An Improved Split-Row Threshold Decoding Algorithm for LDPC Codes

Tinoosh Mohsenin, Dean Truong and Bevan Baas

ECE Department, University of California, Davis

***Abstract*— We present an improved thresholding LDPC decoding algorithm which outperforms the Split-Row and original Split-Row Threshold decoders with a small increase in hardware. Simulation results show that the algorithm provides 0.27–0.50 dB coding gain over Split-Row, 0.10–0.20 dB over Split-Row Threshold, and is within 0.08–0.13 dB of SPA. Compared with the original Threshold algorithm the check node processor's gate count is increased by 3% while total chip area is kept the same.**

***Index Terms*— low density parity check, LDPC, iterative decoder, VLSI, CMOS, 10GBASE-T, reduced complexity decoder, interconnect complexity, Split-Row**

I. Introduction

Low density parity check codes have been shown to perform very close to the Shannon limit when decoded iteratively [1], [2]. Thus, the codes have been considered by many recent communication standards such as 10 Gigabit Ethernet (10GBASE-T) [3], digital video broadcasting (DVB-S2) [4], WiMAX (802.16e) [5], WiFi (802.11n) [6] and WPANs (802.15.3c) [7]. Unfortunately, hardware implementation for high throughput LDPC decoders with typical code lengths and row weights suffers from large interconnect complexity and low area utilization, which result in high cost, large power dissipation, and low performance.

A (W_c, W_r) (N, K) regular LDPC code is a block code defined by a binary parity check matrix with code length N, information length K, column weight W_c which is the number of ones per column, and row weight W_r which is the number of ones per row.

LDPC codes are commonly decoded by an iterative message passing algorithm which consists of two sequential operations: row processing or check node update and column processing or variable node update. In row processing, all check nodes receive messages from their assigned variable nodes, perform parity check operations and send the results back to the variable nodes. For column processing, the variable nodes update their estimates of the decoded bits using the information passed from the check nodes, and send new updates back to the check nodes. This process continues iteratively until all parity check equations are checked.

Sum-Product (SPA) [8], MinSum (MS) [9] and MinSum normalized [10] are near-optimum decoding algorithms which are widely used in LDPC decoders. These algorithms are proposed for different check node processing methods but use the same variable node update.

The major drawback of standard decoding algorithms is that they require communication between a check node and its assigned variable nodes for a single check node update. This communication leads to a greater global interconnect complexity for large row weight codes ($W_r \geq 16$). Considering the fact that even if a decoding message is represented by a few bits (e.g. 6 bit), the interconnect complexity will sharply increase with every additional row weight resulting in larger and slower circuits [11], [12].

The recently proposed Split-Row decoding algorithm provides significant improvements in the throughput, hardware cost and energy efficiency when compared to existing soft decision decoding algorithms at the cost of some error performance loss [13], [14]. In this paper we propose Split-Row Threshold Improved which outperforms the Split-Row algorithm while maintaining the same level of complexity.

The paper is organized as follows: Section 2 reviews Standard MinSum and MinSum Split-Row decoding algorithms. In Section 3, Split-Row Threshold Improved is introduced. Further analysis of threshold and the error performance comparisons for different codes with Split-Row Threshold Improved are shown in Section 4. The hardware complexity and synthesis results for check node processors implemented with different algorithms are presented in Section 5.

II. Standard MinSum and Split-Row Decoding Algorithms

A. MinSum Decoding

MinSum (MS) is a reduced complexity decoding algorithm which simplifies the check node processing of the SPA iterative decoder by using a minimum function in the check node operation. In MinSum normalized [10], a correction factor is applied to the check node processing outputs to improve the error performance. In the following equations, α_{ij} is the message from check node i to variable node j, β_{ij} is the message from variable node j to check node i, and λ_i is the information received from the channel. We define $V(i)\backslash j$ as the set of variable nodes connected to check node C_i excluding variable node j. Similarly, we define the $C(i)\backslash j$ as the set of check nodes connected to variable node V_i excluding check node j. The two phases of message passing algorithm steps for MinSum are described as follows:

1) Row processing or check node update:

$$\alpha_{ij} = S_{MS} \times \prod_{j' \in V(i)\backslash j} sign(\beta_{ij'}) \times \min_{j' \in V(i)\backslash j} (|\beta_{ij'}|) \quad (1)$$

where S_{MS} is the correction factor for MinSum normalized. For each row i, the magnitude of α messages can

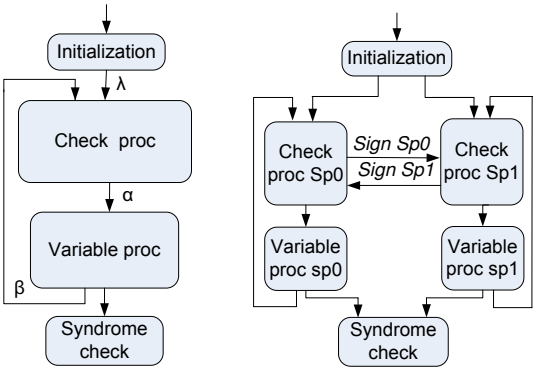

Fig. 1. Block diagram for (a) standard two-phase (b) Split-Row two-phase decoding methods, with passing sign wires between partitions.

also be calculated using the least minimum ($Min1$) and the second least minimum ($Min2$) of the entire set of messages as,

$$Min1_i = \min_{j \in V(i)} (|\beta_{ij}|) \tag{2}$$

$$Min2_i = 2nd \min_{j \in V(i)} (|\beta_{ij}|) \tag{3}$$

Then,

$$|\alpha_{ij}| = \begin{cases} Min2_i, & \text{if } j == Min1\_index \\ Min1_i, & \text{if } j \neq Min1\_index \end{cases} \tag{4}$$

2) Column processing or variable node update:
The MinSum variable node operation is the same as in the standard SPA:

$$\beta_{ij} = \lambda_j + \sum_{i' \in C(j) \setminus i} \alpha_{i'j} \tag{5}$$

B. Split-Row Decoding

The recently proposed Split-Row decoder [13], [14] partitions the check node processing into two or multiple nearly-independent partitions, where each block is simultaneously processed using minimal information from an adjacent partition. The key idea of Split-Row is to reduce communication between check node and variable node processors which is shown to have a major role in the interconnect complexity of existing LDPC decoding implementations.

To illustrate further, Fig. 1 (a) shows the block diagram of a standard two-phase decoder. In Split-Row which is shown in Fig. 1 (b), check node processing is partitioned into two blocks, where each block is simultaneously processed almost independently. With this partitioning, the number of inputs sent to the check processor of each partition is reduced to half which results in less communication between check and variable processors. To improve the error performance the sign computed from each partition is sent to the next partition with a single wire.

Split-Row can be implemented on both MinSum and SPA algorithms. In MinSum Split-Row the check node processing is modified to Eq. 6:

$$\alpha_{ijSplit} = S_{Split} \times \prod_{j' \in V(i) \setminus j} \text{sign}(\beta_{ij'}) \times \min_{j' \in V_{split}(i) \setminus j} (|\beta_{ij'}|) \tag{6}$$

The sign bit is computed using the sign bit of all messages across the whole row of the parity check matrix (because we pass the sign to the next partition). However, the magnitude of α messages in each partition is computed by finding the minimum among $W_r/2$ messages within each partition. S_{Split} is a correction factor which normalizes α values in MinSum Split-Row to improve the error performance.

The major drawback of Split-Row is that it suffers from a 0.4–0.7 dB error performance loss (depending on the number of row partitions) compared to MinSum and SPA decoders. The main reason for its error performance degradation is that in MinSum Split-Row each partition has no information about the minimum value of the other partition. Therefore, when the minimum value in one partition is much larger than the global minimum, the α values in that partition which are calculated by its local minimum are all overestimated when compared to those in the other partition. This leads to a possible incorrect estimation of the bits which reside in that partition.

The recently proposed Split-Row Threshold [15] whose block diagram is shown in Fig. 3, improves the error performance of Split-Row by 0.15–0.2 dB through an additional signal passed between each partition. The main drawback of Split-Row Threshold decoding is that it only compares the local least minimum ($Min1$) with the threshold. To correctly compute the check node messages, both $Min1$ and $Min2$ of the entire set in each partition must be compared with the threshold.

To illustrate the errors caused by the various Split-Row decoding algorithms, Fig. 2 shows the first three rows of the parity check matrix for an LDPC code with $W_r = 4$, and $N = 12$ along with the check node messages (α) updated after the first iteration using (b) MinSum (c) MinSum Split-Row and (d) MinSum Split-Row Threshold. The entries with the largest deviations from MinSum are underlined. For example, in the second row of the Split-Row matrix in (c), the local minimum ($Min1$) in the right side is '2', which is 20 times larger than the local minimum in the left side, '0.1', which is also the global minimum of the entire row. This results in an overestimation of α values for the bits on the right side of the second row, possibly causing an incorrect decision for these two bits. Although Split-Row Threshold in (d) can correct these errors, it cannot correct the major errors in the first and third rows, since it does not compare $Min2$ with threshold T.

III. Proposed MinSum Split-Row Threshold Improved

The proposed Improved Threshold algorithm further improves the error performance while adding negligible hardware to the check node processor, and without the need of any additional global wires to the Split-Row Threshold decoder. In the algorithm, both $Min1$ and $Min2$ are compared with a given threshold (T). Based on this, four conditions will occur: **Condition 1** occurs when both $Min1$ and $Min2$ are less than threshold T, thus they are used to calculate α messages according to Eq. 4. Additionally, a signal ($Threshold\_en$) which goes to the next partition is asserted high, indicating that the least minimum ($Min1$) in this partition is smaller

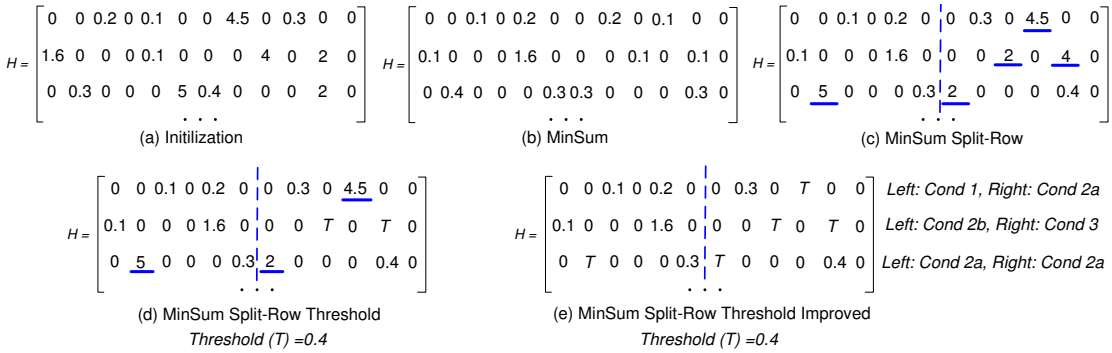

Fig. 2. (a) The first three rows of the parity check matrix for an LDPC code with $W_r = 4$, and $N = 12$, initialized with input values (λ). The α values after the first iteration using (b) MinSum, (c) MinSum Split-Row, (d) MinSum Split-Row Threshold and (e) MinSum Split-Row Threshold Improved

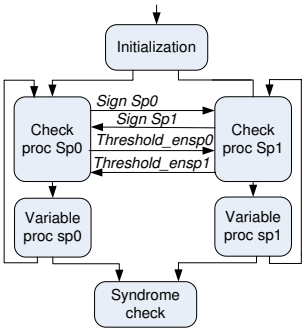

Fig. 3. Split-Row Threshold block diagram

than T. **Condition 2** occurs when only $Min1$ is less than T. Again, the $Threshold\_en$ signal going to the next partition is asserted high. If the $Threshold\_en$ signal from the neighboring partition is also high, indicating that local minimum in the other partition is less than T, then we use $Min1$ and T to update the α messages, while using Eq. 4, in place of $Min1$ and $Min2$, respectively. Otherwise, use Eq. 4 as is. **Condition 3** occurs when the local $Min1$ is larger than T and the $Threshold\_en$ signal from the neighboring partition is high; thus, we only use T to compute all α messages for the partition. **Condition 4** occurs when the local $Min1$ is larger than T and $Threshold\_en$ signal from neighboring partition is low; thus, we again use Eq. 4 unchanged.

The Improved Threshold algorithm for check node update in each row of the $Sp0$ partition is summarized as:
Compute $Min1$ and $Min2$ for the entire $W_r/2$ messages:
Condition 1: if $(Min1 \leq T \;\&\; Min2 \leq T)$
$Threshold\_ensp0 = 1$,
Propagate $Min1$ and $Min2$ to V_j, $j = 1, ..., W_r/2$.
Condition 2: else if $(Min1 \leq T \;\&\; Min2 > T)$
$Threshold\_ensp0 = 1$,
if $(Threshold\_ensp1 == 1)$
Propagate $Min1$ and T to V_j, $j = 1, ..., W_r/2$.
else
Propagate $Min1$ and $Min2$ to V_j, $j = 1, ..., W_r/2$.
Condition 3: else if $(Min1 > T \;\&$
$Threshold\_ensp1 == 1)$
$Threshold\_ensp0 = 0$,
Propagate T to V_j, $j = 1, ...W_r/2$.
Condition 4: else
$Threshold\_ensp0 = 0$,
Propagate $Min1$ and $Min2$ to V_j, $j = 1, ..., W_r/2$.

Note that the variable node operation in Split-Row Threshold Improved is identical to the MinSum and Split-Row algorithms.

The updated check node messages (α) after one iteration using Split-Row Threshold Improved are shown in Fig. 2 (e). Condition 1 and Condition 4 lead to the original MinSum Split-Row decoding, while Condition 2 corrects the large errors left uncorrected by the original Threshold algorithm. Note that this is a simplistic representation of the improvement that the Threshold Improved decoding will have over Split-Row and Split-Row Threshold decoding methods for larger parity check matrices.

IV. Analysis and Impact of Threshold Selection

A. Impact of SNR and Decoding Iterations on Threshold

Clearly, the error performance highly depends on the choice of threshold values. If the threshold T is chosen to be very large, then most local $Min1$ and $Min2$ will be smaller than T. This causes only Condition 1 to be met, which results in the algorithm behaving just as the original MinSum Split-Row. On the other hand if the threshold value is very small, the local minimums will be generally larger than T and only Condition 4 is met which is again the original Split-Row algorithm.

To further illustrate the impact on error performance when choosing the optimal threshold value, Fig. 4 plots the error performance of a (6,32) (2048,1723) RS-based LDPC code [16], which has been adopted for the 10 Gigabit Ethernet (10GBASE-T) standard [3] versus threshold values for different SNRs. As shown in the figure, there are two limits for the threshold value which lead the algorithm to converge to the error performance of the original Split-Row. Also shown is the optimum value for the threshold ($T = 0.2$) in which the algorithm performs best.

In addition, to investigate the impact of decoding iterations on choosing the optimum threshold, Fig. 5 shows the error performance of the same code versus threshold values for different decoding iterations at SNR=4.0 dB, with optimum threshold value of 0.2. Thus, one of the benefits of the MinSum Split-Row Threshold Improved method is that the threshold T is not only independent of the SNR and channel statistics, but is also independent of the number of decoding iterations.

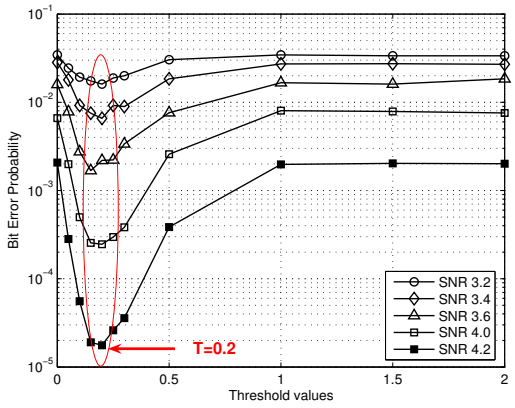

Fig. 4. BER versus threshold for (6,32) (2048,1723) LDPC code for different SNR values with 15 decoding iterations

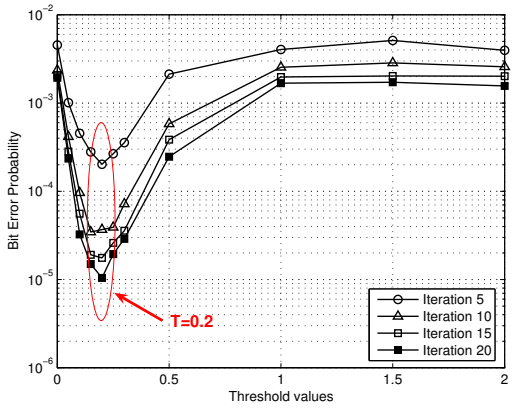

Fig. 5. BER versus threshold for (6,32) (2048,1723) LDPC code for different iterations at SNR=4.0 dB

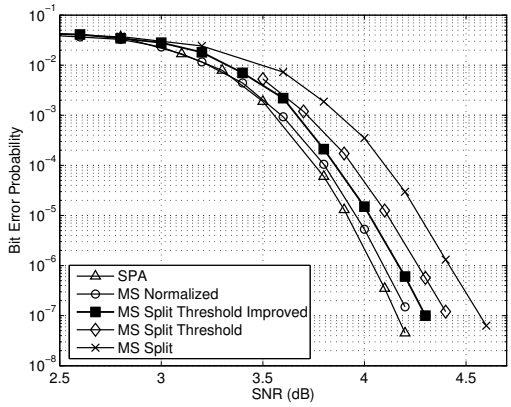

Fig. 6. Error performance results for (6,32) (2048,1723) LDPC code using various decoding algorithms.

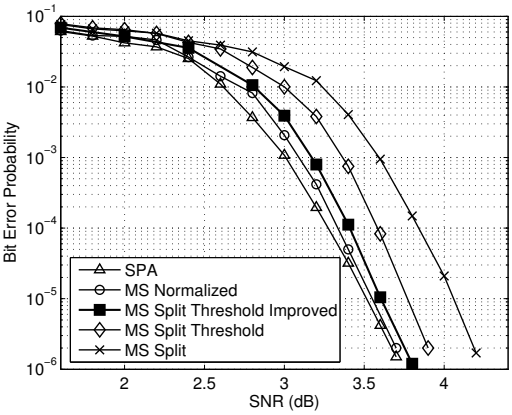

Fig. 7. Error performance results for (4,16) (1536,1155) LDPC code using various decoding algorithms.

B. Error Performance Results

The error performance simulations presented here assume an additive white Gaussian noise channel with BPSK modulation. Simulations were made for 80 error blocks and with either a maximum of 15 decoding iterations or earlier when the decoder converged.

Figure 6 shows the error performance results for a (6,32) (2048,1723) LDPC code for SPA, MinSum normalized, MinSum Split-Row, MinSum Split-Row Threshold, and MinSum Split-Row Threshold Improved with optimal threshold value $T = 0.2$. The threshold is fixed over different SNRs and different decoding iterations. As shown in the figure, the coding gain of MinSum Split-Row Threshold Improved over the original Split-Row is 0.27 dB, and it is only 0.07 dB away from MinSum normalized and 0.13 dB from SPA at BER $= 2 \times 10^{-7}$.

The error performance of a (4,16) (1536,1155) LDPC code is shown in Fig. 7 for different decoding algorithms. As shown in the figure, Split-Row Threshold Improved with optimal threshold $T = 0.3$ performs about 0.5 dB better than original Split-Row and is only 0.04 dB away from MinSum normalized and 0.08 dB away from SPA at BER $= 2 \times 10^{-6}$.

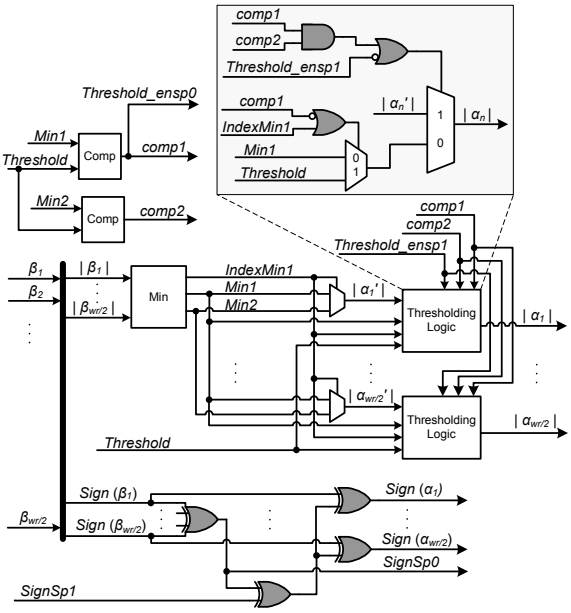

Fig. 8. Check node processor implementation block diagram for partition $Sp0$ using the MinSum Split-Row Threshold Improved method. The threshold logic is shown within the shaded box

V. Hardware Implementation Analysis

The logical implementation of a check node processor in the $Sp0$ partition for our improved threshold algorithm is depicted in Fig. 8. The magnitude update of α is shown along the upper part of the figure while the global sign is determined with the xor logic along the lower part (see Eq. 6). As in MinSum decoding, the first minimum $Min1$ and the second minimum $Min2$ are found alongside the signal $IndexMin1$, which indicates whether $Min1$ or $Min2$ is chosen for a particular α. However, Split-Row Threshold adds a "Thresholding Logic" block for every check node's α output. In addition, two additional comparisons are required between $Min1$ and $Threshold$, and $Min2$ and $Threshold$, which generate the $comp1$ and $comp2$ signals, respectively, which are used by the "Thresholding Logic" block. $Sp0$'s $comp1$ signal is also sent (as $Threshold\_ensp0$) to $Sp0$'s check node processor counterpart in $Sp1$.

The logic of the Improved Split-Row Threshold algorithm (Sec. 3) is encapsulated by the "Thresholding Logic" block (Fig. 8) and only consists of two muxes and their select logic. The final mux selects between α' and the $Threshold—Min1$ mux using: $comp1 \cap comp2$ (Condition 1) $\cup$ $\overline{Threshold\_ensp1}$ (Condition 4 and 2b). The $Threshold—Min1$ mux chooses between Condition 2a and 3.

Table I summarizes the synthesis results for a single check node processor implemented using the original MinSum (MS), MS Split-Row, MS Split-Row Threshold, and (MS) Split-Row Threshold Improved for an LDPC code with row weight $W_r = 32$ in a 65 nm CMOS process using Synopsys Design Compiler. The additional algorithmic complexity of the Improved Thresholding algorithm only incurs a minimal overhead cost of 16% in increased area, 3% in increased gate count and 14% in additional worst case gate delay. Thus, error performance is improved substantially over the original Threshold algorithm at a small increase to check node processor hardware.

A (2048,1723) LDPC decoder chip implemented using Split-Row Threshold Improved algorithm in 65 nm CMOS occupies 8.9 mm^2 and has a throughput of 5.5 Gbps with 15 iterations [17]. This chip is 2 times smaller and 3.3 times faster than MinSum, and has the same area while running at nearly the same clock frequency as MinSum Split-Row.

	Area (μm^2)	Gate Count	Delay (ns)
MinSum (MS)	3578	1018	2
MS Split	1767	541	1.4
MS Split Threshold	1932	586	1.4
MS Split Threshold Improved	2237	604	1.6

TABLE I

Check Node Processor Synthesis Results for Various MinSum Decoding Methods for an LDPC code with Row Weight $W_r = 32$ in 65 *nm* CMOS

VI. Conclusion

In this paper an improved thresholding method for the Split-Row decoding algorithm was proposed, which only requires minor modification to the basic Split-Row architecture. Simulation and synthesis results show that the new method only requires a minor increase in logical complexity and area, while substantially improving the error performance, which can be up to 0.27 dB for a (6,32) (2048,1723) LDPC code in comparison to the Split-Row decoding algorithm. Furthermore, unlike the original Split-Row Threshold, the optimal threshold does not have any dependency on channel statistics and decoding iteration.

VII. Acknowledgments

The authors gratefully acknowledge support from ST Microelectronics, Intel, UC Micro, NSF Grant 0430090 and CAREER Award 0546907, SRC GRC Grant 1598 and CSR Grant 1659, Intellasys, S Machines, and a UCD Faculty Research Grant; LDPC codes and assistance from Shu Lin and Lan Lan; and thank Pascal Urard, Jean-Pierre Schoellkopf and Patrick Cogez.

References

[1] R.G. Gallager, "Low-density parity check codes," *IRE Transaction Info.Theory*, vol. IT-8, pp. 21–28, Jan. 1962.

[2] D.J.C. MacKay and R.M.Neal, "Near shannon limit performance of low-density parity-check codes," *Electronic Letter*, vol. 32, pp. 1635–1646, Aug. 1996.

[3] "IEEE P802.3an, 10GBASE-T task force," `http://www.ieee802.org/3/an`.

[4] "T.T.S.I. digital video broadcasting (DVB) second generation framing structure for broadband satellite applications.," `http://www.dvb.org`.

[5] "IEEE 802.16e. air interface for fixed and mobile broadband wireless access systems. ieee p802.16e/d12 draft, oct 2005.," .

[6] "IEEE 802.11n. wireless lan medium access control and physical layer specifications: Enhancements for higher throughput mar 2006.," .

[7] "Wireless medium access control (MAC) and physical layer (PHY) specifications for high rate wirelss personal area networks (WPANs) 2008.ieee p802.15.3c/d00," .

[8] D.J.MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Transaction Info.Theory*, vol. 45, pp. 399–431, Mar. 1999.

[9] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Transaction Communications*, vol. 47, pp. 673–680, May 1999.

[10] 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.

[11] A. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate 1/2 low-density parity-check code decoder," *JSSC*, vol. 37, no. 3, pp. 404–412, Mar. 2002.

[12] M. Mansour and N.R. Shanbhag, "A 640-Mb/s 2048-bit programmable LDPC decoder chip," *JSSC*, vol. 41, pp. 684–698, Mar. 2006.

[13] T. Mohsenin and B. Baas, "Split-row: A reduced complexity, high throughput LDPC decoder architecture," in *ICCD*, Oct. 2006, pp. 13–16.

[14] T. Mohsenin and B. Baas, "High-throughput LDPC decoders using a multiple split-row method," in *ICASSP*, 2007, vol. 2, pp. 13–16.

[15] T. Mohsenin, P. Urard, and B. Baas, "A thresholding algorithm for improved split-row decoding of LDPC codes," in *ACSSC*, Nov. 2008.

[16] I. Djurdjevic Jun Xu K. Abdel-Ghaffar and Shu Lin, "A class of low-density parity-check codes constructed based on reed-solomon codes with two information symbols," *IEEE Communications Letters*, vol. 7, pp. 317–319, July 2003.

[17] T. Mohsenin, D. Truong, and B. Baas, "Multi-split-row threshold decoding implementations for LDPC codes," in *ISCAS*, May 2009.