LDPC DECODER ARCHITECTURE

Neethu Yesudas*, Anand A**

*II MEECE, Ratnavel Subramaniam College of Engineering and Technology, Dindigul, India

**Assistant Professor-ECE, Ratnavel Subramaniam College of Engineering and Technology, Dindigul, India

*neethuyesudas@hotmail.com,**anandtce@gmail.com

***Abstract*— Low-density parity-check (LDPC) codes have recently emerged as a practicable option for forward error correction (FEC) in future communications systems. In this paper, an LDPC decoding architecture, where the "High throughput LDPC layered decoder architecture" utilizes a split row decoding algorithm along with a block parallel decoding architecture. Split row decoding will increases the throughput without an increase in the complexity. The split row decoding algorithm doubles the row processing, decreases the number of memory accesses per row processor and makes the column processing parallelism easier to exploit. The proposed architecture is simulated for a 12 bit, rate ½ irregular LDPC code on a Xininx sparten-3. The design achieves a throughput of 71.026Mb/s with an iteration of 2.**

Index Terms: LDPC, iterative decoding, , CNBP interconnection complexity, split row.

I. INTRODUCTION

Low Density Parity Check (LDPC) codes, invented by Gllanger in 1962 [2] and rediscovered in middle 1990's, are best known codes that operate near Shannon limit. Compared to Turbo codes, LDPC decoders requires less computational processing and are more suitable for parallelization, low implementation complexity and low latency. LDPC code has been considered in many industrial standards such as WLAN (802.11n), WiMAX (802.16e), DVB-S2, CMMB, and 10BaseT (802.3an) systems. Implementing high throughput and energy efficient LDPC decoder remains a challenging factor due to the high interconnection complexity and high memory bandwidth requirements of the existing decoding algorithm.

An LDPC codes is specified by spares parity check matrix ***H.*** Bipartite graph or also called Tanner graph is used as the graphical representation. There will be two nodes in the graph, check node and variable node. The 1-component in the parity check matrix is associated to edges in the Tanner graph. There are many ways to generate LDPC codes, randomly or by structured. The paper mainly focuses on structured LDPC codes due to practical considerations such as power and area constraints in hardware implementation.

LDPC codes can be decoded by iterative massage passing (Two Phase Massage Passing) algorithm consists of check node update and variable node update schedule. The general algorithms in the decoding are sum of product (SP) algorithm, min-sum (MS) algorithm. The major drawback of standard decoding algorithm is that it requires communication between a check node and its assigned variable nodes for a single check node update. This increases the complexity for large row weight codes. To reduce complexity the two shuffling network in the typical layered decoder is reduced to one and min-sum algorithm is used for further reduction in computational complexity.

In the proposed split row decoder, a reduced complexity method by dividing the row module in to two nearly independent halves. This provides significant improvement in the throughput, wire complexity and energy efficient when compared to existing algorithms.

The paper is organized as follows: Section provides a brief overview of LDPC decoding algorithm and the decoder architecture. Section 3 presents the proposed split-row decoder algorithm for regular LDPC codes. Section 4 shows the implementation result showing the hardware complexity and throughput.

II. LAYERED ARCHITECTURE

LDPC codes are described by M×N binary sparse matrix called parity check matrix **H**. the number of row, M represent the parity check equation for code and the number of columns, N represent the code length. Colum weight W_c is the number of ones per column and row weight W_r is the number of ones per row. An LDPC code is regular (r,c) code if the degree of check node is constant and degree of any variable node is also a constant. Otherwise the code is called irregular code. The number of edges connected to a node is called degree.

LDPC code can also be defined by bipartite graph as shown in figure 1. Each check node C_i correspond to row i in H is connected to variable node V_j corresponding to column j in H. Each variable node corresponds to a received symbol, each check node corresponds to a particular set of parity check constrains, each variable node correspond to a received symbol, and each edge correspond to a nonzero entry in M×N parity-check matrix.

$$\begin{array}{c}
\begin{array}{ccccccccc} V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 & V_8 & V_9 \end{array} \\
\begin{bmatrix}
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\end{array}
\begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{array}$$

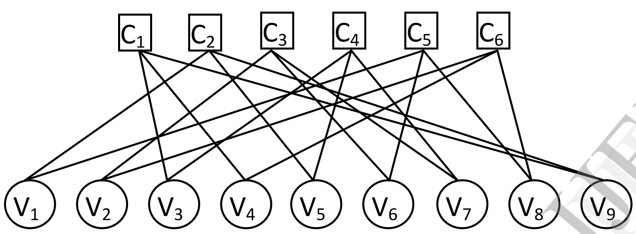


Fig. 1. Parity check matrix and Tanner graph representation of a (Wc = 2, Wr = 3) LDPC code with code length N = 9 bits. Check node Ci represents a parity check constraint in row i and variable node Vj represents bit j in the code.

A. ALGORITHM

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 neighboring variable nodes, perform parity check operations and send the results back to neighboring variable nodes. The variable nodes update soft information associated with the decoded bits using information from check nodes, and then send the updates back to the check nodes, and this process continues iteratively. Sum-Product (SPA) and Min-Sum (MS) are widely-used decoding algorithms which refer to as standard decoders. The following subsections describe these two algorithms in detail.

1. Sum Product Algorithm

Let us assume a binary code word (x_1, x_2, ..., x_N) is transmitted using a binary phase-shift keying (BPSK) modulation. Then the sequence is transmitted over an additive white Gaussian noise (AWGN) channel and the received symbol is (y_1, y_2, ..., y_N).

Let V(i)={j:H_{ij}=1} as set of variable nodes which participate in check equation i. C(j) ={i: H_{ij} =1} denotes the set of check nodes which participate in the variable node j update. Define V(i)\ j as the set of variable nodes connected to check node Ci excluding variable node j. Similarly, C(i)\ j as the set of check nodes connected to variable node Vi excluding check node j. Some important variables using are shown below.

λi: is defined as the information derived from the log likelihood ratio of received symbol yi

αij: is the message from check node i to variable node j. This is the row processing output.

βij: is the message from variable node j to check node i. This is the column processing output.

The SPA decoding can be summarized in to four steps.

1. Initialization: For each i and j, initialize βij to the value of the log-likelihood ratio of the received symbol yj, which is λj. During each iteration, α and β messages are computed and exchanged between variable nodes and check nodes through the graph edges according to the following steps numbered 2–4.

$$\beta_{ij} = \lambda_i \tag{1}$$

$$\lambda_i = \log\left(\frac{P(x_i=0|y_i}{P(x_i=1|y_i}\right) \tag{2}$$

2. Row processing or check node update: Compute αij messages using β messages from all other variable nodes connected to check node Ci, excluding the β information from Vj:

$$\alpha_{ijSPA} = \prod_{j' \in V(i)\backslash j} sign\,(\beta_{ij}) \times \emptyset\left(\sum_{j' \in V(i)\backslash j} \emptyset(|\beta_{ij}|)\right) \tag{3}$$

Where $\emptyset(x) = -\log(tanh\frac{|x|}{2})$

3. Column processing or variable node update: Compute βij messages using channel information *(λj)* and incoming α messages from all other check nodes connected to variable node V_j, excluding check node C_i.

$$\beta_{ij} = \lambda_i + \sum_{i \in C(j)\backslash i} \alpha_{i'jSPA} \quad (4)$$

4. Syndrome check and early termination: When column processing is finished, every bit in column j is updated by adding the channel information (λ_j) and α message from neighboring check nodes.

$$Z_j = \lambda_i + \sum_{i \in C(j)} \alpha_{i'jSPA} \quad (5)$$

From the updated vector, an estimated code vector $\hat{X} = \{\widehat{x_1}, \widehat{x_2}, \dots \dots .., \widehat{x_N}\}$ is calculated by

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

If H. $\hat{X}$=0, then $\hat{X}$ is a valid code word and therefore the iterative process has converged and decoding stops. Otherwise the decoding repeats from step 2 until a valid code word is obtained or the number of iterations reaches a maximum number.

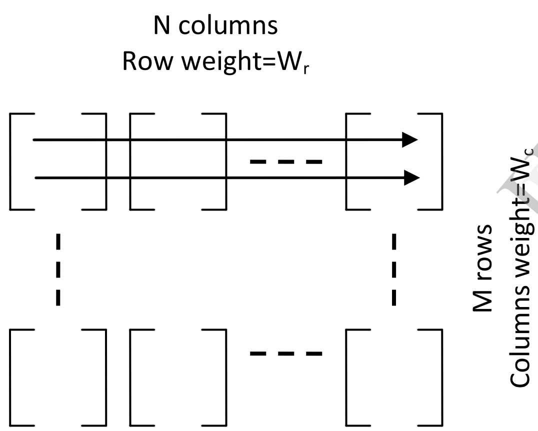


Fig. 2. Parity check matrix of a (*Wc,Wr*) LDPC code with code length*N* for row processing operations using standard decoding.

2. MIN-SUM algorithm

In the SPA, the computational complexity and is very sensitive to finite word length implementation. The magnitude part of check node update in SPA decoding can be simplified by approximating the magnitude computation in the row processing step (Eq. 3), with a minimum function. This algorithm is called Min-Sum (MS) and the row processing output is calculated by:

$$\alpha_{ijMinSum} = \prod_{j' \in V(i)\backslash j} sign\ (\beta_{ij'}) \times \min_{j' \in V(i)\backslash j} (\beta_{ij'}) \quad (7)$$

All other steps are the same as in SPA decoding. The error performance loss of MS decoding can be improved by normalizing the row processor outputs *(α)* in Eq.7 with a correction factor $S \leq 1$, resulting in the Min-Sum normalized algorithm.

$$\alpha_{ijMinSum} = S \times \prod_{j' \in V(i)\backslash j} sign\ (\beta_{ij'}) \times \emptyset\left(\sum_{j' \in V(i)\backslash j} \emptyset(|\beta_{ij'}|)\right) \quad (8)$$

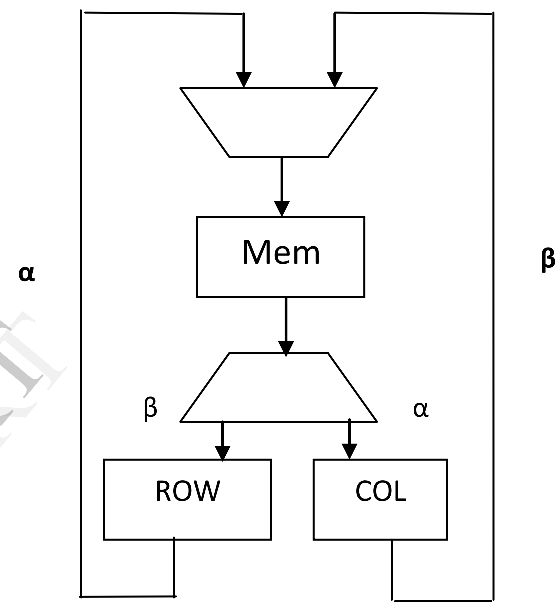


Fig. 3. Block diagram of a typical standard decoder

B. LDPC Decoding architecture

The dataflow of the layered decoder architecture based on the above two characteristics is shown in Figure 3.2. The bit update block is initialized when it received the data. After the initialization, the bit update block works on the updated soft values $P^{(k)}$. The decoder starts the updating of first check to variable massage, α. The shuffle network is an array of cyclic shifters, shuffles the soft values. The shifted soft value, $P^{(1)}$ and the first constituent code α read from memory is used to find the variable to check massage β. The decoding update block calculate the new check to variable massage α' from β. The updated massage is then stored in the memory. The updated posterior messages are computed by adding the recently updated check-to-variable messages to the variable-to-check messages. This updated soft value, $P^{(k+1)}$ is used to compute the next constituent code. Decoding for a constituent code or for the complete H is called one sub-iteration or one iteration, respectively.

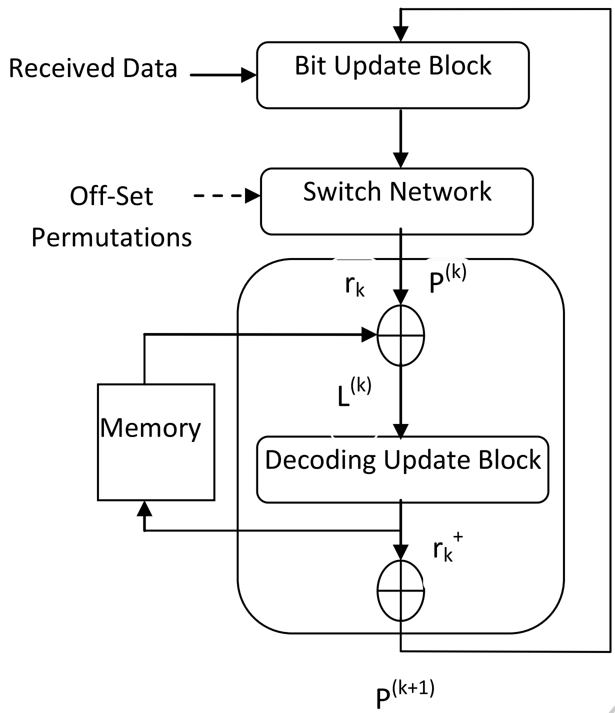


Fig. 4. LDPC layered architecture with offset permutations

In this architecture, all messages can be simultaneously processed in a single clock cycle, which will considerably improve the throughput of the decoder. The block-parallel LDPC decoder mainly consists of two memory blocks for storing messages, check node-based processors (CNBPs) for processing intermediate messages, switching networks (SNs) for routing messages, a parity check module and a decoder control module. The architecture of a CNBP suggests the use of parallel structures for achieving faster decoding on convergence of the layered decoding schedule. Let m(i) (i = 1, 2,…, z) represent the i-th element of each message vector. For each element i of each message vector per row block, the number of inputs in the CNBP depends on the value of dc(maximum check node degree= 3). The CNBP simultaneously processes several block edges adjacent to the M_b th block check node.

A switch network (SN) that implements rotations of the input message vector is available in the Benes network. In the work, 2 × dc SNs are required for switching message outputs from the CNBP to the VN, and for switching messages output from the VN to the CNBP. In addition, a specific memory that is responsible for storing pre-computed routing patterns should be able to provide for different code rates and block sizes.

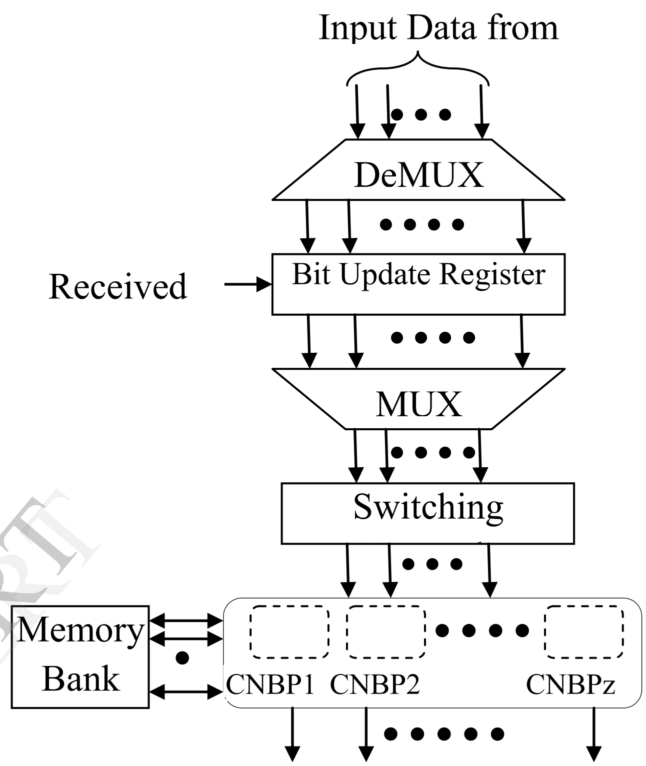


Fig. 5. Block parallel layered decoder architecture

III. PROPOSED SPLIT ROW DECODER

The Split-Row decoding method is proposed to facilitate hardware implementations capable of: high throughput, high hardware efficiency, and high energy efficiency. The row processing stage is divided into two independent halves. This architecture has three major benefits: 1) it doubles parallelism in the row processing stage; 2) it decreases the number of memory accesses per row processor; 3) it makes each row processor simpler. These three factors combine to make row processors (and therefore the entire LDPC decoder) smaller, faster, and more energy efficient. In addition, the Split-Row method makes parallelism in the column processing stage easier to exploit. To reduce performance loss due to errors from this simplification, the sign computed from each row processor is passed to its corresponding "half processor" with a single wire in each direction these are the only wires between the two halves.

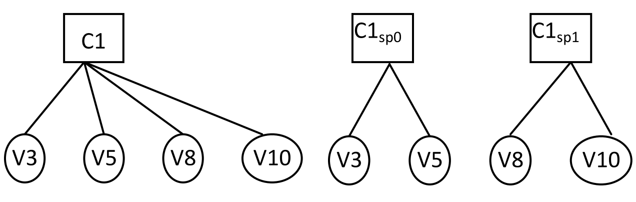
From a mathematical point of view, all steps are similar to the SP algorithm except the row processing step. In each half of the Split-Row decoder's row operation, the parity (sign) bit update is the same as in the SP algorithm. The magnitude part is updated using half of the messages in each row of the parity check matrix. In the Min-Sum split row the check node processing is modified to

$$\alpha_{ijsplit} = S \times \prod_{j' \in V(i) \setminus j} sign\,(\beta_{ij'}) \times \min_{j' \in V_{split}(i) \setminus j} (\beta_{ij'}) \qquad (9)$$

In the Sum Product split row the check node processing is modified to

$$\alpha_{ijsplit} = \prod_{j' \in V(i) \setminus j} sign\,(\beta_{ij}) \times \emptyset\left(\sum_{j' \in V_{Split}(i) \setminus j} \emptyset(|\beta_{ij'}|) \right) \qquad (10)$$

$$\begin{bmatrix} \boxed{0 & 0 & 1 & 0 & 1 & 0} & \boxed{0 & 1 & 0 & 1 & 0 & 0} \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$



(a) (b)

Fig. 6. The parity check matrix example: highlighting the first row processing step using (a) standard decoding (SPA or Min-Sum) and (b) Split-Row decoding.

If the β input messages for a Split-Row decoder and an SPA decoder are the same in a particular decoding step, then,

1. $\alpha ijSPASplit$ and $\alpha ijSPA$ have the same sign, and
2. $|\alpha ijSPASplit| \geq |\alpha ijSPA|$.

Since the sign values are passed between each half, the proof of the first assertion is straightforward. The proof of the second assertion comes from the fact that φ is a positive function and therefore the sum of half of the positive values is less than or equal to the sum of all:

$$\left(\sum_{j' \in V_{Split}(i) \setminus j} \emptyset(|\beta_{ij'}|) \right) \leq \left(\sum_{j' \in V(i) \setminus j} \emptyset(|\beta_{ij'}|) \right) \qquad (11)$$

Also $\emptyset(x)$ is a decreasing function , therefore the following inequality holds:

$$\emptyset\left(\sum_{j' \in V_{Split}(i) \setminus j} \emptyset(|\beta_{ij'}|) \right) \leq \emptyset\left(\sum_{j' \in V(i) \setminus j} \emptyset(|\beta_{ij'}|) \right) \qquad (12)$$

And thus obtain:

$$|\alpha_{ijsplit}| \geq |\alpha_{ijSPA}| \qquad (13)$$

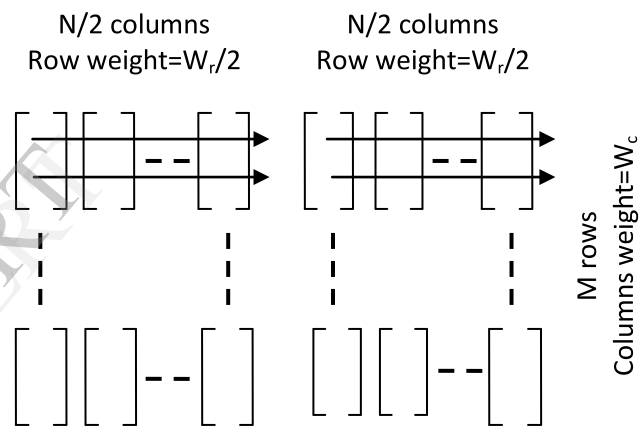


Fig. 7. Parity check matrix for row processing operation with the proposed Split-Row algorithm.

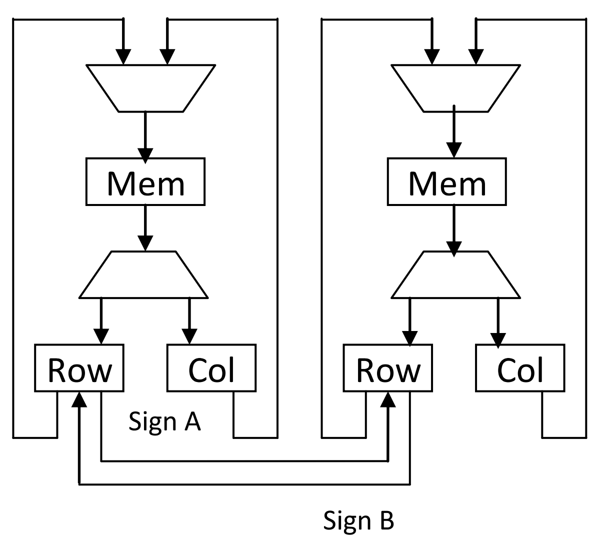


Fig. 8. Block diagram of the proposed Split-Row decoder

IV. IMPLIMENTATION RESULT

The paper design an N=12 , rate=1/2 regular LDPC decoder based on split row decoder algorithm. The simulation is done in modelsim and Xilinx tool is used for synthesis. Spartan 3 is the FPGA used for the implementation process.The throughput can be calculated by the by using the equation

$$\text{Throughput} = \frac{N \times f_{clk} \times R}{N_{clk} \times N_{iter} + N_{latency}} \quad (14)$$

Where f_{clk} is the clock frequency, R is the code rate, N_{clk} is the number of clock cycles for an iteration, N_{iter} is the average number of iterations and $N_{latency}$ is the number of clock ycles due to the pipline latency. In this the throughput is estimated approximatily (580×12×(1/2))/(3×14+7) = 71.026. The synthesis result for both min-sum and split-row decoder is shown below.

TABLE II
Xilinx Spartan3 Xc3s2000 FPGA Synthesis Result

Resource	Min-Sum	Split-Row	Improvement
Slices	1892	1524	19.12%
Slice Flip Flops	359	354	1.39%
4 input LUTs	2426	1986	18.13%
Throughput	61.057	71.026	14%

IV. CONCLUSION

The proposed split-row decoder algorithm is a promising approach for high throughput, low complexity and high speed LDPC decode. Compared to the min-sum decoder algorithm, the throughput is increased by 1.2 times with a degradation in the complexity. The number of iteration for converging is also reduced to 2 from 3.

REFERENCE

[1] Sangmin Kim, Gerald E. Sobelman, and Hanho Lee, “A Reduced-Complexity Architecture forLDPC Layered Decoding Schemes”, IEEE transaction on very large scale integration (VLSI) systems, Vol.19, no. 6,June 2011

[2] R. G. Gallager, Low-Density Parity-Check Codes. Cambridge, MA: MIT Press, 1963.

[3] 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.

[4] T. Zhang and K. K. Parhi, “Joint (3,k)-regular LDPC code and decoder/ encoder design,” IEEE Trans. Signal Process., vol. 52, no. 4, pp. 1065–1079, Apr. 2004.

[5] S. Kim, K. K. Parhi, and R. Liu, “System and method for designing RS-based LDPC code decoder,” U.S. Patent App. US2007-0033484, Feb. 8, 2007.

[6] M. M. Mansour and N. R. Shanbhag, “A 640-Mb/s 2048-bit programmable LDPC decoder chip,” IEEE J. Solid-State Circuits, vol. 41, no. 3, pp. 684–698, Mar. 2006.

[7] T. Mohsenin and B. M. Baas “Split-Row: A Reduced Complexity, High Throughput LDPC Decoder Architecture”, IEEE Trans. Very Large Scale Integr. (VLSI) Syst.,-vol. 17, no. 4, pp. 582–587, Apr. 2006.

[8] Tinoosh Mohsenin, Dean Truong and Bevan Baas ECE Department, University of California, Davis, “An Improved Split-Row Threshold Decoding Algorithm for LDPC Codes”. IEEE Trans. Very Large Scale Integr. (VLSI), Apr. 2010.