# Split-Row: A Reduced Complexity, High Throughput LDPC Decoder Architecture

T. Mohsenin and B. M. Baas

ECE Department, University of California, Davis

***Abstract*— A reduced complexity LDPC decoding method is presented that dramatically reduces wire interconnect complexity, which is a major issue in LDPC decoders. The proposed Split-Row method makes column processing parallelism easier to exploit, doubles available row processor parallelism, and significantly simplifies row processors— which results in smaller area, higher speeds, and lower energy dissipation. Simulation results over an additive white Gaussian channel show that the error performance of high row-weight codes with Split-Row decoding is within 0.3–0.6 dB of the Min-Sum and Sum-Product decoding algorithms. A full parallel decoder for a (3,6) LDPC code with a code length of 1536 bits is implemented in a 0.18 $\mu$m CMOS technology twice: once using the Split-Row method, and once using the Min-Sum algorithm for comparison. The Split-Row decoder operates at 53 MHz and delivers a throughput of 5.4 Gbps with 15 decoding iterations per block. The Split-Row decoder is about 1.3 times smaller, has an average wire length 1.5 times shorter, and has a throughput 1.6 times higher than the Min-Sum decoder.***

## I. Introduction

Low density parity check (LDPC) codes are a class of linear block codes which were first introduced by Gallager in 1963 [1]. Recently, LDPC codes have received a lot of attention because their error performance is very close to the Shannon limit when decoded using iterative methods [2]. 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 (10GBASE-T) [3] and digital video broadcasting (DVB-S2) [4]. Also the next generations of WiFi and WiMAX are considering LDPC codes as part of their error correction systems. Among the greatest barriers to the widespread adoption of high throughput LDPC decoders are their enormous memory bandwidth and interconnect requirements [5], [6], [7]. Previous work has studied efficient implementations of hardware LDPC decoders. *Serial* decoders require less hardware but deliver low decoding throughputs [8]. *Semi-parallel* decoders perform row and column operations partially in parallel [6], [9] and deliver higher throughput than serial decoders. Quasi-Cyclic (QC) LDPC codes [10], [11] are well suited for semi-parallel decoder implementations. In *full parallel* decoders, row and column processors are directly connected to each other according to the Tanner graph [12] of the corresponding parity check matrix [5]. Full parallel decoders can provide very high throughputs while operating at low clock rates. The major challenge in implementing parallel decoders is the high interconnect complexity between row and column processors. There have been studies to reduce the interconnect complexity by using structured LDPC codes [13] or optimizing mapping techniques [14].

In this paper, we propose *Split-Row,* a reduced complexity decoding method which splits each row module into two nearly-independent simplified halves. This method reduces the wire interconnect complexity between row and column processors and increases parallelism in the row processing stage. The Split-Row method also simplifies row processors which results in an overall smaller decoder. We further propose a mapping method to decrease the interconnect complexity in a full parallel Split-Row decoder. The proposed mapping architecture makes the overall decoder yet smaller, which results in higher clock rates and higher energy efficiency than with traditional mapping methods.

This paper is organized as follows: Section 2 provides a brief overview of LDPC codes and their decoding. Section 3 details the Split-Row decoding algorithm. Section 4 presents bit error rate simulation results for the Split-Row decoder. Section 5 proposes a mapping method for a full-parallel decoder with the Split-Row method. Section 6 provides chip design results for a full-parallel decoder chip implementation with Split-Row and Min-Sum algorithms.

## II. Standard Iterative Decoding of LDPC Codes

LDPC codes are defined by an $M \times N$ binary matrix called the parity check matrix $H$. The number of columns, represented by $N$, defines the code length. The number of rows in $H$, represented by $M$, defines the number of parity check equations for the code. Column weight $W_c$ is the number of ones per column and row weight $W_r$ is the number of ones per row. LDPC codes can also be described by a bipartite graph or Tanner graph [12]. The parity check matrix and corresponding Tanner graph of an LDPC code with code length $N = 9$ bits are shown in Fig. 1. Each check node $C_i$ corresponding to row $i$ in $H$ is connected to variable node $V_j$ corresponding to column $j$ in $H$. LDPC codes can be iteratively decoded in different ways depending on the complexity and error performance requirements. Sum-Product (SP) [2] and normalized Min-Sum [15], [16] are near-optimum decoding algorithms which are widely used in LDPC decoders and are known as

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

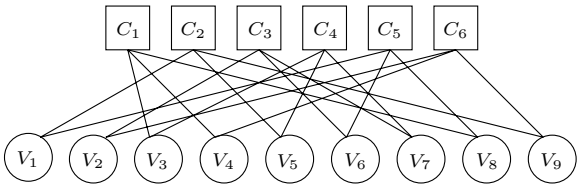


Fig. 1. Parity check matrix and Tanner graph representation of a ($W_c = 2, W_r = 3$) LDPC code with code length $N = 9$ bits. Check node $C_i$ represents a parity check constraint in row $i$ and variable node $V_j$ represents bit $j$ in the code.

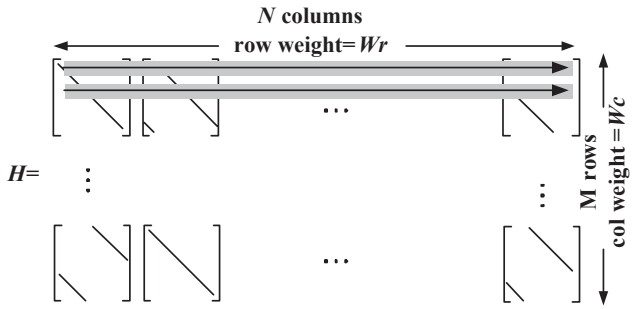


Fig. 2. Parity check matrix of a $(W_c, W_r)$ LDPC code with code length $N$ highlighting row processing operations using standard decoding. For simplicity, a Quasi-Cyclic structure is shown.

*standard* decoders. These algorithms perform row and column operations iteratively using two types of messages: *check node message* $\alpha$ and *variable node message* $\beta$. The parity check matrix and the block diagram of the standard decoding algorithm are shown in Fig. 2 and Fig. 3, respectively.

### A. Sum-Product Decoding

We define $V(i) = \{j : H_{ij} = 1\}$ as the 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. Also $V(i)\backslash j$ denotes all variable nodes in $V(i)$ except node $j$. $C(j)\backslash i$ denotes all check nodes in $C(j)$ except node $i$. In the Sum-Product algorithm during the row processing or *check node update stage,* each check node $C_i$ computes the $\alpha$ message for each variable node $V_j$ based on $\beta$ messages from all other variable nodes $V_{j'}$, $j' \neq j$ which are connected to $C_i$. In this stage, $\alpha$ is computed as follows:

$$\alpha_{ij} = \prod_{j' \in V(i)\backslash j} sign(\beta_{ij'}) \times \phi \left( \sum_{j' \in V(i)\backslash j} \phi(|\beta_{ij'}|) \right) \quad (1)$$

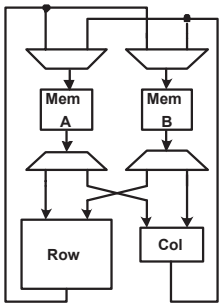


Fig. 3. Block diagram of a typical standard decoder

where,

$$\phi(x) = -\log\left(\tanh\frac{|x|}{2}\right) \quad (2)$$

The first product term in Eq. 1 is called the parity (sign) update and the second product term is the reliability (magnitude) update. In column processing, which is also called the *variable node update stage,* each variable node $V_j$ computes the $\beta$ message for check node $C_i$ by adding the received information from the channel corresponding to column $j$ (called $\lambda$), and $\alpha$ messages from all other check nodes $C_{i'}$, $i' \neq i$ which are connected to $V_j$.

$$\beta_{ij} = \lambda_j + \sum_{i' \in C(j)\backslash i} \alpha_{i'j} \quad (3)$$

### B. Min-Sum Decoding

The check node or row processing stage of SP decoding can be simplified by approximating the magnitude computation in Eq. 1 with a minimum function. The algorithm using this approximation is called Min-Sum (MS) [17], [18]:

$$\alpha_{ij} = \prod_{j' \in V(i)\backslash j} sign(\beta_{ij'}) \times \min_{j' \in V(i)\backslash j} (|\beta_{ij'}|) \quad (4)$$

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. 4 with a scale factor $S \leq 1$ which normalizes the approximations [15], [16].

$$\alpha_{ij} = S \times \prod_{j' \in V(i)\backslash j} sign(\beta_{ij'}) \times \min_{j' \in V(i)\backslash j} (|\beta_{ij'}|) \quad (5)$$

## III. Split-Row LDPC Decoding

The parity check matrix for the row processing stage of the proposed algorithm is shown in Fig. 4. As shown in the figure, 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

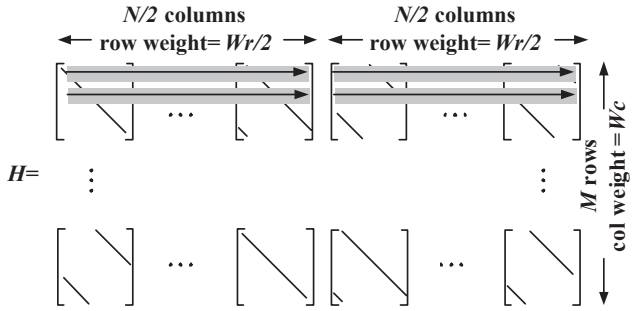


Fig. 4. Parity check matrix highlighting row processing operation with the proposed Split-Row algorithm. For simplicity, a Quasi-Cyclic structure is shown.

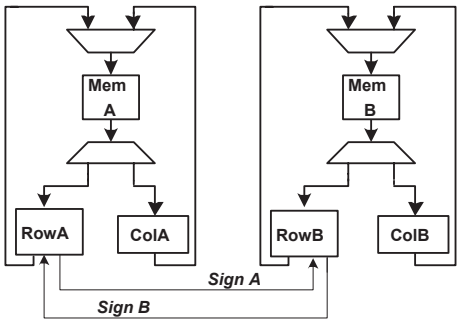


Fig. 5. Block diagram of the proposed Split-Row decoder

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. A block diagram of the Split-Row decoder with two memory blocks is shown in Fig. 5.

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. We denote the parity check matrix $H$ divided into half columnwise by $H_{Split}$. $V_{Split}(i) = \{j : H_{ijSplit} = 1\}$ denotes the set of variable nodes in each half of the parity check matrix which participates in check equation $i$. Therefore, modifying Eq. 1 using half of the messages yields:

$$\alpha_{ijSplit} = \prod_{j' \in V(i)\backslash j} sign(\beta_{ij'}) \times \phi\left(\sum_{j' \in V_{Split}(i)\backslash j} \phi(|\beta_{ij'}|)\right) \quad (6)$$

If the $\beta$ input messages for a Split-Row decoder and an SP decoder are the same in a particular decoding step, $\alpha_{ijSplit}$ and $\alpha_{ij}$ will have the same sign, and $|\alpha_{ijSplit}| \geq |\alpha_{ij}|$. From Eq. 1 and Eq. 6 the proof of the first assertion is clear. The proof of the second assertion comes from the fact that $\phi$ is a positive function and therefore the sum of half of the positive values is less than or equal to all:

$$\sum_{j' \in V_{Split}(i)\backslash j} \phi(|\beta_{ij'}|) \leq \sum_{j' \in V(i)\backslash j} \phi(|\beta_{ij'}|) \quad (7)$$

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

$$\phi\left(\sum_{j' \in V_{Split}(i)\backslash j} \phi(|\beta_{ij'}|)\right) \geq \phi\left(\sum_{j' \in V(i)\backslash j} \phi(|\beta_{ij'}|)\right) \quad (8)$$

Returning to the computation of $\alpha$ in Eq. 1 and Eq. 6, we obtain:

$$|\alpha_{ijSplit}| \geq |\alpha_{ij}|. \quad (9)$$

By normalizing the $\alpha_{ijSplit}$ values with a scale factor $S$ less than one we can improve the error performance of the Split-Row algorithm.

$$\alpha_{ijSplit} = S \times \prod_{j' \in V(i)\backslash j} sign(\beta_{ij'}) \times \phi\left(\sum_{j' \in V_{Split}(i)\backslash j} \phi(|\beta_{ij'}|)\right) \quad (10)$$

Application of the Split-Row method to Min-Sum is equally viable. Similar to the Min-Sum algorithm, the optimal value for $S$ varies over different code rates and SNR values and can be obtained experimentally.

## IV. Error Performance Simulation Results

In this section, the error performance of two regular LDPC codes for the proposed algorithm are presented. The simulations are performed over an additive white Gaussian noise (AWGN) channel with BPSK modulation. The maximum number of iterations is set to $I_{max} = 15$. The following labelings are used for the figures: "MS" for normalized Min-Sum, "MS Split-Row" for the proposed Min-Sum Split-Row algorithm and "$S$" for the scaling factor. Different scaling factors are used to demonstrate the optimal performance. Figure 6 depicts the error performance of a (4,16) Quasi-Cyclic LDPC code with $N = 1536$ bits. As shown in the figure with scaling factor $S = 0.4$, there is about 2 dB gain over the curve with $S = 1.0$. At bit error rate (BER) $= 3 \times 10^{-7}$, the performance gap between Min-Sum and Min-Sum Split-Row $S = 0.4$, is 0.6 dB. Figure 7 shows the error performance of a (6,32) RS-based LDPC code [19], $N = 2048$ bits, adopted by the 10GBase-T Ethernet Standard [3]. At BER $= 3 \times 10^{-7}$, the performance gap between Min-Sum and Min-Sum Split-Row $S = 0.35$ is only 0.3 dB.

### A. Split-Row Decoder Limitation

Error performance simulation results indicate that the Split-Row decoding algorithm performs within a few tenths of a dB from Sum-Product and normalized Min-Sum algorithms for high row-weight ($W_r \geq 10$) LDPC codes at a BER of $10^{-6}$ and lower. However, this method is not as well suited

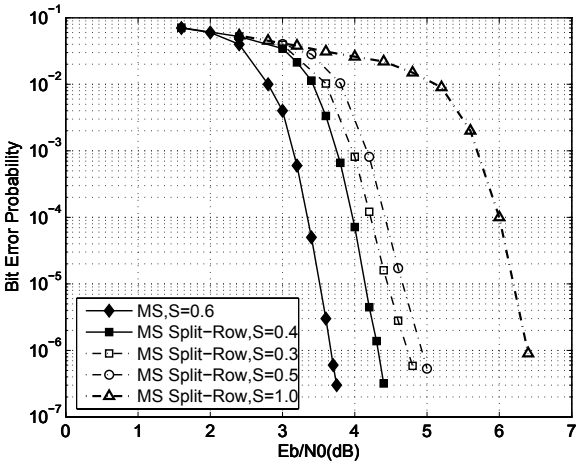


Fig. 6. Error performance of the proposed Min-Sum Split-Row decoder for a $(4, 16)$ QC-LDPC code, $N = 1536$ bits with different scaling factors.

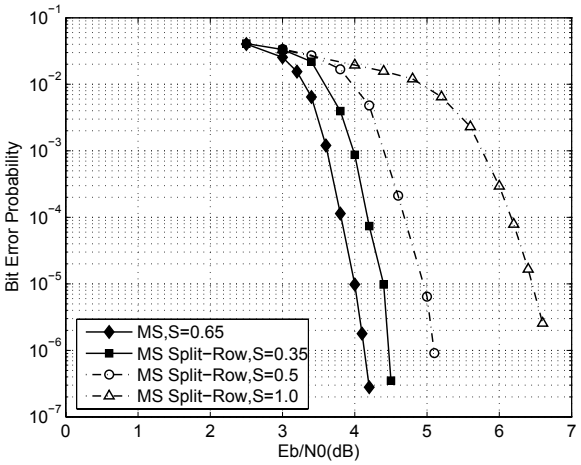


Fig. 7. Error performance of the proposed Split-Row decoder for a $(6, 32)$ RS-based LDPC code, $N = 2048$ bits [19] for different scaling factors.

for low row-weight LDPC codes since the error performance gap for these codes is somewhat larger. The reason for this increased gap is because a low row weight parity check matrix reduces the number of participating variable nodes in each half of the matrix to a small enough number such that check equation computation accuracy is more strongly impacted. For example, a Split-Row decoder for a (3,6) QC-LDPC code, N=1536 experiences approximately 2.0 dB performance loss at the BER of $10^{-6}$.

## V. Full Parallel Decoder Implementation With the Split-Row Algorithm

This section presents the implementation of a full parallel decoder for the proposed Split-Row decoder. Figure 8 shows the block diagram of a full parallel decoder for a $(3, 6)$ LDPC code with the proposed Split-Row decoder. The parity check matrix of the code has 768 rows and 1536 columns. The

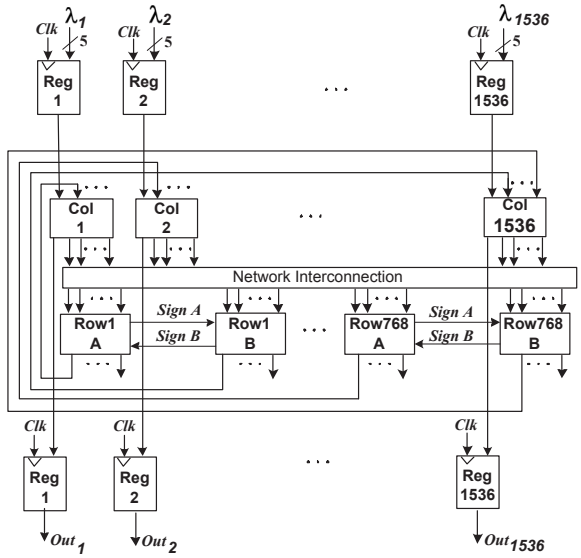


Fig. 8. Full parallel decoder block diagram with the proposed Split-Row method for a $(3, 6)$ LDPC code, $N = 1536$.

number of quantization bits for each message is set to $b = 5$. The row and column processors are connected directly to each other according to the parity check matrix. This connection is shown by the "Network Interconnection" module in the figure. The full parallel decoder takes 1 clock cycle to finish all row and column operations for one iteration. Therefore, the decoding throughput is:

$$Throughput \geq \frac{N}{I_{max}} \times Clock\_Frequency \tag{11}$$

where $I_{max}$ is the maximum number of iterations. The row processor and column processor block diagrams are shown in Figs. 9 (a) and (b), respectively, which are directly implemented from Eq. 10 and Eq. 3.

Figure 10 shows a mapping method of the full parallel decoder with (a) Min-Sum and (b) the proposed Min-Sum Split-Row decoder. As shown in the figure, in the Split-Row decoder, half of the row and column processors can be placed separately and corresponding row processors are connected with two sign wires. A (3,6) QC-LDPC code with code length $N = 1536$ bits is used for the decoder implementation. There are 768 row and 1536 column processors in Min-Sum and 768 row and 768 column processors in each half of the Min-Sum Split-Row decoder.

Each row processor in the Min-Sum decoder connects to 6 column processors while each row processor in the Split-Row decoder connects to 3 column processors. The main advantage of the Split-Row method derives from the fact that it provides significant reductions in circuit area and wire area (length), which results in reduced total area and therefore increased speed and energy efficiency. Its reduction in interconnect complexity enables a compounding benefit with an increase in circuit area utilization.

Both Min-Sum and Min-Sum Split-Row decoders are imple-

TABLE I

COMPARISON OF VLSI IMPLEMENTATIONS IN 0.18 $\mu$m CMOS FOR TRADITIONAL MIN-SUM AND THE PROPOSED MIN-SUM SPLIT-ROW

| Decoder design | Core utilization (%) | Chip size ($mm^2$) | Average wire length (mm) | Maximum clock frequency (MHz) | Decoding throughput (Gbps) | CAD tool P&R run time (minutes) | CAD tool P&R req. RAM (GB) |
|---|---|---|---|---|---|---|---|
| Min-Sum | 40 | 22.08 | 0.224 | 32 | 3.2 | 320 | 3.9 |
| Proposed Min-Sum Split-Row | 50 | 16.80 | 0.142 | 53 | 5.4 | 193 | 2.3 |

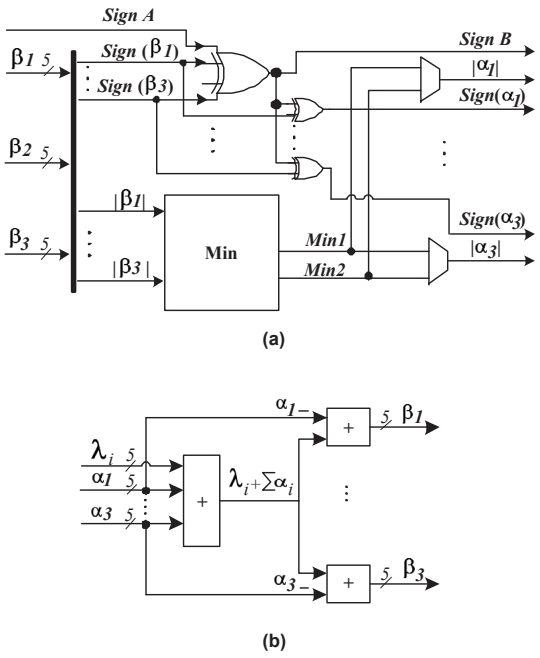


Fig. 9. (a) Row processor and (b) column processor architectures in the Split-Row decoder for a (3,6) LDPC code.

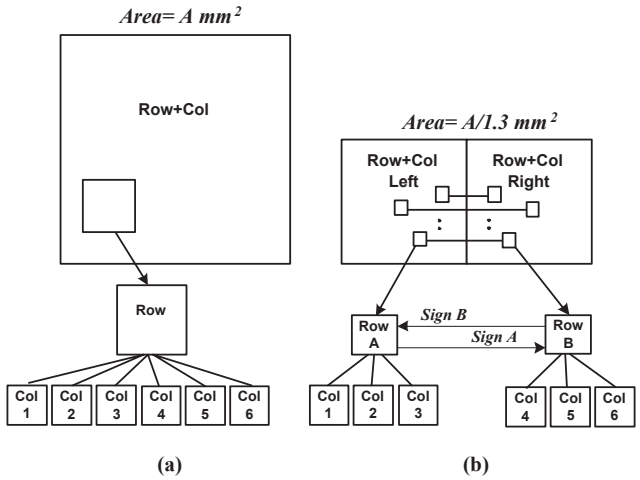


Fig. 10. Mapping of row and column processors for a $W_r = 6$ LDPC code using (a) Min-Sum and (b) Min-Sum Split-Row decoders

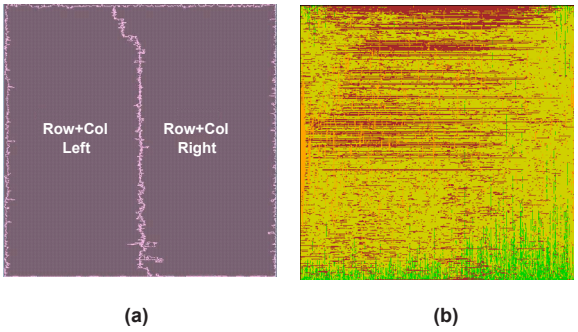


Fig. 11. Split-Row 0.18 $\mu$m CMOS decoder chip (a) floorplan and (b) final layout plot

mented in a 0.18 $\mu$m CMOS technology with 6 metal layers. The netlists were synthesized to a standard cell library from verilog descriptions and the final layout was generated by a place and route (P&R) flow. Each synthesized row processor in the Min-Sum Split-Row decoder is about 2.7 times smaller than the row processors in the Min-Sum decoder, which leads to shorter wires and smaller decoder areas in the Split-Row decoder. Table I summarizes results for the two decoders.

The Min-Sum Split-Row decoder is about 1.3 times smaller than the Min-Sum decoder. The average wire length in the proposed decoder is about 1.5 times shorter than wire length in Min-Sum decoder. Post place-and-route timing analysis indicates that the Split-Row decoder can run at a clock rate of 53 MHz.

With the number of iterations per block set to 15, the Min-Sum Split-Row decoder delivers about 5.4 Gbps decoding throughput, which is about 1.6 times higher than the throughput of the Min-Sum decoder. Figure 11 shows the (a) floorplan, and (b) final layout plot of the chip.

Other major factors for VLSI implementation of large designs are CPU time and memory requirements for the place and route process. As shown in the table, the Split-Row CPU time and memory were 1.65 times shorter and 1.7 times smaller than those for the Min-Sum decoder, respectively. We attempted implementing decoders for the (4,16) $N = 1536$ bits, and (6,32) $N = 2048$ bits LDPC codes; however the CAD tool could not complete the designs because it required more than the 4 GB available memory in our machine.

## VI. CONCLUSION

The proposed Split-Row decoder architecture is a promising approach to tradeoff between hardware complexity and error performance of LDPC decoders. Compared to the near-optimal decoding algorithms Sum-Product and Min-Sum, the error

performance loss of the proposed decoder for high row weight LDPC codes is about 0.3–0.6 dB. The decoder chip implemented with Split-Row has 1.2 times higher core utilization, is 1.3 times smaller, enables 1.5 times shorter wires on average, and delivers 1.6 times higher throughput than the Min-Sum decoder.

## VII. Acknowledgments

The authors thank S. Lin, L. Lan and S. Song for providing parity check matrices of the LDPC codes used in this paper and very helpful discussions; E. Work for enlightening comments in this research; and gratefully acknowledge support from Intel, UC MICRO, NSF Grant No. 0430090, and a UCD Faculty Research Grant.

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

[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] 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, 2002.

[6] M. Mansour and N.R. Shanbhag, "High-throughput LDPC decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, pp. 976–996, Dec. 2003.

[7] E. Yeo, B. Nikolic, and V. Anantharam, "Iterative decoder architectures," *IEEE Communications Magazine*, vol. 41, pp. 132–140, Aug. 2003.

[8] A. Prabhakar and K. Narayanan, "A memory efficient serial LDPC decoder architecture," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, 2005, vol. 5.

[9] H. Zhong and T. Zhang, "Block-LDPC: A practical LDPC coding system design approach," *IEEE Transaction Circuits and Systems I*, vol. 52, pp. 766–775, Apr. 2005.

[10] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near-shannon-limit quasi-cyclic low-density parity-check codes," *IEEE Transaction Communications*, vol. 52, pp. 1038–1042, July 2004.

[11] M. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Transaction Info.Theory*, vol. 50, pp. 1788–1793, Aug. 2004.

[12] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transaction of Information Theory*, vol. 27, pp. 533–547, Sept. 1981.

[13] E.Liao, E. Yeo, and B. Nikolic, "Low-density parity-check code constructions for hardware implementation," in *IEEE International Conference on Communication*, June 2004, vol. 5, pp. 2573–2577.

[14] A. Darabiha, A.C. Carusone, and F.R. Kschischang, "Multi-gbit/sec low density parity check decoders with reduced interconnect complexity," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, 2005, pp. 5194–5197.

[15] 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.

[16] 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.

[17] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of block and convolutional codes," *IEEE Transaction of Information Theory*, vol. 42, pp. 429–445, Mar. 1996.

[18] 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.

[19] S. Lin and Jr. D.J.Castello, *Error Control Coding*, Prentice Hall, Upper Saddle River, NJ, second edition, 2004.