International Journal OF Research Science and Engineering 2019

IMPROVED SECURITY AND LOW ERROR RATE IN LDPC CODES USING XOR ENCRYPTION TECHNIQUE

A.Silambarasan[#1], E.Sangeetha, M.E [#2]

[#1]PG scholar & Tagore Institute of Engineering and technology, [#2] Associate Professor & Tagore Institute of Engineering and technology

Department of ECE & Tagore Institute of Engineering and technology, Tamilnadu, India

[1] asilambarasan175@gmail.com, [2] sangeta.r@gmail.com

ABSTRACT

Low-density parity check (LDPC) codes are an attractive error correction scheme for ensuring data integrity in new generation of memories. A quick assessment of the iterative decoders for LDPC codes reveals a wide range of varying complexities. The message mapping in memory banks and the pipeline-related data hazards in low-density parity-check (LDPC) decoders. We believe a layered hardware construction using particular read/single write port memory banks. The throughput of such architecture is limited by memory access conflicts, due to improper message mapping in the memory banks, and by pipeline data hazards, due to delayed update effect. We solve these issues by: 1) a residue-based layered scheduling that reduces the pipeline related hazards for optimizing the message mapping in memory banks and the message read access scheduling. Our estimates for different LDPC codes indicate that the hardware usage efficiency of our layered decoder is improved by 3%–49% when only the XOR encryption algorithms are employed and by 16%–57% when both the residue-based layered architecture is used.

KEYWORDS: Bank allocation, graph coloring, LDPC QC-LDPC, in-place memory mapping, pipeline conflicts.

INTRODUCTION

Low-Density Parity-Check (LDPC) codes and turbo codes are among the known near Shannon limit codes that can achieve very low bit error rates for low signal-to-noise ratio (SNR) applications. When compare to the decoding algorithm of Turbo codes, LDPC decoding algorithm has more parallelization, low accomplishment complexity, low decoding latency, as well as no error-floors at towering SNRs. Low-Density Parity Check (LDPC) codes and Turbo codes are among the best known near Shannon limit codes. LDPC decoding algorithm has more parallelization when compared to the decoding algorithm of Turbo codes. The major issues surrounding the VLSI implementation of LDPC decoders are the complex interconnects and large memory requirements due to the sparse nature of the parity generator matrix. This paper proposes low complexity architecture with reduced memory requirements for LDPC decoding based on the recent work on structured LDPC codes. LDPC codes are considered for virtually all the next generation communication standards. Low Density Parity Check Codes (LDPC) codes which are among the Shannon limit codes have been given intensive attention in recent few years due to their merits in implementing a high throughput, low latency decoder. A suboptimal decoding, Sum of Product (SP), algorithm has been proposed for near Shannon limit performance and its approximate version, Offset Min- Sum (MS), algorithm has also been proposed. Offset MS reduces the complexity of the decoding by removing the non-linear operations needed in SP.

The LDPC codes are usually decoded by some iterative message-passing algorithms. There are two types of message scheduling schemes for LDPC decoding, i.e., flooding scheduling and sequential (layered) scheduling. Studies show that layered scheduling not only improves the convergence speed in terms of number of iterations but also outperforms the traditional flooding scheduling for a large number of iterations. Current most LDPC decoder implementations are based on layered-decoding. Unlike turbo codes, LDPC codes can be designed in a quasicyclic (QC) manner such that high throughput implementation becomes possible. Fully parallelized architectures have been proposed and designs with multiple-Giga bps throughputs have been reported for certain standards. However, to maintain a reasonable die-size, for fully parallelized architectures, the underlying LDPC codes usually have to be short, such as the codes defined. For moderate and long codes, there always are tradeoffs between the throughput/ latency and the complexity.

The partially parallel architecture is a good trade-off between throughput and hardware cost. Since a PU is shared for a number of rows or columns, the number of PUs becomes much smaller than that of the fully parallel architecture. As decoding operations are parallel in nature, it is important to determine which rows or columns are processed in a PU. In the grouping, the dependencies between rows and columns should be considered to minimize the overall cycles by overlapping the decoding operations. There has been a heuristic scheduling algorithm proposed for quasi-cyclic LDPC codes but it cannot be applied to general LDPC codes. These manuscripts propose an efficient scheduling algorithm that can be applied to general LDPC codes. The proposed algorithm is based on the concept of the matrix permutation.

RELATED WORKS

In [1] Zhenzhi Wu, Dake Liu, and Yanjun Zhang et al presents Layered Decoding (LD) algorithm is widely applied in high throughput QC-LDPC decoders. Amongst every ensure node update algorithms in LD, Turbo-Decoding Message-Passing (TDMP) is adopted by many proposals. A-posteriori memory entrance conflict under pipelined TDMP decoder incur severe throughput decline. In this dissertation, numerous matrix reorder techniques are initiate to diminish the conflict incidence without incurring the performance loss, which includes Row Exchange method, element Sequence Reordering method, and a conflict detector with pipeline stall insertion. They are incorporated in a joint recursive deep-first penetrating process. Test consequences illustrate that the efficiency enhancement reaches up to 60% compared to non-optimized scenarios for 802.11n and 802.16e standards. Matrix reordering techniques are investigated for conventional Two Phase Message Passing (TPMP) LDPC decoding. The upper-right part of the parity check matrix, the Check Node Update (CNU) and Variable Node Update (VNU) in overlapped schedule may suffer less conflict, and therefore less pipeline stalls are required. However, these methods cannot be adopted in LD decoders. For pipelined LD decoders, the conflicts between consecutive layer and inter-layer are analyzed

In [2] Zijing Wu, Kaixiong Su et al presents Due to the overlap of nonzero sub-matrices in the successive layers of check matrix, the pipeline process might introduce data updating conflict in pipelined layered LDPC decoder. A solution to solve this problem by adjusting the decoding order of layers in check matrix and nonzero sub matrices in the same layer is proposed in this paper. Furthermore the corresponding fast algorithm is given. In term of hardware implementation, this method which can be achieved simply by changing the order of the corresponding data in the ROMs will not increase any extra hardware overhead. Experimental results show that due to fewer idle clocks even zero idle clock need to be inserted into decoding pipeline when using this solution, the decoding rate is improved effectively. More significantly, the technique will not humiliate the decoding performance for LDPC codes. So we analyze two kinds of data updating conflicts and put forward a solution which not only can avoid this conflict but also can reduce the decoding delay in this paper. Firstly, we need obtain the optimal sequencing of layers, in which the maximum number of consecutive overlap of nonzero sub-matrices and the sum of overlaps between adjacent layers are all minimized.

In [3] Aroua Briki, Cyrille Chavet, Philippe Coussy et al presents Recent communication standards and storage systems (e.g. wireless access, digital video broadcasting or magnetic storage in hard disk drives) uses error correcting codes such as LDPC (Low Density Parity Check) or Turbo-codes to reliably transfer data between source and destination. For elevated data rate application, Turbo and LDPC codes are translate on parallel architectures. However, parallel architectures suffer from memory access conflicts and efficient memory mapping algorithms are required to design parallel interleaver architectures which are one of the most critical parts of parallel decoders. In this manuscript, we nearby a process that finds a conflict-free memory mapping for whichever interleaving law and associated parallelism constraint. The proposed approach always complies with the interconnection network topology the designer wants to infer. Moreover, contrary to traditional methods, the resulting architecture is optimized by reducing the cost of network and controller (network and memory controller) architectures.

In [4] Awais Hussain Sani, Philippe Coussy, and Cyrille Chavet et al presents To meet the higher data rate requirement of current and future communication standards, numerous techniques to decode Turbo and LDPC codes on hardware structural design are urbanized. Unfortunately, interleaving laws that are used in these codes often result in memory access conflicts when massively parallel architectures are targeted which considerably limits the throughput. In this article, the first dedicated approach that finds conflict free memory mapping for every type of codes and for every type of parallelism in polynomial time is presented. The implementation of this highly efficient algorithm shows significant improvement in terms of computational time compared to state of the art approaches. Ultimately, this could enable memory mapping algorithm to be embedded on chips and executed on the fly to support multiple block lengths and standards. To manage this problem, conflict is resolved either at run time or during definition of interleaving law or at design time. Solving conflict problem at runtime results in huge hardware cost and delays and becomes an impractical solution for high data rate and low power applications. Designing conflict free interleaving law often simplifies the construction of parallel decoder architectures

In [5] Thien Truong Nguyen-Ly, Valentin Savin, Xavier Popon, and David Declercq et al presents The recently introduced class of Non-Surjective Finite Alphabet Iterative Decoders (NS-FAIDs). First, optimization results for an extended class of regular NS- FAIDs are presented. They reveal different possible trade-offs between decoding performance and hardware implementation efficiency. To validate the promises of optimized NS-FAIDs in terms of hardware implementation benefits, we propose two high-throughput hardware architectures, integrating NS-FAIDs decoding kernels. Implementation consequences demonstrate that NS-FAIDs permit significant improvements in terms of both throughput and hardware resources consumption, as compared to a baseline Min- Sum decoder, with even better or only slightly degraded decoding performance. The proposed architectures target high-throughput and efficient use of the hardware resources. Both architectures implement layered scheduled decoding with fully parallel processing units. The first architecture is pipelined, so as to increase throughput and ensure an efficient use of the hardware resources, which in turn imposes specific constraints on the decoding layers1, in order to ensure proper execution of the layered decoding process.

PROBLE DEFINITION

LDPC codes have exposed near-capacity error-correcting performance whilst well-organized hardware implementations have led to acceptance of LDPC mistake coding in high throughput wired and wireless standards. LDPC codes have established outstanding error-correcting ability such that a number of topical wireless standards have opted for their inclusion. A well-designed LDPC decoder is designed and analyzed. This decoder directly employs hard results returned from the voltage detector and utilizes a single XOR gate to calculate likelihood ratio. At the same time right shift and left shift operation is used to provide the better decoding capability process. Results of this work provide insights into a more effective implementation of a high-throughput LDPC decoder for low error rate performance.

PROPOSED SYSTEM

A holistic approach based on a set of off-line algorithms, for increasing resource usage efficiency of layered scheduling LDPC decoders, and *(2)* a modified version for layered decoding updates – residue based layered decoding. Off-line algorithms for both the message mapping and read access scheduling problems, such that the decoder hardware usage efficiency is maximized. A new residue-based layered decoding that relaxes the access scheduling constraints associated to the pipeline related hazards, and implements it using

FPGA's. Our simulation results show that this imprecision induced by errors in the offline prediction of the maximum energy improves the decoding performances

BLOCK DIAGRAM

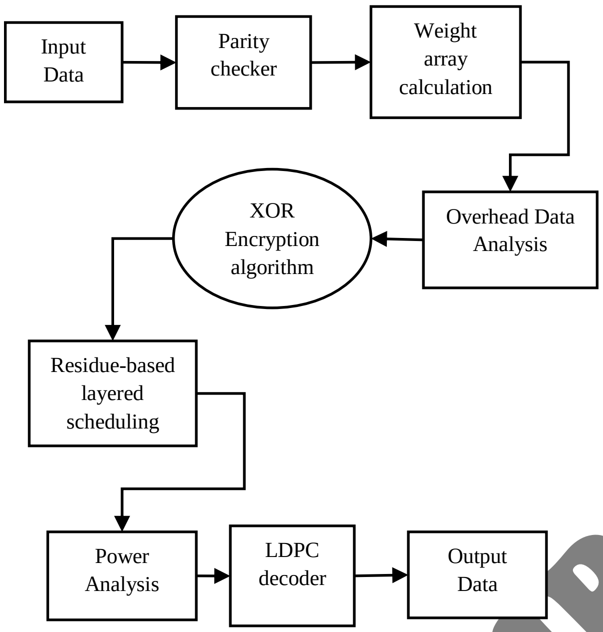


Fig 1 Block diagram

LDPC Codes

LDPC codes are block codes with parity-check matrices that enclose only a very diminutive numeral of non-zero entries. It is the sparseness of H which guarantees both a decoding complexity which increases only linearly with the code length and a least amount distance which also increases linearly with the code length. Aside from the requirement that H be sparse, an LDPC code itself is no diverse to any other block code. Indeed existing block codes can be effectively used with the LDPC iterative decode algorithms if they can be represent by a spare parity-check matrix. Normally, however, judgment a bare parity-check matrix for an existing code is not practical. Instead LDPC codes are calculated by constructing a sparse parity-check matrix first and then influential a generator matrix for the code afterwards. The major dissimilarity connecting LDPC codes and standard block codes is how they are decoded. Classical block codes are normally decoded with ML like decoding algorithms and so are usually short and designed algebraically to make this task less complex. LDPC codes however are decoded iteratively using a graphical representation of their parity-check matrix and so are calculated with the properties of H as a focus

BLOCK DIAGRAM DESCRIPTION

CONSTRUCTION

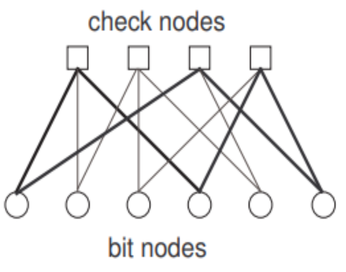


Fig 2 LDPC Construction

The original LDPC codes obtainable by Gallager are regular and defined by a banded structure in H. In this method columns of H are additional one column at a time from left to right. The weight of each column is selected to obtain the accurate bit amount distribution and the position of the non-zero entries in each column chosen arbitrarily from those rows which are not yet full. If at any point there are rows with more positions empty then there are columns outstanding to be added, the row degree distributions for H will not be exact. The development can be started again or back tracked by a little columns, until the accurate row degrees are obtained

LDPC Decoding Process

LDPC codes are decoded iteratively using a message passing algorithm. This algorithm involves exchanging the belief information among the variable nodes and check nodes that are connected by edges in the bipartite graph. Let In be the intrinsic information from the received signal, Ln be the reliable information for variable node n, Ln,m be the information conveyed from variable node n to check node m, and En,m be the extrinsic information generated in check node m that is passed to variable node n. The belief information is updated in an iterative manner and implemented in two phases. In the first phase, the variable nodes send their belief information, Ln,m, to check nodes connected to them; in the second phase, the check nodes send the updated belief information (new En,m) to the variable nodes connected to them for updating Ln. Here, N(m) is the set of variable nodes which are connected with check node m in the bipartite graph. Similarly, M(n) is the set of check nodes which are connected with variable node n.

Variable Depth Pipeline

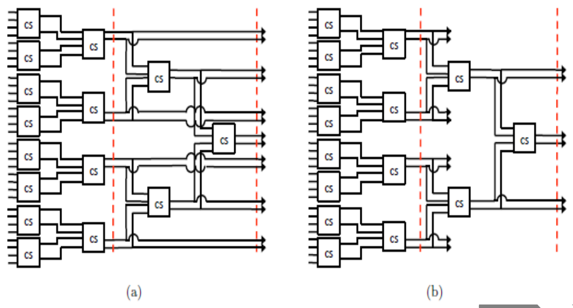


Fig 3 Variable depth pipeline

If the row degree of any of the layers in at least one of the matrices is large, it may be necessary to have pipeline stages within the check node since it needs several CS stages to compute its result. Additionally, if any of the matrices have a large l, the output must be taken early in the tree. By putting the pipeline stage at the same point where an output must be taken, a pipeline stage can be removed for the large l matrices. The pipeline depth varies based on the current code, which reduces the worst-case number of cycles to decode a frame. This lowers the decoder's frequency and saves power

VNU processing block

It consists of m VNUs; the m VNUs perform the variable node and a-posteriori updates corresponding to a column in the B matrix; each VNU process dv β messages in a serial manner; the VNU outputs dv α messages also in a serial manner (one α message per clock cycle); the VNU processing blocks requires the reading of m β messages in one clock cycles and perform m α messages write operations in one clock cycle.

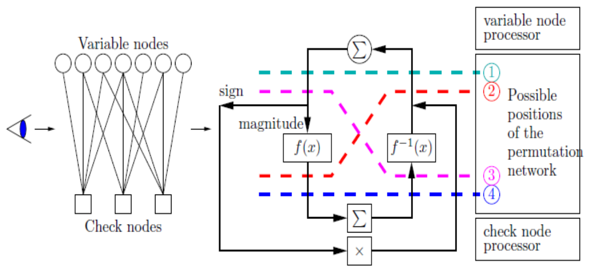


Fig 4 VNU processing block

XOR Encryption and Decryption

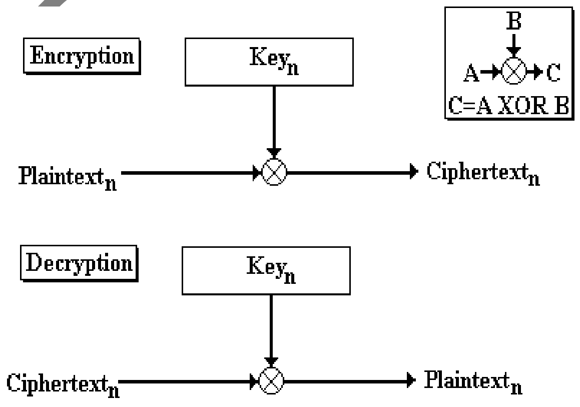


Fig 5 XOR encryption process

XOR encryption (or Exclusive-OR encryption) is an ordinary way of encrypting book keen on a format that cannot be trivially cracked by the average person. XOR encryption is huge for store cog like laughter save details, and extra statistics kind to be stored locally on a user’s computer, that while not a big deal if they are tamper with, you would like to discourage citizens from doing so.

XOR encryption is as fine used frequently as a division of extra composite encryption algorithms. The idea behind it is that if you don't identify the novel disposition or the XOR encryption key, it is impossible to determine what either one is. However, the reason that it is not exclusively make safe is that data nearly at all times contain patterns (JSON uses '{' and '}' characters, XML contains plenty of '<' and '>' characters, etc.) so if a big shot is able to decide the prototype and undo still one nature, they will contain the key to unlocking everything else. But protected or anxious XOR encryption in truth is it has bounty of applicable use bags. Any kind of deterrent added to data that you don't want users to tamper with but that they will have easy access to is a prime candidate, so long as security isn't paramount. The concept is effortless, you label a key spirit, and for each nature in the cord you want to encrypt, you apply the key. Once you want to unencrypted the encrypted data, you simply go through the string and apply the key again.

Residue Based Layered LDPC Decoder

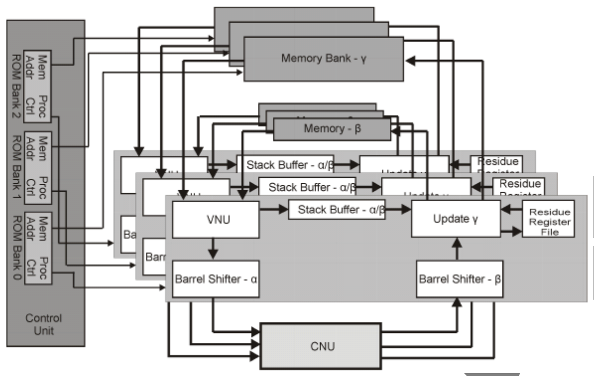


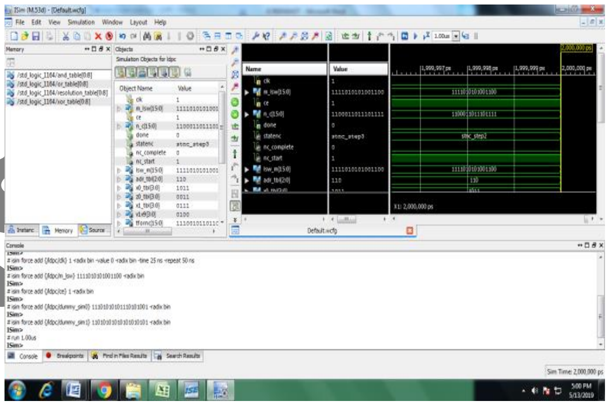
Fig 6 Residue Based Layered LDPC Decoder

It used separate processing units – VNU, CNU and AP-LLR update –, while the BS are used to route the α and β messages, instead of the AP-LLRs, used in most layered decoding architectures. This way, cost savings are obtained, due to the lower quantization of the α and β messages with respect to the AP-LLR messages. Due to simpler control when implementing decoders for irregular LDPC codes, we use baseline architecture with reverse write-back strategy. The stack buffer is a simple bidirectional shift register, suitable for both dc regular and quasi-regular codes (i.e. ±1 variation). Pipelining is applied in the data-processing block.

ERROR CORRECTION AND PARITY CHECKS

Here we will only believe binary messages and so the broadcast messages consist of strings of 0's and 1's. The necessary thought of forward error control coding is to supplement these communication bits with deliberately introduce redundancy in the form of extra check bits to produce a codeword for the message. These check bits are added in such a way that codeword's are sufficiently distinct from one another that the transmitted message can be correctly inferred at the receiver, even when various bits in the codeword are despoiled during broadcast over the channel.

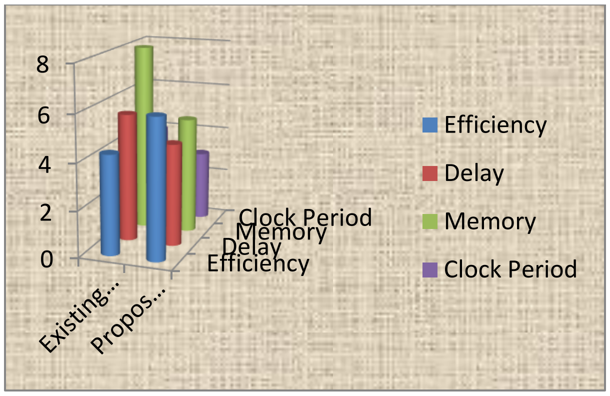
RESULT AND DISCUSSION


```
# isim force add {/ldpc/clk} 1 -radix bin -value 0 -radix bin -time 25 ns -repeat 50 ns
ISim>
# isim force add {/ldpc/m_lsw} 1111010101001100 -radix bin
ISim>
# isim force add {/ldpc/ce} 1 -radix bin
ISim>
# isim force add {/ldpc/dummy_sim0} 11101010101110101001 -radix bin
ISim>
# isim force add {/ldpc/dummy_sim1} 11010101010101010101 -radix bin
ISim>
# run 1.00us
ISim>
```

COMPARISON

PARAMETER	EXISTING SYSTEM	PROPOSED SYSTEM
EFFICIENCY	93.0%	95.6%
DELAY	4.103 ns	3.283ns
MEMORY	242256 kilobytes	187740 kilobytes
COMBILATION TIME	3.52 sec	3.52 sec
FREQUENCY	508.647 MHz	71.045 MHz
CLOCK PERIOD	1.966 ns	1.950 ns

CHART



CONCLUSION

A holistic approach based on a set of off-line algorithms, for increasing resource usage efficiency of layered scheduling LDPC decoders, and *(2)* a modified version for layered decoding updates – residue based layered decoding. The evaluation of the off-line mapping and scheduling algorithms performed for 6 QC-LDPC codes indicates an increase in HUE of 3% to 49% when no overlap is used, of 24% to 57% for a one layer overlap and of 25% to 57% for unlimited overlaps with respect to the case when no optimization is employed. FPGA implementation results of the selected 6 decoding architectures corresponding to WiMAX rate 3/4 irregular code, and DVB-S2X rate 140/180 code suggest that by jointly using the proposed residue-based scheduling with unlimited update overlaps, can lead to up to 85% TAR improvements with respect to the architecture employing conventional layered scheduling. The application not only considers the most reliable symbol in the syndrome computation, but also takes at least the second most reliable symbol of each incoming message into account. An extended information set is available for the parity-check node update and this allows introducing the concept of weak and strong votes performed by the check node unit. Each variable node can receive two kinds of votes, whose amplitudes can be tuned to the reliability of the syndrome that produces the vote.

REFERENCE

[1] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11-2016.

[2] Wireless Medium Access Control (MAC) and Physical layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs) Amendment 2: Millimeter-Wave-Based Alternative Physical Layer Extension, IEEE Standard 802.15.3c-2009, 2009. [Online]. Available:m http://standards.ieee.org/getieee802/download/802.15.3c-2009.pdf

[3] Air Interface for Fixed and Mobile Broadband Wireless Access Systems, IEEE Standard 802.16e-2005, 2005.

[4] DVB—The Family of International Standards for Digital Video Broadcasting, Second Generation Framin Structure, Channel Coding and Modulation Systems for Broadcasting, Interactive Services, News Gathering, and Other Broadband Satelite Applications, Part 2: DVB-S2 Extensions (DVB-S2X), ETSI Standard, Oct. 2014.

[5] K.-C. Ho, C.-L. Chen, and H.-C. Chang, "A 520k (18900, 17010) array dispersion LDPC decoder architectures for NAND flash memory," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 24, no. 4, pp. 1293–1304, Apr. 2016.

[6] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in Proc. IEEE Workshop Signal Process. Syst., Oct. 2004, pp. 107–112.

[7] A. Tarable, S. Benedetto, and G. Montorsi, "Mapping interleaving laws to parallel turbo and LDPC decoder architectures," IEEE Trans. Inf.Theory, vol. 50, no. 9, pp. 2002–2009, Sep. 2004.

[8] E. Amador, R. Pacalet, and V. Rezard, "Optimum LDPC decoder: A memory architecture problem," in Proc. 46th ACM/IEEE Des. Autom. Conf., Jul. 2009, pp. 891–896.

[9] A. Briki, C. Chavet, and P. Coussy, "A conflict-free memory mapping approach to design parallel hardware interleaver architectures with optimized network and controller," in Proc. SiPS, Oct. 2013, pp. 201–206.

[10] S. U. Reehman, C. Chavet, P. Coussy, and A. Sani, "In-place memory mapping approach for optimized parallel hardware interleaver architectures," in Proc. Des. Autom. Test Eur. Conf. Exhib. (DATE), Mar. 2015, pp. 896–899.
[11] Z. Wu and K. Su, "Updating conflict solution for pipelined layered LDPC decoder," in Proc. IEEE Int. Conf. Signal Process. Commun. Comput. (ICSPCC), Sep. 2015, pp. 1–6.
[12] C. Marchand, J. B. Dore, L. Conde-Canencia, and E. Boutillon, "Conflict resolution for pipelined layered LDPC decoders," in Proc. IEEE Workshop Signal Process. Syst., Oct. 2009, pp. 220–225.
[13] M. Rovini, G. Gentile, F. Rossi, and L. Fanucci, "A scalable decoder architecture for IEEE 802.11n LDPC codes," in Proc. IEEE Global Telecommun. Conf., Nov. 2007, pp. 3270–3274.
[14] Z. Wu, D. Liu, and Y. Zhang, "Matrix reordering techniques for memory conflict reduction for pipelined QC-LDPC decoder," in Proc. IEEE/CIC Int. Conf. Commun. China (ICCC), Oct. 2014, pp. 354–359.
[15] M. P. C. Fossorier, "Quasicyclic low-density parity-check codes from circulant permutation matrices," IEEE Trans. Inf. Theory, vol. 50, no. 8, pp. 1788–1793, Aug. 2004.
[16] O. Boncalo, P. F. Mihancea, and A. Amaricai, "Template-based QCLDPC decoder architecture generation," in Proc. 10th Int. Conf. Inf., Commun. Signal Process. (ICICS), Dec. 2015, pp. 1–5.
[17] T. T. Nguyen-Ly, V. Savin, X. Popon, and D. Declercq, "High throughput FPGA implementation for regular non-surjective finite alphabet iterative decoders," in Proc. IEEE Int. Conf. Commun. Workshops, May 2017,pp. 961–966.
[18] T. Nguyen-Ly et al., "FPGA design of high throughput LDPC decoder based on imprecise offset min-sum decoding," in Proc. IEEE 13th Int. New Circuits Syst. Conf. (NEWCAS), Jun. 2015, pp. 1–4.
[19] M. Rovini, G. Gentile, F. Rossi, and L. Fanucci, "A minimum-latency block-serial architecture of a decoder for IEEE 802.11n LDPC codes," in Proc. IFIP Int. Conf. Very Large Scale Integr., Oct. 2007, pp. 236–241.
[20] M. Gomes, G. Falcao, V. Silva, V. Ferreira, A. Sengo, and M. Falcao, "Flexible parallel architecture for DVB-S2 LDPC decoders," in Proc. IEEE Global Telecommun. Conf. (GLOBECOM), Nov. 2007, pp. 3265–3269.