 1)} = 0$

$$Min_{j' \in V(i) \setminus j}(|\beta_{ij}'|) = \begin{cases} Min1_i & \text{if } j \neq \text{argmin}(Min1_i) \\ Min2_i & \text{if } j = \text{argmin}(Min1_i) \end{cases} \quad (15)$$

end if.

The kernel of the threshold multi split-row algorithm is given in Algorithm 1. As shown, four conditions will occur:

**Condition 1:** both  $Min1$  and  $Min2$  are less than threshold  $T$ . In this case these 2 parameters are used to calculate  $\alpha$  messages according to Eq.(12). In addition,  $Threshold\_ensp(i)_{out(i \mp 1)}$ , which represents the general threshold-enable signal of a partition with two neighbors, is asserted high indicating that the least minimum ( $Min1$ ) in this partition is smaller than  $T$ .

**Condition 2:** only  $Min1$  is less than  $T$ . As Condition 1  $Threshold\_ensp(i)_{out(i \mp 1)} = 1$ . If at least one  $Threshold\_ensp(i \mp 1)$  signal from the nearest neighboring partitions is high, indicating that the local minimum in the other partition is less than  $T$ , then we use  $Min1$  and  $T$  to calculate  $\alpha$  messages according to Eq.(13). Otherwise, we use Eq.(12).

**Condition 3:** when the local  $Min1$  is larger than  $T$  and at least one  $Threshold\_ensp(i \mp 1)$  signal from the nearest neighboring partitions is high; thus, we only use  $T$  to compute all  $\alpha$  messages for the partition using Eq.(14).

**Condition 4:** when the local  $Min1$  is larger than  $T$  and if the  $Threshold\_ensp(i \mp 1)$  signals are all low; thus, we again use Eq.(12).

The variable node operation in minsum threshold multi split-row algorithm is identical to the minsum normalized and minsum multi split-row algorithms.

### 3.2 BER Simulation Results

The error performance simulations presented here assume an additive white gaussian noise (AWGN) channel with binary phase-shift keying (BPSK) modulation. The maximum number of iterations is set to  $I_{max} = 7$  or earlier when the decoder converged [11]. We fix the threshold  $I_{max}$  to satisfy a tradeoff between the performance correction and the speed of the decoder.

The following labelings are used for the figures: “MS Standard” for normalized minsum, “MS Multi Split-Row” for the method minsum multi split-row algorithm, and “S” for the scaling factor. The performances of irregular LDPC codes are illustrated in the figure given below.

The error performance depends strongly on the choice of threshold  $T$  values and the scaling factor  $S$ . The optimum values for  $T$  and  $S$  are obtained by empirical simulations as illustrated in the following figures.

The figure 3 shows the simulation to determine experimentally the threshold for minsum threshold multi split-row of the LDPC code (6,18) (1536,1152). with:

Code length  $N = 1536$ .

Information length  $K = 1152$ .

Column weight  $W_c = 6$  which is the number of ones per column.

Row weight  $W_r = 18$  which is the number of ones per row.

The threshold optimum value for an SNR ranging from 2.7 dB to 4.2 dB, with the value of the normalization factor  $S = 0.25$  obtained in figure 6, is 1.5.

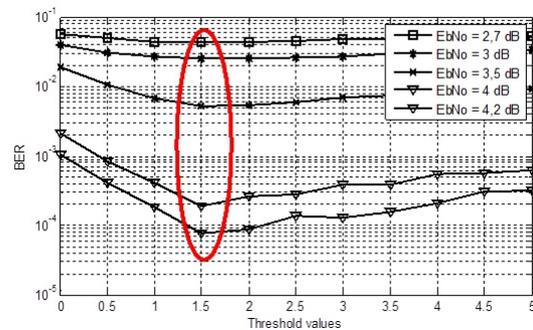


Figure 3. BER (Bit error rate) performance in function of the threshold for different SNR (Signal to noise ratio) values.

Figures 4 and 5 show the simulation to determine experimentally the scaling factor for normalized minsum and minsum multi split-row, respectively, of the LDPC code (6,18) (1536,1152). Optimal scale factor,

for normalized minsum, is 0.6. And optimal scale factor, for minsum multi split-row, is 0.2.

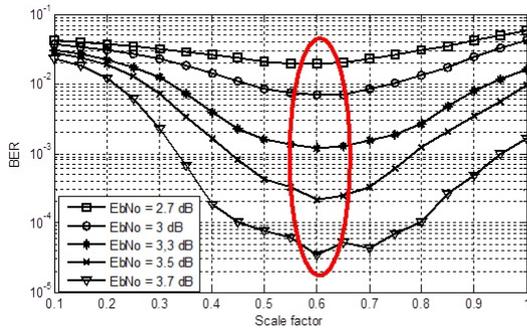


Figure 4. BER (Bit error rate) performance in function of the scale factor using normalized minsum for different SNR (Signal to Noise Ratio) values.

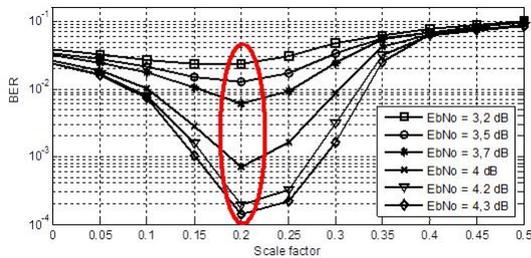


Figure 5. BER (Bit error rate) performance in function of the Scale factor using minsum multi split-row for different SNR (Signal to noise ratio) values.

Figure 6 shows the simulation to determine experimentally the scaling factor for minsum threshold multi split-row of the LDPC code (6,18)(1536,1152). Optimal scale factor is 0.25 with the threshold  $T=1.5$ .

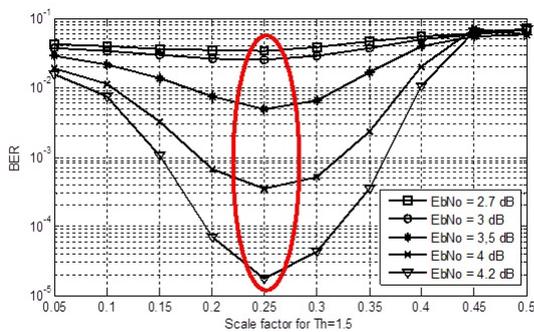


Figure 6. BER (Bit error rate) performance in function of the scale factor using minsum threshold multi split-row for different SNR (Signal to noise ratio) values.

Figure 7 shows the error performance results for a (6,18) (1536,1152) LDPC code for different decoding algorithms (minsum normalized, minsum multi split-row and minsum threshold multi split-row). The code rate is  $R=0.75$  and the maximum number of iteration is set to  $I_{max} = 7$ , with threshold values obtained in Fig.3 and scale factors obtained in Fig.4, Fig.5 and Fig.6.

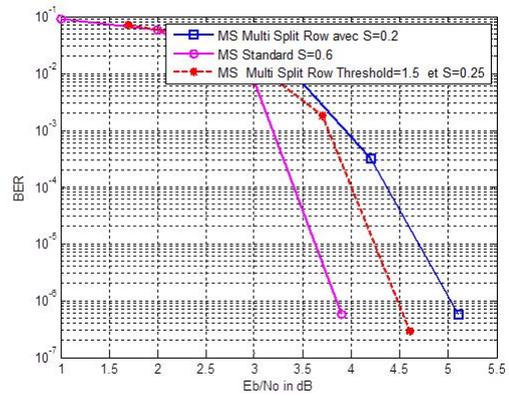


Figure 7. BER performance of (6,18) (1536,1152) irregular code using various decoding algorithms.

As shown in the figure, minsum threshold multi split-row with optimal threshold  $T = 1.5$  performs about 0.6 dB better than minsum multi split-row and is only 0.7 dB away from MinSum normalized at  $BER = 6.10^{-7}$ .

The minsum threshold multi split-row algorithm utilizes a threshold-enable signal to compensate for the loss of  $\min()$  interpretation information in multi-split-row algorithm. It provides at least 0.6 dB error performance over the multi-split-row algorithm with  $Spn = 4$ . Partitioning the parity matrix in  $Spn$  blocs allows us to increase the decoder's parallelism. Therefore, the partitions are performed simultaneously and we obtained a parallel decoder. That means, the complexity decreases and the decoding becomes faster.

As shown in Figure 7, the better performance gap indicated between the minsum threshold multi split-row and minsum multi split-row algorithm revert to the information passed from one partition to another (*Threshold\_ensp*) in algorithm threshold multi split-row, which reduces difference between the local minimum in each partition and the first and second global minimums.

## 4 Conclusion

In this paper we have extended threshold multi split-row decoding method used for regular LDPC code to the irregular LDPC code. The aims was to improve the errors performances of the decoder. Simulation results show that the threshold multi split-row outperforms the multi split-row algorithm for 0.6 dB while maintaining the same level of complexity (only the comparison between the threshold and the first and second minimums is added). Our simulation results show that for a given LDPC code keeping threshold  $T$  constant at any SNR does not cause any error performance degradation.

## References

1. R. G. Gallager, "Low-Density-Parity-Check," IRE Trans. Information Theory, vol. II, no. 8, pp. 21-28, Jan 1962.
2. D. MacKay et R. M. Neal, "Near Shannon Limit Performance of Low-Density-Parity-Check Codes," Electron. Lett., vol. 32, pp. 1645-1946, Aug. 1996

3. D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, p. 399-432, Mar. 1999.
4. "IEEE 802.3an, standard for information technology-telecommunications and information exchange between systems".
5. Cohen, A.E., Parhi and K.K, "A Low-Complexity Hybrid LDPC Code Encoder for IEEE 802.3an (10GBase-T) Ethernet," *IEEE Transactions on Signal Processing*, vol. 57, no. 10, pp. 4085-4094, 2009.
6. "T.T.S.I. digital video broadcasting (DVB) second generation framing structure for broadband satellite applications," [Online].
7. S.-M. Kim, C.-S. Park and S.-Y. Hwang, "A Novel Partially Parallel Architecture for High-Throughput LDPC Decoder for DVB-S2," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 2, pp. 820-825, May 2010.
8. Y.-L. Wang, Y.-L. Ueng, C.-L. Peng and C.-J. Yang, "A Low-Complexity LDPC Decoder Architecture for WiMAX Applications," in *Proc. of the Int. Symp on VLSI Design, Automation and Test, Hsinchu, April 2011*.
9. K.-W. Shin and H.-J. Kim, "A Multi-Mode LDPC Decoder for IEEE 802.16e Mobile WiMAX," *Journal of Semiconductor Technology and Science*, vol. 12, no. 1, pp. 24-33, Mar. 2012.
10. R. ElAlami and M. Mrabti, "Reduced Complexity of Decoding Algorithm for Irregular LDPC Codes Using a Multiple Split Row Method," *International Journal of Research and Reviews in Mechatronic Design and Simulation*, vol. 1, no. 1, pp. 12-17, March 2011.
11. R. ElAlami, H. Qjidaa, Y. Mehdaoui and M. Mrabti, "A Thresholding Algorithm for Improved Split-Row Decoding Method of Irregular LDPC Codes," in *Proc. of the The IEEE International Conference on Intelligent Systems and Computer Vision 2015, Fez, Morocco, March 2015*.
12. M. Fossorier, M. Mihaljevic and H. Imai, "Reduced Complexity Iterative Decoding of Low-Density Parity-Check Codes Based on Belief Propagation," *IEEE Transactions on Communications*, vol. 47, no. 5, pp. 673-680, May 1999.
13. Mohsenin T, Baas B. "Split-row: a reduced complexity, high throughput LDPC decoder architecture," *Proceedings of the ICCD*, October. 2006.
14. J. Chen and M. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Transaction Communications*, vol. 50, pp. 406-414, Mar. 2002.
15. J. Chen, A. Dholakia, E. Eleftheriou, and M. Fossorier, "Reduced complexity decoding of LDPC codes," *IEEE Transaction Communications*, vol. 53, pp. 1288-1299, Aug. 2005.
16. R. ElAlami, M. Mrabti and C. B. Gueye, "Reduced Complexity of Decoding Algorithm for Irregular LDPC Codes Using a Split Row Method," *Journal of Wireless Networking and Communications*, vol. 4, no. 2, pp. 29-34, 2012.
17. T. Mohsenin, D. Truong, and B. Baas, "Multi-split-row threshold decoding implementations for LDPC codes," in *Proc. ISCAS*, May 2009, pp. 2449-2452.