High Throughput and Energy Efficient LDPC Decoders Using Multi-Split-Row Threshold Method

Tinoosh Mohsenin and Bevan Baas

ECE Department, University of California, Davis

***Abstract*— Low density parity check (LDPC) codes have received significant attention due to their superior error correction performance, and have been considered by emerging communication standards such as 10 Gigabit Ethernet (10GBASE-T), digital video broadcasting (DVB-S2), WiMAX (802.16e), Wi-Fi (802.11n) and WPANs (802.15.3c). Due to the codes' inherently irregular and global communication patterns, high-speed systems that require many processing nodes typically suffer from large wire dominated circuits with low clock rates. The recently introduced Split-Row Threshold decoding algorithms and architectures increase parallelism, significantly reduce wire interconnect complexity, and have a small increase in bit error rate compared to the standard MinSum decoding algorithm. Several Multi-Split-Row Threshold decoders have been implemented in 65 nm CMOS for a (2048,1723) LDPC code compliant with the 10GBASE-T Ethernet standard. The impact of different levels of partitioning on error performance, wire interconnect complexity, decoder area, power dissipation and speed are investigated. A 16-way Split-Row Threshold decoder occupies 3.8 mm^2, runs at 101 MHz, delivers a throughput of 13.8 Gbps at 15 iterations, and dissipates 318 mW at 1.3 V. Compared to a standard MinSum decoder implemented in the same technology and physical design flow, the presented chip is 3.9 times smaller, has a clock rate and throughput 6 times higher, is 4.4 times more energy efficient, and has an error performance degradation of only 0.22 dB.**

I. Introduction

The significant error correction capability of low density parity check (LDPC) codes [1] has led to their adoption by recent communication systems such as 10 Gigabit Ethernet (10GBASE-T) [2], digital video broadcasting (DVB-S2) [3], WiMAX (802.16e) [4], Wi-Fi (802.11n) [5] and 60 GHz WPAN (802.15.3c) [6]. While there has been much research on LDPC decoders, the implementation of high-speed systems that require many processing nodes still remains a challenge mainly due to their high interconnect complexity and large circuit area.

A $(W_c, W_r)(N, K)$ LDPC code is characterized by an $M \times N$ binary matrix which is called the *parity check matrix* or *H matrix*, a column weight W_c, and a row weight W_r. LDPC codes can also be defined by a bipartite graph or Tanner graph consisting of two sets of nodes: M check nodes and N variable nodes.

The common decoding method is the message passing algorithm, which performs iterative check node and variable node message exchanges along the edges of the graph. Although check node and variable node processing steps do not require very sophisticated operations, the major challenge is the wire interconnection between nodes for large codes with large row weights. Thus, even though an inherent parallel decoding realization has the highest theoretical throughput, full-parallel LDPC decoders suffer from low circuit utilization and large circuit area due to their high interconnect complexity and large number of processing nodes [7].

This paper gives an overview of recently proposed Multi-Split-Row Threshold decoding [8], [9], which significantly reduces wire interconnect complexity and considerably improves the error performance compared to non-threshold Multi-Split decoding [10].

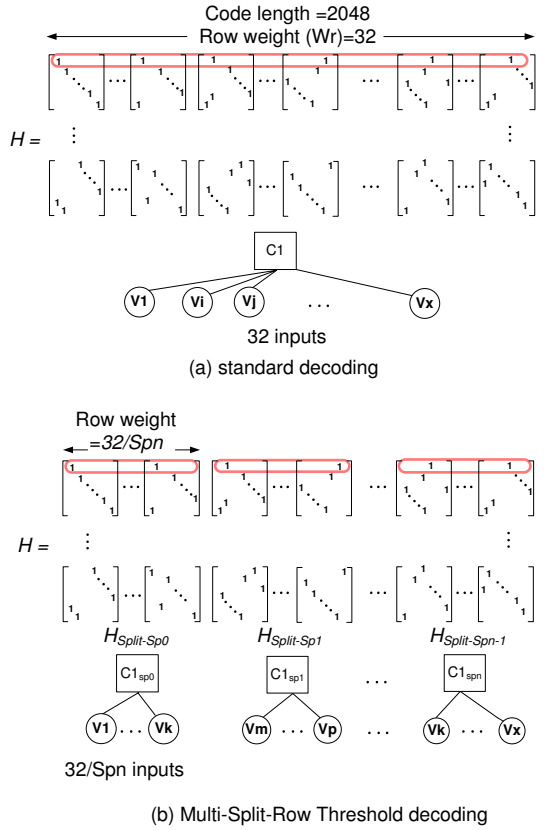

Fig. 1. An example 2048-bit parity check matrix with $W_r = 32$ highlighting the first check node processing step using (a) standard decoding (SPA or MinSum) and (b) Multi-Split-Row Threshold decoding. The check node C_1 and its connected variable nodes are shown for each method.

II. Multi-Split-Row Threshold Decoder Design Goals and Key Features

The key goals of this work are to design a very high throughput and high energy efficiency decoder with small area which is well suited for long codes with large row weights and is easy to be implemented using automatic place and route CAD tools and also has a good error performance. To achieve these goals, we developed a reduced complexity decoding algorithm called the Multi-Split-Row Threshold decoding method which increases parallelism compared to standard decoding methods and significantly reduces the processor and wire interconnect complexity.

To further illustrate, Fig. 1(a) shows the parity check matrix of a 2048-bit LDPC code with row weight of 32, highlighting the first check node (row) processing using a standard decoding method (Sum Product Algorithm (SPA) [11] or MinSum (MS) [12]). As shown in the figure, in standard decoding the information from all variable nodes connected to a check node (32 in Fig. 1) is passed to the check node. The key idea of the Multi-Split-Row Threshold method

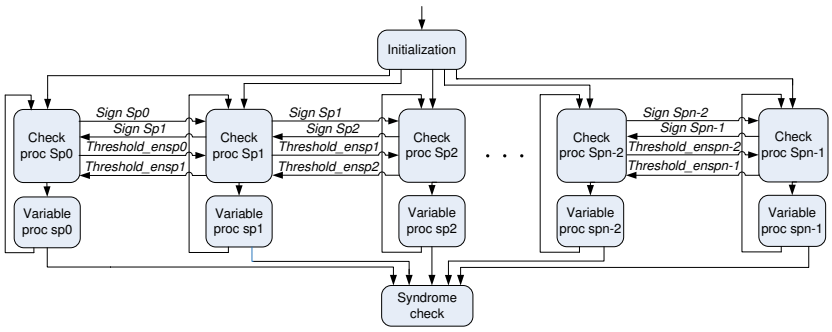

Fig. 2. Block diagram of Multi-Split-Row Threshold decoding with Spn partitions, highlighting *sign* and *threshold_en* signals between partitions.

which is shown in Fig. 1(b) is to partition the check node processing into multiple blocks, where each block is simultaneously processed almost independently. So with this partitioning, the number of inputs to each check node or row processor is reduced which results in less interconnect complexity. In addition each check node processor area is reduced because it requires to process less number of inputs.

More details of Multi-Split-Row Threshold decoding are shown in the block diagram in Fig. 2 with Spn partitions. There are only four inter-partition wires. Two wires are the sign bits calculated in each partition and passed to the next one. The other two wires are the *Threshold_en* signals which are for compensating the minimum between partitions. If a partition has a minimum less than a predefined threshold value T, it asserts the *Threshold_en* signal, otherwise it sets it to zero. This makes all check nodes to take the minimum of their own local min or T. Thus, any large deviations from the true minimum because of the partitioning is reduced. Ideally, the actual minimum value computed in each partition should be sent to all partitions. However, this results in large number of wires passing between partitions. In addition, each check node processor area increases since it must include additional comparison levels.

Eq. 1 and Eq. 2 show the check node processing equations of the MinSum normalized [13] and MinSum Multi-Split-Row Threshold decoding methods. In the following equations, β is the input to check node processing and α is the output of check node processing. $V(i)\backslash j$ is defined as the set of variable nodes connected to check node C_i excluding variable node j. Also, $V_{split}(i)\backslash j$ is defined as the set of variable nodes in partition Spi, which are connected to check node C_i excluding variable node j. In MinSum Multi-Split-Row Threshold 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 the α message in each partition is computed by finding the minimum among the messages within each partition. Then the minimum is each partition is compared against a predefined threshold T. If the *Threshold_en* signal from any of the neighboring partitions is one then the minimum between the local min and T is chosen to compute α messages. Otherwise, the local min is chosen to compute α messages. S_{MS} and S_{Split} are correction factors which normalize α values in MinSum and MinSum Split-Row Threshold to improve the error performance.

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

$$\begin{aligned} \alpha_{ijSplit} &= S_{Split} \times \prod_{j' \in V(i)\backslash j} \text{sign}(\beta_{ij'}) \\ &\quad \times \min \left(\min_{j' \in V_{split}(i)\backslash j} (|\beta_{ij'}|), T \right) \end{aligned} \tag{2}$$

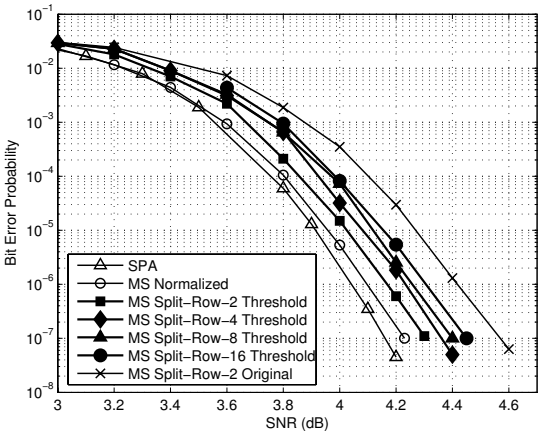

Fig. 3. BER comparison of Multi-Split-Row Threshold Improved with SPA and MinSum Normalized

Our chip implementation results indicate a significant improvement in area, speed, and energy dissipation when using Multi-Split-Row Threshold decoding method for long and large row weight codes.

Multi-Split-Row Threshold partitioning can be arbitrary so long as there are at least two variable nodes per partition. Fig. 3 shows the error performance results for a (6,32) (2048,1723) LDPC code for SPA, MinSum Normalized, and MinSum Split-Row Threshold with different levels of splitting and with optimal correction factors. The error performance simulations 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 fewer when the decoder converges early. As the figure shows, MinSum Split-Row-2 Threshold is about 0.13 dB and 0.07 dB away from SPA and MinSum Normalized, respectively. From Split-2 Threshold through Split-4, Split-8 and Split-16 Threshold the error performance losses are less than 0.05 dB and total loss from Split-Row-16 Threshold to Split-Row-2 Threshold is 0.15 dB at BER $= 10^{-7}$. Also shown in the plot is the Split-Row-2 [14] original algorithm which is 0.12 dB away from Split-Row-16 Threshold algorithm.

III. Multi-Split-Row Threshold Hardware Implementation

A. Check Node Processor Implementation

Fig. 4 shows the sign and magnitude implementations of $Sp1$ check node processor according to Eq. 2. 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. The sign bit calculated from partition $Sp1$ is passed to $Sp0$ to correctly calculate the global sign bit according to the check node processing equation

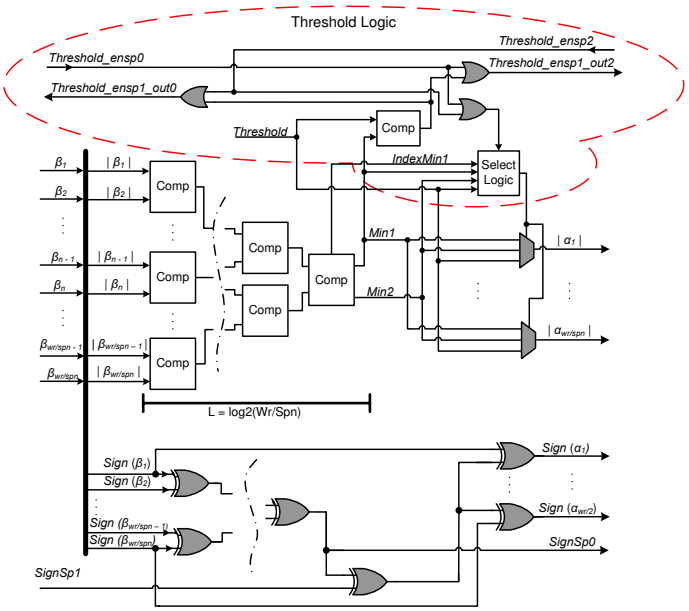

Fig. 4. Check node processor implementation block diagram for partition $Sp1$ using the MinSum Multi-Split-Row Threshold method.

Eq. 2. 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. The thresholding logic is shown within the dashed line, which consists of three OR gates, one comparator and a few logic gates and a mux in the "Select Logic" block. Since there are only W_r/Spn inputs to each check node processor of Multi-Split-Row, the number of comparator stages (L) is reduced to $\log_2(W_r/Spn)$, compared to that of a MinSum decoder which is $\log_2(W_r)$. This results in lower area and smaller critical path delay.

B. Physical Design Flow and Implementation

The left column in Fig. 5 shows the standard-cell-based physical design flow for a single block in a Multi-Split-Row Threshold decoder. The bottom right figure shows the Split-16 Threshold decoder floorplan, highlighting wires passing between the Spi blocks. One of the key benefits of Multi-Split-Row Threshold decoder is that it reduces the time and effort for a full-parallel decoder implementation of large LDPC codes using automatic CAD tools. The major obstacle for building a full-parallel decoder is the high congestion caused by the long wire interconnection between several check nodes and variable nodes in long LDPC codes with large row weights. Due to this congestion, CAD tools become very inefficient in place and route. Since Multi-Split-Row Threshold reduces the interconnection between check nodes and variable nodes per block, then each block can be implemented independently and connected to the neighboring blocks with nearly zero length wires (*sign* and *threshold_en* pins), therefore the physical design flow is greatly simplified.

C. Implementation Results

To further investigate the impact on the hardware implementation due to partitioning, several Multi-Split-Row Threshold full-parallel decoders are implemented for the (6,32) (2048,1723) LDPC code in 65 nm CMOS.

Figure 6 shows the decoder area after synthesis and layout. As shown in the figure, by increasing the number of partitioning the area of Split-Row Threshold decoder is reduced. Notice that for

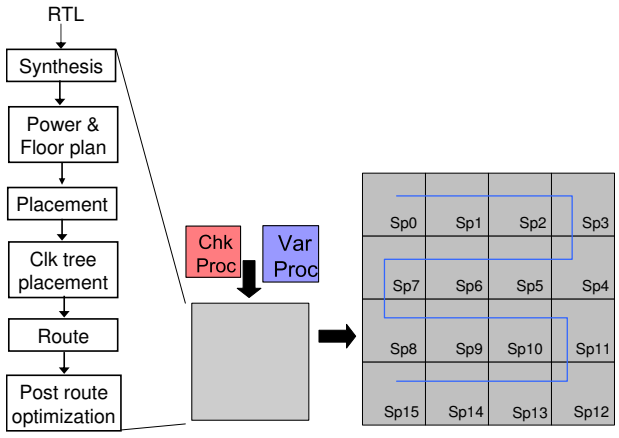

Fig. 5. Standard cell-based physical flow implementation of Multi-Split-Row Threshold decoder

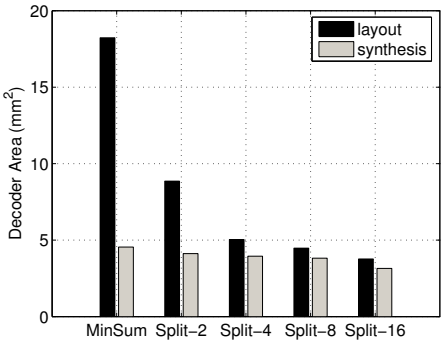

Fig. 6. The core area from synthesis and layout for MinSum Normalized and Multi-Split-Row Threshold decoders. The effect of wire interconnect complexity is shown by the area difference between layout and synthesis results for MinSum Normalized decoder.

the MinSum Normalized decoder the area of the layout deviates significantly from the synthesized area. The reason is because of the inherent interdependence between the large number of check and variable nodes for large row weight LDPC codes, the number of timing critical wires that the automatic place and route tool must constrain becomes an exponentially challenging problem. Typically, the layout algorithm will try to spread standard cells apart because of the increased metal density between gate, drain and source connections of transistors and the upper metal layers. This results in a lower logic (i.e. transistor) utilization and a larger overall area.

Figure 7 shows the layout of five decoders implemented using MinSum Normalized and Multi-Split-Row Threshold decoding with different levels of partitioning in 65 nm, 7 metal layer CMOS.

Table I summarizes the chip implementation results of MinSum and Multi-Split-Row Threshold decoders. For a high Spn such as Split-16 the utilization is 98% which is about 4 times higher, while its average wire length is 7 times shorter than MinSum. It occupies 3.8 mm^2 which is 4.8 times smaller, it runs at 101 MHz and with 15 decoding iterations, it delivers 13.6 Gbps which is 6 times higher and dissipates 23 pJ per bit which is 4.4 times more energy efficient than MinSum. Thus, although Split-16 runs slightly slower than Split-8, it is the smallest decoder—this represents the inflection point along the tradeoff curve between area and speed. Also notice that the time for implementation of all 16 blocks of the Split-16 decoder is much less (about 15 times) when compared

	Normalized MinSum	Split-2 Threshold MinSum	Split-4 Threshold MinSum	Split-8 Threshold MinSum	Split-16 Threshold MinSum
Logic Utilization (%)	25	40	85	95	**98**
Area (mm^2)	18.2	8.9	5	4.5	**3.8**
Avg. wire length per sub-block (μm)	142.5	88.1	59.1	27.1	**20.3**
Worst case speed (MHz)	17	40	53	**112**	**101**
Throughput @ 15 iterations (Gbps)	2.3	5.5	7.2	**15.3**	**13.8**
Energy per bit @ 15 iterations (pJ/bit)	103	87	68	27	**23**
CAD tool CPU time (hour)	78	36	18	10	**5**

TABLE I

COMPARISON OF FULL-PARALLEL DECODERS IN 65 *nm* CMOS, FOR A (6,32) (2048,1723) CODE IMPLEMENTED USING MINSUM NORMALIZED AND MINSUM SPLIT-ROW THRESHOLD WITH DIFFERENT LEVELS OF SPLITTING.

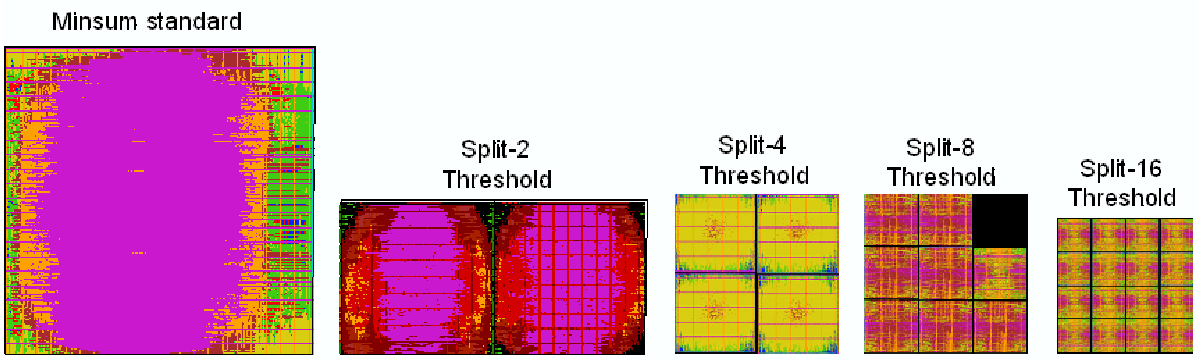

Fig. 7. Scale layout implementations of MinSum and Multi-Split-Row Threshold decoders for the same code and design flow

with the MinSum Normalized decoder. For all five designs, we used the exact same CAD tool automatic place and route flow. But since Multi-Split-Row Threshold reduces check node processor area and eliminates significant communication between check and variable node processors, layout becomes much more compact. As a result, the automatic place and route tool is very efficient.

IV. CONCLUSION

In this work we presented an overview of the recently proposed Multi-Split-Row Threshold decoder, which achieves a better error performance for a chip that has a high level of partitioning when compared to the original Split-Row algorithm. Split-16 Threshold outperforms the original Split-2 by as much as 0.12 dB while being only 0.22 dB away from MinSum Normalized. The Split-16 Threshold decoder achieves an increased throughput of 6 times and a circuit area reduction of 4.8 times compared to the MinSum Normalized decoder. The highest throughput is obtained by a Split-8 implementation with a throughput of at least 15.3 Gbps at 15 iterations. These results demonstrate that LDPC decoders utilizing the Multi-Split-Row Threshold algorithm and architecture can meet the demands of high speed applications while requiring small circuit areas compared to standard decoding methods.

V. ACKNOWLEDGMENTS

The authors gratefully acknowledge support from ST Microelectronics, NSF Grant 0430090 and CAREER Award 0546907, SRC GRC Grant 1598 and CSR Grant 1659, Intel, UC Micro, Intellasys, SEM, and a UCD Faculty Research Grant; and thank Shu Lin and Lan Lan for providing LDPC codes and assistance.

REFERENCES

[1] R.G. Gallager, "Low-density parity check codes," *IRE Transaction Info.Theory*, vol. IT-8, pp. 21–28, Jan. 1962.
[2] "IEEE P802.3an, 10GBASE-T task force," `http://www.ieee802.org/3/an`.
[3] "T.T.S.I. digital video broadcasting (DVB) second generation framing structure for broadband satellite applications.," `http://www.dvb.org`.
[4] "IEEE 802.16e. air interface for fixed and mobile broadband wireless access systems. ieee p802.16e/d12 draft, oct 2005.," .
[5] "IEEE 802.11n. wireless lan medium access control and physical layer specifications: Enhancements for higher throughput mar 2006.," .
[6] "Merged proposal: New phy layer and enhancement of mac for mmwave system proposal. ieee 802.15 wpan millimeter wave alternative phy task group 3c (tg3c), november 2007.," .
[7] 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.
[8] T. Mohsenin, D. Truong, and B. Baas, "An improved Split-Row Threshold decoding algorithm for LDPC codes," in *ICC*, June 2009.
[9] T. Mohsenin, D. Truong, and B. Baas, "Multi-Split-Row Threshold decoding implementations for LDPC codes," in *ISCAS*, May 2009.
[10] T. Mohsenin and B. Baas, "High-throughput LDPC decoders using a multiple Split-Row method," in *ICASSP*, 2007, vol. 2, pp. 13–16.
[11] D.J.MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Transaction Info.Theory*, vol. 45, pp. 399–431, Mar. 1999.
[12] 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.
[13] 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.
[14] T. Mohsenin and B. Baas, "Split-row: A reduced complexity, high throughput LDPC decoder architecture," in *ICCD*, Oct. 2006, pp. 13–16.