LDPC decoder architecture for DVB-S2 and DVB-S2X standards

Cédric Marchand, Emmanuel Boutillon

LDPC decoder architecture for DVB-S2 and DVB-S2X standards

Cédric Marchand and Emmanuel Boutillon
Université Européenne de Bretagne, Lab-STICC, CNRS, UBS, BP 92116 56321 Lorient, France.
Email: cedric.marchand@univ-ubs.fr

***Abstract*—A particular type of conflict due to multiple-diagonal sub-matrices in the DVB-S2 parity-check matrices is known to complicate the implementation of the layered decoder architecture. The new matrices proposed in DVB-S2X no longer use such sub-matrices. For implementing a decoder compliant both with DVB-S2 and DVB-S2X, we propose an elegant solution which overcomes this conflicts relying on an efficient write disable of the memories, allowing a straightforward implementation of layered LDPC decoders. The complexity and latency are further reduced by eliminating one barrel shifter. Compared with the existing solutions, complexity is reduced without performance degradation.**

***Keywords*—Low-Density Parity-Check (LDPC) code, memory conflict, layered decoder, DVB-S2, DVB-S2X.**

I. Introduction

Low Density Parity-Check (LDPC) codes [1] have gained a great deal of attention due to their remarkable error correcting capabilities. The design of structured codes allows practical hardware implementations of LDPC decoders [2]. This family of codes has been adopted in several standards, such as the 2^{nd} Generation Satellite Digital Video Broadcast (DVB-S2) [3] ratified in 2005. In 2014, the DVB project standardized DVB-S2X [4] as an optional extension of the DVB-S2 standard.

Even if the DVB-S2 standard specifies structured parity-check matrices, they are not perfectly structured for layered decoding, due to the overlapped sub-matrices or Multiple-Diagonal Sub-Matrices (MDSM). These MDSMs lead to memory update conflicts in the a posteriori memories.

Many papers in the literature consider the MDSM problem [5], [6], [7], [8], [9]. However these solutions require a modified layered decoder with extra hardware and the use of two barrel shifters. In the reminder of this paper, we classify these solutions as “patched-hardware”. In contrast to the “patched-hardware”, in [10], [11], “patched-control” solutions require no overhead hardware, but lead to performance degradations or reduced throughput. These solutions do not require extra hardware because only the control sequence is modified. In [10], [9], [11], the number of MDSM is first significantly reduced by decreasing the parallelism. In [10], the remaining MDSMs are solved using intermediate dummy variable nodes. However, simulations showed Frame Error Rate (FER) degradation for a parallelism higher than $P = 45$. In [11], an appropriate memory control is combined with the repetition of layers to cancel the MDSM effects. However, the layer repetition significantly constraints the resolution of conflicts due to pipeline [12] when the parallelism is high. With the growing demand in throughput, an efficient solution for full parallelism ($P = 360$) is required. Hardware-patched solutions lead to significant area overhead while control-patched solutions lead to performance degradation.

In this paper, the control proposed in [11] is significantly simplified, and no layer repetition is required. As J.Hadamard said about mathematics, “simple ideas usually come last”. The main contribution of this paper is an elegant solution to generate the appropriate memory control to counteract the MDSM effects, without introducing observable performance degradations, even with full parallelism. The complexity and latency are further reduced compared with patched-hardware solutions by saving one barrel shifter. Finally, this solution allows an efficient implementation for DVB-S2X, while remaining backward compatible with DVB-S2.

This paper is organized as follows: Section II is dedicated to describe the memory update conflicts. In Section III, the write disable process is explained. In Section IV, the implementation of the write disable process is discussed. Finally, FER performances are presented in Section V.

II. Memory update conflicts due to the DVB-S2 matrix structure

After a short review on the layered decoding algorithm, the memory update problem is explained.

A. Layered decoding algorithm

The layered algorithm divides each iteration into 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 Log Likelihood Ratio (LLR) and messages from check node to variable nodes ($M_{c\to v}$) are initialized to zero. For each sub-iteration:

$$M_{v\to c}^{old} = SO_v^{old} - M_{c\to v}^{old}, \qquad (1)$$

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

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

where v_c/v is the set of variables connected to check c without variable v.

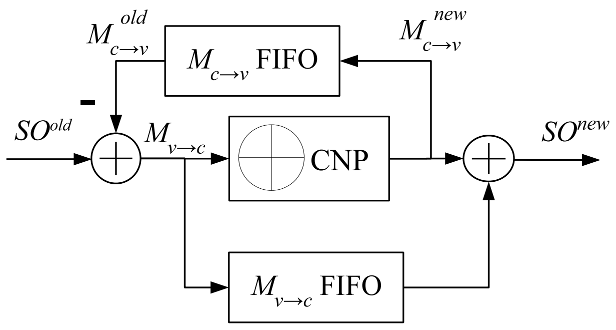

Fig. 1. Node processor in layered decoder architecture

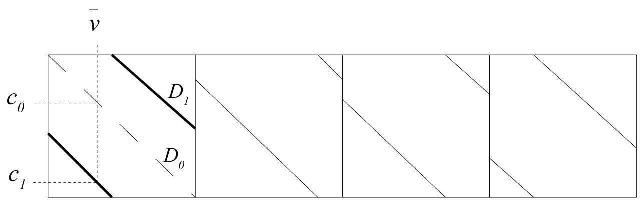

Fig. 2. Example of a layer with one DDSM

From these equations, the node processor architecture can be derived, as depicted in Fig. 1. The left adder of the architecture performs (1), while the right adder implements (3). The central block is in charge of the serial $M_{c\to v}$ updates (2).

Combining (1) and (3), we obtain:

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

where

$$\Delta M_{c\to v} = M_{c\to v}^{new} - M_{c\to v}^{old}. \tag{5}$$

B. Memory update conflicts problem

Fig. 2 shows an example of one layer with one Double-Diagonal Sub-Matrix (DDSM). Each square with one diagonal line represents a shifted Identity Matrix (IM). The square with two diagonals, D_0 and D_1, corresponds to a DDSM. Let us consider two check nodes (c_0 and c_1), which are connected through D_0 and D_1 to the same VN, denoted $\bar{v}$. During the processing of this layer, the $SO_{\bar{v}}$ value are updated twice, according to:

$$SO_{\bar{v}}^{new_0} = SO_{\bar{v}}^{old} + \Delta M_{c_0\to\bar{v}} \tag{6}$$

$$SO_{\bar{v}}^{new_1} = SO_{\bar{v}}^{old} + \Delta M_{c_1\to\bar{v}} \tag{7}$$

However, since the SO is updated serially in the layered architecture, the $SO_{\bar{v}}^{new_1}$ will overwrite the $SO_{\bar{v}}^{new_0}$ value. As a consequence, the contribution of the $M_{c_0\to\bar{v}}$ message is left out, leading to a significant performance degradation.

	Standard frame			Short frame		
CR	IM	DD	TD	IM	DD	TD
1/4	540	3	0	135	4	0
1/3	600	13	0	150	4	0
2/5	648	8	0	162	8	0
1/2	630	8	0	135	8	0
3/5	792	29	3	162	0	0
2/3	600	12	0	150	14	0
3/4	630	21	1	132	9	0
4/5	648	32	1	125	9	0
5/6*	660	32	2	137	19	1
8/9	540	30	0	135	20	0
9/10	540	36	0	x	x	x

TABLE I. NUMBER OF DDSM AS A FUNCTION OF CODE RATE

Table I shows the number IM, the number of DDSM (DD) and triple-diagonals sub-matrices (TD) as a function of code rates (CR) and frame size (normal or short) for the DVB-S2 standard. The asterisk for code rate 5/6 indicates the presence of one quadruple-diagonal sub-matrix for normal frame. The DVB-T2 standard uses same matrices as the DVB-S2 standard for code rates 1/2, 3/5, 2/3, 3/4, 4/5, 5/6 at normal frame and for code rate 1/4, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6 at short frame size. Also, the DVB-C2 standard uses same matrices as the DVB-S2 matrices for code rate 3/4, 4/5, 5/6, 9/10 at normal frame size and code rates 1/2, 2/3, 3/4, 4/5, 5/6, 8/9 at short frame size. A new matrix common to DVB-C2 and DVB-T2 with 12 DDSM has been defined for code rate 2/3, at normal frame size. Note that, due to the splitting process [10], all the triple diagonals are removed with a parallelism less than or equal to 180, and the number of DDSM is reduced significantly. The DVB-S2X [4] defines 24 optional matrices for normal frame size, 7 optional matrices for short frame size and 3 optional matrices for a new medium frame of size N=32400. None of these matrices have a DDSM allowing straightforward and efficient implementations. Thanks to the write disable process, a design optimized for DVB-S2X matrices would only require a control of the write disable memory signal to be backward compatible with DVB-S2 matrices.

III. RESOLUTION OF THE MEMORY UPDATES CONFLICTS

A. Existing patched-hardware solutions

In [6], the proposed Delta-based architecture compute the $\Delta M_{c\to v}$ value and then SO is updated. (7) is modified as follows:

$$SO_{\bar{v}}^{new_1} = SO_{\bar{v}}^{new_0} + \Delta M_{c_1\to\bar{v}}, \tag{8}$$

and replacing $SO_{\bar{v}}^{new_0}$ by (6) gives:

$$SO_{\bar{v}}^{new} = SO_{\bar{v}}^{old} + \Delta M_{c_0\to\bar{v}} + \Delta M_{c_1\to\bar{v}}. \tag{9}$$

The SO value thus benefits from c_0 and c_1. To make possible the consecutive SO update, a specific SO memory is required to allow two read accesses (1 and 8) and one write access (8). In [7], the same principle is used but only when a MDSM is detected and a local memory is dedicated to save intermediate SO results (7). In [8], (6) and (7) are used as intermediate results to compute SO as follow:

$$SO_{\bar{v}}^{new} = SO_{\bar{v}}^{new_0} + SO_{\bar{v}}^{new_1} - SO_{\bar{v}}^{old} \tag{10}$$

giving same result as in (9).

In [9], layers with Double Diagonal Sub-Matrice (DDSM) are divided in two sub-layers which are decoded concurrently with appropriate scheduling. The result of the first sub-layer is used by the second sub-layer. To make this possible the serial scheduling of sub-layers are modified and two idle cycles are added. This solution mainly relies on control and only a minimal hardware is added to bypass SO values from first sub-layer to second sub-layer. However, this solution cannot deal with Triple Diagonal Sub-matrices limiting maximum parallelism to $P = 180$ and two barrel shifters are required.

B. Existing patched-control solutions

In [10], a design without modification of the layered decoder architecture is described, using only one barrel shifter. The MDSMs are solved using intermediate dummy variables slowing down the belief propagation process. Simulations showed significant degradations when targeting a parallelism higher than $P = 45$.

In [11] a solution based on layer repetition and write disable of the memory is presented. The layer repetition increases the decoding latency while improving the FER performance. Simulation at normalized decoding duration shows slight performance degradation. Also, the repetition of layers complicates the resolution of memory update conflicts due to pipeline [12] because repeated layers have to be separated by at least one layer to avoid conflicts.

C. Principle of the Write Disable process

Let us consider the layer with one DDSM in Fig. 2 where $SO_{\bar{v}}$ is updated twice, first by c_0 (6), then by c_1(7). If nothing is done, (7) overwrite (6) and $SO_{\bar{v}}$ only benefits from c_1. If a process disables the $SO_{\bar{v}}$ update during (7), then $SO_{\bar{v}}$ only benefits from c_0. With a simple disable process, one can decide either $SO_{\bar{v}}$ benefits from c_0 or c_1.

The idea is to differentiate odd and even iterations. During the even iterations (iterations 0, 2, 4 ...), the writing operation in the SO RAM of (7) is skipped thanks to a Write Disable (WD) signal. SO_{even} is thus updated according to:

$$SO_{\bar{v}}^{even} = SO_{\bar{v}}^{old} + \Delta M_{c_0 \to \bar{v}} \tag{11}$$

During the odd iterations, the writing operation in the SO RAM of (6) is skipped and SO_{odd} is updated according to:

$$SO_{\bar{v}}^{odd} = SO_{\bar{v}}^{old} + \Delta M_{c_1 \to \bar{v}} \tag{12}$$

One can conclude that the write disable process enables the contribution of diagonals D_0 and D_1.

It is important to note that in order to avoid inconsistencies during the next iteration, the $M_{c \to v}$ memory must also be Write Disabled synchronously with the SO memory.

To summarize, the Write Disable signal *WD* connected to the SO and $M_{c \to v}$ memories are active in case of even iterations and diagonals D_0 or in case of odd iterations and diagonals D_1. For TD and QD, the WD process can be described using Algorithm 1 where D is the number of diagonals (2 for DD, 3 for TD and 4 for QD) and d be the diagonal number.

Algorithm 1 pseudocode for write disable signal

```
WD ← 0
for all $d \in D$ do
    if $it \mod D \neq d$ then
        WD ← 1
    end if
end for
```

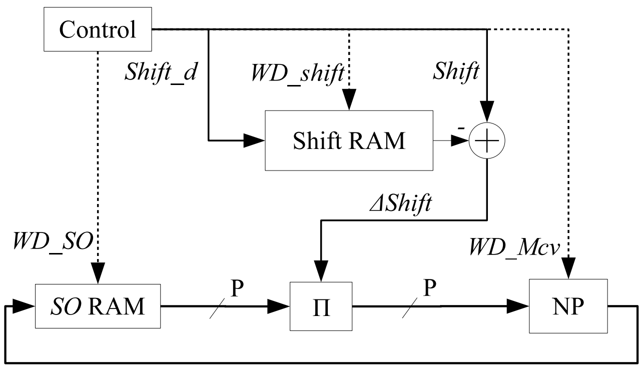

Fig. 3. Layered decoder architecture

This control over the *WD* signal is added to the existing control process and does not impact the layered architecture complexity. This control enables the implementation of a high speed and high parallelism layered decoder.

D. Architecture

In Fig. 3, the layered decoder architecture is presented. The Node Processor (NP) computes (1),(2) and (3) as shown in Fig. 1. Only one barrel shifter (Π) is implemented thanks to the computation of the shift variation $\Delta Shift = Shift - Shift^{old}$ [10]. As a reminder, the P SO values are not shifted back in the SO RAM, but the shift value is saved in the *Shift* RAM. During the next access to the P SO values, the SO values are shifted of the required shift value minus the previous shift saved in the *shift* RAM. The modifications required to make the layered decoder architecture compliant with the WD process are control signals depicted using dashed lines in Fig. 3. The *WD* signal from Algorithm 1 generates the *WD_SO*, *WD_shift* and *WD_Mcv* signals. These signals are connected respectively to the WD control input of memories SO RAM, *Shift* RAM and $M_{c \to v}$ FIFO with the appropriate delays. These signals are processed in parallel to the layered decoder and do not increase the latency, nor the critical path of the layered decoder. Thus, the modifications retain the layered decoder efficiency.

The $M_{c \to v}$ FIFO should be modified such that the *WD_Mcv* signal write disable the FIFO memory while allowing incrementing the write pointer. In case of Min-Sum algorithm and its variations, the $M_{c \to v}$ values connected to one check node can be compressed. A specific memory for the write disabled values should then be added as shown in Fig. 4, where the *WD_Mcv_d2* signal is a delayed version of *WD_Mcv* and the size of the FIFO is given by the number of MDSM.

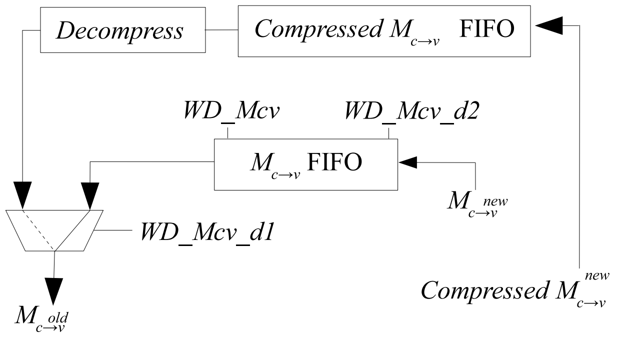

Fig. 4. Modified $M_{c\to v}$ memory for compresed $M_{c\to v}$

Stratix V	LC Combinational	LC Registers	Memory bits
1 Node Processor	257	388	0
1 FIFO $m_{c\to v}$	0	0	11088
180 Nodes	46298	69841	1995840
Control	415	45	0
ROM_VG_Shift	5100	17	0
ping-pong SO RAM	3361	1081	1036800
Barrel shifter	8320	4328	0
Total	63495	75460	3032640

TABLE II. SYNTHESIS RESULTS FOR DVB-S2 LDPC DECODER

In our multi-rate compliant decoder implementation, we consider uncompressed $M_{c\to v}$ memory mainly for two reasons. Fist, the width and length of compressed $M_{c\to v}$ memory vary significantly with code rate, which reduce memory saving in case of a multi-rate compliant decoder [13]. Second, P compression and decompression units are saved when using uncompressed $M_{c\to v}$ memory.

IV. SIMULATION RESULTS AND COMPLEXITY COMPARISON

A. Simulation results

Fig. 5 illustrates bit-true C simulation results for normal frame, P=360, 3-min algorithm, QPSK modulation, intrinsic quantized on 6 bits, SO quantized on 8 bits, extrinsic quantized on 6 bits, saturation process as in [13] and outer BCH decoder. Simulations are performed on code rates of 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 5/6 and 9/10 from DVB-S2 standard with maximum number of iterations (it_{max}) equal to 45, 40, 35, 35, 35, 35, 35, 35 and 35, respectively and on code rates of 13/45, 9/20 and 11/20 from DVB-S2X standard with it_{max} equal to 45, 45 and 35 respectively. The DVB-S2 and DVB-S2X ideal E_s/No performance requirement at Quasi Error Free (QEF) are also presented. In the DVB-S2 standard, QEF is defined at Packet Error Rate (PER) = 10^{-7} where one packet is a 188×8 bits MPEG transport stream packet. For practical reasons, we represent here the QEF as defined in the DVB-S2X standard at FER=10^{-5}. Simulation shows performances greater than or equal to the standard requirements.Simulation showed no performance degradation compared with "patched-hardware" simulations based on architecture described in [6].

B. Synthesis results

The architecture presented in Fig. 3 was synthesized on a Stratix-V FPGA (5SGXEA7N2F45C2) from Altera, for

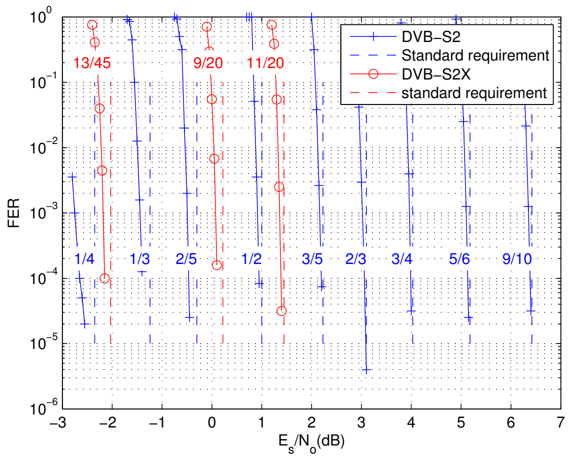

Fig. 5. Normal frame simulation results

validation purposes. The system decodes long frames of all DVB-S2 code rates with $P = 180$. Table II gives the hardware resources required by entities. In the ROM_VG_Shift entity, the Variable Group number and shift value is stored for each IM of each code rates. The barrel shifter is pipelined in three clock cycles. The "ping-pong SO RAM" instantiates two SO RAM allowing the decoder to decode a frame on one SO RAM while the other RAM read new intrinsic value for next frame and give sign of SO values decoded during previous frame. The SO update latency L_{SO} is equals to the read SO latency (1 cycle) plus the shifter latency (3 cycles) plus the NP latency(d_c + 4 cycles) plus one cycle to write the new SO value giving $L_{SO} = 9$.

The maximum clock frequency (F_{clk}) is 250 MHz, the decoder is fully pipelined and the maximum throughput is given by

$$D = \frac{K \times F_{clk}}{d_c \times \frac{M}{P} \times it_{180} + L_{SO} + L_{init}} bit.s^{-1} \qquad (13)$$

Where L_{init} is the latency to write intrinsic to the SO RAM and read hard decision results. This process is pipelined thanks to the "ping-pong RAM" and takes just one cycle delay. For code rate 1/4, 1/2, 3/4 and 9/10, the obtained throughput is respectively 129, 335, 792 and 1405 $Mbit.s^{-1}$.

C. Complexity comparison

In terms of complexity, the "patched-control" method eliminates the use of one barrel shifter and a hardware patch.

In [5] the conflict is solved by recalculation which requires additional hardware and delay. In [6], [7], [8], the conflict resolution strategies are based on $\Delta M_{c\to v}$ computation and specific hardware to update the SO with the $\Delta M_{c\to v}$ value. The $\Delta M_{c\to v}$ value is computed using two subtractions (1) and (5), then the SO value is updated using a third adding/subtraction operation (4). In this paper, the SO update is done in one subtraction (1) and one addition (3). "Patched-control" eliminates for each check node at least one addition/subtraction operation. We modified our NP as in [8] to obtain a "patched-hardware" decoder. The patch overhead is limited to 8 LC Combinational and 16 LC Registers per NP. However, with P=180 and adding one barrel shifter the overhead is 9760(15%) LC Combinational and 7208(9.5%) LC Registers. The complexity reduction also leads to power reduction. The SO update latency is also increased by the hardware patch (2 cycles) and one barrel shifter giving $L_{SO} = 14$. The increased SO update complicates resolution of conflicts due to pipeline [12].

V. Conclusion

In this paper, we have presented an efficient LDPC decoder design for the DVB-S2 and DVB-S2X standards. The main contribution is an elegant control process which solves the memory conflict problem inherent to the DVB-S2, DVB-T2 and DVB-C2 standards. The improvement is achieved without undergoing modifications to the layered architecture, and without constraining resolution of conflicts due to pipelining. The write disable process is also combined with the delta shift process to further save one barrel shifter. The proposed solution is interesting when designing a DVB-S2 decoder, a DVB-S2X decoder and a decoder compatible with both DVB-S2X and DVB-S2 matrices.

References

[1] R. Gallager, "Low-density parity-check codes," Ph.D. dissertation, Cambridge, 1963.

[2] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *IEEE Workshop on Signal Processing Systems*, Austin, USA, Oct. 2004, pp. 107–112.

[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] ——, "Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications ; part ii: S2-extensions (DVB-S2X) - (optional)," *DVB Document A83-2*, 2014.

[5] 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, Oct. 2010, pp. 2985–2989.

[6] 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, Sep. 2006, pp. 537–544.

[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] 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)*, april 2012, pp. 1–4.

[10] 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, Nov. 2009, pp. 1–6.

[11] C. Marchand, L. Conde-Canencia, and E. Boutillon, "High-speed conflict-free layered ldpc decoder for the dvb-s2, -t2 and -c2 standards," in *Signal Processing Systems (SiPS), 2013 IEEE Workshop on*, Oct 2013, pp. 118–123.

[12] 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, Nov. 2009, pp. 220–225.

[13] C. Marchand, L. Conde-Canencia, and E. Boutillon, "Architecture and finite precision optimization for layered LDPC decoders," *Journal of Signal Processing Systems*, vol. 65, pp. 185–197, 2011.