HIGH-SPEED CONFLICT-FREE LAYERED LDPC DECODER FOR THE DVB-S2, -T2 AND -C2 STANDARDS

C. Marchand, L. Conde-Canencia and E. Boutillon

Université Européenne de Bretagne, Lab-STICC, UBS, UMR 6285 , 56100 Lorient, France.
Email: cedric.marchand@univ-ubs.fr

ABSTRACT

Layered decoding is known to provide efficient and high-throughput implementation of LDPC decoders. However, the implementation of layered architecture is not always straightforward because of memory update conflicts in the *a posteriori* information memory. In this paper, we focus our attention on a particular type of conflict that is due to multiple-diagonal sub-matrices in the DVB-S2, -T2 and -C2 parity-check matrices. We propose an original solution that combines repetition of the concerned layers and the write disable of the *a posteriori* information memory. The implementation of this solution on an FPGA-based LDPC decoder led to an average air throughput of 200 Mbit/s with a parallelism of 45 and a clock frequency of 300 MHz. Increasing the parallelism to 120 led to an average air throughput of 720 Mbit/s with a clock frequency of 400 MHz on CMOS technology.

***Index Terms*—** DVB-S2, layered decoder, Low-Density Parity-Check (LDPC) code, memory conflict, VLSI implementation,

1. INTRODUCTION

Low Density Parity-Check (LDPC) codes [1] have gained a lot of attention due to their remarkable error correcting capabilities. The design of structured codes [2] allows practical hardware implementation of LDPC decoders. This family of codes have been adopted in several standards such as the 2^{nd} Generation Satellite Digital Video Broadcast (DVB-S2) [3] ratified in 2005. More recently, the DVB-T2 and DVB-C2 standards have adopted the same family code as the DVB-S2 standard. In the following, DVB-X2 stands for DVB-S2, -T2 and -C2.

Even if the DVB-X2 standards define structured parity-check matrices, they are not perfectly structured for layered decoding because of the overlapped sub-matrices or Multiple-Diagonal Sub-Matrices (MDSM). These MDSMs lead to memory update conflicts in the *a posteriori* or Soft Output (SO) memories. The main idea of this paper is to solve the problem by processing twice the layers containing MDSMs and generating the appropriate memory control to cancel the MDSM effects. Layers with MDSM are repeated in such a way that the extra layers contribute also to the convergence of the code. The obtained scheduling can be viewed as a combination of the horizontal scheduling and the replica shuffle scheduling proposed by Zhang and Fossorier [4].

This paper is organized as follows: Section II is dedicated to describe the memory update conflicts and their resolution in the state-of-the-art. Section III describes the repeat process and its implementation. Finally, Section IV presents BER performance and synthesis results.

The authors thank NXP Semiconductors Caen for the funding of the study, the "Région Bretagne" and the "European Funds for Regional Development"(FEDER) for funding materials used in the study in the frame of the PALMYRE II project.

2. MEMORY UPDATE CONFLICTS DUE TO THE DVB-X2 MATRIX STRUCTURE

After a short review on the layered decoding algorithm, the memory update problem is explained and the existing work on this subject is presented. For further information on layered decoders, readers are invited to read [2] and [5].

2.1. Layered decoding algorithm

The layered algorithm divides each iteration in sub-iterations or layers and uses the intermediate Soft Output (SO) for each variable node $v = 1, \dots, N$. First, the SO_v are initialized with the channel LLR and messages from check node to variable nodes ($M_{c\to v}$) are initialized to zero. For each sub-iteration:

$$M_{v\to c}^{old} = SO_v^{old} - M_{c\to v}^{old}, \tag{1}$$

$$M_{c\to v}^{new} = 2\tanh^{-1}\Big(\prod_{v_c/v}\tanh(\frac{M_{v\to c}^{old}}{2})\Big), \tag{2}$$

$$SO_v^{new} = M_{v\to c}^{old} + M_{c\to v}^{new}, \tag{3}$$

where v_c/v is the set variable node indices connected to check c excluding v. Combining (1) and (3), we obtain:

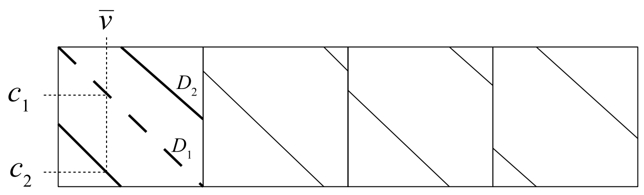

Fig. 1. One layer with one DDSM

$$SO_v^{new} = SO_v^{old} + \Delta M_{c\to v}, \tag{4}$$

where $\Delta M_{c\to v}$ is equal to $M_{c\to v}^{new} - M_{c\to v}^{old}$.

2.2. Memory update conflicts problem

Fig. 1 shows a layer ($\bar{L}$) with one Double-Diagonal Sub-Matrix (DDSM). Each square with one diagonal line represents a shifted Identity Matrix (IM). The first square has two diagonals D_1 and D_2 corresponds to a DDSM. Let us consider two check nodes (c_1 and c_2) which are connected through D_1 and D_2 to the same variable node denoted $\bar{v}$. During the processing of this layer, the $SO_{\bar{v}}$ value should benefits from c_1 according to

$$SO_{\bar{v}}^{c_1} = SO_{\bar{v}}^{old} + \Delta M_{c_1\to \bar{v}}, \tag{5}$$

and from c_2 according to

$$SO_{\bar{v}}^{c_2} = SO_{\bar{v}}^{old} + \Delta M_{c_2\to \bar{v}}, \tag{6}$$

and finaly contribution of c_1 and c_2 should give

$$SO_{\bar{v}} = SO_{\bar{v}}^{old} + \Delta M_{c_1\to \bar{v}} + \Delta M_{c_2\to \bar{v}}. \tag{7}$$

Since the SO are updated serially in the layered architecture, the $SO_{\bar{v}}^{c_2}$ will overwrite the $SO_{\bar{v}}^{c_1}$ value. As a consequence, the contribution of $M_{c_1\to \bar{v}}$ message is cut. This problem is usually called a “cutting edge”. The existance of “cutting edge” leads to performance degradation.

Many papers in the literature consider this problem [6, 5, 7, 8, 9, 10, 11, 12, 13, 14] and their solutions are briefly exposed in the following subsection.

2.3. Existing work

In [6, 5], the authors compute the $\Delta M_{c\to v}$ values and add an access to the SO^{new} value allowing for concurrent updates of the SO values. This solution requires an additional access to the SO value which significantly constrains the implementation. In [7] and [8] the $\Delta M_{c_1\to \bar{v}}$ value is computed only when there are DDSM. The selective $\Delta M_{c_1\to \bar{v}}$ calculation is efficiently applied to the Min-Sum algorithm with an horizontal layer decoder in [9] and with a vertical layered decoder in [10]. All the presented solutions based on the $\Delta M_{c_1\to \bar{v}}$ calculation require a modified layered decoder and the use of two barrel shifters.

In [11] the conflicts due to DDSM are efficiently solved by a parallel computation of the horizontal layers. However, this implementation is only possible with the Chinese Mobile Multimedia Broadcasting (CMMB) standard which provides matrices with a constant variable node degree of connection. In [12], an architecture supporting MDSM is designed for CMMB, however, when dealing with a single diagonal submatrix, the architecture is underused.

In [13, 14] the parallelism is first reduced to eliminate most of the DDSM, then layers with DDSM are modified with specific scheduling. In [13] the presented design is efficient but limited to a parallelism of $P = 40$. In [14] the authors describe a design without modification of the layered decoder architecture and using only one barrel shifter. However, simulations showed some degradation for a parallelism higher than $P = 45$. For this reason, we propose a new technique that successfully solves the matrix-structure conflicts, even when keeping the maximum level of parallelism (i.e. $P = 360$). The main objective of this work is to keep the data-path of the decoder as simple as possible (a single barrel shifter, a simple layered node processor) in order to obtain both high speed clock frequency and low area design. Then, starting from this constraint, we solved the DDSM memory conflicts using an appropriate control.

3. RESOLUTION OF THE MEMORY UPDATE CONFLICTS

In this section the repeat process dedicated to solve the DDSM problem is explained and an efficient architecture is deduced.

3.1. Principle of the repeat process

The idea is to perform twice the layers with one DDSM to allow update of the two diagonals serially. Let us consider the layer in Fig. 1 with two consecutive processing. $\bar{L}_1$ will refer to the first processing of the layer and $\bar{L}_2$ the second processing. The result of $\bar{L}_x$ on $SO_{\bar{v}}$ is defined as $SO_{\bar{v}}^{\bar{L}_x}$. The repeat process principle is exposed hereafter by focusing on the evolution of the $SO_{\bar{v}}$ value, then control signals required for the architecture are deduced.

During the first occurrence $\bar{L}_1$, the conflict occurs as described in Section 2.2: $SO_{\bar{v}}^{old}$ is replaced first by $SO_{\bar{v}}^{c_1}$ (5), then overwritten by $SO_{\bar{v}}^{c_2}$ (6). Thus $SO_{\bar{v}}^{\bar{L}_1}$ is updated according to

$$SO_{\bar{v}}^{\bar{L}_1} = SO_{\bar{v}}^{old} + \Delta M_{c_2\to \bar{v}}. \tag{8}$$

During $\bar{L}_2$, we consider a specific control process that disables the $SO_{\bar{v}}^{c_2}$ update of equation (6). As a consequence,

$SO_{\bar{v}}^{\bar{L}_2}$ is updated by applying equation (5) to $SO_{\bar{v}}^{\bar{L}_1}$ which gives

$$SO_{\bar{v}}^{\bar{L}_2} = SO_{\bar{v}}^{\bar{L}_1} + \Delta M_{c_1 \to \bar{v}}. \qquad (9)$$

The result of the layer repeat process is given by combining the result of $\bar{L}_1$ and $\bar{L}_2$ respectively (8) and (9):

$$SO_{\bar{v}}^{\bar{L}_2} = SO_{\bar{v}}^{old} + \Delta M_{c_1 \to \bar{v}} + \Delta M_{c_2 \to \bar{v}}. \qquad (10)$$

From (10), one can conclude that the repeat process allows the contribution of both diagonals D_1 and D_2 to SO_v during a decoding iteration.

The specific control process that disables $SO_{\bar{v}}^{c_2}$ update during $\bar{L}_2$ can be easily implemented by an on time Write Disable (WD) signal on the SO memory. To avoid inconsistency at the next iteration, during $\bar{L}_2$ and update of $M_{c_2 \to \bar{v}}$ in the $M_{c \to v}$ memory, the $M_{c \to v}$ memory must also be Write Disabled to keep $M_{c_2 \to \bar{v}}$ unchanged. The same important remark is valid during the $\bar{L}_1$ with $M_{c_1 \to \bar{v}}$. Note that equation (8) can also be obtained by a WD of the SO memory during D_1 update. To summarize, a Write Disable (WD) of the SO and $M_{c \to v}$ memories must be active during the processing of the diagonal D_1 of $\bar{L}_1$ and during the processing of diagonal D_2 of $\bar{L}_2$.

Note that during a decoding iteration, the repeat process implies that all the variables connected to a layer containing a DDSM are updated two times. If the repeated layer are separated by several layers, then the second processing will use more updated SO and thus, contribute more efficiently to the convergence of the decoder. With a well chosen layer scheduling, an iteration with $x\%$ of layers repetition will perform almost equivalent with $1 + x$ conventional iterations.

3.2. Conflict free decoder

The resolution of memory update conflicts due to pipeline [15] can be efficiently combined with the repeat process to design a conflict free decoder. This solution is performed in three steps: the first step applies the split process [15] up to the minimum level of parallelism needed to fulfill the speed requirement of the application; the second step solves the problem of the remaining DDSMs by repeating the concerned layers. Finally, the third step applies an efficient layer scheduling (including the duplicated layers) to solve conflicts due to pipeline. In case of remaining pipeline conflicts, an optional fourth step consisting in scheduling the variable group inside the layers [16] can be considered.

The complexity added to the control process to drive the WD signals does not impact the layered architecture. This allows the implementation of a high speed and high parallelism layered decoder such as the one presented in [17].

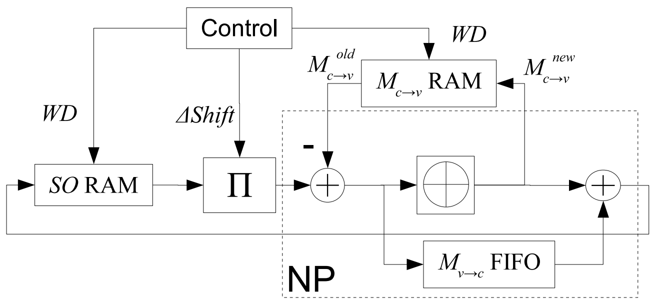

Fig. 2. Layered decoder architecture

3.3. Architecture

In Fig. 2, the layered decoder architecture is presented. The Node Processor (NP) computes (1), (2) and (3). Only one barrel shifter (Π) is implemented thanks to the computation of the shift variation $\Delta Shift$ as described in [14]. The modifications required to make the layered decoder architecture compliant with a WD architecture are described hereafter. In a layered decoder, the $M_{c \to v}$ memory can be made of a FIFO memory as a $M_{c \to v}$ update occurs only once during one iteration. In case of repeated layers, the $M_{c \to v}$ are updated more than once and the $M_{c \to v}$ read and write should be at the same address as during the first call. This can be implemented using a RAM and an ad hoc address generator. When a layer is repeated, an address identical to the first occurrence should be provided.

The WD signal connected to the SO RAM and the $M_{c \to v}$ RAM can be generated by detecting DDSM and layer repeat occurrence. In the proposed implementation, when two consecutive IM are connected to a common variable group then a DDSM is detected. A DDSM counter is then incremented to access in a specific ROM. The output of the ROM is a single bit that indicates if the current layer of this DDSM is the $\bar{L}_1$ or the $\bar{L}_2$ layer. The DDSM counter is set to zero at each new decoding iteration.

The presented modifications are included in the control block to generate appropriate control signals. These signals are processed in parallel to the layered decoder and do not increase the latency or the critical path of the layered decoder. Thus, the modifications retain the layered decoder efficiency. The splitting process [14], the layer scheduling [15] and the repeat process can be combined and give the designer the freedom to choose the check node update algorithm. Nevertheless, if the $M_{c \to v}$ are stored in a compressed way, as can be the case for the Min-Sum algorithm and its variations, then a special adaptation of the architecture is required.

3.4. Conflict resolution applied to compressed memories

If the $M_{c \to v}$ messages are compressed in the $M_{c \to v}^{MEM}$ memory, there are no more direct accesses to the $M_{c_1 \to \bar{v}}$ and $M_{c_2 \to \bar{v}}$ in the $M_{c \to v}$ memory to apply the WD signal.

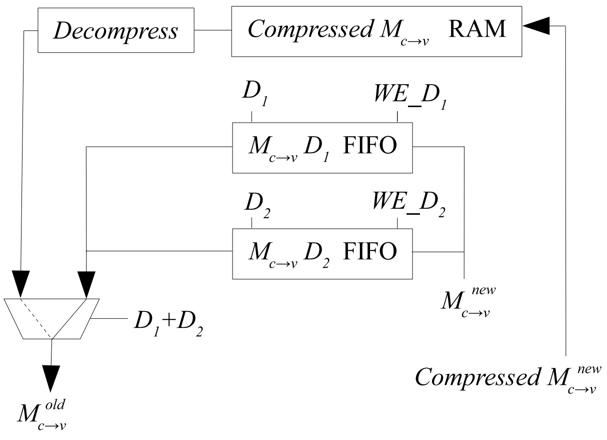

Fig. 3. Modified $M_{c\to v}$ memory

The proposed solution is to store in a compressed way all the $M_{v\to c}$ messages that do not belong to a DDSM, and to store individually the messages that belong to a DDSM. The modifications required to the $M_{c\to v}$ memory are presented in Fig. 3. Two additional memories, D_1 FIFO and D_2 FIFO are responsible to memorize respectively the $M_{c_1\to\bar{v}}$ and $M_{c_2\to\bar{v}}$ messages. D_1 FIFO stores $M^{new}_{c_1\to\bar{v}}$ during the process of layer L_2 (no overwrite of D_1)with signal WE_D_1 while D_2 FIFO stores $M^{new}_{c_2\to\bar{v}}$ during the process of layer L_1 with signal WE_D_2. The read process in the $M^{MEM}_{c\to v}$ implies three kinds of memory accesses:

1. **Normal access**: generation of the $M^{old}_{c\to v}$ message based on the compressed information and the current index of the message.
2. **D1 access**: the $M^{old}_{c_1\to\bar{v}}$ message is read in D_1 FIFO.
3. **D2 access**: the $M^{old}_{c_2\to\bar{v}}$ message is read in D_2 FIFO.

The over cost due to added memory depends on the number of DDSM. A decoder with $P = 360$ and compatible with all DVB-S2 code rates will require a maximum of 35 DDSM (code rate=5/6). The number of $M_{c\to v}$ messages ($\#M_{c\to v}$) which are required to be stored is given by $\#M_{c\to v} = 35 \times 360 \times 2 = 25200$ (over 285120 edges) . With $P = 120$, this number is reduced to $\#M_{c\to v} = 16 \times 120 \times 2 = 3840$ and with $P = 45$, $\#M_{c\to v} = 4 \times 45 \times 2 = 360$.

4. APPLICATION CASE

The WD architecture was simulated and implemented on an FPGA platform for validation purpose and synthesized on CMOS technology for comparison with state of art.

4.1. Simulation results

Fig. 4 illustrates fixed-point simulation results for a rate-2/3, standard frame (i.e. long frame) with the Normalized Min-

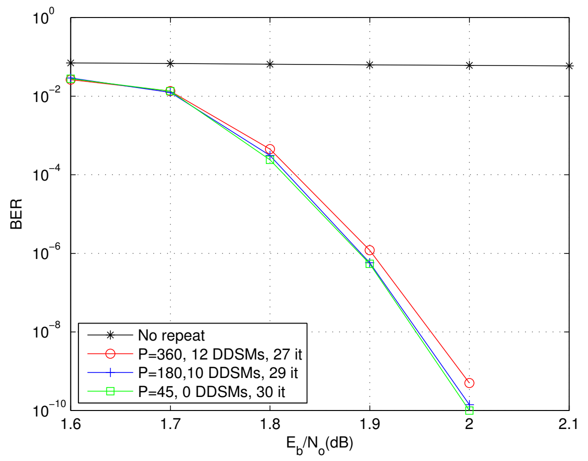

Fig. 4. BER for R=2/3 long frame DVB-T2 LDPC code for a decoding throughput equivalent to 30 decoding iterations without layer repetition

Sum algorithm. The first curve shows a layered decoder not taking into account the DDSMs. The performance obtained with a parallelism of $P = 45$ (no conflict) and $n_{it} = 30$ decoding iterations will be used as the reference curve for the LDPC code alone (the outer BCH code of the DVB-X2 standards is not considered in the simulation result). If n_{it} is constant for P=45, 180 and 360, then $P = 360$ gives best performances due to repeated layers, but throughput is also reduced due to repeated layers. To obtain a fair comparison, the number of iterations is reduced as a function of the proportion of added layers. With $P = 180$, 10 layers are repeated over the 240 layers. To obtain a constant throughput, the number of iterations is normalized by a factor $240/(240 + 10)$, which gives 29 decoding iterations. With a parallelism of 360 (12 DDSM and 120 layers, 27 iterations) at a BER of 10^{-10}, the curve is less than 0.05 dB away from the reference and at less than 1 dB from the Shannon limit. Note that for a parallelism of 360, there are sub-matrices with triple diagonals or even quadruple diagonals for some rates. In the case of triple (respectively quadruple) overlapped diagonals, the layer must be repeated three (respectively four) times with an appropriate WD of the memory to allow the effective update of a different diagonal for every repeated layer. Note that, thanks to the splitting process, all the triple diagonals are removed with a parallelism less than or equal to 180 [14].

4.2. Synthesis

The WD architecture with finite precision options described in [17] was synthesized on a Xilinx Virtex-V Pro FPGA (XQ5VLX110), for validation purposes. The system decodes long frames of code rates 1/2, 2/3, 3/4, 4/5 and 5/6. The average number of decoding iterations is reduced to 15 thanks

Paper	[8]	[19]	This
Parallelism	180	180	120
Algorithm	3-min	?	NMS
Throughput[Mbit/s]	180	135	720
Frequency[MHz]	270	174	400
Extrinsic [bits]	6	6	5
SO_{ram} [bits]	10	8	6
Channel [bits]	6	6	5
Buffer	?	?	yes
Capacity[Mbits]	2.68	3.18	2.2
BCH	yes	yes	no
Technologie[nm]	65	65	65
Total area[mm^2]	6.03	6.07	5.89

Table 1. Layered decoder CMOS implementation comparison

to the addition of an input buffer and the implementation of the stopping criteria [18]. For code rate 3/5, which has the most of edges, the average air throughput reaches 200 Mbit/s considering the clock frequency of 300 MHz after place and route. For comparison purposes, the decoder was also synthesized with the Synopsis Design Compiler based on Chartered CMOS 65nm technology.

The implementation of a layered decoder for the DVB-S2 standard was considered in [8] and [19]. Table 1 compares these implementations in terms of parallelism, air throughput, signal width and total area. In the proposed architecture, a buffer of size two (i.e. two RAM of size 64800 to store temporarily two input messages) is added to store the channel LLR values to halve the average number of iterations as described in [18]. To avoid conflicts due to pipeline, reduce the number of DDSM, reduce area and routing congestion, a parallelism of 120 is chosen. The memories used in the design have been synthesized using memory cells giving a total area of 5.54 mm^2 working at 500 MHz. The area occupied by the barrel shifter, the NPs processing elements and the control is only 0.35 mm^2 working at 500 MHz. We estimated that the full design can be clocked at 400 MHz which allows an average air throughput of 720 Mbit/s.

5. CONCLUSION

In this paper, we presented an efficient design of an LDPC decoder for the DVB-S2, -T2 and -C2 standard. The main contribution is to repeat the layer that contains DDSMs in such a way that each layer repetition contributes efficiently to the convergence of the code. This method allows the use of a layered decoder with one barrel shifter and no additional patch in the critical path. We performed two designs, the first targeted a Xilinx Virtex 5 FPGA with a parallelism of $P = 45$. Emulation results showed a decoding throughput of 200 Mbit/s. The second design targeted a CMOS 65 nm technology with a parallelism of $P = 120$. The preliminary results gave a total area of $5.89mm^2$ and a clock frequency of 400 MHz, which corresponds to a decoding throughput of 720 Mbit/s. For the maximum parallelism of $P = 360$, the repeat layer technique can solve the DDSM conflicts at a cost of a 0.05 dB when no throughput degradation is accepted.

6. REFERENCES

[1] R. Gallager, *Low-Density Parity-Check Codes*. PhD thesis, Cambridge, 1963.

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

[3] ETSI, "Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications (DVB-S2)," *EN 302 307 V1.2.1*, 2009.

[4] J. Zhang, Y. Wang, M. P. C. Fossorier, and J. S. Yedidia, "Iterative decoding with replicas," *IEEE Transactions on Information Theory*, vol. 53, pp. 1644–1663, May 2007.

[5] M. Rovini, F. Rossi, P. Ciao, N. L'Insalata, and L. Fanucci, "Layered decoding of non-layered LDPC codes," in *9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools*, (Dubrovnick, Croatia), pp. 537–544, Sept. 2006.

[6] E. Boutillon, J. Tousch, and F. Guilloud, "Ldpc decoder, corresponding method, system and computer program," *US Patent 7,174,495*.

[7] A. Segard, F. Verdier, D. Declercq, and P. Urard, "A DVB-S2 compliant LDPC decoder integrating the horizontal shuffle schedule," in *IEEE International Symposium on Intelligent Signal Processing and Communication Systems. ISPACS 2006.*, (Tottori, Japan), Dec. 2006.

[8] S. Muller, M. Schreger, M. Kabutz, M. Alles, F. Kienle, and N. Wehn, "A novel LDPC decoder for DVB-S2 IP," in *Conference & Exhibition on Design Automation & Test in Europe.*, (Nice, France), Apr. 2009.

[9] W. Ji, N. Hamaminoto, H. Nakayama, and S. Goto, "Data conflict resolution for layered LDPC decoding algorithm by selective recalculation," in *International Conference on Image and Signal Processing (CISP2010)*, (Yantai, China), pp. 2985–2989, Oct. 2010.

[10] L. Meng, C. Abdel Nour, C. Jego, and C. Douillard, "Design and FPGA prototype of a bit-interleaved coded modulation receiver for the DVB-T2 standard," in *IEEE workshop on Signal Processing System*, (Sam Francisco, USA), pp. 162–167, Oct. 2010.

[11] K. Guo, Y. Hei, and S. Qiao, "A parallel-layered belief-propagation decoder for non-layered ldpc codes," in *Journal of Communications*, vol. 5, pp. 400–408, May 2010.

[12] Y. Sun and J. Cavallaro, "Vlsi architecture for layered decoding of qc-ldpc codes with high circulant weight," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, 2012.

[13] X. Zhao, Z. Chen, X. Peng, D. Zhou, and S. GOTO, "DVB-T2 LDPC decoder with perfect conflict resolution," in *International Symposium on VLSI Design, Automation, and Test(VLSI-DAT)*, pp. 1–4, april 2012.

[14] C. Marchand, J.-B. Doré, L. Conde-Canencia, and E. Boutillon, "Conflict resolution by matrix reordering for DVB-T2 LDPC decoders," in *IEEE Internationa Conference on communications*, (Honolulu, USA), pp. 1–6, Nov. 2009.

[15] C. Marchand, J.-B. Doré, L. Conde-Canencia, and E. Boutillon, "Conflict resolution for pipelined layered LDPC decoders," in *IEEE Workshop on Signal Processing Systems*, (Tampere, Finlande), pp. 220–225, Nov. 2009.

[16] 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 *IFIP International Conference on Very Large Scale Integration, VLSI - SoC 2007*, (Atlanta, USA), pp. 236–241, Oct. 2007.

[17] C. Marchand, L. Conde-Canencia, and E. Boutillon, "Architecture and finite precision optimization for layered LDPC decoders," in *IEEE Workshop on Signal Processing Systems*, (San Francisco, USA), Oct. 2010.

[18] M. Rovini and A. Martinez, "On the addition of an input buffer to an iterative decoder for LDPC codes," in *IEEE 65th Vehicular Technologie Conference,VTC2007*, (Dublin, Ireland), pp. 1995–1999, Apr. 2007.

[19] P. Urard, L. Paumier, V. Heinrich, N. Raina, and N. Chawla, "A 360mw 105b/s DVB-S2 compliant codec based on 64800b LDPC and BCH codes enabling satellite- transmission portble devices," in *IEEE International Solid-State Circuits Conference (ISSCC)*, (San Francisco, USA), pp. 310–311, Feb. 2008.