Cost-Efficient FPGA Layered LDPC Decoder With Serial AP-LLR Processing

Oana Boncalo , Alexandru Amaricai, Andrei Hera
University Politehnica Timisoara
Computer Engineering Department
Timisoara, Romania
{alexandru.amaricai, oana.boncalo}@cs.upt.ro

Valentin Savin
CEA-LETI,
MINATEC Campus,
Grenoble, France
valentin.savin@cea.fr

***Abstract*—This paper proposes an FPGA based layered architecture for quasi-cyclic (QC) irregular LDPC decoder. Our approach is based on merging variable and check node processing into one single variable-check node (VCN) unit. Layer message computation is done using a parallel scheme of a number of VCNs equal to the expansion factor of the QC matrix. The proposed architecture is characterized by the serial processing of the a posteriori LLRs by an FPGA specific high frequency VCN unit implementation using ROM memories. In our approach data conversions as well as additions and comparators are replaced by look-up-tables implemented using distributed RAM. In addition to this, other techniques such as: efficient packaging of LLRs messages and check-node message compression as well as the configurable port width of the FPGA's BRAM are used to reduce BRAM block utilization. Throughput increase is achieved by utilizing techniques such as pipelining, parallel processing of multiple VCNs, as well as relatively high working frequency. Implementation results for the WiMAX (1152, 2304) QC irregular LDPC code indicate that the proposed architecture has up to 3x less slices resource utilization and up to 1 order of magnitude less BRAM blocks with respect to other approaches, while maintaining a throughput of several hundreds of Mbps (800 Mbps coded bits). We achieved this without sacrificing flexibility; therefore we can easily adapt our design to accommodate different code rates.**

Keywords—LDPC; Min-Sum; Layered scheduling; FPGA

I. INTRODUCTION

The interest for Low-Density Parity-Check codes (LDPCs) has grown during the last decades. Since they were rediscovered in the late '90s they became backbone on which multiple standards were developed such as IEEE 802.16 WiMAX data transmission protocol and the DVB-S2 video encoding standard [1]. Because of the high demand for ever-increasing throughput, hardware implementations of LDPC codes, such as [2-8] are a natural choice being favored by the parallel nature of LDPC decoders. Field programmable gate arrays (FPGAs) integrated circuits are highly versatile and can offer high degree of parallel processing for custom solutions. This makes them suitable platforms in areas such as telecommunications, video processing and of course coding applications.

This paper aims at utilizing FPGA specific resources to build a cost-efficient FPGA layered Quasi-Cyclic (QC) irregular LDPC decoder architecture with serial A Posteriori Logarithmic Likelihood Ratio (AP-LLR) processing. The QC-LDPC codes addressed are built of permutated identity sub-matrices and zero sub-matrices. The algorithm implemented is called "Min-Sum" (MS) and is known to offer a very good trade-off between hardware complexity and error-correction performance. For this purpose, on the one hand we combine existing techniques for reducing cost, while trying to increase throughput such as: reduced quantization scheme in variable-check-nodes (VCNs) [2], variable-to-check node message compression [7], take advantage of block RAM (BRAM) features - configurable width, and use message packaging for using less BRAM resources [4], several parallel processing datapath units [4][6][8] and pipelining. On the other hand, in order to obtain an increase in throughput, we propose a new approach for building datapaths by replacing groups of arithmetic and logic operators with ROM memories implemented in Xilinx FPGAs using look-up-table (LUT) resources. Hence, it reduces the critical path, and results in higher operating frequencies for the designed VCN units. Another important design feature – flexibility – is also ensured, as we can easily adapt our design to accommodate different code rates. Furthermore, the efficient FPGA resource usage makes the underlying design principles of the proposed architecture suitable for larger codes.

This paper is organized as follows: Section I presents a short introduction of QC-LDPC codes, Section III presents related work as well as this paper's contributions, Section IV discusses the proposed method for the LDPC decoder architecture using serial AP-LLR message processing, Section V presents a case study for the WiMAX (1152, 2304) code, and conclusive results in Section VI.

II. OVERVIEW OF LDPC CODES

LDPC codes [9] are a class of linear codes that can be advantageously represented by sparse bipartite graphs. The bipartite-graph of an LDPC code [10] contains two types of nodes – variable-nodes, corresponding to codeword bits, and check-nodes, corresponding to parity-check equations – which are connected according to the non-zero entries of the parity check matrix H defining the code. LDPC decoding is performed by message-passing (MP) algorithms, which consist of an exchange of messages along the edges of the bipartite graph, taking place in several rounds or iterations.

Quasi-Cyclic (QC) LDPC codes are a class of algebraic constructed LDPC codes that feature a highly structured parity check matrix, defined by blocks of circulant matrices [11]. The structure of the parity-check matrix allows for the efficient hardware implementation of the MP decoder, which explains the wide use of QC-LDPC codes in many telecommunication standards. Precisely, a QC-LDPC code is defined by a base matrix *B*, with integer entries $b_{i,j} \geq -1$. The parity-check matrix *H* is obtained by expanding the base-matrix *B* by an expansion factor *m*; thus, each entry of *B* is replaced by a square matrix of size $m \times m$, defined as follows: -1 entries are replaced by the all-zero matrix, while $b_{i,j} \geq 0$ entries are replaced by a circulant matrix, obtained by right-shifting the identity matrix by $b_{i,j}$ positions.

MP decoders may deal with different scheduling strategies, according to the order in which variable and check-node messages are updated during the message passing iterative process. The classical convention is that, at each iteration, all check-nodes and subsequently all variable-nodes pass new messages to their neighbors. This message-passing schedule is usually referred to as flooding *scheduling* [12]. A different approach is to split the parity check matrix in several horizontal layers, then process horizontal layer sequentially, while check-nodes (rows) within the same layer are processed by using a flooding schedule strategy. Each time a layer is processed the decoder updates the neighbor variable-nodes, so as to profit from the propagated messages, and then proceeds to the next layer. This strategy is known as *layered scheduling* [13]. Its main advantage is that it propagates information faster and converges in about half the number of iterations compared to the flooding schedule [14]. For applications requiring a small number of decoding iterations, the layered schedule provides superior decoding performance, thanks to its faster convergence.

While flooding scheduling is suitable for a fully parallel implementation, enabling high throughput at the cost of an increased area, layered scheduling is suitable for a cost-effective implementation and allows a flexible trade-off between throughput and hardware resources. We note that the layer decoding is the natural choice for QC-LDPC, as their parity check matrix *H* has a built-in layered structure, with each layer composed by *m* consecutive rows that correspond to one row of the base matrix *B*.

Concerning the decoding algorithm, the Min-Sum (MS) decoding [15] is the most suitable candidate for hardware implementation, due to its low computational complexity. The computations required by MS decoding are additions and comparisons. The MS layered decoding algorithm starts with the initialization of the with the a-posteriori log likelihood ratios (AP-LLR, denoted γ_n) with the received values of log-likelihood ratios (denoted $\overline{\gamma_n}$). Each iteration is comprised of the following steps (where $\beta_{m,n}$ denotes check-node messages, and $\alpha_{m,n}$ denotes variable-node messages):

```
For L = 1,…, Layers_Number
    1. Variable-node messages:
        $\alpha_{z,n} = \gamma_n - \beta_{z,n};\ \forall z \in M_L\ \forall n \in H(z)$
    2. Check-node messages:
        $\beta_{z,n} = \left( \prod_{n' \in H(z) \setminus n} \operatorname{sgn}(\alpha_{z,n'}) \right) \times \min\left(\alpha_{z,n'}\right);\ \forall z \in M_L\ \forall n \in H(z)$
    3. AP-LLR update:
        $\gamma_n = \alpha_{z,n} + \beta_{z,n};\ \forall z \in M_L\ \forall n \in H(z)$
End
```

M_L stands for the horizontal group of rows from the *H* matrix corresponding to layer *L*, while $H(z)$ represents the indexes of the non-zero entries of the row *z* of layer *L*.

The process continues until the codeword is found (which can be verified by computing the syndrome) or the maximum number of iterations is reached. In practical implementations, exchanged messages are quantized on a finite number of bits, typically 4 bits for variable and check-node messages (and $\alpha_{z,n}$ and $\beta_{z,n}$) and 6 bits for the AP-LLR (γ_n).

Finally, we will denote by d_c and d_v the number of non-zero entries (1's) per column and row of H, respectively. If both d_c and d_v are constant the code is said to be regular, otherwise it is said to be irregular.

III. Related Work

Several strategies have been employed when building QC-LDPC architectures based on the targeted optimizations: cost, and/or throughput. Very high throughput solutions usually rely on flooding scheduling, which demands a substantial amount of resources. Hence, they compromise decoding performance by employing lower message quantization schemes in order to reduce the cost [2]. Other solutions that are less area hungry, use layered scheduling and only a number of parallel units, and are typically referred to as partially parallel architectures. FPGA designs use several techniques to efficiently utilize the resources available: (i) *mimicking multiport by using separate BRAM blocks for increased bandwidth*. This solution typically results in a large number of BRAM blocks. One possibility for alleviating this problem is by employing message packaging techniques that take advantage of the configurable BRAM port width [4]. However, as stressed out in [4], efficient packaging proposed in the folded technique introduces overhead when virtualizing memory ports. Compression schemes to reduce area requirements and port width size for variable-to-check node messages for the MS algorithm has been proposed in [7]. (ii) *pipelining*: for increasing frequency used in most of the solutions ; (iii) *parallel processing units* inside a layer of the matrix *H* (horizontal parallelization) [4][5], or between layers (vertical parallelization) [8]. These solutions rely on various degrees of parallelism and overlapping for layer processing [4], so as to speed up the processing of a layer, as well as overlapping between different layers [8]; (iv) *Datapath optimizations* by algorithm modifications [3], or unit operation handcrafting [2][6]; (v) *Reduced quantization schemes* that aim at reducing memory requirements, interconnect complexity and area, as well as area for the processing units [2][3]. The later

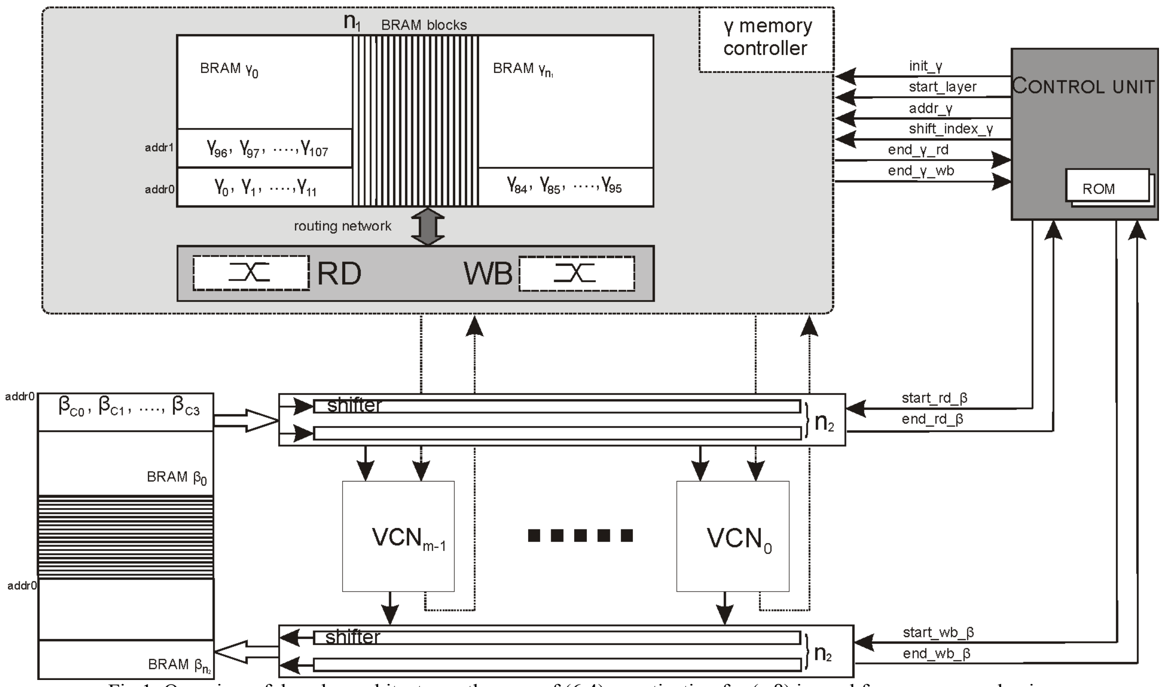

Fig.1. Overview of decoder architecture - the case of (6,4) quantization for (γ,β) is used for message packaging

trades area and complexity for decoder error correction performance. Other architecture features is flexibility. A highly flexible solution is presented in [5], with the price of sacrificing throughput.

The solution employed in this paper is a layered scheduled QC irregular LDPC decoder with serial AP-LLR message processing, which targets a tradeoff cost/throughput/ design flexibility: a low cost flexible FPGA architecture that can support design and runtime parameters (see section IV) while achieving very good throughput. The contributions of this work are as follows:

- Combining already established techniques such as: configurable BRAMs and message packaging for efficient BRAM utilization, pipelining, parallel units for layer processing, compressed variable-to-check node messages for reduced memory storage size and reduced memory width size, serial AP-LLR processing to reduce requirements for routing network and the numbers of access ports for the AP-LLR memory.
- Novel approach for building VCN processing units by replacing design sections with ROMs implemented using Distributed RAM. The VCN unit is required to perform certain arithmetic operations and comparisons. Instead of describing the logical blocks required to perform these operations, our solution replaces these blocks with ROMs altogether. All messages are represented in sign magnitude (SM), since the arithmetic units have been removed. Hence, the critical path is reduced and an increase of up to 15% frequency is obtained, thus increasing the decoder throughput with a minimal increase of FPGA resources.
- Case study presenting implementation results and decoder performance evaluation for the proposed approach for WiMAX (1152, 2304), rate $1/2$.

Hence, we target a low-cost FPGA implementation of QC irregular LDPC codes with different expansion factors and different d_c and d_v degrees, while trying to maximize throughput.

IV. METHOD FOR EFFICIENT SERIAL AP-LLR PROCESSING

The proposed design method for QC LDPC codes tries to build architectures that exploit FPGA resources efficiently, while trying to optimize throughput for layer scheduling decoding. Maximizing throughput requires a lot of resources to be committed in order to parallelize the CNU and VNU processing. The characteristics of the architecture presented in this paper are:

1. The decoder relies on layer scheduling.
2. It utilizes a parallelization factor of m, where m denotes the size of the matrix expansion factor and is a design parameter. Hence, m processing elements for processing one layer are employed.
3. The proposed approach relies on a merged VNU and CNU unit into what we call a VCN unit. It is able to process one AP_LLR message at a time. The implementation offered tries to maximize frequency by using an FPGA specific approach for building the datapath (see subsection B for more details). By doing so, it tries to minimize the number of memory access ports for the AP-LLR memory.
4. It tries to minimize BRAM block utilization for the AP-LLRs (from herein referred as γ) and check node messages (from herein referred as β-messages) by:
   a. Packaging a maximum number of γ and β-messages by using the BRAM configurable port width (8-72 bits);

b. Using compressed β-messages. A VCN unit requires a total of d_c β-messages for each layer. For d_c β-messages only the sign bits, the first and second minimum magnitudes and the index for the β corresponding to the first minimum are stored. We refer to it as a compressed β word and denote it with β_C .The reason behind using a compressed scheme is to lessen the requirements for the memory width and size for storing the β-messages.

5. Flexible, in that it can be extended to accommodate different code rates. This is the result of the serial processing. Through selecting a different maximum value for the decoder CU counter of the γ-messages a different number of γ can processed per layer. An input control signal can select a different ROM memory for the γ indexes and addresses for each layer.

6. Aggressive pipelining for increased throughput and frequency.

The design parameters for the proposed architecture are: *(i)* m – number of VCNs; *(ii)* w_β - size of the compressed β - message word (denoted by β_C); *(iii)* w_γ - size of the γ-message; *(iv)* n_1 - number of γ BRAM blocks, where $n_1 = \left\lceil \frac{m \times w_\gamma}{w_{\max BRAM}} \right\rceil$; *(v)* n_2 - number of β BRAM blocks, representing the minimum integer value that satisfies two criteria: $n_2 \le \frac{w'_\beta \times m}{\lceil \log_2 m \rceil \times w_{BRAM}}$, and $\frac{m}{n_2}$ is a multiple of $\frac{w_{BRAM}}{w'_\beta}$. Here, w_{BRAM} denotes the maximum configurable value of the BRAM bock that satisfies the criteria, while w'_β is the smallest multiple of 8 bits, greater than or equal to w_β (β messages are aligned to multiple of 8 bits for a BRAM memory location).

For the case of WiMAX (1152, 2304) QC irregular LDPC, with rate ½, the expansion factor m is 96, message quantization (γ, β) size is (6,4) bits. The width for the γ memory is 576 bit (for m γ-messages). This results in $n_1 = 576/72 = 8$ BRAM blocks. The size of the compressed β-message for each VCN unit is $w_\beta = 16$ bits (d_c signs, *3* bits minimum$_1$, *3* bits minimum$_2$, *3* bits index for minimum$_1$). A total of 1536 bits need to be read in $\lceil \log_2 m \rceil$ cycles (the amount of time needed for routing γ at the inputs of the corresponding CU units); hence, the data width of the β memory is 256. This results in $n_2 = 256/64 = 4$ BRAM blocks for $w_{BRAM} = 64$. The total BRAM block usage is 12 for this example.

A. Architecture overview

The proposed solution can be employed to implement QC-LDPC decoders that are defined by parity check matrices *H*, containing cyclically shifted identity submatrices, by an offset given by the generator matrix *B*, and zero submatrices. An example of such a code is WiMAX (1152, 2304).

The architecture comprises four major units (see Fig. 1): (i) *datapath processing*: uses m parallel VCN processing units (ii) *γ memory and routing network*: reads m γ-messages, corresponding to the *B* matrix column that is being currently processed; the m γ-messages are shifted according to the offset given by the generator matrix *B*; and at the end of layer computation, it updates the m γ-messages, re-alingning them through the writeback routing network ; (iii) *β memory and routing network*: responsible with preparing the required compressed β-messages for each VCN; (iv) *decoder control unit (CU)*: generates read and writeback requests for the γ, and *β* units, it is responsible for providing γ memory address as well as the offset from matrix B for aligning the γ words corresponding to the circulant sub-matrix that is processed within the layer.

B. VCN architecture

The architecture uses merged VNU and CNU units into a VCN unit. The proposed solution is FPGA specific, in that it utilizes the dedicated ROM memories in place of conventional arithmetic logic in a bid to optimize VCN datapath working frequency. In addition to this classical solution such as pipelining has been employed. The main idea has been to take advantage of the relatively small number of bits for the VCN operands and to group them in computational blocks that are replaced by small ROM memory implemented using distributed RAM. Furthermore, for this approach both input messages (γ and *β*), as well as intermediate results can be in sign magnitude representation. Fig. 2 describes the computational blocks that have been replaced by ROM memories, as well as the pipeline registers that have been introduced. Three ROM memories are used: α computation and saturation to w_γ bits, updated γ value computation memory, and one that computes the minimum$_1$ and minimum$_2$ values. For message quantization (γ, β) size equal to (5,3) bits the input sizes are: 8 bits (3 bits SM β , 5 bits SM γ) for γ and α computation memories, with 5 bits data output width, and 6 bits inputs for the ROM that replaces the comparator with 6 bits output. The inputs for the comparator ROM are: 2 bits magnitude current minimum$_1$, 2 bits magnitude current minimum$_2$, 2 bits magnitude saturated α . The 6 bits output represents: 2 bits magnitude new minimum$_1$, 2 bits magnitude minimum$_2$, 2 bits selection for the minimum indexes.

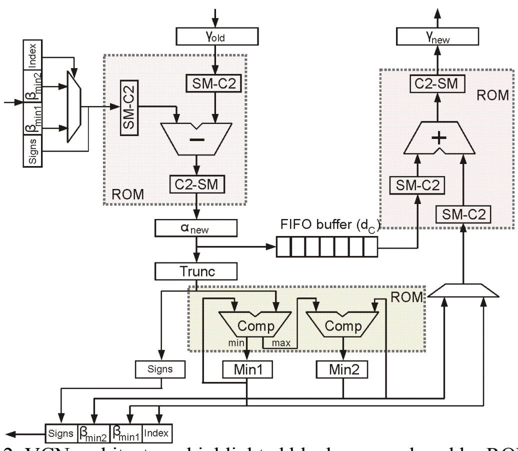

Fig. 2. VCN architecture: highlighted blocks are replaced by ROMs .

C. Memory design

The proposed solution tries to efficiently utilize FPGA BRAM block resources by selecting the maximum configurable width that complies with the architecture parameters.

The γ module has the following building blocks: (i) n_1 BRAM storage blocks; (ii) γ CU that receives read and writeback requests from decoder CU and signals the end of the read operation for the layer messages, and the end of the writeback for the new layer γ values through *end_gamma_rd*, *end_gamma_wb* signals; (iii) routing module responsible for preparing the γ messages for the VCN units, and for writing their updated values $\bar{\gamma}$ at the end of a layer processing. The γ routing unit is made of two barrel shifters: one for reading the γ's corresponding to the currently processed sub-matrix from layer that is processed, and one that is responsible for re-arranging the updated γ in order for writeback. Both barrel shifters are pipelined with $\lceil \log_2 m \rceil$ stages. The shift amount index for arranging the γ messages is given by the current sub-matrix offset for the layer being processed.

The β module is made of: (i) n_2 BRAM storage blocks; (ii) β CU that receives read and writeback requests from decoder CU for reading and routing the compressed β words to the appropriate VCN unit and signals end of operation through *end_beta_rd*, *end_beta_wb* signals (ii) routing module made of two sets of shift registers that prepare the data at the VCN input and writeback the result at the end of layer processing. For read, n_2 serial in/parallel out shift registers of $m/n_2 \times w_\beta$ bits have been used. For writeback, n_2 parallel in/serial out shift registers of $m/n_2 \times w_\beta$ bits have been employed. The shift amount for both cases equals w_{BRAM} bits.

The addressing scheme for both memories is simple: counter for β memory, and ROM memories for γ. Each ROM memory location contains information related to a layer: the sub-matrix offset for the read barrel shifter, and index of the generator B matrix column that gives the address of the m γ-messages for processing. One additional field for each entry is needed in the case of irregular codes to code which d_c value is to be used for the current layer. For the example considered WiMAX (1152, 2304) QC irregular LDPC , rate ½, two d_c values are possible: 6 and 7. For this case, 1 bit suffices to encode this layer information in the decoder CU ROM memory.

D. Performance analysis

Three parameters have been considered when evaluating the proposed architecture: throughput, cost and flexibility. The architecture cost is dictated by: the two γ barrel shifters - w_γ units of m bit barrel shifters with $\lceil \log_2 m \rceil$ pipeline stages; m VCN units; the two $m/n_2 \times w_\beta$ bits shift registers for β read and writeback; $n_1 + n_2$ BRAM blocks. Throughput is theoretically computed by the following formula:

$$\mathrm{T}_{DEC} = T_{CLK}\left[\begin{array}{l} n_{INIT} + \\ \left(1+\lceil \log_2 m \rceil + (d_c - 1)\left(n_{VCN} + (d_c - 1)\right)\right) + \\ d_c\left(n_{LAYERS} \times n_{IT} - 1\right) \end{array}\right]$$

Three factors can be identified: initialization time, the time required for processing one layer, and the time required to process the rest of the layers. The dominant factor is the last product term, which is a consequence of the serialized processing of the γ-messages, and it gives the limiting bounds for our QC-LDPC architecture. n_{INIT} denotes the number of cycles required to load/offload the AP-LLRs in the γ memory, n_{IT} the number of iterations, n_{VCN} equals the number of pipeline stages of the VCN unit, and n_{LAYERS} the number of layers from the H matrix.

Regarding flexibility, the proposed architecture can be used for various expansion factors, message quantization sizes – design time flexibility, and it can easily be extended to support different code rates, providing that the base matrix is of same length and expanded by the same expansion factor – runtime flexibility. It is worthwhile emphasizing that runtime flexibility only impacts the β module parameters and the decoder's CU. For the CU, ROM memories for each code rate need to be provided, as well as d_c values. For the β module, the design parameters w_β' and n_2 are the worst case values.

V. Case Study

We have implemented a baseline architecture for WiMAX (1152, 2304) QC irregular LDPC [16], rate ½, the expansion factor m is 96, message quantization (γ,β) size is (6,4) bits with the classical approach for building the VCN unit. Furthermore, we have implemented the same code with message quantization (γ,β) size is (5,3) bits with an error correction capability degradation of less than 0.25 dB for a bit error rate (BER) of 10^{-6} with respect to the conventional (6,4) quantization (Fig. 3). The two decoders have been implemented on Xilinx XC5VLX50T device, (Digilent

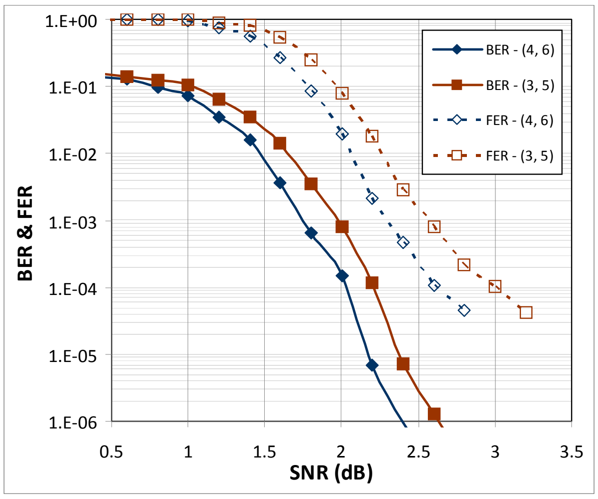

Fig. 3. BER& FER for the proposed architecture, with quantization (4,6) and (3,5) .

TABLE I. RESULTS COMPARISON FOR DIFFERENT FPGA ARCHITECTURES

	Code	Quantization	Device	Frequency (MHz)	Throughput (Mbps)	Resources	
						Slices	BRAM
Chandrasetty2012 [3]	(576,1152)	γ-4 bit, β-2 bit	Virtex-5	138	11400	10823	
Balatsoukas-Stimming2012 [2]	(1152,2304)	γ-4 bit, β-3 bit γ-3 bit, β-2 bit	Virtex-5	154 211	8900 12200	21688 11700	
Vector overlapped Folded [Chen2011] [4]	(768,1536) (3213,3969)	γ-6 bit, β-4 bit γ-6 bit, β-4 bit	Virtex-4	149-162 200-226	223-888 101-460	1472-6100 6600-14100	24 330
Kim2011 [6]	(336,672)	γ-6 bit, β-4 bit	Virtex-4	335	822	27000	32
Beuschel2008 [5]	(1152,2304)	γ-6 bit, β-4 bit	Virtex-4	75	75	9877-19500	122
Proposed (baseline)	(1152,2304)	γ-6 bit, β-4 bit	Virtex-5	281	719 (coded)	6529	12
Proposed (new VCN)	(1152,2304)	γ-5 bit, β-3 bit	Virtex-5	312	800 (coded)	6326	11

Genesys board) speed grade -3 using the Xilinx ISE 14.4 tool.

The proposed solution presents the most efficient BRAM utilization with respect to other solutions which use memories for message storing blocks. With respect to the vector overlapped solution in [4] and the solution in [6], the proposed architecture uses at least 50% less BRAM blocks (12 BRAM for proposed baseline and 11 BRAM for the (5,3) quantization with respect to 24 in [4] and 31 in [6]), although the used codeword is larger (2304 bits for the proposed with respect to 1536 for [4] and 768 for [6]). With respect to folded architecture in [4] and the solutions in [5], the proposed solution uses one order of magnitude less BRAM. Regarding the slice usage, the proposed architectures present the most efficient utilization for decoders implementing the (1156, 2304) codes, with around 50% less with respect to [2] (although it uses more bits for quantization) and with 35% less then [5]. Regarding the working frequency, the ROM based datapath implementation is higher with respect to [2][3][4][5], and slightly lower than the one in [6].

VI. CONCLUSIONS

This paper presents a cost efficient layered architecture for QC LDPC decoders for FPGA implementations. The main advantages of the proposed architecture: ***(1)*** *efficient BRAM utilization* for both AP-LLR and check-to-variable memories, due to message packaging and the compression of variable-to-check node messages; ***(2)*** *efficient slice based utilization* due to the serial nature the processing of AP-LLRs which reduces both the number of memory access ports and the routing network requirements; ***(3)*** *new approach for building the datapath – VCN units*: replacing arithmetic operations by ROM memories, which yields high working frequency (312 MHz); ***(4)*** *pipelining and parallelization* - *m* VCN units, with *m* representing the matrix expansion factor. The proposed decoder targets applications of QC irregular LDPC codes with variable d_c and d_v degrees in cost sensitive communication systems.

ACKNOWLEDGMENT

This work has been supported by the Franco-Romanian (ANR-UEFISCDI) Joint Research Programme "Blanc–2013" – project DIAMOND and by Seventh Framework Program of European Union under Grant Agreement 309129 (i-Risc project).

REFERENCES

[1] T. Richardson and R. Urbanke, "The Renaissance of Gallager's Low-Density Parity-Check Codes", *IEEE Comm. Magazine*, Aug. 2003.

[2] A. B. Stimming and A. Dollas, "FPGA-based design and implementation of a multi - GBPS LDPC decoder", *FPL*, 2012.

[3] V. A. Chandrasetty and S. M. Aziz, "An area efficient LDPC decoder using a reduced complexity min-sum algorithm", *VLSI Journal*, 2012.

[4] X. Chen, J. Kang and S. Lin and V. Akella, "Memory System Optimization for FPGA Based Implementation of Quasi-Cyclic LDPC Codes Decoders", *IEEE Trans. on CAS*, 2011.

[5] C. Beuschel, H.-J. Pfleiderer, "FPGA implementation of a Flexible LDPC decoder", *FPL,* 2008.

[6] S. Kim, G. E. Sobelman, and H. Lee, "A Reduced-Complexity Architecture for LDPC Layered Decoding Schemes", *IEEE Trans. On VLSI,* 2011.

[7] Zhongfeng Wang and Zhiqiang Cui, "A Memory Efficient Partially Parallel Decoder Architecture for Quasi-Cyclic LDPC Codes*", IEEE Trans. on VLSI Systems*, Vol. 15, No. 4, April 2007.

[8] Kai Zhang, Xinming Huang, Zhongfeng Wang, "High-Throughput Layered Decoder Implementation for Quasi-Cyclic LDPC Codes", *IEEE J. on Selected Areas in Communications*, Vol. 27, No. 6, August 2009.

[9] R. G. Gallager, Low Density Parity Check Codes, M.I.T. Press, 1963, Monograph.

[10] R. M. Tanner, "A Recursive Approach to Low Complexity Codes," *IEEE Trans. Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[11] M.P.C. Fossorier, "Quasicyclic Low-Density Parity-Check Codes from Circulant Permutation Matrices," *IEEE Trans. on Information Theory*, vol. 50, no. 8, pp. 1788–1793, 2004.

[12] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," IEEE Journal on Selected Areas in Communications, vol. 16, no. 2, pp. 219–230, 1998.

[13] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. of IEEE Workshop on Signal Processing Systems (SIPS)*, 2004, pp. 107–112.

[14] J. Zhang, Y. Wang, M. P. C. Fossorier, and J. S. Yedidia, "Iterative decoding with replicas," *IEEE Transactions on Information Theory*, vol. 53, no. 5, pp. 1644–1663, 2007.

[15] M.P.C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. on Communications*, vol. 47, no. 5, pp. 673–680, 1999.

[16] IEEE-802.16e, "Physical and medium access control layers for combined fixe and mobile operation in licensed bands," 2005, amendment to Air Interface for Fixed Broadband Wireless Access Systems.