Buffers optimization for multi-core decoders

E. Boutillon, Cédric Marchand

Buffers optimization for multi-core decoders

Emmanuel Boutillon, Cédric Marchand
Université Bretagne Sud, Lab-STICC UMR 6285, CNRS
Email: {emmanuel.boutillon,cedric.marchand}@univ-ubs.fr

***Abstract*—For very high-speed satellite communication (up to 10 Gbit/s), the natural level of parallelism of a single decoder might be insufficient to achieve the decoding throughput. A known solution is to implement several decoder cores working in parallel. This solution entails efficient control and design of the input and output buffers to regulate the varying number of decoding iterations of each decoder. This paper presents a methodology to build such a system effectively for iterative decoders with stopping criteria. As an application, we present the result of the implementation of 3 DVB-S2/S2X decoders in a single FPGA. Simulation results of the whole system show performance within (or very close to) the standard requirements. The implementation can handle code rates from 13/45 (3.3 Gbit/s air throughput) up to 9/10 (10 Gbit/s air throughput) for several modulation sizes.**

***Index Terms*—Buffer, Multi-core, decoder, DVB-S2, architecture**

I. Introduction

Low-Density Parity-Check (LDPC) codes [1] have been adopted in several standards, such as the 2^{nd} Generation Satellite Digital Video Broadcast (DVB-S2) [2] and its optional extension in DVB-S2X [3]. DVB-S2 was designed for up to $n_c = 360$ processing units in parallel, thus enabling hardware-friendly layered implementation for up to a few hundred Mbit/s throughput [4], [5]. However, once the code parallelism is fully exploited, increasing the throughput becomes much more challenging. A designer can maximize the operating clock rate by pipelining, but layered decoding suffers from data dependency between successive layers [12]. Further increasing the parallelism is possible with a fully parallel solution for a single Code Rate (CR) and a short frame length [6]. However, in the case of DVB-S2 frames of length $N = 64,800$ bits, a fully parallel solution would lead to routing congestion and loss of flexibility required to achieve adaptive coding modulation. A straightforward and efficient solution is to increase the decoding throughput by a factor P by implementing P decoders in parallel [7], [8].

Using a stopping criterion in an iterative decoder allows stopping the decoding process as soon as a codeword is decoded. The average time to process a frame is thus reduced by a factor i_a/i_m, where i_m is the maximum number of iterations and i_a is the average number of iterations before convergence. In a real-time system, such a technique requires input and output buffers to smooth the variation between the constant input/output frame throughput and the variable time for decoding each frame. The size of the input/output buffer can be determined formally as a function of three parameters: the maximum probability of input buffer overflow, the effective frame throughput, and the distribution of the number of decoding iterations at a given Signal-to-Noise Ratio (SNR) point [9], [10], [11].

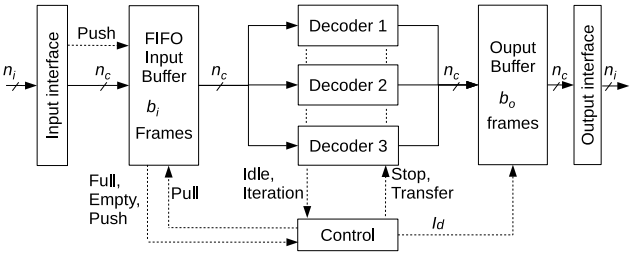

Fig. 1. Multi-core architecture.

This paper proposes a pragmatic extension of the buffering technique defined for a single decoder [9], [10], [11] to the case where P decoders work in parallel. It presents a model of the whole system which allows for determining the input buffer's size. It also proposes a mechanism to re-order the frame after the decoding process using the output buffer and defines a lower limit on the output buffer size. Finally, we provide the FPGA implementation results of a multi-Gbit/s Adaptive Coded Modulation DVB-S2 decoder.

The paper is outlined as follows. Section II presents the multi-core architecture. Then, in section III, an algorithm based on a Markov chain is proposed to modelize the input buffer behavior and determine its size. Section IV describes the output buffer and a simple equation to determine its size. Section V provides the simulation results and complexity analysis. Finally, section VI concludes this paper.

II. Multi-core architecture

When targeting a high-throughput LDPC decoder, a designer can maximize the operating clock frequency or increase the number of parallel processing units of the decoder up to a given point. However, increasing the parallelism further can be counterproductive, especially when code flexibility is required. An efficient, strategy is to process multiple decoders in parallel.

Fig. 1 shows the global architecture of the multi-core architecture option. The iterative decoder of this study is an LDPC decoder for the DVB-S2 standard but the architecture is valid for any iterative decoder with stopping criteria. The Input Interface (II) receives data from the channel with a parallelism of n_i data per clock cycle and re-orders the frame by blocks of n_c data, with n_c the level of parallelism of one decoder. The internal architecture of the II is out of the scope of the

paper. However, it is worth mentioning that its architecture is not trivial. In fact, it performs a configurable (several code rates) and parallel interleaving of the parity bits of the received frame. Once the II processes a frame, the frame is sent to the FIFO Input Buffer (FIB) of size b_i frames. An identification number I_d is associated with each frame entering the FIB. The FIB sends the oldest received frame as soon as one of the iterative decoders is available. Once decoded, the frame is sent to the output buffer of size b_o frames. Due to the unknown number of decoding iterations of a decoder, the initial order of arrival of the frames may not be respected after the decoding process. Thanks to the frames I_d, the output buffer is able to restore the original arrival order of the frames. Implementing this architecture would leave the designer with two unknown variables b_i and b_o that significantly impact the multi-core decoder's performance and complexity. The behavior, control, and size of input and output buffers are discussed in the following sections.

III. Input buffer

The FIB is implemented with a Random Access Memory (RAM) using circular write/read addresses. The n^{th} received frame is given an Identification number I_d equal to $I_d = n \mod b_o$, as shown in the toy example of Fig. 2 where $b_o = 5$ and $P = 2$. As soon as a decoder is idle, the control sends a pull signal to the FIB. If the FIB contains at least one frame, the oldest frame is transmitted to the idle decoder; otherwise, the decoders stay idling until a new frame arrives in the FIB. Before the input buffer gets full, the buffer sends a full signal to the control unit. The control unit checks the decoding status of the P LDPC decoders and stops the decoder having the greatest number of decoding iterations. This situation is illustrated in Fig. 2 by the received frame number 12 (with I_d = 2) where the control stops decoder 2 before the convergence of the decoder. This stopping process frees space in the FIB just in time to avoid a buffer overflow. The choice of stopping the decoder with the greatest number of iterations relies on the fact that most frames are decoded in a low number of iterations, and only a few frames will be decoded in a high number of iterations up to i_m. The bet is that the frame with the greatest number of iterations is a frame stuck in a trapping set configuration that uses unnecessary processing time. Furthermore, in the case of the DVB-S2 standard, the outer BCH code may suppress the remaining errors of the stuck frame, even if the decoder has not yet converged [13]. In the case of a standard with a re-transmission protocol, the control can send a re-transmission request for the stopped frame.

A trade-off between hardware complexity and performance degradation determines the FIB size b_i. On the one hand, the memory required to store a received frame is high. The size of a frame stored in FIB is equal to N times 6 (number of bits to quantize the soft information in our implementation) bits. Thus, one should minimize b_i to save area. On the other hand, reducing the buffer frame storage capacity would increase the risk of a buffer-full scenario leading to significant performance degradation. A patch that simulates the FIB state can be added to an existing decoder simulation to estimate the degradation due to the buffer-full scenario. However, the patch design is challenging when considering multiple decoders, and the simulation time can be prohibitive when targeting very low FER for different buffer sizes and throughput scenarios.

A Markov chain model can describe the buffer state to estimate the minimum buffer size required to avoid performance degradation. When considering only one decoder, the problem is known as the $D/G/1/B$ queuing problem, where D stands for Deterministic inter-arrival time, G for General distribution of the service time, 1 is for one decoder, and B is the input buffer size. In [10], a case study with DVB-S2 shows that the throughput may be doubled with $b_i = 2$ frames. A single input buffer of size b_i shared between P decoders in parallel gives the $D/G/P/b_i$ queuing problem. A discrete-time Markov chain model of the parallel decoders has been developed to estimate the input buffer occupancy. In this model, the time unit is one decoding iteration. The state of a decoder is either idle or decoding. The Probability Density Function (PDF) of the number of decoding iterations of a decoder gives the general distribution G. A Monte-Carlo simulation of the decoder at the targeted SNR allows for estimating the G distribution.

The simulation of the Markov model showed that the buffer size is driven by the ratio ρ (with $\rho \leq 1$ to meet the real-time constraint) between the effective input throughput T_e (in Gbit/s) and the average global decoding throughput of the system, i.e., the number P of parallel decoders multiplied by the average decoding throughput T_a of each decoder, thus,

$$T_e = \rho P T_a \text{ Mbits/s.} \tag{1}$$

The air throughput T_a is determined as

$$T_a = \frac{N \times F_{clk}}{E \times i_a + L} \text{ Mbits/s,} \tag{2}$$

where N is the codeword size (here 64,800), F_{clk} (in MHz) is the clock frequency, E is the number of clock cycles to perform a decoding iteration and i_a is the average number of decoding iterations. L is a constant that includes the pipeline latency, the time to input a frame to a decoder, and the time to output the frame from a decoder. In the study, to allow simplifications in the Markov model, L is approximated to E, and T_a becomes

$$T_a = \frac{N \times F_{clk}}{E \times (i_a + 1)} \text{ Mbits/s.} \tag{3}$$

Equations (1) and (3) give T_e as a function of F_{clk}/E. Since the duration of a decoding iteration is F_{clk}/E, each decoding iteration, the fraction F of a frame that enters the II of the multi-core architecture is defined as

$$F = \frac{\rho \times P}{i_a + 1}. \tag{4}$$

Algorithm 1 presents the Markov model used to estimate the FIB occupancy statistics, through Monte-Carlo simulation, according to the problem's parameters. In this algorithm, the

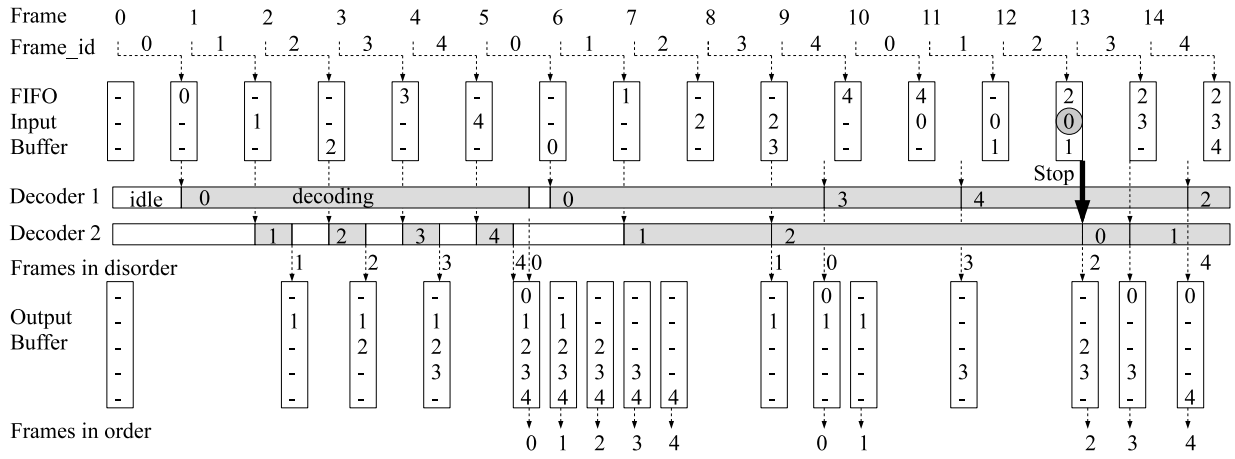

Fig. 2. Example of the behavior of the input and output buffers.

function $G()$ is a random variable that gives the number of decoding iterations required to decode the received codeword. A Monte-Carlo simulation of one decoder at the targeted SNR provides the $G()$ distribution estimation. The state of the decoder k is represented as $S(k)$, with the convention that $S(k) = 0$ presents the decoder at the idling state. $S(k) > 0$ indicates the number of remaining decoding iterations to be performed before the end of the decoding process. Each time a decoder starts a decoding process, $S(k)$ is initialized to $G() + 1$. When decoding, $i(k)$ indicates the number of iterations already done plus one. When a decoder is idling, $i(k)$ is set to 0.

For a given code rate and SNR, one can obtain G() and i_a by simulation, and F is obtained for a given ρ value by applying equation (4). Then, Algorithm 1 provides $P(n)$, the probability that the FIB contains n frames. Fig. 3 shows the obtained $P(n)$ for the code rate $r = 9/10$ and different ρ values. Considering a buffer of size $b_i = 6$, the state $n = 6$ corresponds to a buffer full state that leads to a preemptive stop of a frame. The DVB-S2X standard [3] requires a Quasi Error Free (QEF) performance defined at a FER equal to 10^{-5}. Thus, to avoid performance degradation, we retrieve from Fig. 3 the couples (ρ, b_i) that have a $P(n = b_i)$ value below 10^{-5}. The four couples (0.84, 3), (0.86, 4), (0.87, 5), and (0.88, 7) are valid couples. The couple with the best compromise between complexity (targeting low n value) and throughput (targeting high ρ value) is then selected. Based on the available memory on the targeted FPGA, we set $b_i = 5$ to target $\rho = 0.87$.

To prove the efficiency of the shared FIB, we consider one decoder ($P = 1$) with the same G(), computing one-third of the frame input ($F = F/3$) and applying Algorithm 1 for different ρ values. The simulation results show that a FIB of size $b_i = 2$ and $\rho = 0.83$ are required to get a probability of overflow below 10^{-5}. Thus, ρ is improved and overall input buffer size is reduced when the input buffer is shared among the 3 decoders compared with 3 decoders that embed their own buffer.

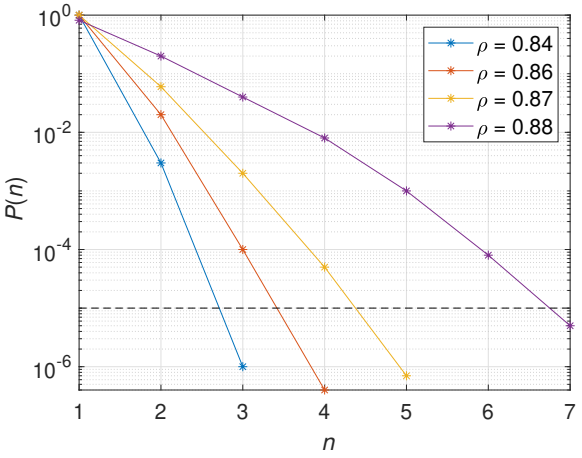

Fig. 3. Probability of input buffer occupancy for several ρ factors.

IV. Output Buffer

Let us consider the example of Fig. 2 where decoder 1 starts decoding frame 0 and then decoder 2 decodes frame 1. Due to the unknown number of decoding iterations of a decoder, the initial order of arrival of the frames may not be respected after the decoding process. For example, in Fig. 2, frame 0 is output after frame 1. The output buffer is in charge of reordering the frame outputs in the same order as they were at the input.

Before entering one of the decoders, each new frame is associated with an identification number, denoted I_d. I_d is determined as the number of frames already received modulo the output buffer size b_0 (I_d takes its values between 0 and b_0-1). When decoded, the frame is written in the output buffer at the address given by its I_d. The initial order of arrival is reconstructed by the output buffer by reading circularly the stored frames from frame addresses 0 to $b_0 - 1$. In our case study, without loss of generality, the output of a frame is also controlled by an external control signal indicating that the outer BCH decoder is available.

Input : r_{max}, $G()$, F, b_i
Output: B: Buffer size statistics
Initialization $i([1,2,...,P]) = 0$; // nb of iterations done
$S([1,2,...,P]) = 0$; // Idle Decoders
$B([1,2,...,b_i]) = 0$; // for statistics
$\mu = 0$; // II set to 0
$n = 0$; // Nb frames in FIB
$r = 0$; // Number of received frame
while $r < r_{max}$ **do**
  $\mu = \mu + F$; // update II state
  if $\mu > 1$ **then**
    $\mu = \mu - 1$; // output from II
    $n = n + 1$; // input in FIB
    $r = r + 1$; // one more frame received
    $B(n) = B(n) + 1$; // Update FIB statistics
  end
  // Process case of FIB overflow
  if $n == b_i$ **then**
    $l = \arg\max_k \{i(k)\}$
    $S(l) = 0$; // l^{th} decoder idle
  end
  // Update iterative decoders state
  $l = 0$; // no idle decoder by default
  for $k = 1, 2, \ldots, P$ **do**
    if $S(k) > 0$ **then**
      $S(k) = S(k) - 1$; // Remaining iteration
      $i(k) = i(k) + 1$; // Iterations done
    else
      $l = k$; // index of an idle decoder
      $i(k) = 0$; // idle: no iteration done
    end
  end
  // Frame pulled from FIB to l^{th} decoder
  if *($n > 1$) **and** ($l > 0$)* **then**
    $n = n - 1$; // From FIB to l^{th} decoder
    $S(l) = G() + 1$; // Remaining nb of iterations
  end
end
$P = B/r_{max}$

Algorithm 1: Markov model of FIB behavior

The output buffer has another specific behavior that impacts the buffer size b_o estimation. Let us work on the scenario illustrated in Fig. 2 where frame 0 is decoded in $i_m = 19$ iterations on decoder 1. Let us assume that a new frame arrives every 4 decoding iterations, i.e.: $F = 1/4$. Starting with empty buffers, and frames 1 to 4 decoded in 2 iterations. While the first decoder decodes frame 0, the second decoder decodes the next 4 frames. The output buffer needs to store at least the 4 decoded frames plus frame 0 to output frames in the same order as they have arrived at the decoders. The proposed solution consists in increasing b_o in such a way that the output buffer can store all the fast decoded frames while one decoder decodes in i_m iterations. Considering a fast decoded frame decoded in $1/F$ iterations or less, b_o is lower bounded by

$$b_o \geq i_m \times F. \tag{5}$$

Note that the output buffer stores hard decisions (N bits per frame), reducing the constraint on the output buffer size compared with the input buffer size. Even at high SNR, LDPC codes have trapping sets that may prevent the early stopping of the LDPC decoder. In that case, a decoder can reach i_m iterations while other decoders decode much faster. Thus, the scenario of Fig. 2 is possible, especially at high SNR, which is a problem when targeting QEF performance.

In the DVB-S2X implementation, each decoded frame is output to an external BCH decoder that may be unavailable. Thus, the size of the output buffer is increased to allow additional buffering capacity to smooth the variation of the time processing of the BCH code. The size of the output buffer b_o has been set to $b_o = 12$ to maintain a comfortable margin.

V. Implementation and Simulation results

A. Synthesis results

The architecture presented in Fig. 1 is synthesized on a Zynq UltraScale+ ZU11eg FPGA from Xilinx. Three LDPC cores are implemented in parallel to reach 10 Gbit/s throughput. The multi-core decoder is part of the prototype of an ultra-wideband DVB-S2X satellite MODEM [14].

xczu11eg	LUT	REG	BRAM	URAM
1 LDPC core	60k	58k	111.5	0
Input Buffer	26	28	0	30
Output Buffer	37	38	0	5
Control	157	188	0	0
Input Interface	9k	18k	0	15
Output Interface	794	448	0	0
Total multi-core	196k	193k	350	35

TABLE I
Synthesis results.

Table I shows the synthesis results of the multi-core decoder architecture and the results of each block synthesized alone. The LDPC decoder is a layered decoder based on the architecture presented in [15] and soft input values coded on 6 bits to obtain good complexity over FER performance compromise. The system decodes long frames of code rates 3/5, 3/4, 5/9, and 9/10 from DVB-S2 and code rates 13/45, 9/20, 23/36, 28/45, 128/180, 135/180, 140/180, and 22/30 from DVB-S2X. The hardware resources required by the multi-core architecture on a ZU11eg FPGA are 66% of LUT, 32% of Registers, 58% of Block RAMs, and 44% of Ultra RAMs. The input parallelism is $n_i = 30$, the input buffer stores $b_i = 6$ frames, and the output buffer stores $b_o = 12$ frames. The input and output buffers use Ultra RAM (URAM), which are memory blocks in the UltraScale+ family dedicated to buffering and storage.

The maximum clock frequency F_{clk} is 335 MHz. The effective air throughput T is thus given by

$$T = \min(T_i, T_e), \tag{6}$$

where T_e is given by (1) and $T_i = n_i \times F_{clk}$ is the maximum capacity of the input throughput. For $n_i = 30$ and $F_{clk} = 335$ MHz, $T_i = 10.05$ Gbit/s.

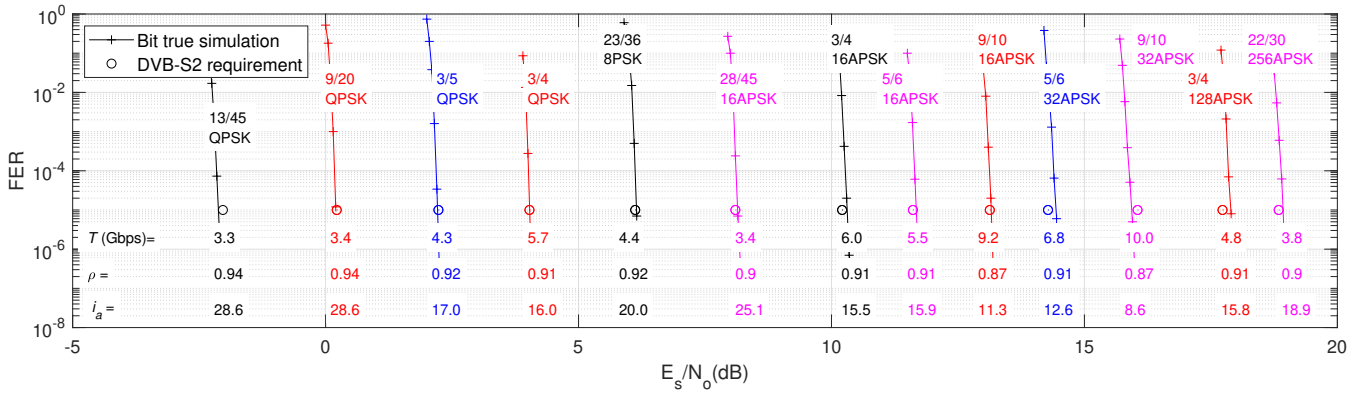

Fig. 4. Bit true simulation results for soft input values coded on 6 bits and the air throughput T for FER below 10^{-5}.

B. Simulation results

The stopping criteria used in simulations is described in [16]: when all the parities of all layers are satisfied during a decoding iteration, convergence is almost obtained. After an additional decoding iteration, the "decoding stop" flag rises, and the decoder outputs its results. Fig. 4 illustrates the bit-true C simulation results for normal frame size with a parallelism of $n_c = 360$. The decoding algorithm is an optimized offset Min-Sum algorithm with input information quantized with 6 bits. The maximum number of iterations is $i_m = 35$. Note that the performance is given considering the correction of the outer BCH code (not included in the design). For each code rate, the obtained air throughput (see (6)) and the average number of iterations i_a are given for the SNR given by DVB-S2 requirements. For code rate $r = 9/10$, the input air throughput is equal to $T = 9.2$ Gbps with the 16APSK modulation and to 10.05 Gbps (11.7 Gbps saturated by T_i) with the 32APSK modulation.

A patch based on algorithm 1 can be added to the bit-true C simulation to compute the statistics of the input buffer occupancy. However, once the $G()$ is generated by the C simulation, algorithm 1 computes the input buffer statistics on one million frames at least a thousand times faster than the bit-true C simulation with the patch. This fast computing allows testing all code rates with many throughputs (ρ) and complexity (buffer size) compromises in a reasonable time.

VI. Conclusion

In this paper, a high throughput DVB-S2 decoder is obtained by processing 3 decoders in parallel. The 3 decoders share the same input buffer of size 6 frames. The input buffer size is simulated thanks to a provided Markov model that allows the optimization of the input buffer size such that the frame error rate is not impacted by a buffer-full scenario. The output buffer regulates the variable decoding time of the frames and restores the initial frame order arrival. The output buffer is sized to cope with dramatic scenarios that may occur at high SNR. Simulations show high throughput and performance that conform with the standard requirements for a wide range of code rates. Although the multi-core architecture is implemented for the DVB-S2 standard, the presented solutions for input and output buffer control and size optimization are valid for any iterative decoder with stopping criteria.

Acknowledgement

This paper is based on work performed for WideNorth (www.widenorth.com) as part of their work in the ARTES 4.0 technology and product developments contract 'SSDR Applications' with the European Space Agency.

References

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

[2] 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

[3] "Digital Video Broadcasting (DVB); 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.

[4] S. Muller, M. Schreger, M. Kabutz, M. Alles, F. Kienle and N. Wehn, "A novel LDPC decoder for DVB-S2 IP," 2009 Design, Automation & Test in Europe Conference & Exhibition, 2009, pp. 1308-1313, doi: 10.1109/DATE.2009.5090867.

[5] C. Marchand and E. Boutillon, "LDPC decoder architecture for DVB-S2 and DVB-S2X standards," 2015 IEEE Workshop on Signal Processing Systems (SiPS), 2015, pp. 1-5, doi: 10.1109/SiPS.2015.7345034.

[6] C. -C. Cheng, J. -D. Yang, H. -C. Lee, C. -H. Yang and Y. -L. Ueng, "A Fully Parallel LDPC Decoder Architecture Using Probabilistic Min-Sum Algorithm for High-Throughput Applications," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 61, no. 9, pp. 2738-2746, Sept. 2014, doi: 10.1109/TCSI.2014.2312479.

[7] M. Li, V. Derudder, K. Bertrand, C. Desset and A. Bourdoux, "High-Speed LDPC Decoders Towards 1 Tb/s," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 68, no. 5, pp. 2224-2233, May 2021, doi: 10.1109/TCSI.2021.3060880.

[8] B. Le Gal and C. Jego, "High-Throughput Multi-Core LDPC Decoders Based on x86 Processor," in IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 5, pp. 1373-1386, 1 May 2016, doi: 10.1109/TPDS.2015.2435787.

[9] G. Bosco, G. Montorsi and S. Benedetto, "Decreasing the complexity of LDPC iterative decoders," in IEEE Communications Letters, vol. 9, no. 7, pp. 634-636, July 2005, doi: 10.1109/LCOMM.2005.1461688.

[10] M. Rovini and A. Martinez, "On the Addition of an Input Buffer to an Iterative Decoder for LDPC Codes," 2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring, 2007, pp. 1995-1999, doi: 10.1109/VETECS.2007.413.

[11] S. L. Sweatlock, S. Dolinar and K. Andrews, "Buffering requirements for variable-iterations LDPC decoders," 2008 Information Theory and Applications Workshop, pp. 523-530, doi: 10.1109/ITA.2008.4601025.
[12] C. Marchand, J. -B. Dore, L. Conde-Canencia and E. Boutillon, "Conflict resolution for pipelined layered LDPC decoders," 2009 IEEE Workshop on Signal Processing Systems, 2009, pp. 220-225, doi: 10.1109/SIPS.2009.5336255.
[13] Cédric Marchand and E Boutillon, "Before convergence early stopping criterion for inner LDPC code in DVB standards," Electronics Letters, IET, 2015, 51 (1), pp.114 - 116.
[14] Bjarne Risløw, Helge Fanebust, Cédric Marchand, Matthieu Arzel and Jean-Noël Bazin, "User terminal wideband modem for very high throughput satellites," in Ka and Broadband Communications Conference, Italy, October 2022.
[15] C. Marchand and E. Boutillon, "LDPC decoder architecture for DVB-S2 and DVB-S2X standards," 2015 IEEE Workshop on Signal Processing Systems (SiPS), 2015, pp. 1-5, doi: 10.1109/SiPS.2015.7345034.
[16] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," IEEE Workshop on Signal Processing Systems (SiPS), 2004, pp. 107-112, doi: 10.1109/SIPS.2004.1363033.