ARCHITECTURE AND FINITE PRECISION OPTIMIZATION FOR LAYERED LDPC DECODERS

Cédric Marchand[*†], *Laura Conde-Canencia*[†], *Emmanuel Boutillon*[†]

[*]NXP Semiconductors, Campus Effiscience, Colombelles BP20000 14906 Caen cedex 9, France.
[†]Université Européenne de Bretagne, Lab-STICC, CNRS, UBS, BP 92116 56321 Lorient, France.
Email: cedric.marchand@univ-ubs.fr

ABSTRACT

Layered decoding is known to provide efficient and high-throughput implementation of LDPC decoders. In the practical hardware implementation of layered decoders, the performance is strongly affected by quantization. The finite precision model determines the area of the decoder, which is mainly composed of memory, especially for long frames. To be specific, in the DVB-S2,-T2 and -C2 standards, the memory can occupy up to 70% of the total area. In this paper, we focus our attention on the optimization of the number of quantization bits. Message saturation and memory size optimization are considered for the case of a DVB-S2 decoder. We show that the memory area can be reduced by 28% compared to the state-of-the-art, without performance loss.

***Index Terms*—Low-density parity-check (LDPC) code, layered decoding, VLSI implementation, DVB-S2.**

I. INTRODUCTION

Low Density Parity-Check (LDPC) codes were initially proposed by Gallager in the early 60's [1] but they were not used for three decades mainly because the technology was not mature enough for practical implementation. Rediscovered by MacKay [2] in 1995 with a moderate decoding complexity, LDPC codes are now included in many standards. Among the existing standards, we can distinguish standards using short frames (648, 1296 and 1944 bits for Wi-Fi) and standards using long frames (16200 and 64800 bits for DVB-S2). The use of long frames makes it possible to get closer to the Shannon limit, but leads to delays that are not suitable for internet protocols or mobile phone communications. On the other hand, long frames are suitable for streaming or Digital Video Broadcasting (DVB). The 2^{nd} Generation Satellite Digital Video Broadcast (DVB-S2) standard was ratified in 2005, the 2^{nd} Generation Terrestrial DVB (DVB-T2) standard was adopted in 2009 and the 2^{nd} Generation Cable DVB (DVB-C2) will be adopted during 2010. These three DVB standards include a common Forward Error Correction (FEC) block. The FEC is composed of a BCH codec and an LDPC codec. The FEC supports eleven code rates for the DVB-S2 standard and is reduced to six code rates for the DVB-T2 standards. The LDPC codes defined by the DVB-S2,-T2,-C2 standards are structured codes or architecture-aware codes (AA-LDPC [3]) and can be efficiently implemented using the layered decoder architecture [4], [5] and [6]. The layered decoder benefits from three architecture improvements: parallelism of structured codes, turbo message passing, and Soft-Output (SO) based Node Processor (NP) [4], [5] and [6].

Even if the state-of-the-art of the decoder architecture converges to the layered decoder solution, the search of an efficient trade-off between area, cost, low consumption, high throughput and high performance make the implementation of the LDPC decoder still a challenge. Furthermore, the designer has to deal with many possible choices of algorithm, parallelism, quantization parameters, code rates and frame lengths. In this article, we study the optimization of the memory size. We consider the DVB-S2 standard to compare results with the literature but our work can also be applied to the Wi-Fi and WiMAX LDPC standards or, more generally, to any layered LDPC decoder.

A first step concerning the memory reduction is the choice of a sub-optimal algorithm. The already well-known Min-Sum algorithm [7] and its variants significantly reduce the memory needs by the compression of the extrinsic messages. Another way to reduce the memory needs is to limit the word size by saturation. In the state-of-the-art, the way the SO and the extrinsic messages are saturated is rarely explicitly explained. In this article, we provide some discussion on efficient saturation of the extrinsic messages and the SO values. We present some ideas leading to significant memory savings. To complete the discussion, we also introduce a methodology to optimize the size of the extrinsic memory.

The paper is organized as follows: Section II presents the layered decoder and the Min-Sum sub-optimal algorithm. In Section III, we explain the saturating process. Section IV deals with the optimization of the size of the extrinsic memory. Finally, simulation and synthesis results are provided in Section V.

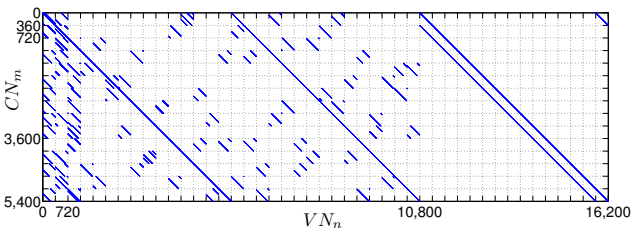

Fig. 1. Block-structured rate-$2/3$ DVB-S2 matrix (N=16200)

II. LDPC LAYERED DECODER

An LDPC code is defined by a parity check matrix $\mathbf{H}$ of M rows by N columns. Each column in $\mathbf{H}$ is associated to a bit of the codeword or Variable Node (VN), and each row corresponds to a parity check equation or Check Node (CN). A nonzero element in a row means that the corresponding bit contributes to this parity check equation. Fig. 1 shows the structure of the rate-$2/3$ short-frame DVB-S2 LDPC parity check matrix. This structured matrix is composed of shifted identity matrices, allowing for efficient parallel decoding.

II-A. Horizontal layered decoder

In the horizontal layered decoder, a VN is represented by its SO value (SO_v). This value is first initialized by the channel Log-Likelihood Ratio (LLR$= \log(P(v=0)/P(v=1))$). Then the decoding proceeds iteratively until all the parity checks are verified or a maximum number of iterations is reached. For layered decoding, one iteration is split into sub-iterations, one for each layer. A layer corresponds to one or several CNs and a sub-iteration consists in updating all the VNs connected to the CNs of the layer. The update of the VNs connected to a given CN is done serially in three steps. First, the message from a VN to a CN ($M_{v\to c}$) is calculated as:

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

The second step is the serial $M_{c\to v}$ update, where $M_{c\to v}$ is a message from CN to VN, and is also called extrinsic. Let v_c be the set of all the VNs connected to CN c and v_c/v be v_c without v. For implementation convenience, the sign and the absolute value of the messages $|M^{new}_{c\to v}|$ are updated separately:

$$sign(M^{new}_{c\to v}) = \prod_{v'\in v_c/v} sign(M_{v'\to c}) \tag{2}$$

$$|M^{new}_{c\to v}| = f\left(\sum_{v'\in v_c/v} f(|M_{v'\to c}|)\right) \tag{3}$$

where $f(x) = -\ln\tanh\left(\frac{x}{2}\right)$. The third step is the calculation of the SO_{new} value:

$$SO^{new}_v = M_{v\to c} + M^{new}_{c\to v} \tag{4}$$

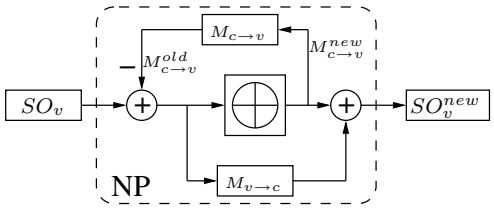

Fig. 2. SO based Node Processor

The updated SO^{new}_v value can be used in the same iteration by another sub-iteration leading to convergence which is twice as fast as the flooding schedule [3].

II-B. Architecture overview

From equations (1) to (4), the Node Processsor (NP) architecture shown in Fig. 2 can be derived. The left adder of the architecture performs equation (1) and the right adder performs equation (4). The central part is in charge of the serial $M_{c\to v}$ update.

As the structured matrices are made of identity matrices of size P, the computation of P CNs is done in parallel. Hence, the layered decoder architecture is based on P NPs that first read serially the Groups of P VNs linked to one layer and then write back the SO^{new}_v in the VNs.

II-C. The normalized Min-Sum algorithm and other related algorithms

Equation (3) can be implemented using a sub-optimal algorithm such as the Min-Sum algorithm [7], the normalized Min-Sum algorithm, the Offset Min-Sum algorithm, the A-min* algorithm, the λ-min algorithm [8] and related [9]. The advantages of these algorithms are the simplified computation of equation (3) and the compression of the $M_{c\to v}$ messages. Although all these algorithms present different performances, the memory space they require to store the $M_{c\to v}$ messages is identical (considering $\lambda = 2$ for the λ-min algorithm). Hence, without loss of generality, for the rest of the paper, we will consider in the normalized Min-Sum algorithm. With this algorithm, equation (3) becomes:

$$|M^{new}_{c\to v}| = \alpha \min_{v'\in v_c/v} |M_{v'\to c}| \tag{5}$$

where α is the normalization factor, $0 < \alpha \leq 1$.

The CN generates two different values: min and $submin$. The min value is the normalized minimum of all the incoming $M_{v\to c}$ values and the $submin$ is the second normalized minimum. Let ind_{min} be the index of the minimum. For each $|M^{new}_{c\to v}|$ values, if the index of $M^{new}_{c\to v}$ is ind_{min} then $|M^{new}_{c\to v}| = submin$ else $|M^{new}_{c\to v}| = min$. The $M_{c\to v}$ from one CN can be compressed with four elements, i.e. min, $submin$, ind_{min} and $sign(M^{new}_{c\to v})$. For matrices with a check node degree greater than four, this compression leads to significant memory saving.

III. SATURATION

An SO value is the sum of the channel LLR with all the incoming extrinsic messages. Considering the case of an LDPC code of the DVB-S2 standard, the maximum variable node degree (d_v) is 13. If the channel LLR and the $M_{c\to v}$ are quantized on 6 bits, the SO values must be quantized on at least 10 bits to prevent overflows. However, to avoid prohibitive word size, efficient saturation of the $M_{c\to v}$ and SO values should be considered.

III-A. The problem of SO saturation

Let us consider the saturation case where $SO_{max} < SO_v^{new}$ during the SO update (4). A saturation process will bound SO_v^{new} to the SO_{max} value. This will introduce an error ϵ_v in the SO_v^{new} value ($\epsilon = SO_v^{new} - SO_{max}$). During the next iteration, the new $M'_{v\to c}$ value will be $M'_{v\to c} = SO_v - M_{c\to v} = M_{v\to c} - \epsilon_v$.

Let us consider the worst case: during an iteration, SO_v is saturated at $+SO_{max}$, each CN confirms a positive $M_{c\to v}$ value, and d_v=13 (i.e. SO_v is saturated 13 times). At the beginning of the next iteration, $SO_v = SO_{max}$. From (1) and (4), we can deduce that $SO_{new} = SO_{old} + \Delta M_{c\to v}$ where $\Delta M_{c\to v} = M_{c\to v}^{new} - M_{c\to v}^{old}$. If $\Delta M_{c\to v} < 0$, the SO value decreases. The SO value can even decrease 13 times and change its sign. To summarize, when there is saturation, the SO value cannot increase, but can decrease. The saturation introduces a nonlinearity that can produce pseudo-codewords and an error floor. A solution has to be found to overcome this problem.

III-B. A solution for SO saturation

The solution that we propose was first introduced in [10] and relies partially on the A Priory Probability (APP) based decoding algorithm [7]. The APP-variable decoding algorithm simplifies equation (1) to:

$$M_{v\to c} = SO_v \tag{6}$$

which greatly reduces the architecture complexity but introduces significant performance loss. The idea is to use equation (6) only when there is saturation. This leads to the APP-SO saturation algorithm, which is described as follows:

Algorithm 1 APP-SO saturation algorithm

```
if SO_v = SO_max then
    M_{v→c} = SO_v
else
    M_{v→c} = SO_v − M_{c→v}
end if
```

$Rate$	M	W_{Sign}	W_{Ind}	$W_{Mc\to v}$	Memory
1/4	48600	4	2	14	680400
1/3	43200	5	3	16	691200
2/5	38880	6	3	17	660960
1/2	32400	7	3	18	583200
3/5	25920	9	3	21	544320
2/3	21600	10	4	22	475200
3/4	16200	14	4	26	421200
4/5	12960	18	5	31	401760
5/6	10800	22	5	35	378000
8/9	7200	27	5	40	288000
9/10	6480	30	5	43	278640

Table I. Memory size of extrinsic

III-C. Saturation of the extrinsic messages

The $M_{c\to v}^{new}$ value is directly used to compute SO_v^{new} from equation (4); any saturation on this value would not produce area savings and would degrade performance. On the contrary, the $M_{c\to v}^{old}$ value is calculated from a stored $M_{c\to v}^{new}$ value and is used only once during an iteration. Saturation of the stored $M_{c\to v}^{old}$ value is much less critical and leads to memory savings.

IV. OPTIMIZING THE SIZE OF THE EXTRINSIC MEMORY

The extrinsic memory size requirements strongly depend on the coding rate. This section focuses on the design of an optimal implementation for eleven different code rates.

IV-A. Memory size

The memory requirements of each CN is determined by the $M_{c\to v}^{old}$ messages needed for the CN computation. In the case of the normalized Min-Sum algorithm, the $M_{c\to v}^{old}$ values are compressed with min, $submin$, ind_{min} and $signM_{c\to v}$. In terms of memory, one address must be allocated for every CN which means that the RAM address range (R_{RAM}) is given by the number of CNs (M). The RAM word size (W_{RAM}) is given by the size of the compressed $M_{c\to v}^{old}$ values. If we denote by W_h the word size of h, then $W_{Mc\to v} = W_{|min|} + W_{|submin|} + W_{ind} + W_{sign}$. Table I presents the required memory capacity ($M \times W_{Mc\to v}$) for each rate. To calculate $W_{Mc\to v}$, we fix the value of $W_{|min|}$ and $W_{|submin|}$ to 4. To deal with the eleven code rates of the standard, a simple implementation would define R_{RAM} with the maximum M value, and W_{RAM} with the maximum $W_{Mc\to v}$ in Table I. Here, the total memory capacity would give: $48600 \times 43 = 2089800$ bits. For rate 1/4, 67% of word bits are wasted but addresses are fully used. On the other hand, for rate 9/10, word bits are fully used but 86 % of the addresses are wasted. Theoretically, a memory size of 691200 bits would be enough to cover all the rates. An implementation solution has to be found for a better utilization of the memory.

$Rate$	M	$W_{Mc\to v}$	n_{cycles}	R_{RAM}
1/4	48600	14	2	97200
1/3	43200	16	2	86400
2/5	38880	17	2	77760
1/2	32400	18	2	64800
3/5	25920	21	3	77760
2/3	21600	22	3	64800
3/4	16200	26	3	48600
4/5	12960	31	4	51840
5/6	10800	35	4	43220
8/9	7200	40	5	36000
9/10	6480	43	5	32400

Table II. Memory capacity of the extrinsic message with $W_{RAM} = 9$

IV-B. Optimization principle

The idea is to add flexibility to both the address range and the word size. For this, we benefit from the fact that the RAM that stores the compressed $M_{c\to v}^{old}$ value is needed only once per layer. As the delay to compute the next layer is of d_c cycles, we can use up to d_c cycles to fetch the data in the memory. A word can be split into two if we take two cycles to fetch the data, and split in three if we take three cycles. If we consider a single port RAM to implement the memory, up to $\lfloor d_c/2 \rfloor$ cycles can be used to read data, and $\lfloor d_c/2 \rfloor$ cycles to write new data.

Let us consider the example of a memory bank of size $48600(R_{RAM}) \times 22(W_{RAM})$. In a first configuration, where one cycle is used, we have a memory size of 48600×22 which fits to rates $1/4$, $1/3$, $1/2$, $3/5$, and $2/3$. In a second configuration, where two cycles are used, and two words of size 22 are fetched at consecutive addresses, we have the equivalent of a memory of size 24300×44 which fits to rates $3/4$, $4/5$, $5/6$, $8/9$ and $9/10$. The total memory size for the two-cycle option is equal to $48600 \times 22 = 106920$ bits. This constitutes a memory savings of 50% compared to the straightforward implementation.

IV-C. Results

The previously described process can be used for different word sizes. Table II gives an example with $W_{RAM} = 9$. For each rate, the number of cycles is given by $n_{cycles} = \lceil W_{Mc\to v}/W_{RAM} \rceil$, and R_{RAM} is deduced from $R_{RAM} = n_{cycles} \times M$. The global RAM range ($R_{RAM}^{global}$) is given by the maximum R_{RAM} in Table II and the total memory capacity is $R_{RAM}^{global} \times W_{RAM} = 97200 \times 9 = 874800$ bits.

Fig. 3 shows the total memory capacity as a function of the word length W_{RAM}. There are local minima for word sizes 1, 9, 14, 18 and 21 bits. As the number of clock cycle to fetch $M_{c\to v}^{old}$ is bounded by $\lfloor d_c/2 \rfloor$, the possible solutions are limited to W_{RAM} greater than 7. A word size of 9 bits gives the best memory optimization of 874800 bits. This is only 26 % more than the theoretical minimum.

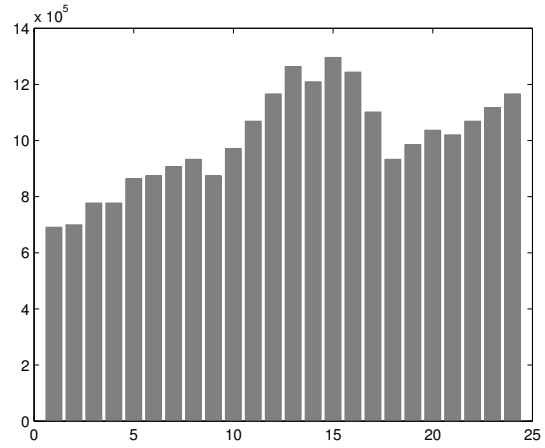

Fig. 3. Memory capacity as a function of W_{RAM}

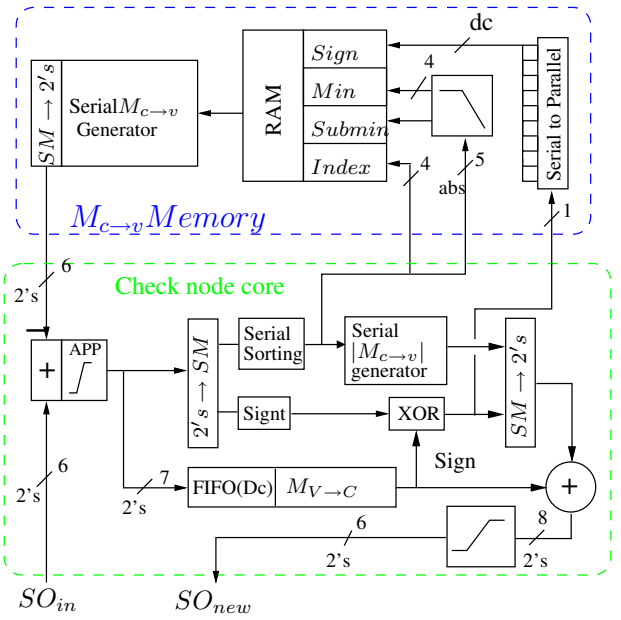

Fig. 4. Finite precision of the NP architecture

V. FINITE PRECISION ARCHITECTURE OF THE LAYERED DECODER

Fig. 4 presents the finite precision architecture of the NP (Fig. 2). Word size, type (signed or absolute) and direction are detailed for every signal connection. The architecture implements the normalized Min-Sum algorithm. The sorting block generates the Min, $Submin$ and Ind values. These values are stored in a RAM to generate $M_{c\to v}^{old}$ during the next iteration. A single port RAM is used to store the $M_{c\to v}$ values.

Fig. 5 is an overview of the proposed layered decoder architecture (see [11] and [12] for a detailed description). In

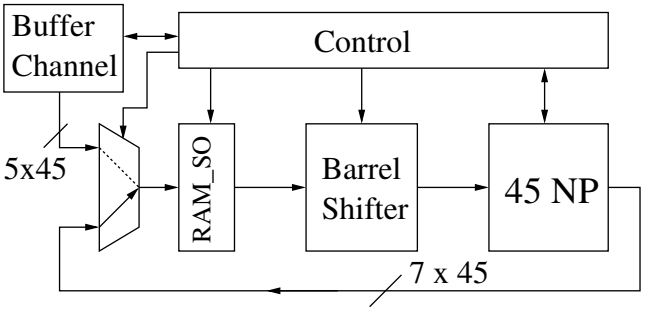

Fig. 5. Layered decoder architecture

XQ5VLX85	LUT	LUT RAM	BRAM
Node Processor	143	2	0
sorting	37	0	0
gen $m_{c\to v}$	34	0	0
fifo $m_{v\to c}$	12	2	0
$m_{c\to v}$ memory	46	0	3
Total 1 node	189	2	3
Total 45 nodes	8586	90	135
Control	667	3	0
block SO RAM	360	0	22
Channel RAM	48	0	25
Barrel shifter	945	0	0
Total	11005	93	182
Percentage [%]	5	1	50

Table III. Synthesis Results for DVB-S2 LDPC decoder

this figure, the NP block is made of 45 NP (Fig. 4) working in parallel. The Barrel Shifter shifts seven words of size 45. The RAM_{SO} block stores the SO values.

VI. RESULTS

VI-A. Synthesis

The architecture presented in Fig. 5 was synthesized on a Virtex-V Pro FPGA (XQ5VLX110) from Xilinx, for validation purposes. The system decodes long frames of code rate 2/3. Table III gives the hardware resources required. The clock frequency is 200 Mhz, the average number of iterations is 20 and the throughput is 90 Mbit/s, which allows for the decoding of two simultaneous High-Definition Television (HDTV) streams.

VI-B. Simulation results

Fig. 6 shows the simulation results for a normalized Min-Sum fixed point layered decoder, with a maximum of 30 iterations, long frame, code rates 2/3 in Additive White Gaussian Noise channel. The normalization factor is 0.75. Let us consider the following notation: a 5-6-5 configuration refers to a channel LLR quantized on 5 bits, an SO value word size of 6 bits and a $M_{c\to v}$ word size of 5 bits. We depicted the standard limit at 1 dB from the Shannon limit in E_b/N_0 for rate 2/3. Simulations show the robustness of

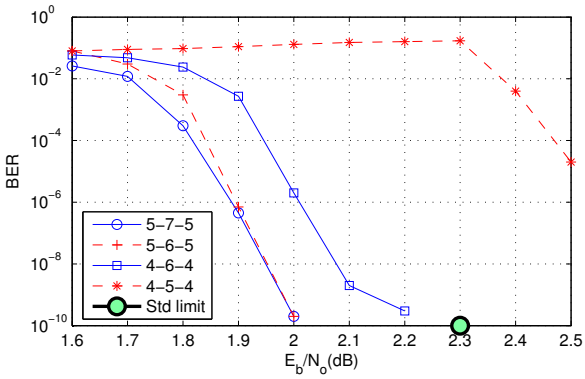

Fig. 6. BER simulation for a rate 2/3

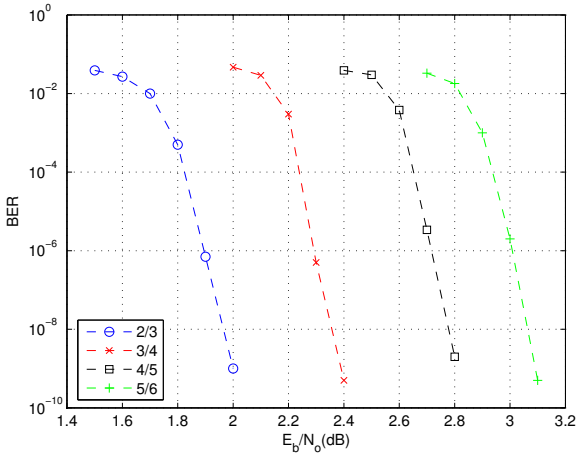

Fig. 7. BER for long frames with parallelism of 45

our saturations allowing for the use of fewer bits than the usual 6-8-6 configuration.

Fig. 7 shows simulation results for code rates 2/3, 3/4, 4/5, 5/6 and 5-6-5 configuration. Code rates with check node degree smaller than 7 (1/4, 1/3 and 2/5) present an error floor with the normalized min-sum algorithm. This problem should be solved by implementing an A-min* or λ-min algorithm instead of the normalized Min-Sum in the CNP, with no change in the rest of the architecture.

VI-C. Memory capacity comparison

Table IV shows the number of bits for the main memory units in the latest published DVB-S2 decoder IPs [9], [13], [14]. Note that no information is provided on the ROM memories that store the matrices for every rate. In our architecture, the ROM capacity is 0.8 Mbits for a parallelism of 45 [11]. A buffer of size two is used for the channel LLR values. Our architecture provides memory saving of 28% compared to [9], for the 5-6-5 configuration (for the 4-6-4 configuration the memory savings is 40%).

Paper	[9]	[13]	[14]	This
Parallelism	180	360	180	45
Air Throughput[Mb/s]	180	135	135	90
Extrinsic [bits]	6	6	6	5
SO_{ram} [bits]	10	8	8	6
Channel [bits]	6	6	6	5
Capacity[Mbits]	2.8	2.83	3.18	2.0

Table IV. Memory capacity comparison

VII. CONCLUSION

In this paper we have presented a memory optimization for a layered LDPC decoder. A first approach to memory savings was to analyze the saturation problem in the layered decoder. A second approach relied on the word split of the extrinsic memory. We developed a finite precision layered decoder architecture that implements the proposed saturation process. This architecture outperforms the state-of-the-art in terms of memory needs while satisfying the standard requirements in terms of performances. Even if we have considered the DVB-S2 standard in our study, the proposed techniques can be extended to DVB-T2,-C2 and, more generally, to any layered LDPC decoder. Future work will be dedicated to the hardware implementation optimization (area and frequency) of the proposed decoder architecture and to the evaluation of its performance at low BER.

VIII. REFERENCES

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

[2] D. MacKay, "Good error-correcting codes based on very sparse matrices," *Information Theory, IEEE Transactions on*, vol. 45, pp. 399–431, Mar. 1999.

[3] 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.

[4] T. Brack, M. Alles, F. Kienle, and N. Wehn, "A synthesizable IP core for WIMAX 802.16e LDPC code decoding," in *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium on*, (Helsinki, Finland), pp. 1–5, Sept. 2006.

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

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

[7] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Transactions on communications*, vol. 47, pp. 673–680, May 1999.

[8] F. Guilloud, E. Boutillon, and J.-L. Danger, "lambda-min decoding algorithm of regular and irregular LDPC codes," *Proceedings of the 3rd International Symposium on Turbo Codes and Related Topics*, Sept. 2003.

[9] S. Muller, M. Schreger, M. Kabutz, M. Alles, F. Kienle, and N. Wehn, "A novel LDPC decoder for DVB-S2 IP," in *Design, Automation & Test in Europe Conference & Exhibition,2009. Date '09.*, (Nice, France), Apr. 2009.

[10] J.-B. Doré, *Optimisation conjointe de codes LDPC et de leurs architecture de decodage et mise en oeuvre sur FPGA*. PhD thesis, INSA, Rennes, France, 2007.

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

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

[13] P. Urard, E. Yeo, L. Paumier, P. Georgelin, T. Michel, V. Lebars, E. Lantreibecq, and B. Gupta, "A 135mb/s DVB-S2 compliant codec based on 64800b LDPC and BCH codes," in *Solid-State Circuit Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, (Sam Francisco, USA), pp. 446–609, Feb. 2005.

[14] 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 *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, (San Francisco, USA), pp. 310–311, Feb. 2008.