Multi-Gbit/sec Low Density Parity Check Decoders with Reduced Interconnect Complexity

Ahmad Darabiha, Anthony Chan Carusone and Frank R. Kschischang
Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto
Email: {ahmadd,tcc}@eecg.utoronto.ca, frank@comm.utoronto.ca

***Abstract*— A 3.2-Gbit/sec 2048-bit parallel LDPC decoder is implemented in a 0.18μm CMOS process. We employ two new techniques to address the interconnect problem: A broadcasting technique reduces the total amount of check-to-variable interconnect wires by more than 40%. A hierarchical placement algorithm places the variable and check nodes in the top-level hierarchy of the design and reduces the maximum wire length by up to 50%.**

I. Introduction

Superior error correction performance and parallelizable decoding algorithms have made Low-Density Parity Check (LDPC) codes [1] a powerful competitor to turbo codes [2] for reliable high speed communication applications such as long-haul optical channels [3] and magnetic storage [4]. Two standards have been recently proposed to adopt LDPC codes: Gigabit Ethernet [5] and Digital Video Broadcast (DVB) satellite communications [6]. In spite of all their desirable properties, one characteristic of LDPC codes, namely their random parity-check matrix, makes implementation of LDPC decoders a difficult task as this leads to complex interconnect wiring and routing congestion for practical codes, and hence to a significantly larger and slower decoder. As an example, 50% of the chip area is unused in [7] because of the routing congestion. In this paper, we introduce two new techniques for the design of the internal architecture of nodes and also for the physical design of the decoder circuit to alleviate the random interconnect problem for fully parallel LDPC decoder implementations. We focus on fully parallel architecture in this paper because it is the only suitable choice for high throughput applications. At the node level, we introduce the *broadcasting* technique that reduces the total amount of top-level check-to-variable interconnect wires by more than 40% and allows the decoder to be implemented in a smaller area and with more relaxed routing requirements. At the physical level, we propose *column reordering* placement algorithm that places variable and check nodes to shorten the longest wires in the post-layout design by up to 50%. These new techniques do not impose any extra hardware cost, nor do they degrade the performance of the error correction.

This paper is organized as follows. In Section II, we briefly review the commonly used iterative message passing algorithm for decoding LDPC codes and discuss previous work in the design of LDPC codes/decoders. In Sections III and IV the broadcasting and column reordering techniques are explained respectively. Section IV provides the results achieved by applying the above techniques in the implementation of a length-2048 parallel LDPC decoder. Finally, Section VI concludes the paper.

II. Background

A. LDPC codes and message passing decoding

Low-density parity check codes are a sub-class of linear block codes that are defined as the null space of a very sparse binary parity check matrix $H_{M\times N}$. LDPC codes can also be represented using a bipartite graph, or Tanner graph, where one set of nodes represents data symbols, also known as variable nodes, and the other set represents parity check constraints. Each edge in the Tanner graph corresponds to a '1' in the parity check matrix H. Fig. 1 shows the Tanner graph for an LDPC code with $N = 10$ variable nodes and $M = 5$ check nodes. This code is called (3,6)-regular LDPC because all the variable nodes participate in a fixed number of checks (i.e. variable degree $d_v = 3$) and each check node is connected to the same number of variable nodes (i.e. check degree $d_c = 6$).

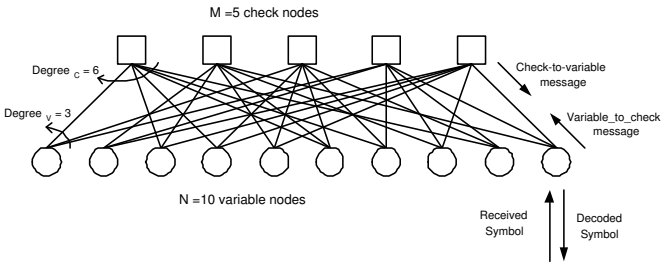

Fig. 1. Tanner graph for a (3-6)-regular LDPC code and information flow for message passing algorithm.

Message passing (MP) is an iterative algorithm commonly used in decoding LDPC codes [8]. Each iteration of MP consists of updating outgoing messages from both variable and check nodes. Each outgoing message is calculated using an update rule applied to all the received messages from all the edges except the edge for which the message is being calculated. Different update rules are used for hard decision or soft decision decoding algorithms. Fig. 2 shows a parallel MP decoder block diagram where for simplicity purposes only one check node and one variable node is present in the diagram. The variable and check update rules in this diagram are shown with $\odot$ and $\boxplus$ respectively. To exclude the effect of each incoming message on its corresponding output, operators $\oslash$ and $\boxminus$ are used that indicate the inverse of the variable and check update rules respectively. For simplicity, the extra blocks required for initializing the variable messages in the first iteration and outputting the variable messages in the last iteration are not present in the diagrams.

B. Previous work

The majority of the previous LDPC code/decoder research has been on new methods of designing serial decoders. These techniques usually generate a "hardware aware" parity-check matrix or adopt a "decoder-first code design" strategy [9] to reduce the required number of clock cycles to complete one decoding iteration [10] [11]. While some of the above techniques introduce a good trade-off between coding performance and hardware cost, they are not usually applicable for high throughput parallel decoder implementations

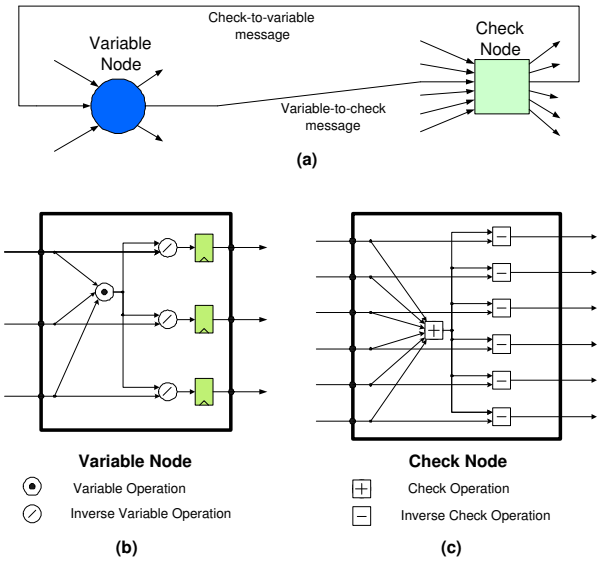

Fig. 2. Original fully parallel LDPC decoding with message passing (a) Global architecture (b) Variable node (c) Check node.

where the critical issues are complex wiring and interconnection delay.

Among the works related to parallel LDPC decoder, in [12] and [13] algorithms are proposed that optimize H matrix attributes such as "cut-size" or "loopiness" to reduce wiring complexity. These techniques, however, do not consider a realistic floorplan (physical arrangement of variable and check nodes on the chip layout) which is usually set at the initial design stages of an integrated circuit. One common problem in the design of practical LDPC codes is that the designer is constrained to using general purpose CAD tools for synthesis and physical design (placement and routing). Performing P&R for an LDPC code of length more than few hundreds is very time consuming and more importantly does not usually converge to a viable decoder both in terms of timing and area. This fact necessitates development of specially tailored CAD tools that are aware of the characteristics of LDPC codes to replace the general purpose tools. As an example, in [7] a special buffer placement strategy has been developed to reduce the routing congestion.

III. BROADCASTING

To mitigate the interconnect problem, we are proposing a scheme which to the best of our knowledge has not been employed in any previous parallel LDPC decoders. This scheme is shown in Fig. 3. The main idea is that we move the inverse check functions (shown with '⊟') from check nodes to inside variable nodes without affecting the functionality of the iterative MP decoding algorithm.

The advantage of this new scheme is that now each check node *broadcasts* one outgoing message to all its adjacent variable nodes. From the hardware implementation point of view, this property allows for sharing a lot of wires that can not be shared in the original scheme where separate messages are sent to individual neighboring variable nodes. Fig. 4 shows an example of broadcasting and how it reduces the total amount of wires.

Broadcasting technique saves a significant amount of interconnect wires without introducing any extra computational hardware cost. It is a new way of partitioning variable and check nodes and shows its effect in hierarchical design methodology where variable and check nodes are synthesized in node level and the global routing is performed in top level.

Fig. 5 shows a zoomed-in portion of the interconnects for a 2048-bit Reed Solomon-based LDPC code [14] where the total length of top-level check-to-variable nets is reduced by more than 40% after applying the broadcast scheme. This figure is generated by Matlab simulation and assumes that wires can be in any arbitrary direction. However, we can observe similar congestion effect in the layouts where only vertical and horizontal wiring is used. We have used a floorplan similar to [7] where check nodes are located in the center of the layout and the variable nodes are surrounding them, however the broadcasting idea can be applied to any arbitrary floorplan.

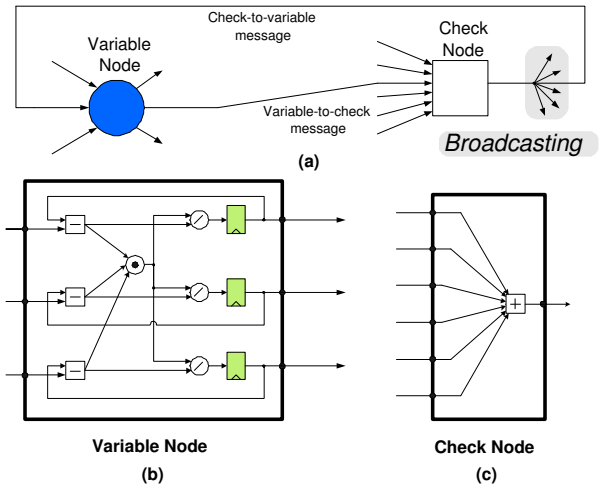

Fig. 3. Broadcast architecture (a) Global architecture (b) Variable node (c) Check node.

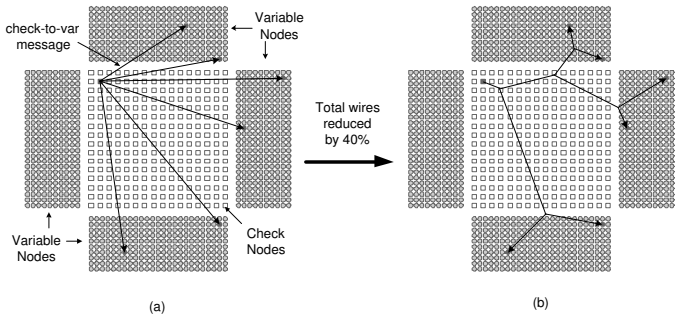

Fig. 4. Broadcasting reduces the total top-level wirelength by sharing the wires. (a) Output messages of a check node without broadcasting (b) Sharing interconnect wires of a check node with broadcasting

IV. COLUMN REORDERING

Using generic CAD tools for designing parallel LDPC decoders usually leads to top level net lengths with a Gaussian-like histogram. The long wires in the tail of the histogram have a major effect on the timing of the circuit because they are the only part of timing paths that differ between different nodes as the logic delays inside variable nodes and check nodes are almost identical among all the nodes. This effect is especially visible for longer codes as both net resistance and net capacitance is proportional to its length. In addition to limiting the timing performance, the switching activity of these long wires strongly influences the power dissipation of the decoder.

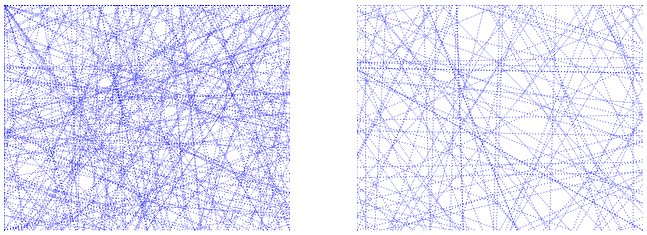

Fig. 5. A small section of interconnects for a length-2048 LDPC code (a) before broadcast (b) after broadcast. There is a 40% reduction in total wirelength.

To address the problem of long wires explained above, we have developed a top-level node placement algorithm, called *column reordering*. The column reordering placement algorithm ensures that the Manhattan distance between those variable and check nodes that are communicating with each other is less than a desired threshold. This leads to a wire length histogram that does not suffer from a long tail and has a maximum length much less than what is achievable with general purpose placement tools.

This algorithm takes two inputs: A parity-check matrix, $H_{M\times N}$, along with a floorplan of nodes. The floorplan is a set of coordinates at which a variable or a check node can be placed. The floorplan is imported to our algorithm in the form of two sets of coordinate vectors $\{v_1, ..., v_N\}$ and $\{c_1, ..., c_M\}$, where v_i is the X-Y coordinate of ith variable *slot* in the grid shown in Fig. 4 and, similarly, c_j is X-Y coordinate of jth check *slot* in that figure. The numbering of variable and check nodes does not affect the performance of our placement algorithm, however for simplicity we have labeled variable slots starting from the top left corner of the left block and moved counter-clockwise. The check slots were numbered starting from the top left corner and going through the rows from left to right.

The output of the algorithm is another parity check matrix, $\hat{H}_{M\times N}$, which is the same as H except its columns are reordered.

As columns in a parity check matrix correspond to the variable nodes, H and $\hat{H}$ both represent the structure of the same LDPC code. In the next few paragraphs, we explain how to generate $\hat{H}$ and where to place its variable or check nodes to reduce the maximum chick-to-variable distance.

The column reordering algorithm consists of the following steps:

1) From the given floorplan vectors $\{v_1, ..., v_N\}$ and $\{c_1, ..., c_M\}$ create a cost map matrix, $D_{M\times N}$, where D_{ij} is the Manhattan distance between c_i and v_j and then find $L_{max} = Max(D_{ij}.H_{ij})$. Fig. 6(a) shows a visualized D matrix in gray scale such that the brighter pixels indicate longer distance.
2) Pick a length reduction ratio, α, less than but close to 1. (The initial value of α does not affect the final result but $\alpha = 0.7$ is a good choice for fast convergence)
3) Create $K_{M\times N}$, where K_{ij} is 0 if $D_{ij} < \alpha \times L_{max}$ and 1 otherwise. Fig. 6(b) shows a black and white K matrix for $\alpha = 0.70$, where white pixels indicate 1's.
4) The 1's in K can be interpreted as locations in which a candidate parity-check matrix should not have any '1' in order to have a maximum var-to-check Manhattan distance of less than $\alpha \times L_{max}$. In the following steps, we reorder the columns of input $H_{M\times N}$ matrix to create $\hat{H}_{M\times N}$ that does not have any '1' in the white regions of K:
   a) Initialize $\hat{H}$ with zeros.
   b) Find K_j, the most constrained column of K, i.e., the column with the largest number of '1's.
   c) Find a column of H that has no '1' in those coordinates where K_j has '1' and set this column as jth column of $\hat{H}$. If there are more than one columns in H meeting this constraint, then pick the column that has maximum inner product with jth column of D. If no column of H satisfies K_j then go to step 2 and choose a larger value for α.
   d) Repeat steps $4(b)$ and $4(c)$ for the remaining columns of D and H matrices until all the columns of D are satisfied and all columns of H are set to a column of $\hat{H}$.
5) Restart from step 2. Each time pick a slightly smaller value of α until a point at which further reduction of α stalls the algorithm in step $4(c)$.

After completing the above algorithm, the value of 1– α represents the maximum possible percentage reduction in the maximum length of top-level wires. Also, matrix $\hat{H}$ is the same as H but only with its columns reordered. Now, the optimum location for each variable and check node of $\hat{H}$ is the same as those slot coordinates given in the floorplan vectors, i.e., the variable node corresponding to the ith column of $\hat{H}$ is placed at coordinate v_i and similarly the check node corresponding to jth row of $\hat{H}$ is placed at coordinate c_j.

The column reordering placement algorithm deals with a high-level abstract of the decoder, and hence is at least one order of magnitude faster than generic placement tools and at the same time generates more satisfying results. Fig. 7 shows the effect of column reordering on the histogram of top-level nets for a length-2048 irregular RS-based LDPC code using the floorplan of Fig. 4 resulting in more than a 30% reduction in the length of the longest wire as it has pushed back the tail of the originally-Gaussian histogram.

The column reordering algorithm can be applied to any arbitrary regular or irregular H matrix and with any desired floorplan. It can also be applied in conjunction with other optimizations in the internal architecture of the nodes or the H matrix design as it does not change the structure of the code.

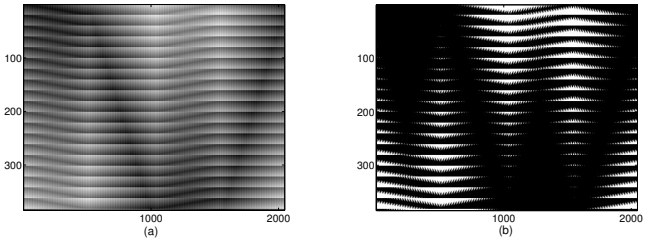

Fig. 6. (a) Cost map matrix (b) Cost map matrix after threshold.

V. Results

To illustrate our proposed techniques, we have implemented a parallel decoder for the (2048,1723) RS-based Gallager (6,32)-regular LDPC code using a hard decision message passing algorithm. This code is chosen because it is one of the only two candidate codes proposed for 10GBase-T Ethernet standard [5]. We have used a bottom-up design methodology where broadcasting is used at the node level and column reordering is used for the placement of the nodes at the top level of the physical design. Figure 8 shows the layout of the decoder chip using a 0.18μm CMOS process with a die size of 4.2 mm $\times$ 4.2 mm. The decoder performs 32 message passing iterations per block and each iteration takes two clock cycles. The maximum clock frequency of the decoder is above 100 MHz and decoder achieves a throughput of 3.2 Gbit/sec. No other parallel

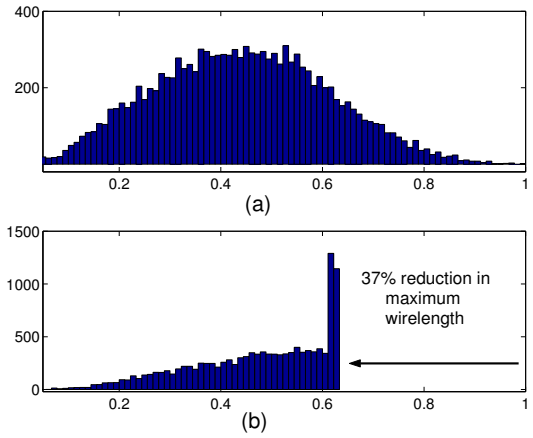

Fig. 7. Top-level net length histogram (a) before and (b) after column reordering.

RS-based LDPC decoder implementation has been reported in the literature. As a comparison with other high throughput parallel LDPC decoders, in [7], a 1024-bit irregular LDPC decoder with 4-bit message passing is described which has a die size of 52.5 mm^2 and operates at 64 MHz and has a throughput of 1 Gbit/sec.

VI. Discussion and Conclusion

Using the broadcasting technique introduced in Section III, we reduced the outgoing messages of check nodes to one common message whereas the variable node outgoing messages are left intact. In fact, we can extend the above broadcasting technique to *full-broadcasting* where the variable node outgoing messages are also reduced to one common message for an even greater savings in interconnect wires. It can be shown that employing full-broadcasting requires additional inverse check and inverse variable functions and it also requires extra storage elements. Depending on the choice of check and variable update functions there may be cases where this hardware overhead negates the interconnect advantage. Based on the above argument and due to our choice of variable and check functions, we have implemented the broadcast scheme as explained in Section III in the LDPC decoder presented in this work.

Similarly, in the column reordering placement algorithm explained in Section IV, the location of check nodes were fixed while the location of variable nodes were optimized. We have also considered other approaches such as fixing the variable nodes and optimizing the location of check nodes (row reordering). Our experience shows that column reordering results are usually slightly superior to those from row reordering because the number of variable nodes are usually more than check nodes, providing the column reordering with a larger search space, hence better solutions. Still, if for any reason the order of variable nodes has to be kept unchanged, then row ordering can replace column reordering. The second option is to optimize the location of both check and variable nodes simultaneously. We did not choose this approach because it makes the processing time impractical and in fact we do not expect to get any significant improvement compared to our column reordering results.

The broadcasting architecture and column reordering algorithm proposed in this paper addressed the interconnect problem in the design of parallel high throughput LDPC decoders. The broadcasting technique reduced check-to-variable wires by more than 40% in our 2048-bit example. In addition, the maximum length of top-level wires were reduced by 37% using the column reordering placement algorithm. We implemented a 3.2 Gbit/sec 2048-bit RS-based LDPC decoder in a 0.18μm CMOS process using the above techniques. The performance of this parallel decoder illustrates the effectiveness of our proposed approaches.

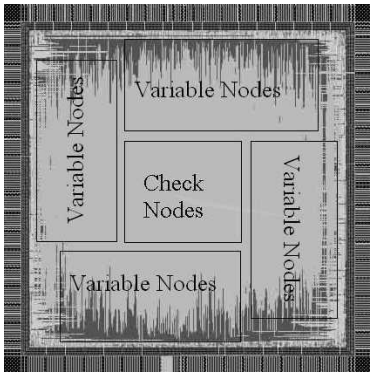

Fig. 8. Layout and floorplan of the 2048-bit LDPC decoder.

Acknowledgment

The authors would like to thank Canadian Microelectronics Corporation for providing the design tools for this project.

References

[1] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT press, 1963.

[2] C. Berrou and A. Glavieux, "Near optimum error correcting coding: Turbo codes," *IEEE Trans. on Communications*, vol. 44, pp. 1261–1271, Oct 1996.

[3] B. Vasic and I. B. Djordjevic, "Low-density parity check codes for long-haul optical communication systems," *IEEE Photonics Technology Letters*, vol. 14, pp. 1208–1210, Aug 2002.

[4] T. Mittelholzer, A. Dholakia, and E. Eleftheriou, "Reduced-complexity decoding of low density parity check codes for generalized partial response channels," *IEEE Trans. on Magnetics*, vol. 37, no. 2, pp. 721–728, March 2001.

[5] IEEE 802.3 10GBase-T Study Group Meeting, World Wide Web, http://www.ieee802.org/3/10GBT/public/jul04/rao-1-0704.pdf, July 2004.

[6] European Telecommunication Standards Institue, World Wide Web, http://www.dvb.org/documents/white-papers/wp06.DVB-S2.final.pdf.

[7] A. J. Blanksby and C. J. Howland, "A 690-mw 1-Gb/s 1024-b, rate-1/2 low-density parity-check decoder," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 3, March 2002.

[8] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. on Information Theory*, vol. 47, pp. 498–519, Feb 2001.

[9] E. Boutillon, J. Castura, and F. R. Kschischang, "Decoder-first code design," in *Proceedings of Turbo Codes*, Brest, France, 2000, pp. 459–462.

[10] M. M. Mansour and N. R. Shanbhag, "High-throughput LDPC decoders," *Trans. on VLSI Systems*, vol. 11, no. 6, Dec 2003.

[11] T. Zhang and K. K. Parhi, "Joint code and decoder design for implementation-oriented (3, k)-regular LDPC codes," in *IEEE Asilomar Conference*, Nov 2001, pp. 1232–1236.

[12] M. Mohiyuddin, A. Prakash, A. Aziz, and W. Wolf, "Synthesizing interconnect-efficient low density parity check codes," in *Design Automation Conference*, June 2004, pp. 488–491.

[13] J. Thorpe, "Design of LDPC graphs for hardware implementation," in *International Symposium on Information Technology*, Lausanne, Switzerland, July 2002.

[14] I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin, "A class of low-density parity-check codes constructed based on Reed-Solomon codes with two information symbols," *IEEE Comm. Letters*, vol. 7, no. 7, July 2003.