Architecture and Finite Precision Optimization for Layered LDPC Decoders

Cédric Marchand, Laura Conde-Canencia, Emmanuel Boutillon

Signal Processing Systems manuscript No.
(will be inserted by the editor)

Architecture and finite precision optimization for layered LDPC Decoders

Cédric Marchand · Laura Conde-Canencia · Emmanuel Boutillon

Received: date / Accepted: date

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 first focus our attention on the optimization of the number of quantization bits. The effect of quantification on the channel values, Extrinsic and A-Posteriori Probabilities are studied separately. We propose solutions to optimize the quantification by an efficient saturation considering 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.

Then, we optimize the size of the extrinsic memory considering implementation constraints when decoding multiple code rates. Finally, fixed point architecture is presented with implementation results.

Keywords
Low-density parity-check (LDPC) code, layered decoding, VLSI implementation, DVB-S2.

1 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) has been adopted during 2010. These three DVB standards include a common Forward Error Correction (FEC) block. The FEC is composed of an LDPC inner code and BCH outer code. The FEC supports eleven code rates for the DVB-S2 standard frames and is reduced to six code rates for the DVB-T2 standard frames. The LDPC codes defined by the DVB-S2,-T2,-C2 standards are structured codes or architecture-aware codes (AA-LDPC) [3]) and they 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

C. Marchand · L. Conde-Canencia · E.Boutillon
Université Européenne de Bretagne
Lab-STICC, CNRS, UBS
BP 92116 56321 Lorient, France.
E-mail: firstname.surname@univ-ubs.fr

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 layerd decoders. 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.

The well-known Min-Sum algorithm [7] and its variants significantly reduce the memory needs by the compression of the extrinsic messages. This memory compression makes naturaly the Min-Sum algorithm a good condidate as a mean of memory reduction and is choosen for this studies althought we will show that in case of the DVB-S2 standard, the sum-product algorithm can gives better performance for a resonnable overcost.

Another way to reduce the memory needs is to limit the word size by saturation. The other advantages of reducing the number of quantification bit are a reduction in complexity of the interconnection routing scheme and the processing units. In the state-of-the-art, the way how the SO and the extrinsic messages are saturated is rarely explicitly explained. In this article, we provide some discussion on efficient saturation of the channel LLR values, the extrinsic messages and the SO values. We present some ideas related on the efficient use of saturation leading to significant memory savings. To complete the discussion, we also introduce a methodology to optimize the implementation of the extrinsic memory with the constraint of 11 code rates and single port RAM.

The paper is organized as follows: Section 2 presents the layered decoder and the Min-Sum sub-optimal algorithm. In Section 3, we explain the saturating process. Section 4 deals with the optimization of the size of the extrinsic memory. Finally, simulation and synthesis results are provided in Section 5.

2 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 with a bit of the codeword or Variable Node (VN), and each row corresponds to a parity check equation or Check Node (CN). A non-zero 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.

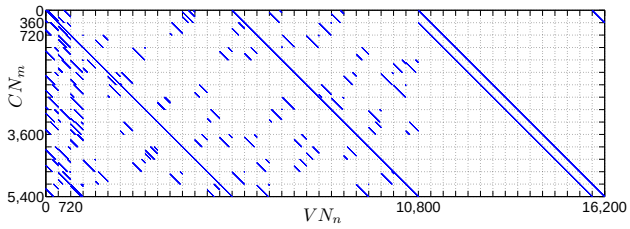

Fig. 1 Block-structured rate-2/3 DVB-S2 matrix (N=16200)

2.1 Horizontal layered decoder

In the horizontal layered decoder, an SO value is associated to each VN. 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 v to a CN c ($M_{v\to c}$) 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 it 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:

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

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].

2.2 Architecture overview

From equations (1) to (4), the Node Processsor (NP) architecture shown in Fig. 2 can be derived. The left

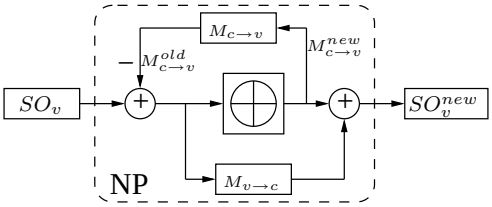

Fig. 2 Check Node centric Node Processor

adder of the architecture performs equation (1) and the right adder performs equation (4).

As the structured matrices are made of identity matrices of size P, then P CNs can be computed in parallel without memory access conflicts. 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_v^{new} in the VNs. The central part is in charge of the serial $M_{c\to v}$ update described in equation 3. The $f(.)$ function used in this equation is difficult to implement. This function can be implemented using look up table or linear pieceware approximation as in [8] but can also be implemented more efficiently by using a sub-optimal algorithm.

2.3 The normalized Min-Sum algorithm and other related algorithms

The most used sub-obtimal algorithms are improved version of the well known Min-Sum algorithm [7], such as the normalized Min-Sum algorithm, the Offset Min-Sum algorithm, the A-min* algorithm [9], the λ-min algorithm [10] and related [11]. 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 the normalized Min-Sum algorithm. With this algorithm, equation (3) becomes:

$$|M_{c\to v}^{new}| = \alpha \min_{v'\in v_c/v} |M_{v'\to c}| \quad (5)$$

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 $|M_{c\to v}^{new}|$ values, if the index of $M_{c\to v}^{new}$ is ind_{min} then $|M_{c\to v}^{new}| = submin$ else $|M_{c\to v}^{new}| = min$. The $M_{c\to v}$ from one CN can be compressed with four elements, i.e. *min*, *submin*, ind_{min} and $sign(M_{c\to v}^{new})$. For matrices with a check node degree greater than four, this compression leads to significant memory saving.

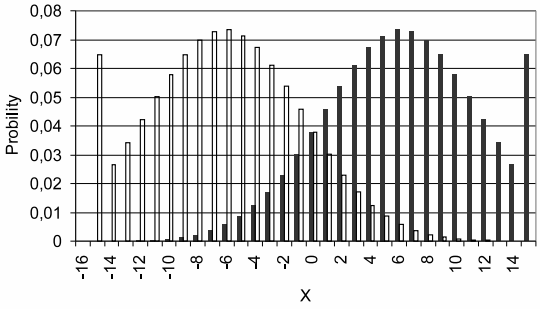

Fig. 3 Resulting distribution of a quantified BPSK modulation

3 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. Even if the channel LLR and the $M_{c\to v}$ are quantized on 6 bits, the SO values must be quantized on 10 bits to prevent overflows. However, to avoid prohibitive word size, efficient saturation of channel LLRs lead to a reduction of the overall quantization. Then a saturation of the SO values and the $M_{c\to v}$ are considered.

3.1 Channel LLR saturation

Figure 3 shows the Probability Density Fonction of a quantized BPSK modulation (-1 and +1) which is perturbed by an Additive White Gaussian Noise (AWGN) of variance $\sigma = 0.866$ (corresponding to Eb/No = 2 dB). The channel is quantization on 5 bits and the saturation threshold at $1+\beta = 2,47$. The distribution filled in black shows the +1 offset, and the unfilled distribution is the -1 offset. The quantized distribution varies from $X_{min} = -(2^4-1)$ to $X_{max} = 2^4-1$. The problematic is to find for a given quantization, the saturation threshold which will provide the best performances. If the saturation threshold is low, then the informations given by the tail of the gaussian curve are saturated, and if the threshold is hight then the quantization error increase.

For floating point simulation, it is known that the decoders using the Normalized Min-Sum algorithm are not sensitive to scaling in the LLR_{in} values. During the initializing process, the equation $LLR_{in} = 2y/\sigma^2$ can

be simplified to $LLR_{in} = y$, saving the need to compute the variance. However, it is important to scale the y value so that the LLR_{in} values fit the range imposed by the quantification. We define

$$LLR_{in} = \omega \times y$$

where ω is the scaling factor. Considering the BPSK modulation and LLR_{in} quantified on n_{LLR} bits, we saturate y at $1+\beta$ and compute ω using the following equation:

$$\omega = \frac{2^{n_{LLR}-1}-1}{1+\beta} \tag{6}$$

The saturation limit $\beta+1$ can be calculated so that the proportion of saturated values is equal to the average proportion of the other values. The average proportion of a given value is $1/(2^{n_{LLR}}-1)$(probability of a value in an uniform distribution). On the other side of the eguality, the Cumulate Distributive Function (CDF) of a -1 offset distribution applied to the negative saturation limit will give the proportion of saturated values for a -1 offset signal. The eguality can be written as:

$$\frac{1}{2}\left[1+erf\left(\frac{-\beta}{\sqrt{2}\sigma}\right)\right] = \frac{1}{2^{n_{LLR}}-1} \tag{7}$$

From equation (7), one can deduce β:

$$\beta = \sigma \times \sqrt{2}\left(erf^{-1}\left(\frac{2^{n_{LLR}}-1}{2^{n_{LLR}}+1}\right)\right) \tag{8}$$

Thus the β value is function of n_{LLR} and is proportional to σ. By applying equations (8) and (6), the optimum saturation treashold and the scaling factor can be computed. The problem of this solution is that an adaptative quantization of y is needed wich requires a channel estimation of σ .

In fact, to prevent the σ computation, an optimal scaling factor is calculated for a given SNR. If the performance requirement are reach for a given SNR, then for higher SNR, the quantification would be sub-optimal but well within the performance requirement. For each code rates, a constant scaling factor ω and saturation value $1+\beta$ can be pre-computed saving the need for the σ value.

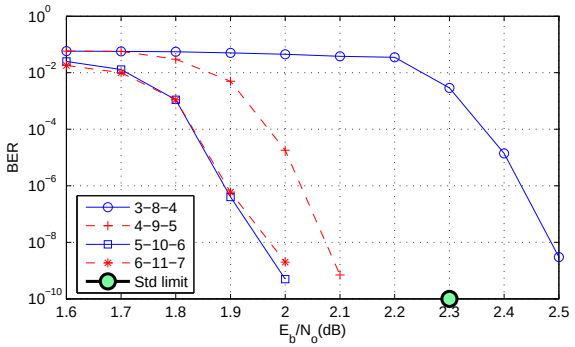

Fig. 4 BER simulation for a rate 2/3

3.1.1 Effects of LLR_{in} saturation on BER performance

Fig. 4 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 3-8-4 configuration refers to a channel LLR quantized on 3 bits, an SO value word size of 8 bits and a $M_{c\to v}$ word size of 4 bits. We also depicted the standard limit at 1 dB from the Shannon limit in E_b/N_0 for code rate 2/3.

The quantification values of the $M_{c\to v}$ and SO are not optimized and chosen large enough to not affect the results of the channel quantification. Fig. 4 shows that a quantification on 4 or 5 bit of the LLR_{in} is enough to fulfill the standard requirements.

3.2 SO saturation

Once the LLR_{in} quantized, they are stored in an SO memory wich will evoluate with the iteration process. This SO memory need to be saturated to limit their size.

3.2.1 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 $SO_{new} =$

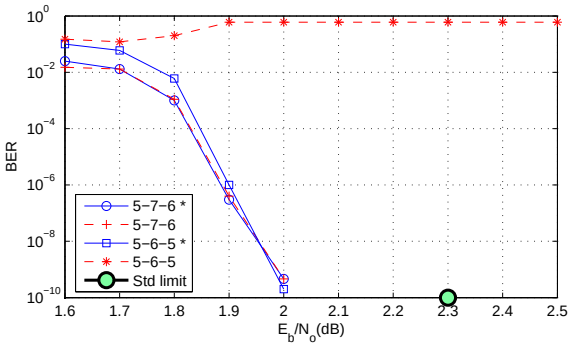

Fig. 5 BER simulation for a rate 2/3

$SO_{old} + \Delta M_{c\to v}$ where $\Delta M_{c\to v} = M^{new}_{c\to v} - M^{old}_{c\to v}$. 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 SO is saturated, the SO value cannot increase but it can decrease. The saturation introduces a non-linearity that can produce pseudo-codewords and an error floor. A solution has to be found to overcome this problem.

3.2.2 A solution for SO saturation

The solution that we propose was first introduced in [12] and relies partially on the A Priory Probability (APP) based decoding algorithm [7]. The APP-variable decoding algorithm simplifies equation (1) to:

$$M_{v\to c} = SO_v \tag{9}$$

which greatly reduces the architecture complexity but introduces significant performance loss. The idea is to use equation (9) 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**
  $M_{v\to c} = SO_v$
else
  $M_{v\to c} = SO_v - M_{c\to v}$
end if

3.2.3 Effects of SO saturation on BER performance

Fig. 5 shows the simulation results in the same conditions as in Fig. 4. An asterisk symbol in the legend means that the APP-SO algorithm is used. The results shows that the APP-SO algorithm, compared to a standard solution shows perfomance improvement only if

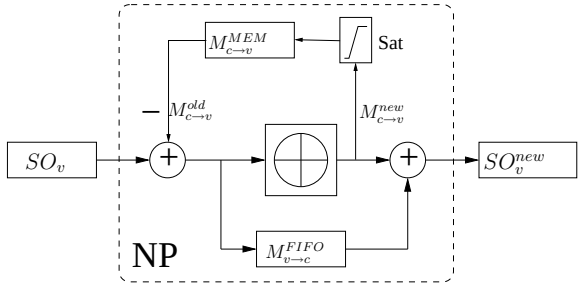

Fig. 6 NP with saturation of the extrinsic messages

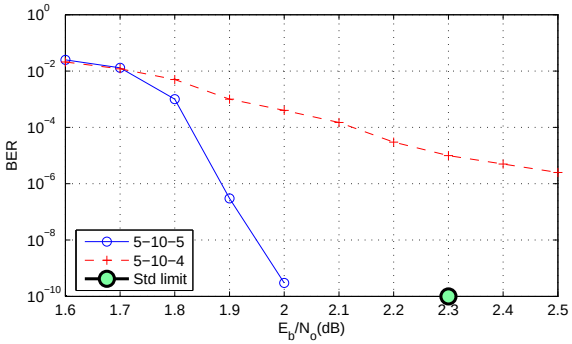

Fig. 7 BER simulation for a rate 2/3

the extrinsic values are satured (curves 5-6-5* and 5-6-5).

3.3 Saturation of the extrinsic messages

Figure 6 shows the SO based node processor. One can see that the newly updated extrinsic $M^{new}_{c\to v}$ is used to compute SO^{new}_v from equation (4) and $M^{new}_{c\to v}$ is also stored in the extrinsic memory for the calculation of $M_{v\to c}$ (equation 1) at the next iteration. Any saturation on the value $M^{new}_{c\to v}$ responsible for the SO update would not produce area savings and would degrade performance. This is the reason why we do not saturate this value. On the other hand, saturation of the $M^{new}_{c\to v}$ message that are stored in a memory would lead to significant area saving. Furthermore, the saturation of the $M^{new}_{c\to v}$ stored in the extrinsic memory is much less critical because it will be used only once during an iteration to compute an $M_{v\to c}$ and this $M_{v\to c}$ will affect SO^{new} only if it is a minimum value (see equation 5).

3.3.1 Effects of extrinsic message saturation on BER performance

Fig. 7 shows the simulation results in the same conditions as in Fig. 4. Simulations show that the $M_{c\to v}$ with the same quantification as the LLR_{in} ($n_{LLR} = n_{ext}$) gives result equivalent to higher quantification. A

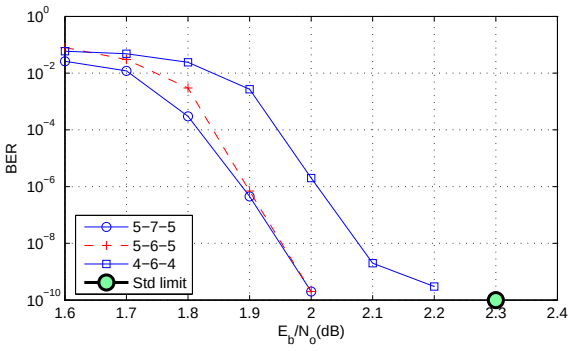

Fig. 8 BER simulation for a rate 2/3

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 1 Memory size of extrinsic

reduction of the quantification lower than the LLR_{in} quantification lead to dramatic performance loss.

3.4 Combining the saturation processes

Simulations show that the combinaison of the SO saturation process and the $M_{c\to v}$ saturation process lead to better performance that when they are implemented separately. The combinaison of our saturation process allows the use of fewer bits than the usual 6-8-6 configuration.

3.5 Saturation optimization conclusion

The analysis of the saturation process shows that a better trade-off between word size and performance can be obtained with an efficient saturation of the LLR_{in},the SO and the extrinsic values. Simulations show the robustness of our saturations allowing for the use of fewer bits than the usual 6-8-6 configuration. Simulations with channel values quantified on 5 bits, SO on 6 bits and $M_{c\to v}^{old}$ on 5 bits gives the same performance as other known implementation with fewer bits.

4 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 DVB-S2 standard which provide eleven different code rates for standard frames.

4.1 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 [7], 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 1 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 1. 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.

4.2 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

Rate	M	$W_{Mc\rightarrow v}$	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 2 Memory capacity of the extrinsic message with $W_{RAM} = 9$

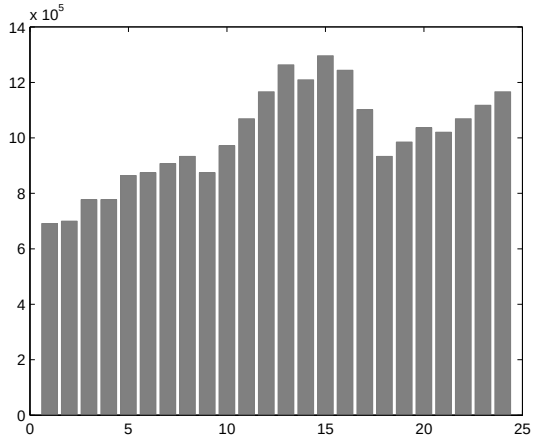

Fig. 9 Memory capacity as a function of W_{RAM}

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.

4.3 Results

The previously described process can be used for different word sizes. Table 2 gives an example with $W_{RAM} = 9$. For each rate, the number of cycles is given by:

$$n_{cycles} = \lceil W_{Mc\rightarrow v}/W_{RAM} \rceil$$

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 2 and the total memory capacity is $R_{RAM}^{global} \times W_{RAM} = 97200 \times 9 = 874800$ bits.

Fig. 9 shows the total memory capacity as a function of the word length W_{RAM}. There are local minimum for word sizes 1, 9, 14, 18 and 21 bits. As the number of clock cycle to fetch $M_{c\rightarrow v}^{old}$ 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.

4.4 Case of the sum-product algorithm

When using a sum-product algorithm [13] [14] [15] instead of a Min-sum algorithm, the check node update equation (3) is computed for each $M_{c\rightarrow v}$ value and each $M_{c\rightarrow v}$ value are stored. The same process as the previously described can by used but a simpler and more efficient implementation can be used. We can consider a dual port ram which is read every cycle to fetch a $M_{c\rightarrow v}^{old}$ value and write simultaneously a $M_{c\rightarrow v}^{new}$ value to be stored. With the constraint of 11 code rates, 5 bits quantification and dual port ram, the memory requirement is given by the code rate that require the maximum number of $M_{c\rightarrow v}$ values. The code rate 5/6 requires storing 237600 values of 5 bit which give 1.2 MBits memory requirement. Although this size is 26% higher than the solution that we proposed for the Min-Sum algorithm, the over cost can worst it, considering the performance increase especially at low code rate. Implementation of a FIFO memory with single port modules for allowing simultaneous read and write operations is presented in [16]. This solution requires one memory banks for even addresses and another memory banks for odd addresses and lead to area saving compared to the use of dual port ram.

4.5 $M_{c\rightarrow v}$ memory optimization conclusion

The $M_{c\rightarrow v}$ memory optimization shows that the implementation of LDPC decoder for multiple code rate and single port RAM to store the $M_{c\rightarrow v}$ is an issue.

A careful implementation of the Min-sum algorithm gives result only 26 % more than the theoretical minimum compared to a straight forward implementation with 200% over cost.

The sum-product algorithm lead to a 26% over cost compared to the Min-Sum algorithm, but considering performance increase with low code rate, the implementation of the sum-product can be considered.

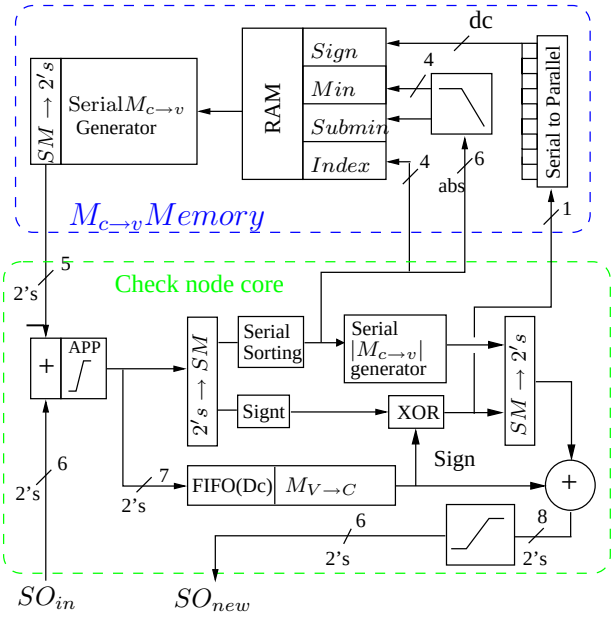

Fig. 10 Finite precision of the NP architecture

5 Finite precision architecture of the layered decoder

Fig. 10 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.

In the Check node core, The $M_{v\rightarrow c}$ values arrive serially in two's complement representation from the adder modified to implement algorithm 1. They are transformed to sign and magnitude representation so that the sign and magnitude of the messages can be computed separately. The serial sorting of the incoming magnitude values is implemented, in order to give Min, $SubMin$ and $Index$ values until all the incoming messages are read. In the serial $M_{c\rightarrow v}$ block, the Min, SubMin and index values previously sorted are used to serially compute the new outgoing messages. An XOR function is computed recursively on the incoming sign bits in the $Signt$, until all the incoming messages are read. Next, an XOR function is computed in XOR block, between the output of $Signt$ block and the sign of the outgoing message $M_{v\rightarrow c}$ from the FIFO memory of size d_c. The sign and magnitude from XOR block and serial $M_{c\rightarrow v}$ block, respectively, are transformed into a two's complement representation ready for subsequent computations.

In the $M_{c\rightarrow v}$ memory block, the Min, $Submin$, $index$ and $sign$ of each $M_{c\rightarrow v}$ linked to one check node are stored in a RAM. From the Min, $Submin$, $index$ and $sign$ values, the serial $M_{c\rightarrow v}$ generator computes the two's complement representation of the $M_{c\rightarrow v}$ value.Min and $Submin$ are quantified on 4 bits (absolute values) which gives $M_{c\rightarrow v}$ values quantified on 5 bits (with the sign bit). Note that as $signt$ gives the result of the parity check equation of line and the syndrome can easily be computed by using the result of $signt$. The computation of the syndrome allows deciding an early termination of the decoding process leading to a significant reduction of the number of iterations.

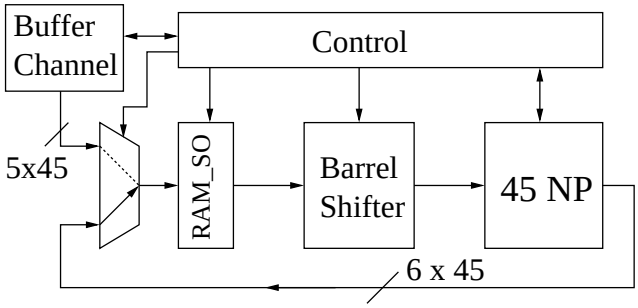

Fig. 11 Layered decoder architecture

Fig. 11 is an overview of the proposed layered decoder architecture (see [17] and [18] for a more detailed description). In this figure, the NP block is made of 45 NP (Fig. 10) working in parallel. The Barrel Shifter shifts seven words of size 45. The RAM_{SO} block stores the SO values. Thanks to a systematic syndrome calculation, it is possible to use a built-in stopping rule while decoding in a variable-iteration decoder. The addition of a buffer on the decoder input allows the exploitation of the variations in the decoding time in the different frames. A preemptive buffer control as in [19] is used to reduce the buffer size. Note that the quantification reduction described in Section 3 for earea reduction of the memory also lead to area reductions of the node processor and the barrel shifter. The latency in the Check node core is also reduced due to complexity reduction of the addition and comparison computation.

6 Results

6.1 Simulations

Fig. 12 shows simulation results for code rates 1/4, 1/2, 2/3, 3/4, 4/5, 5/6 and 5-6-5 configuration. The code rates 1/2, 2/3, 3/4, 4/5 and 5/6 show results that fulfil the standard requirements. The code rate 1/4 shows poor performances. The code rates 1/4, 1/3 and 2/5 have a check node degree smaller than 7 (4, 5 and 6 respectively) which lead to an error floor with the normalized min-sum algorithm. One can note that code rate 1/2 with a check node degree of 7 start producing an error floor that can be corrected with the BCH

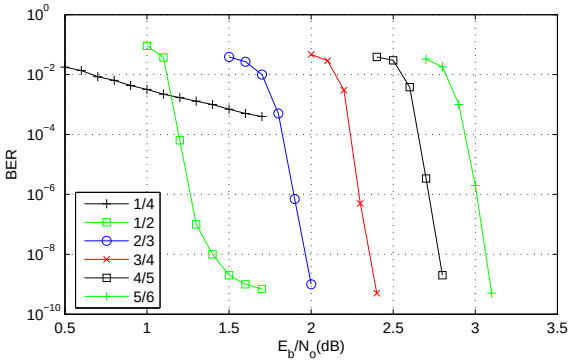

Fig. 12 BER for long frames

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 3 Synthesis Results for DVB-S2 LDPC decoder

outer decoder. This error floor problem can 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. One can also implement a Sum-product algorithm as in [8] combined with the proposed saturation processes.

6.2 Synthesis

The architecture presented in Fig. 11 was synthesized on a Virtex-V Pro FPGA (XQ5VLX110) from Xilinx, for validation purposes. The system decodes long frames of code rate 2/3 and a parallelism of 45. Table 3 gives the hardware resources required. The clock frequency is 300 Mhz, the average number of iterations is 20 and the throughput is 120 Mbit/s, which allows for the decoding of two simultaneous High-Definition Television (HDTV) streams.

Paper	[11]	[20]	[21]	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 4 Memory capacity comparison

6.3 Memory capacity comparison

Table 4 shows the number of bits for the main memory units in the latest published DVB-S2 decoder IPs [11,20,21]. Note that no information is provided on the ROM memories that store the matrices for every rate. In our architecture, a buffer of size two is added to store the channel LLR values to reduce by two the average number of iteration as in [19]. Our architecture provides memory saving of 28% compared to [11], for the 5-6-5 configuration and the memory savings is up to 40% for the 4-6-4 configuration.

7 Conclusion

In this paper we have presented optimization solutions for a layered LDPC decoder. A first approach was to analyze the saturation problem in the layered decoder.An efficient saturation lead meanly to a reduction of memory area and also a reduction of latency in the computing elements. We developed a finite precision layered decoder architecture that implements the proposed saturation processes. This architecture outperforms the state-of-the-art in terms of memory needs while satisfying the standard requirements in terms of performances. In a second approach, we studied the problematic of implemanting efficiently multiple code rate decoder. Our solution relie on the word split of the extrinsic memory for Min-Sum algorithm implementation. This solution allows the use of single port RAM and lead to significant memory reduction compared to a straight forward implementation. 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.

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. X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes," in *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, vol. 2, pp. 1036–1036E vol.2, 2001.
9. C. Jones, E. Valles, M. Smith, and J. Villasenor, "Approximate-min* constraint node updating for LDPC code decoding," in *IEEE Military Communication Conference*, pp. 157–162, Oct. 2003.
10. 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.
11. 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.
12. J. Doré, *Optimisation conjointe des codes LDPC et de leurs architecture de décodage et mise en oeuvre sur FPGA*. PhD thesis, INSA, Rennes, France, 2007.
13. S. Papaharalabos and P. Mathiopoulos, "Simplified sum-product algorithm for decoding LDPC codes with optimale performance," *Electronics letters*, vol. 45, pp. 536–539, June 2009.
14. M. Gones, G. Falcao, J. Goncalves, V. Silva, M. Falcao, and P. Faia, "HDL library of processing units for generic and DVB-S2 LDPC decoding," in *International Conference on Signal Processing and Multimdia Applications (SIGMAP2006)*, (Setubal, Portugal), 2006.
15. O. Eljamaly and P. Sweeney, "Alternative approximation of check node algorithm for DVB-S2 LDPC decoder," in *Second International Conference on Systems and Networks Communications (ICSNC 2007)*, pp. 157–162, Oct. 2007.
16. Andreev, Alexander, Bolotov, Anatoli, Scepanovic, and Ranko, "Fifo memory with single port memory modules for allowing simultaneous read and write operations," *US patent 7181563*, Feb. 2007.
17. 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.
18. 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.
19. 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.
20. 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. 446–447, Feb. 2005.
21. 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.