IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 1 December 2022, accepted 3 January 2023, date of publication 10 January 2023, date of current version 25 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3235966

APPLIED RESEARCH

CCSDS 131.2-B-1 Serial Concatenated Convolutional Turbo Decoder Architecture for Efficient FPGA Implementation

MIGUEL ÁNGEL PÉREZ NARANJO AND VÍCTOR P. GIL JIMÉNEZ, (Senior Member, IEEE)
Department of Signal Theory and Communications, Universidad Carlos III de Madrid, Leganés, 28911 Madrid, Spain

Corresponding author: Miguel Ángel Pérez Naranjo (mpnaranjo@tsc.uc3m.es)

This work was supported in part by the Project "IRENE" through MINECO/AEI/FEDER, UE, under Grant PID2020-115323RB-C33; and in part by the Project Madrid Flight on Chip (MFOC)-Innovation Cooperative Projects Comunidad of Madrid-HUBS 2018/MFOC.

ABSTRACT Most of the turbo encoding schemes at standards are parallel-based, so different architectures for efficient implementation are common in the literature. However, a serial turbo decoder is not that common. This scheme is used in CCSDS 131.2-B-1 standard, which is attracting much of attention recently due to its higher performance for satellite communications. In this paper, an efficient architecture for the decoder is proposed and analyzed. It is intended to show an architecture that can be modeled in a circuit description language (such as VHDL and Verilog) in such a way that it can be easily implemented on a Field Programmable Gate Array (FPGA). This work describes in detail this architecture explaining the encoding operations that are performed at the transmitter and then, how to undo them at the receiver. The proposed algorithm works by using independent components to divide the tasks and to obtain a pipeline architecture to improve the efficiency. The results of simulating and implementing the proposed architecture on a Xilinx Zynq UltraScale+ RFSoC ZCU28DR board with XCZU28DR-2FFVG1517E RFSoC are shown. The final results presented demonstrate how the hardware operations give equivalent results to the software simulation and do not consume board resources aggressively as usually the turbodecoder does.

INDEX TERMS DSP, coding, FEC, serial turbodecoder, pipeline, convolutional, BCJR, VHDL, synthesis, implementation, hardware, FPGA, CCSDS.

I. INTRODUCTION

Forward Error Correction (FEC) is a mandatory technique nowadays for the design of a high performance communications system, as it allows to detect and correct errors on the transmitted information, allowing to reach the well-known Shanon limit [1].

Among all the channel coding techniques, turbo encoding/decoding [2] is one of the most promising strategies for improving performance [3] although the complexity increases.. For this reason, there are many standards using turbo encoding/decoding [4], [5], [6]. The turbo encoding idea is the concatenation of two simple convolutional encoders, preferably those based on a recursive systematic codes (RSC) model, to improve the error correction capacity by means of an iterative algorithm, in order to achieve very low error rates without the need to use a large number of shift registers in the encoder architecture, as this would exponentially increase the complexity of the decoding process [3].

The origins of turbo decoder technology dates back to late 1980s. In 1989, Alain Glavieux proposed a modification to the Viterbi algorithm called Soft-Output Viterbi Algorithm (SOVA) [7], [8] that allowed working with soft outputs after decoding a single convolutional encoder, which led to the observation that working with soft-input and soft-output (SISO) decoders [9], [10], [11] improved the signal-to-noise ratio (SNR). When the next phases of the general structure of the turbo decoder were developed, the concatenation of the encoders made the use of SOVA for decoding unfeasible

The associate editor coordinating the review of this manuscript and approving it for publication was Jonathan Rodriguez.

and the BCJR algorithm [12], [13], also known as the forward-backward algorithm, which follows the Maximum a Posteriori (MAP) criterion, began to be used. It was created in 1974 but adapted and improved in 1993 by its inventors, Bahl, Cocke, Jelinek, and Raviv.

The first commercial use of turbo codes occurred in 1997 with Inmarsat's M4 multimedia service by satellite. This new service used the component of Turbo4 circuits [14] (CAS 5093 successor) with a 16-QAM modulation and enables the user to communicate with Inmarsat-3 spot-beam satellite from a terminal at 64 kbit/s. The narrowband technology based on a 16 QAM constellation mapping and turbo coding provides significant reduction (>50 %) in the required bandwidth for mobile satellite channels, at the same time improving the satellite power efficiency [15]. This was the beginning of turbo codes in commercial applications which took a lot of effort from several teams in hardware implementation [16], [17] and the adaptation of communications standards to these technologies in a way that would make them achievable in commercial electronics [18], [19]. However, as hardware development technology evolved and it became possible to increase the number of resources of the devices where such algorithms are to be implemented, it also became possible to increase the complexity of the turbo decoder variant to be implemented, as increasingly higher transmission rates are required and decoding operations are applied to arrays of symbols of much longer lengths, as is the case with IEEE 802.16 family of wireless communications standards [20] or HomePlug AV became an IEEE standard in 2010 [21]. It has also been a key player in more recent technologies such as faster-than-Nyquist (FTN) signaling [22], [23], [24] or coherent decoding [25], [26].

For the case of satellite communications, the performance of the turbo decoder combined with Frequency-Hopped Spread Spectrum (FH-SS) has been analysed in [27] and compared with the combined performance of several dynamic power allocation algorithms, where a modification of the classical structure is made to develop a new iterative algorithm for channel variance and carrier phase estimation (side information), which was shown to provide superior performance to the case where no side information is available [28]. A turbo trellis coded modulation in conjunction with continuous phase modulation was used in a frequency-hopping packet radio structure to further reduce the error probability too [29]. Also very interesting is the high throughput that can be achieved with error correction algorithms based on newer techniques such as Turbo-Hadamard coding methods [30]. However, all these algorithms are not yet validated in direct hardware implementation, that is, they are still complex to implement by transcribing them directly into a circuit description language, either VHDL or Verliog, without using other elements such as a microprocessor or Universal Software Radio Peripheral (USRP) that allow certain parts of the algorithm to be implemented in software. That is why the standard when using a turbo decoder scheme is to try not to add additional blocks beyond those used in classical schemes [31], which are the RSCs, an interleaver and a puncturing block in some cases to achieve the rates recommended in the standards.

Thus we come to the standard specifications recommended by the Consultative Committee for Space Data Systems (CCSDS) in [32] where optimal combinations of coding rates and frame lengths are pursued to make efficient use of bandwidth and maximising spectral efficiency.

The aim of this work is to present a valid and efficient architecture to perform decoding of the Serial Concatenated Convolutional Code (SCCC) block, described as a Serial Concatenated Convolutional (SCC) Turbo Coding Scheme, proposed in the aforementioned standard and described in [33], showing the pipelined components of the complete decoder assembly and the connections of its integrated components. In addition, detailed simulation results are shown on the code produced in VHDL, corresponding to the waveforms that would be obtained from the proposed circuit, validating the likelihoods computed in each iteration and the final bits obtained after the hard decision process, to compare those results with the software simulation using MATLAB. In addition, the synthesis and implementation performance values are included on the evaluation board, in this case, on a Xilinx Zynq UltraScale+ RFSoC ZCU28DR with XCZU28DR-2FFVG1517E RFSoC, to evaluate the resource management on it and the efficiency of the proposed architecture.

In other words, this paper presents a novel architecture not explored in the literature to provide a customized decoding scheme to [32], and optimized for electronics to facilitate its implementation in real systems. So, the main contributions of this paper are summarized as follows:

- Complete description of the proposed architecture, including the description of each component with its corresponding block diagram and the connections and signals between them for their subsequent hardware implementation. In addition, the optimized algorithm is specified for decoding each RSC individually.
- A software simulation comparing the theoretical and optimized versions of the algorithm, to demonstrate that using the appropriate number of iterations only 0.5 dB of Eb/N_0 is lost between the two versions, the latter being infinitely simpler to implement in hardware and more efficient in terms of FPGA resource consumption.
- Exhaustive comparison of the results in the software simulation with the hardware simulation to demonstrate that the values are the same and to verify that the architecture works correctly.
- Show the synthesis results of the proposed architecture to evaluate the resources it consumes on the evaluation board.
- Present the results of carrying out the implementation of the synthesised circuit to evaluate the timing and temperature performance and certify that it can be realised on the FPGA.

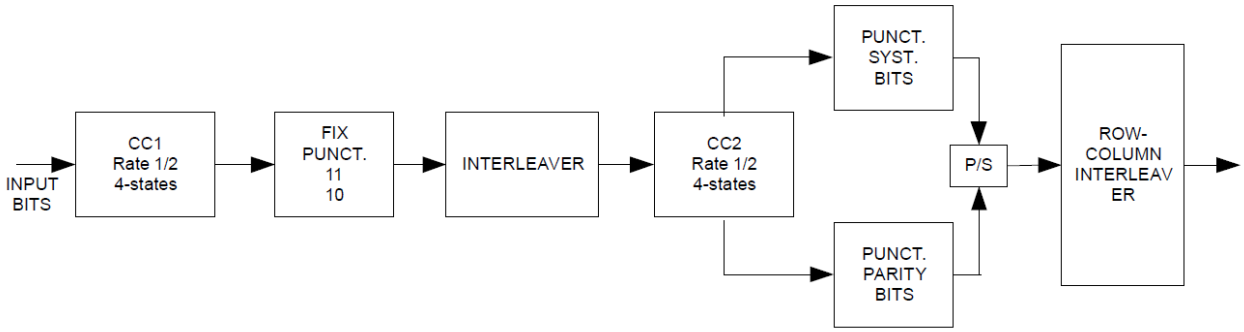

FIGURE 1. Block Diagram of the SCC Turbo Coding Scheme in the transmitter. Extracted from [32].

This paper is organized as follows: Section II describes the SCC Turbo Coding block collected in the reference standard, an overview of the BCJR algorithm for a single RSC and a the description of methods of simplifying this algorithm, which makes it possible to do such signal processing in hardware in an efficient way; section III shows the proposed architecture of the turbo decoder, detailing the pipelined structure that builds it, showing each component that makes it up and how the flow is controlled to avoid mismatches of information in the memory blocks to generically fit the different proposed lengths of the input and output frames; section IV presents hardware simulation, synthesis and implementation commented results; conclusions are presented in section V.

II. SYSTEM MODEL

A. SCC TURBO CODING SCHEME DESCRIPTION

Turbo encoder classic architecture is based on two RSCs concatenated through a memory called interleaver which is responsible for scrambling the input data to avoid bursts of consecutive encoding errors. The most common is to find the encoders in parallel, i.e. the original input and the input modified by the interleaver are encoded at the same time, although there is also the possibility of the turbo encoder appearing with the RSC in series with the interleaver in between, in such a way that the interleaver messes up the original encoded information, which means that it has a larger memory than in the parallel case. Each variant has its advantages and disadvantages, which are discussed in [34], but one important advantage that the serial configuration has over the parallel model is that the data and parity bits can exploit the extrinsic information. This advantage is one of the main reasons why in the reference standard of this work a modification of the series turbo decoder,the above mentioned SCCC configuration, is chosen.

The use of SCCC is intended mainly for high data rate applications. The Forward Error Correction (FEC) scheme is based on the concatenation of two simple four-state encoder structures. The SCCC scheme implies a Physical Layer frame of constant length, with pilots inserted in fixed positions. This architecture simplifies the synchronization procedure, thus further allowing fast and efficient acquisition at very high rates for the receiver [32]. The following sub-subsections describe the different blocks that make up the complete encoder. As it has been exposed, the turbo encoding proposed in this standard is a serial-based one, which needs a completely different architecture at the decoder side. As it can be seen in the figure, the puncturing operation after CC1 only removes one bit out of 4 of redundancy, thus the systematic bits are always transmitted.

Fig. 1 shows the block diagram for the complete encoder. It consists of two convolutional encoders, the outer one is referred to as CC1 and the inner one as CC2, a fixed puncturing block, an interleaver block, one de-multiplexer item to split the systematic and parity bits produced by encoder CC2, so that the inputs are finally reorganised to enter another interleaver, named as Row-Column interleaver, with its own puncturing patterns that would only affect the systematic or parity information separately, again preventing possible concatenation errors and making the system more robust. It should be emphasised that the puncturing block pattern, the interleaving block pattern and the additional puncturing patterns of the Row-Column interleaver depend on the mode of operation followed from those available in the reference standard. It should be noted also that it is these modes of operation that determine the lengths of the frames that are sent between the different blocks that make up the SCC Turbo Coding Scheme. Although it is not in the scope of this paper to explain in detail how the individual blocks work, it is considered necessary to provide some additional information about the encoders and the Row-Column interleaver, in order to facilitate the subsequent understanding of the proposed architecture for the decoding of the complete structure.

About the encoders, they have the same features, with coding rates 1/2 and 4 possible states in the encoding process. Fig. 2 shows the architecture of this type of encoder, where the boxes with a D inside symbolise a shift register, and the circles enclosing the '+' symbol refer to the logical operation "Exclusive OR". Initially the registers are initialised to '0',

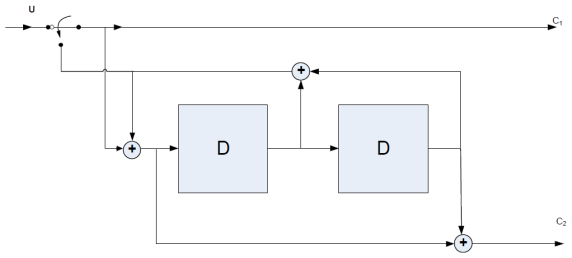

FIGURE 2. Convolutional Encoder Block Diagram for CC1 and CC2. Extracted from [32].

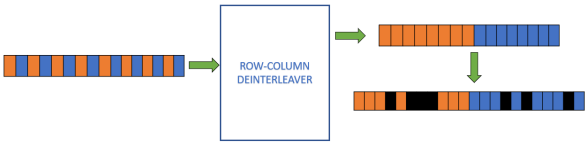

FIGURE 3. Schematic representation of the action performed by the Row-Column de-interleaver block.

and the coding takes place with the switch in upper position. Once the frame is encoded, for the final two bit times, the switch moves to the lower position to receive feedback from the registers. This feedback cancels the same feedback sent (unswitched) to the leftmost exclusive OR gate, and causes all two registers to become filled with zeros after the final two bit times. In this way, a terminated trellis has been obtained, which simplifies the decoding process.

In the case of Row-Column interleaver, its goal is to reorganize the information in such a way that the output consists first of getting all the systematic bits, applying a certain puncturing pattern to them, and then all the parity bits of CC2 also having done an additional puncturing operation. Fig. 3 presents a graphical representation of this action in the reverse way to be implemented in the receiver, in a block that is called Row-Column de-interleaver. It can be seen how the systematic and parity bits of CC2, orange and blue positions in Fig. 3 respectively, are interleaved, and after passing through the block first all the systematic bits are grouped together, and then all the parity bits. At this point, the block has independent puncturing patterns too for the systematic or parity information, which it applies to recover the corresponding position that was deleted in the transmitter when the original puncturing was applied, reflected as black filled positions in Fig. 3. Depending on the transmission mode, the amount of systematic or parity bits are different.

B. BCJR ALGORITHM OVERVIEW

As mentioned in the introduction, in a turbo decoder design the algorithm should be selected as a trade off between the decoding performance and implementation complexity. The BCJR algorithm must work with soft inputs and soft outputs, which allows it to obtain more accurate results than applying a hard method to the output. Besides, a soft output allows to make the structure more flexible to develop an iterative architecture.

Let's consider an input sequence $\mathbf{x} = x_1 x_2 \dots x_N$ of N n-bit symbols, and let u_i be a binary random variable with possible values $\{0, 1\}$ which represent the information or message input bit corresponding to estimated value according to x_k symbol. From now, if u_k is '1', it is mapped as $+1$ and '0' is mapped as -1 otherwise. Thus, taking into account the input bit a priori probability $P(u_i)$, we define the log-likelihood ratio (LLR)

$$L(u_i) = \log \frac{P(u_i = +1)}{P(u_i = -1)}, \tag{1}$$

which, at the beginning of the algorithm is zero. The reason is because we do not have any previous information and we assume the input bits are i.i.d, so $P(u_i = +1) = P(u_i = -1) = 1/2$.

The BCJR algorithm needs certain information to estimate the correct bit. This information corresponds to the sequence of symbols received, denoted as $\mathbf{y}$ for which the algorithm computes the a posteriori LLR

$$L(u_i|\mathbf{y}) = \log \frac{P(u_i = +1|\mathbf{y})}{P(u_i = -1|\mathbf{y})}. \tag{2}$$

Considering the transition from the current state ψ' to the next state ψ, and defining two sets denoted as U_1 and U_0 representing the set of transitions from state $S_{i-1} = \psi'$ to state $S_i = \psi$ originated by $u_i = -1$ or $u_i = +1$, with $i = 1, 2, \dots, N$, respectively. Thus, a posteriori LLR can be expressed as

$$L(u_i|\mathbf{y}) = \log \frac{P(u_i = +1|\mathbf{y})}{P(u_i = -1|\mathbf{y})} = \log \frac{\sum_{\mathrm{U}_1} P(\psi', \psi|\mathbf{y})}{\sum_{\mathrm{U}_0} P(\psi', \psi|\mathbf{y})}. \tag{3}$$

Applying Bayes theorem over (3) due to transitions are mutually exclusive, we finally obtain

$$\begin{aligned} L(u_i|\mathbf{y}) &= \log \frac{\sum_{\mathrm{U}_1} P(\psi', \psi, \mathbf{y}) P(\mathbf{y})}{\sum_{\mathrm{U}_0} P(\psi', \psi, \mathbf{y}) P(\mathbf{y})} \\ &= \log \frac{\sum_{\mathrm{U}_1} P(\psi', \psi, \mathbf{y})}{\sum_{\mathrm{U}_0} P(\psi', \psi, \mathbf{y})}. \end{aligned} \tag{4}$$

In (4), it is shown the joint probability of receiving the N-bit sequence $\mathbf{y}$ and being in state ψ' at time i-1 and in state ψ at the current time i. It can be seen how the final expression for calculating the LLR is a ratio of two joint probabilities, with the numerator being the joint probability of receiving $\mathbf{y}$ and being in state ψ' at time $i-1$ and in state ψ at the current time i for the set originated by $u_k = +1$, and the numerator same case for the set originated by $u_k = -1$. In turn, this joint probability can be disassembled as the product of three temporally differentiable probabilities, associated to the temporal character that reflects the computed trellis diagram of a convolutional encoder. Assuming that we are in the

i-th position of the trellis, we can define these probabilities as referring to the past state, present state and future state of that position in the diagram. Reapplying Bayes' theorem and assuming a memory-less channel where the current symbol does not depend on past information, the joint probability decomposition in the three temporal subsequences is as follows

$$\begin{aligned} P(\psi', \psi, \mathbf{y}) &= P(\mathbf{y}_{>i}|\psi', \psi, \mathbf{y}_{<i}\mathbf{y}_i)P(\psi', \psi, \mathbf{y}_{<i}, \mathbf{y}_i) \\ &= P(\mathbf{y}_{>i}|\psi)P(\psi', \psi, \mathbf{y}_{<i}, \mathbf{y}_i) \\ &= P(\mathbf{y}_{>i}|\psi)P(\mathbf{y}_i, \psi|\psi', \mathbf{y}_{<i})P(\psi', \mathbf{y}_{<i}) \\ &= P(\mathbf{y}_{>i}|\psi)P(\mathbf{y}_i, \psi|\psi')P(\psi', \mathbf{y}_{<i}) \\ &= \beta_i(\psi)\gamma_i(\psi', \psi)\alpha_{i-1}(\psi'). \end{aligned} \tag{5}$$

Attending to (5), the above-mentioned time subsequences are therefore described by those conditional and joint probabilities, in such a way that $\beta_i(\psi) = P(\mathbf{y}_{>i}|\psi)$ is the conditional probability that, given the current state is ψ, the future sequence will be $\mathbf{y}_{>i}$, $\gamma_i(\psi', \psi) = P(\mathbf{y}_i, \psi|\psi')$ defines the probability that next state is ψ, and the received symbol is $\mathbf{y}_i$ given the previous state is ψ' and $\alpha_{i-1}(\psi') = P(\psi', \mathbf{y}_{<i})$ determines the joint probability that at time *i-th* the state is ψ' and the received sequence until then is $\mathbf{y}_{<i}$. Associating the definitions of the probabilities to the grid diagram, therefore, $\beta_i(\psi)$ refers to the future transitions for moments after *i-th* instant and it is denoted as backward metric, $\gamma_i(\psi', \psi)$ refers to current transitions values and it is denoted as branch metric, and $\alpha_{i-1}(\psi')$ refers to previous transitions for moments before *i-th* instant and it is denoted as forward metric. Finally, by inserting (5) in (4) we arrive at the final LLR expression that we will implement in the FPGA to calculate the a posteriori probabilities of each decoder of the proposed architecture

$$L(u_i|\mathbf{y}) = \log\frac{\sum_{\mathrm{U}_1}\beta_i(\psi)\gamma_i(\psi', \psi)\alpha_{i-1}(\psi')}{\sum_{\mathrm{U}_0}\beta_i(\psi)\gamma_i(\psi', \psi)\alpha_{i-1}(\psi')}. \tag{6}$$

The process by which an approximate closed expression is derived for each of the time subsequences is shown below in order to facilitate the simplification process discussed in the next section where the final expressions to be implemented in VHDL are shown.

1) BRANCH METRIC COMPUTATION

Applying the definition of conditional probability, the branch metric expression can be written as

$$\begin{aligned} \gamma_i(\psi', \psi) &= P(\mathbf{y}_i, \psi|\psi') \\ &= P(\mathbf{y}_i|\psi, \psi')P(\psi, \psi') \\ &= P(\mathbf{y}_i|\psi, \psi')P(u_i). \end{aligned} \tag{7}$$

Referring to the first factor, $P(\mathbf{y}_i|\psi, \psi')$, it should been noted that the joint occurrence of the consecutive states $S_{i-1} = \psi'$ and $S_i = \psi$ is equivalent to the occurrence of the corresponding coded symbol $\mathbf{x}_i$ in the transmitter, so, $P(\mathbf{y}_i|\psi, \psi') = P(\mathbf{y}_i|\mathbf{x}_i)$. Substituting this in (7) we obtain

$$\gamma_i(\psi', \psi) = P(\mathbf{y}_i|\mathbf{x}_i)P(u_i). \tag{8}$$

To define an expression for $P(\mathbf{y}_i|\mathbf{x}_i)$, it is taking into consideration that in a memory-less channel the successive transmissions are statistically independent, so it can be written than

$$P(\mathbf{y}_i|\mathbf{x}_i) = \prod_{m=1}^{N} P(\mathbf{y}_{im}|\mathbf{x}_{im}). \tag{9}$$

In this work, in order to facilitate the hardware implementation, we consider the approximation that the channel has been modeled under the conditions of an Additive and White Gaussian Noise (AWGN) channel, therefore in [35] it is shown that the expression for (9) is as follows

$$P(\mathbf{y}_i|\mathbf{x}_i) = \mathcal{C}_i^{(0)}\exp\left\{2\mathcal{F}R\frac{E_b}{N_0}(\mathbf{x}_i \cdot \mathbf{y}_i)\right\}, \tag{10}$$

where $\mathcal{C}_i^{(0)}$ is a constant computed from channel characteristics such as fading or measure with the received sequence, $\mathcal{F}$ is the channel fading amplitude, R is the coding rate and E_b/N_0 is the energy per bit-to-noise ratio in dB of the system. The $P(u_i)$ calculation case is much simpler, since by defining

$$P(u_i = \pm 1) = \frac{\exp\{u_i L(u_i)\}}{1 + \exp\{u_i L(u_i)\}}, \tag{11}$$

and substituting this value in (1), we obtain, after regrouping the terms

$$P(u_i) = \mathcal{C}_i^{(1)}\left\{\frac{u_i L(u_i)}{2}\right\}. \tag{12}$$

Finally, substituting (10) and (12) into (8), we arrive at the final expression of the branch metrics

$$\begin{aligned} \gamma_i(\psi', \psi) &= \mathcal{C}_i^{(0)}\mathcal{C}_i^{(1)}\exp\left\{2\mathcal{F}R\frac{E_b}{N_0}(\mathbf{x}_i \cdot \mathbf{y}_i)\frac{u_i L(u_i)}{2}\right\} \\ &= \mathcal{C}_i \exp\left\{\mathcal{F}R\frac{E_b}{N_0}(\mathbf{x}_i \cdot \mathbf{y}_i)\, u_i L(u_i)\right\}. \end{aligned} \tag{13}$$

In the case of other channel models the metric can be adapted to accordingly or even used as it is in eq. 13 assuming a slight degradation in the performance.

2) FORWARD AND BACKWARD METRIC COMPUTATION

Let us define the expressions to be implemented to refer to past and future transitions with respect to the motion in the *i-th* state. Previously, the subsequence corresponding to the past had been defined as

$$\alpha_{i-1}(\psi') = P(\psi', \mathbf{y}_{<i}) \Leftrightarrow \alpha_i(\psi) = P(\psi, \mathbf{y}_{<i}, \mathbf{y}_i), \tag{14}$$

and applying definitions of probability theory [36] and assuming again the action on a memory-less channel, the expression for the forward metrics computation expression

remains in a recursive calculation as defined below

$$\begin{aligned}\alpha_i(\psi) &= P(\psi, \mathbf{y}_{<i}, \mathbf{y}_i)\\ &= \sum_{\psi'} P(\psi, \psi', \mathbf{y}_{<i}, \mathbf{y}_i)\\ &= \sum_{\psi'} P(\psi, \mathbf{y}_i|\psi')P(\psi', \mathbf{y}_{<i})\\ &= \sum_{\psi'} \gamma_i(\psi', \psi)\alpha_{i-1}(\psi'). \end{aligned} \tag{15}$$

It is worth nothing to say that constraints of memory-less channel is not mandatory for good performance. It is only for ease on the explanation. For the case of the sub-sequences corresponding to the future, the process is analogous. From the probabilistic definition

$$\beta_i(\psi) = P(\mathbf{y}_{>i}|\psi) \Leftrightarrow \beta_{i-1}(\psi') = P(\mathbf{y}_{>i-1}|\psi'), \tag{16}$$

and following the constrains of the previous case, the recursive expression for calculating future transitions is defined as follows

$$\begin{aligned}\beta_{i-1}(\psi') &= P(\mathbf{y}_{>i-1}|\psi')\\ &= \sum_{\phi} P(\psi, \mathbf{y}_i, \mathbf{y}_{>i}|\psi')\\ &= \sum_{\psi} P(\mathbf{y}_{>i}|\psi', \psi, \mathbf{y}_i)P(\psi, \mathbf{y}_i|\psi')\\ &= \sum_{\psi} P(\mathbf{y}_{>i}|\psi)P(\psi, \mathbf{y}_i|\psi')\\ &= \sum_{\psi} \beta_i(\psi)\gamma_i(\psi', \psi). \end{aligned} \tag{17}$$

Due to the fact that (15) and (17) are recursive expressions, it is necessary to define an initial value, in the case $i = 0$ for the forward metrics and $i = N$ in the case of the backward metrics. In both cases, the initial value is

$$\alpha_0(\psi) = \begin{cases} 1 & if\, \psi = 0,\\ 0 & if\, \psi \neq 0. \end{cases} \tag{18}$$

$$\beta_N(\psi) = \begin{cases} 1 & if\, \psi = 0,\\ 0 & if\, \psi \neq 0. \end{cases} \tag{19}$$

However, as discussed in [37], all these expressions are too expensive to be implemented in hardware, and therefore numerical methods have been developed that are able to present similar results to those achieved with the previous expressions but with lower computational cost.

C. BCJR ALGORITHM SIMPLIFICATION METHODS

In order to implement the convolutional decoder for a RSC based on the BCJR algorithm, it is necessary to first apply (13), and then recursively and preferably with a paralleled structure, (15),(17), (18) and (19).

In order to achieve this goal efficiently, in [38] a method based on natural logarithms is proposed, by which the expressions mentioned above are altered, thus obtaining

$$\begin{aligned}\mathcal{L}_i^{(\gamma)}(\psi', \psi) &= \log\gamma_i(\psi', \psi)\\ &= \mathcal{C}_i + \mathcal{F}R\frac{E_b}{N_0}(\mathbf{x}_i \cdot \mathbf{y}_i) + u_i L(u_i). \end{aligned} \tag{20}$$

$$\begin{aligned}\mathcal{L}_i^{(\alpha)}(\psi) &= \log\alpha_i(\psi)\\ &= \max_{[\psi]}\left(\mathcal{L}_{i-1}^{(\alpha)}(\psi') + \mathcal{L}_i^{(\gamma)}(\psi', \psi)\right). \end{aligned} \tag{21}$$

$$\begin{aligned}\mathcal{L}_{i-1}^{(\beta)}(\psi') &= \log\beta_{i-1}(\psi')\\ &= \max_{[\psi']}\left(\mathcal{L}_i^{(\beta)}(\psi) + \mathcal{L}_{N-i}^{(\gamma)}(\psi', \psi)\right). \end{aligned} \tag{22}$$

From (20), (21) and (22) it can be seen that the greatest advantage obtained is that the multiplications have been transformed into sums evaluated on the $\max_{[S_i]}(\cdot)$ function, which is nothing more than the ordinary maximum function evaluated in the current trellis state, where the highest value is sought, which is simple to implement in VHDL'93 and already compiled if VHDL'08 is used. This simplification is known as the max-log-MAP algorithm. Since logarithms are applied to recursive expressions, we must also apply logarithms to the expressions (18) and (19), obtaining

$$\mathcal{L}_0^{(\alpha)}(\psi) = \begin{cases} 0 & \text{if } \psi = 0,\\ -\infty & \text{otherwise.} \end{cases} \tag{23}$$

$$\mathcal{L}_N^{(\beta)}(\psi) = \begin{cases} 0 & \text{if } \psi = 0,\\ -\infty & \text{otherwise.} \end{cases} \tag{24}$$

There is a disadvantage with this approximation, and that is that the numerical values are slightly worse than values obtained with the original expressions. However, such degradation in performance are not very large and therefore in practice this sub optimal solution makes it perfectly viable for implementation due to the versatility it offers in terms of simplicity of implementation and computational efficiency, since in hardware the computational expressions would use auxiliary Digital Signal Processors (DSPs) that with this implementation are replaced by adders, much less expensive and faster, leaving the DSPs free for other tasks of the receiver.

Original BCJR algorithm has a disadvantage which is the problem of numerical instability that occurs in (15) and (17), since a normalization of these expressions is required for each time i and avoid overflow. Max-log-MAP algorithm solves this problem and does not require such normalization, which translates as a splitting over previous results that must be saved for subsequent iteration due to recursion, and therefore, entails extra memory cost and auxiliary DSPs in hardware.

It is important to note that in [39] another simplification method is proposed that gives the same numerical results as the original solution but with less computational expense, using the max* function, based on the Jacobian logarithm operation and defined as

$$\max^*(\theta, \phi) = \max(\theta, \phi) + \log\left(1 + \exp\{1 - |\theta - \phi|\}\right). \tag{25}$$

Algorithm 1: Max-Log-MAP Algorithm Steps

Data: u_i, $L(u_i)$, $i = 0$
Result: $L(u_i|\mathbf{y}) \in \mathbb{R}$
; /* Branch metrics computation */
while $i \neq N$ **do**
  Compute (20) for each input u_i and $L(u_i)$;
  i++;
i=0;
; /* Forward and backward metrics computation */
while $i \neq N$ **do**
  Compute (21) with $\mathcal{L}_i^{(\gamma)}(\psi', \psi)$;
  Compute (22) with $\mathcal{L}_{N-i}^{(\gamma)}(\psi', \psi)$;
  i++;
i=0;
; /* A posteriori probability computation */
while $i \neq N$ **do**
  Compute (26) with $\mathcal{L}_i^{(\alpha)}(\psi')$, $\mathcal{L}_i^{(\gamma)}(\psi', \psi)$ and $\mathcal{L}_{N-i}^{(\beta)}(\psi)$;
  i++;
i=0;

This variant is known as log-MAP algorithm, but despite giving the exact results of the original BCJR algorithm, it requires making external auxiliary modules based on Look-Up-Tables (LUTs) that would consume more board resources and would also add additional clock cycles delays. Due to that the integration of this block with the rest of the structure would harder the complete turbo decoder and the receiver itself, since the latter also uses LUTs based on counters to synchronize the sending of information from one block to another. For these reasons we have preferred to use the max-log-MAP version, since it is simpler to implement and allows the calculations to be performed in the same clock cycle.

Finally, changes made in (20), (21) and (22) have an effect on (6) where the a posteriori probabilities computation is simplified as

$$L(u_i|\mathbf{y}) = \max_{[U_1]}\left(\mathcal{L}_i^{(\alpha)}(\psi') + \mathcal{L}_i^{(\gamma)}(\psi', \psi) + \mathcal{L}_{N-i}^{(\beta)}(\psi)\right) - \max_{[U_0]}\left(\mathcal{L}_i^{(\alpha)}(\psi') + \mathcal{L}_i^{(\gamma)}(\psi', \psi) + \mathcal{L}_{N-i}^{(\beta)}(\psi)\right). \tag{26}$$

For implementation purposes, max-log-MAP algorithm summarized steps are described in Algorithm 1 box. It must be emphasised that while statements in Algorithm 1 box are not apply to do a loop in hardware, they are just to note that in the implementation should be a pipelined structure with flow control.

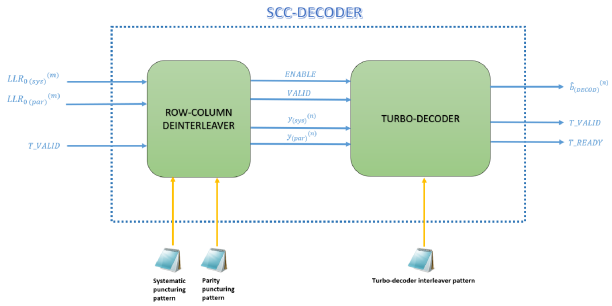

FIGURE 4. SCC-Decoder full hardware block schematic.

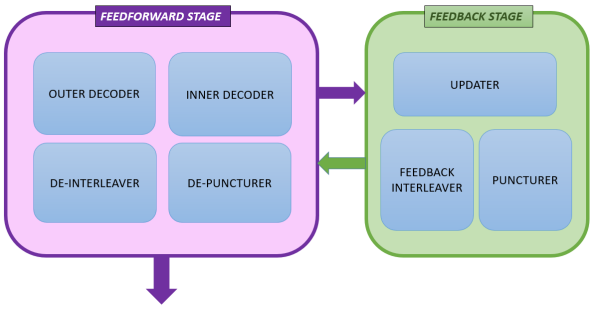

FIGURE 5. Main block diagram of the two major stages of the proposed turbo decoder architecture.

III. PROPOSED PIPELINED ARCHITECTURE

This section shows the schematics of the proposed design to implement the complete turbo decoder structure in hardware.

First, Fig. 4 shows a schematic of the inputs and outputs of the above modules. The Row-Column de-interleaver input, which is the input of the complete SCC-decoding block, takes the output values from the previous soft demodulator and a trigger signal that is activated while calculating those soft values a priori. The demodulator is designed to, on the same clock edge, provide the soft bit corresponding to the systematic bit and the parity bit coming out of the CC2 encoder, according to the transmission scheme shown in Fig. 2. Once these soft bits are obtained, while the trigger signal arriving from the demodulator is enabled, the storing circuit is also active, which is responsible for storing the incoming information, until finally the maximum of the frame has been reached and the demodulator is deactivated. Therefore, the incoming trigger signal to our block is not more enabled and the storing circuit is stopped to perform the interleaving and puncturing operations corresponding to that block.

The proposal for this next block can be simplified in the diagram in Fig. 5, where the decoder operation has been divided into 2 main stages. The first stage consists of decoding by inverting the steps of the proposed scheme, where a block with the VHDL implementation of the max-log-MAP algorithm corresponding to CC2 would be applied, then the effect of the interleaver and the fixed puncturing

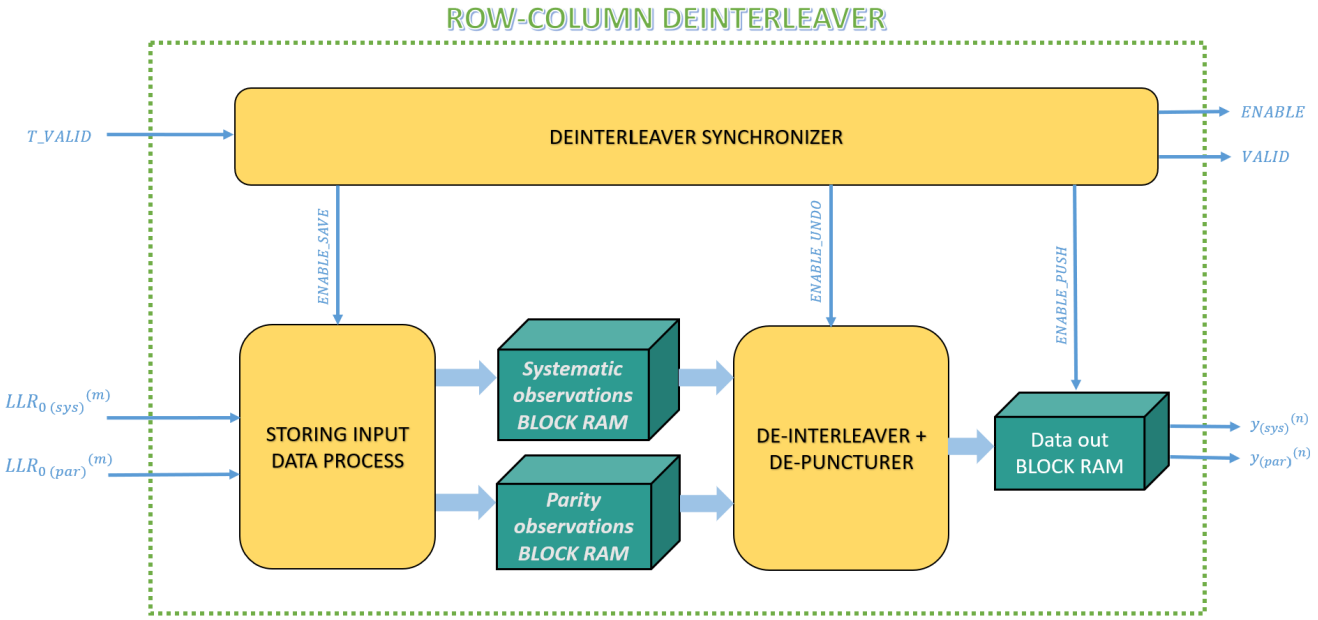

FIGURE 6. Full block diagram for the hardware components and connections corresponding to the Row-Column de-interleaver block.

block would be undone, and finally the max-log-MAP algorithm corresponding to CC1 would be applied again on the processed information. These four big blocks are collected in the purple area of Fig. 5, named as feedforward stage, where the next step is to check the iteration on which we are, if it has already been the last one, we take out the calculated likelihoods to apply a hard decisor and obtain the estimated bits, otherwise we go to the second stage. The second step consists of an update of the a priori observations from the first decoding block, obtaining the inputs from the second decoding block, updating their values and applying puncturing to them to adjust the size of the first decoding block frame, which is fixed, due to the first decoding block inputs always are the observations from the Row-Column de-interleaver, which are themselves stored in auxiliary block RAMs. Also, updated a priori observations are modified by the original interleaver. These stage refers to green area in Fig. 5, named as feedback stage. Data in this stage always returns to feedforward stage.

Further details of the Row-Column de-interleaver can be seen in Fig. 6, where the pipelined components that compose it are shown in yellow. Also, in same figure, it is shown several rectangular prisms in dark turquoise blue that represent the RAM blocks where the information used in the blocks is stored for processing. In Fig. 6, it can be seen how the block consists of a master process, called *deinterleaver synchronizer*, which is responsible for activating each component at the right time, as it has followed a parallel architecture and the blocks are independent, so that when implementing on the FPGA resources are allocated more efficiently. The master process activates the storing circuit in such a way that the input information is stored in two RAM blocks. Subsequently, when no more data enters, the *de-interleaving + puncturing* process is activated to perform the behavior shown in Fig. 3. For this, the data is stored in another auxiliary RAM block in the appropriate order, and finally, reading the puncturing patterns corresponding to the operation mode, implemented by LUTs where '1' is stored if the corresponding position is not deleted and '0' if a position on which puncturing has been performed in transmission has to be added, the corresponding positions are added. Finally, when the end of the LUT corresponding to the puncturing patterns is reached, the information stored in the auxiliary RAM block is sent to the block corresponding to the turbo decoder.

The more detailed separate components that make up the turbo decoder block (hardware-oriented component implementation of the complete decoding algorithm) and the connections between them to control the data flow between the two stages are presented in Fig. 7. As with the Row-Column de-interleaver block architecture, a master process called *turbodecoder synchronizer* is also used here, which is responsible for enabling each of the independent components that perform the operations mentioned above (activation of decoders, interleaver and puncturer circuits for the feedforward stage, and updater, puncturer and interleaver circuit for the feedback stage). As in the previous case, it is shown in yellow every pipelined component used in this

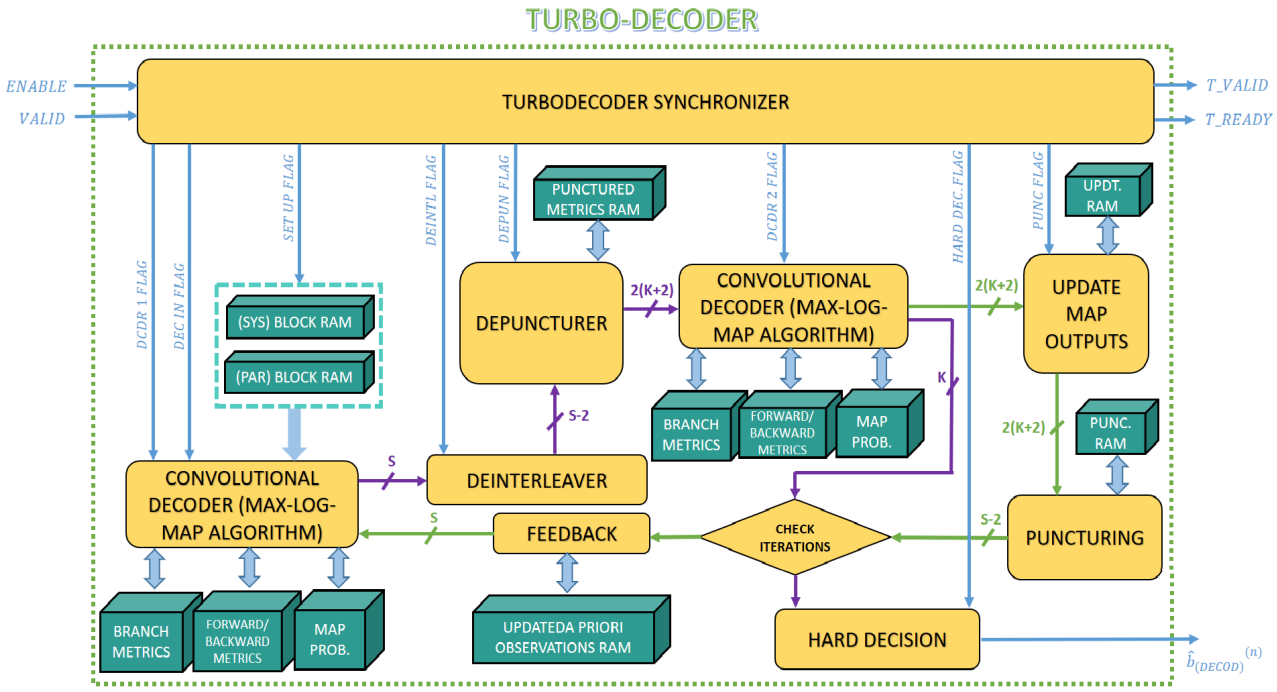

FIGURE 7. Complete block diagram of the proposed hardware architecture for the corresponding turbo decoder block.

architecture and as a rectangular prism in dark turquoise blue for RAM blocks, but in addition, two marked data flows have been added, a purple-coloured path reflecting the route followed in the feedforward stage and a green-coloured path showing the route followed in the feedback stage, in accordance with the diagram presented in Fig. 5. The rest of the blue markers refer to the trigger signals of the independent components, which are handled by the *turbodecoder synchronizer* process.

Before explaining the control logic over the data flow, it should be noted that depending on the used transmission mode, the dimensions of the frames within the block will vary. These dimensions are given in the reference standard and can be summarised in up to two single variables in this block: an integer variable S representing the number of systematic bits generated in the complete encoder, and another integer variable K representing the encoder input frame size [32], both referring to the transmitter components. Therefore, in the block diagram in Fig. 7, the dimensions of the frames according to these variables have been added to all the paths corresponding to the two main stages of the block.

The operation is therefore as follows: in the case of the first iteration, the synchronising signals between the Row-Column de-interleaver and the turbo decoder are only activated while information is sent from the former to the latter, where that information is stored in RAM blocks. Once these signals are deactivated, the 1st convolutional decoder (corresponding to CC2 applying max-log-MAP algorithm) starts to operate taking a priori observations as null, since i.i.d. inputs are considered and therefore their LLR is zero, which produces the first set of observations a posteriori with respect to the initial input information. These observations are reordered according to reverse the original interleaving pattern assigned by a LUT depending on the transmission mode, which generates a frame of 2 observations less than the input, as the interleaver length is S-2 according to the standard, regardless of the transmission modes. The de-interleaved observations are then passed through the depuncturer block, where the dimension of the incoming frame is increased to undo the effect of the fixed puncturing pattern applied in transmission and match the length of the output frame of CC1 in transmission. As explained above, this puncturing pattern is fixed in the standard, and as in the transmitter it consisted of removing one observation every four coming out of CC1, this block in reception generates an extra observation every 3 received from the de-interleaver block. Next, the 2nd convolutional decoder (corresponding to CC1 and also applying the max-log-MAP algorithm version) is applied and the first set of a posteriori observations of the complete turbo decoder is obtained, finishing the operations corresponding to first iteration of the feedforward stage. This decoder also works with null a posteriori likelihoods, but unlike the first one, these are not going to be updated in any iteration, since the information about the changes of the interleaver

and puncturer blocks are the ones used to update only the 1st decoder at the beginning of a new iteration, i.e. the modifications are included in the first decoder. This bring us a simpler and efficient architecture, due to it is not necessary to do extra computation for this block. At this point, it is evaluated whether the maximum number of iterations set has been reached. If this is the case, the hard decision is made on the last a posteriori likelihoods obtained, but as we are still in the first iteration, we move on to the feedback stage, and therefore the a posteriori likelihoods are discarded (frame of length K) and the updated 2nd convolutional decoder input (frame of length 2(K+2)) is sent to the 2nd convolutional decoder block. In this way the original puncturing pattern is performed and the interleaving operation is applied to these observations, which finally return to the 1st convolutional decoder block and will be the a priori observations to be added to the original inputs in the next iteration.

This process will be repeated until the maximum number of iterations is reached, and as mentioned above, it will be applied in a hard decision on the a posteriori likelihoods coming from the 2nd decoder, such that a '0' is chosen if the likelihood is negative, and '1' otherwise. An efficient way to perform this operation in hardware is just to invert the sign bit of the observation value to be evaluated, i.e. if we obtain an observation whose value as a real number is negative, its sign bit will always be '1', and therefore the result of the circuit is simply to apply a NOT gate to this bit, obtaining a '0'. The same operation would be done in the case of a positive likelihood.

IV. RESULTS AND ANALYSIS

This section describes the results obtained with the proposed architecture. Before analysing the simulation and implementation results. It is important to note the degradation suffered with the max-log-MAP algorithm approximation with respect to the optimal version described by (25). For this purpose, Fig. 8 shows the software simulation results of the Bit Error Rate (BER) obtained using the complete architecture for a different number of iterations and for a low range of values over the Eb/N_0. This figure is intended to show the difference in error correction gain achieved by applying the optimized algorithm with the theoretical, or optimal, algorithm (much more expensive and difficult to implement in hardware) since both versions will correct all frame errors but with different values of Eb/N_0 (in dB).

Two groups of curves can be seen in the figure, the solid line curves corresponding to the optimal version of the BCJR algorithm and the dashed line curves corresponding to the results of applying the max-log-MAP algorithm. For the case in which only one iteration of the turbodecoder (associated to both blue curves) is performed, the difference between the optimal version and the simplified approach is null, since the system has not been fed back by updating the information a priori, so the differences between the two versions of the algorithm cannot be appreciated. However, when a second iteration is performed, the difference in system performance

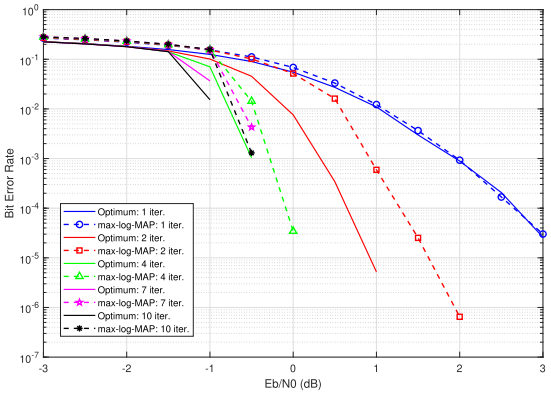

FIGURE 8. BER results using the optimal version of the BCJR algorithm (solid lines) and using the simplified version based on the max-log-MAP algorithm (dashed line) for the complete turbo decoder architecture.

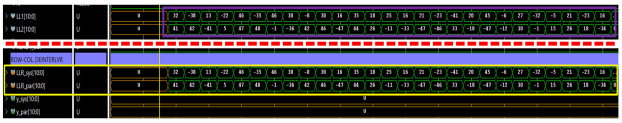

FIGURE 9. Beginning of the Row-Column de-interleaver input frame in the hardware simulation.

is already noticeable, as shown by the red curves. This trend is maintained as the number of iterations increases, being the system able to correct all errors more drastically for a higher number of iterations than the previous cases, until in the case shown by the magenta and black solid lines, corresponding to use 7 and 10 iterations respectively, the system converges and corrects all errors for an $Eb/N_0 = -1$ dB, and its simplified version (associated to the dashed curves with the same colors) shows the same behavior in the case of -0.5 dB. Therefore, it has been decided to implement the system using 7 iterations instead of 10 in order to realize a more efficient system that spends less processing time and FPGA resources.

A. HDL SIMULATION

This subsection shows the hardware simulation, after having applied the architecture described in this paper in VHDL. Simulation is the processing of a complete frame corresponding to having used transmission mode 1, consisting of 16000 observations from the soft demodulator, will be established with the turbo decoder set to run at 7 iterations and operating at a clock frequency of 100 MHz. To verify the results shown and for simplicity for the reader, integer arithmetic is considered. For this, a random synthetic signal of logic vectors is generated, but it will be evaluated using the Vivado converter to signed integers, and the input signal will be cloned with MATLAB to show that the results of the likelihoods and the final logic signal are correct, obtaining the same results in both tools.

The result of the hardware simulation of the Row-Column de-interleaver block is shown in Figures 9 and 10. Fig. 9 shows the Row-Column de-interleaver own input, which

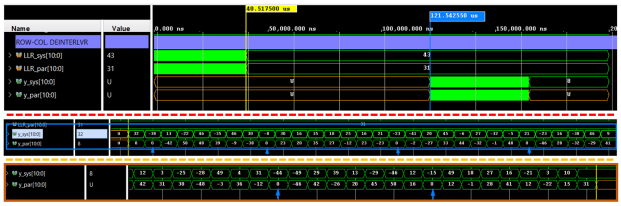

FIGURE 10. Action time of the block corresponding to the Row-Column interleaver.

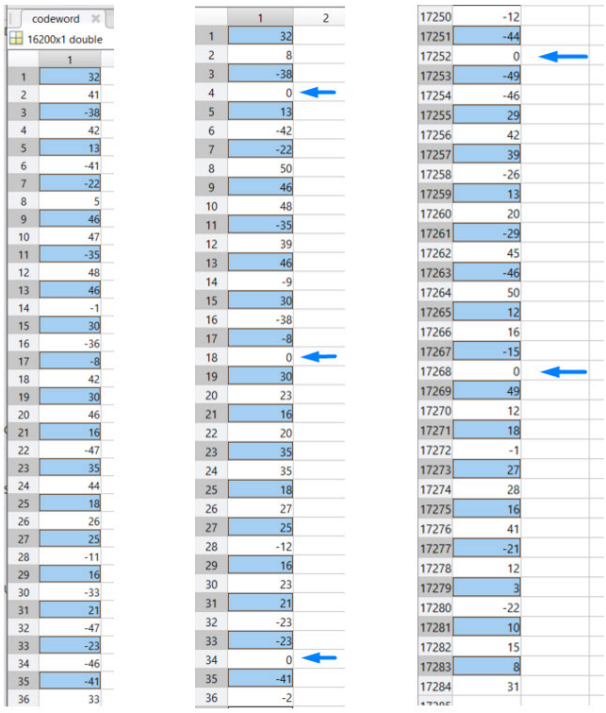

FIGURE 11. Numeric values of the beginning of the Row-Column de-interleaver input frame (left) and of the beginning (middle) and (right) of the Row-Column de-interleaver output frame.

is mapped to samples received from the soft demodulator. It can be seen in this figure how the inputs of this block (marked with a yellow marker) correspond to the outputs of the demodulator (named as 'LL1' and 'LL2' in the figure). Fig. 10 presents the time transition between all the operations corresponding to this complete block. The yellow vertical time marker indicates the beginning of the de-interleaving and puncturing process, and the blue marker indicates the end and the beginning of the ejection of the modified information to the turbo decoder. Also in this figure, it is mentioned the beginning and the end of the modified Row-Column de-interleaver frame, respectively, marking with blue arrows also the positions where extra values have been added as a result of the depuncturer process. The beginning of the frame is the figure divided by a red dash line and the end of the same frame is signalized with an orange dash line, all in the same figure.

The results shown can be validated with the software results shown in Fig. 11, where the systematic observations have also been highlighted in blue, leaving the unmarked ones

FIGURE 12. Waveforms corresponding to the turbo decoder 1st iteration process.

as those corresponding to parity. Blue arrows are also used to indicate the action of the depuncturer on the corresponding positions.

Fig. 12 shows all the trigger signals corresponding to the subsystems involved in the decoding for a single iteration. The signals 'y1' and 'y2' are the outputs of the Row-Column interleaver, while the signals highlighted with an orange box symbolize the activation of each of the blocks that make up the turbo decoder architecture. It is shown how these signals are activated sequentially to maintain the flow of the complete turbo decoding algorithm, and also the end of the second decoder (where the max-log-MAP algorithm has been applied for the second time in this first iteration) has been marked with a circle and a red arrow to symbolize that this is where the likelihoods of interest to be evaluated are obtained after a fixed number of iterations (7 according to the gain simulation measurements as indicated at the end of the second paragraph of section IV). Finally, separated by a red and orange dashed line, the numerical values of the beginning and end of the plot, respectively, of final likelihoods obtained in the first iteration of the architecture are shown. These values are to be compared with those obtained in the software simulation to check the correct functioning of the architecture in hardware.

These trigger signals are responsible for enable each of the sub-circuits marked in yellow in Fig. 7 in the appropriate order to maintain the correct flow of the algorithm. According to the label used for each signal in Fig. 12, it is summarized in Table 1 which sub-circuit is activated and which function each activation signal performs.

Fig. 13 presents the numerical values of the output of the first decoder for each of the iterations in software simulation, to validate the data of the algorithm written in VHDL. The left column shows the values at the beginning of the frame and the right column shows the values at the end of the frame.

Fig. 14 shows the waveforms obtained when the complete decoding process has been completed, reaching the maximum number of configured iterations. In the figure itself, an orange grid has been added in the part according to the trigger signals that divide the activation's of each iteration. It can be seen how the patterns of the activation signals are repeated during all iterations, according adapting to the behavior seen in Fig. 12, where in addition a trigger signal is activated which is held at '1' for the rest of the iterations and returns to '0' when

TABLE 1. Associated circuits and functions of each trigger signal shown in Fig. 12.

Stage	Signal name	Fig. 7 sub-circuit	Operation done
FEEDFORWARD	SET-UP-F	SET UP BRAMs	Start the process, save the initial observations in memory to use at the beginning of each iteration.
	DCDR-1-F	OUTER DECODER	Enables the outer decoder circuit.
	DEC-IN-F	OUTER DECODER	It allows avoiding losing information after the feedback stage.
	DEINTL-F	DEINTERLEAVER	Enables the De-Interleaver circuit.
	DEPUN-F	DEPUNCTURER	Enables the De-Interleaver circuit.
	DCDR-2-F	INNER DECODER	Enables the inner decoder circuit.
	HARD-D-F	HARD DECISION	Enables the hard decision circuit at the end of the algorithm.
FEEDBACK	NEW-IT-F	NEW ITERATOR FLAG	Signaling when a new iteration is on (Non first or final iteration)
	UPDATE-F	UPDATE	Enable the full feedback action with INNER DECODER results (a posteriori estimated likelihoods).
	PUNCTU-F	PUNCTURER	Apply original puncturing pattern
	FEEDBK-F	FEEDBACK INTERLEAVER	Apply original interelaver pattern and feedback OUTER DECODER

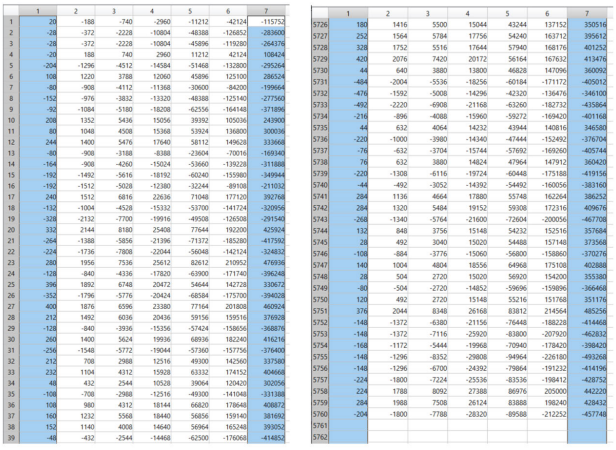

FIGURE 13. Beginning (left) and end (right) of the numerical values frame for the feedforward stage output observations for all iterations.

FIGURE 14. Waveforms corresponding to performing the turbo decoder iterations.

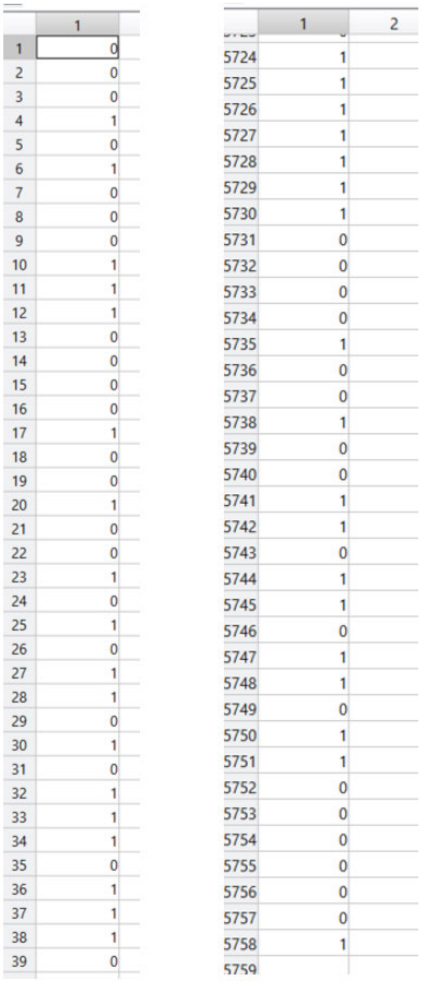

FIGURE 15. Bits obtained after performing the hard decision process in software simulation. Initial (left) and final values (right).

the iteration is finished and the hard decision is about to be made. Below, separated with a dashed red line, the update of the values of the likelihoods of interest on which to do in the last iteration the hard demodulation and obtain the estimated bits of the frame is shown. It can be seen how the behavior of the previous figure is maintained without anomalies in each iteration. At the beginning, it has been marked with a red circle the activation of the signal called in the figure as 'SET-UP-F', which is activated only in the first iteration, and remains off in the rest of iterations, since this signal only indicates that the original input values on which it is necessary to iterate have been stored in memory. Likewise, with a purple circle we have marked the end of the last iteration, where only the signal 'HARD-D-F' is activated, which applies hard demodulation on the last likelihoods calculated to obtain bits, with the method explained in the last paragraph of Section III. Therefore, the behavior is shown to be correct since there are markers at the beginning and end of the architecture.

Finally, the results of the hard decision process are shown in figures 20-22. Fig. 20 simply shows the results of applying the proposed software architecture on the signed integer

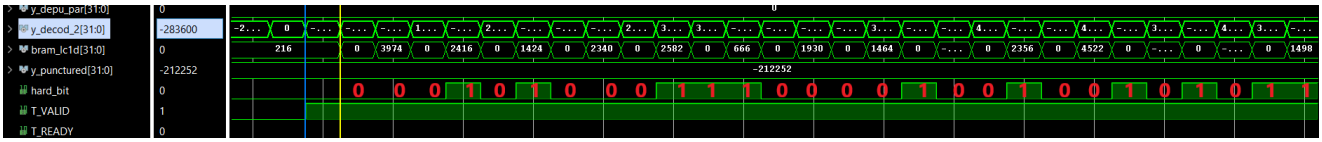

FIGURE 16. Beginning of the bit frame obtained at the output of the SCC-decoder block.

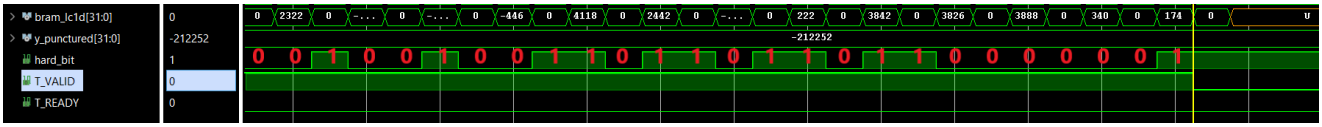

FIGURE 17. End of the bit frame obtained at the output of the SCC-decoder block.

TABLE 2. Resources used by the FPGA after synthesising the proposed architecture.

Resource	Estimation	Available	Utilization (%)
LUT	6914	425280	1.63
FF	2542	850560	0.3
BRAM	273.5	1080	25.33
IO	28	347	8.07
BUFG	1	696	0.14

results used in this simulation. Figures 21 and 22 present the start and end of the bit frame obtained after the complete decoding process, respectively. Also, in these figures, the value of the corresponding bit at each clock edge has been marked to make it easier to compare the results.

B. SYNTHESIS RESULTS

Once it has been shown that the hardware description of the proposed architecture shows waveforms that suit our objectives, we move on to synthesise the circuit to see the resources it consumes. The synthesis tool used is Vivado and, as mentioned before, the FPGA is the Xilinx Zynq UltraScale+ RFSoC ZCU28DR model. The results of the synthesis are summarised in Table 2.

This table 2 shows that the proposed architecture as a whole expends 6914 LUTs, 2542 Flip-Flops (FF), 273.5 RAM blocks (BRAM), 28 input-output (IO) nets and 1 clock buffer (BUFG). It should be emphasised that the BRAM components used in this design refers to the set of memories used in the design, as the architecture in general uses 39 memories, distributed as follows: 12 RAMs of 207 Kbits, 12 RAMs of 180 Kbits, 2 RAMs of 174 Kbits, 3 RAMs of 168 Kbits, 3 RAMs of 126 Kbits, 2 RAMs of 92 Kbits, 2 RAMs of 87 Kbits and 3 RAMs of 84 Kbits. These sizes are related to the turbo decoder parameters in the standard, which define the frame lengths with which it operates. In this way, it is possible to customise the RAMs to be used in the circuit and use the right amount of bits to implement the hardware. Thus, a component such as the decoder that usually consumes a lot of receiver resources has been optimized to

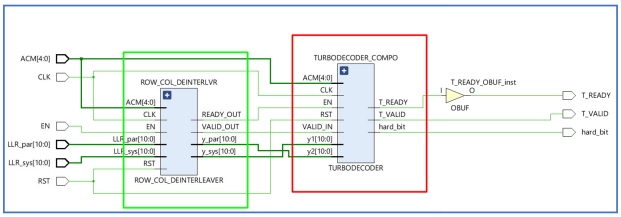

FIGURE 18. Main view of synthesized circuit. The Row-Column De-inteleaver is marked in green and the turbodecoder block is marked in red.

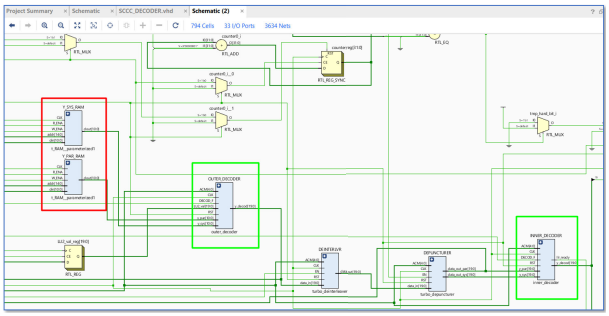

FIGURE 19. Some implemented and encapsulated sub-circuits of Figure 17 inside the turbodecoder block.

use around 25% of the FPGA's available BRAMs. The rest of the resources used are residual considering all those available.

The synthesis results shown can be graphically complemented with the RTL models that Vivado generates from the VHDL code written to implement the equations corresponding to the proposed architecture. Figure 18 shows the two major components of the architecture, which are the Row-Column de-interleaver (marked in green) and the turbodecoder block (marked in red).

Within the turbodecoder block, the submodules of Figure 7 are implemented, some of which are shown in Figure 18. However, it should be noted that for the system to function correctly, a large number of primitives are generated that clutter the canvas, so a zoom of the complete block is provided. In this figure the initial RAMs have been marked in red, used to store the initial frames coming from the Row-Column de-interleaver so that they can be used at the

TABLE 3. Summary of results of timing test on the proposed implemented architecture running at 100 MHz.

Design timing summary					
Setup		**Hold**		**Pulse Width**	
Worst Negative Slack (WNS)	2.354 ns	Worst Hold Slack (WHS)	0.004 ns	Worst Pulse Width Slack (WPWS)	4.458 ns
Total Negative Slack (TNS)	0 ns	Total Hold Slack (THS)	0 ns	Total Pulse Width Slack (TPWS)	0 ns
Number of Failing Endpoints	0	Number of Failing Endpoints	0	Number of Failing Endpoints	0
Total number of Endpoints	16447	Total number of Endpoints	16447	Total number of Endpoints	3119

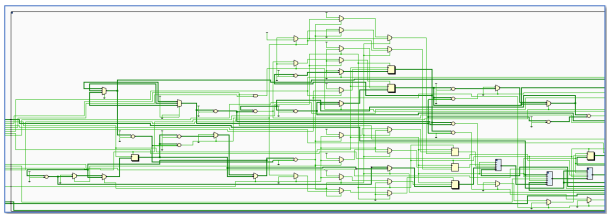

FIGURE 20. Full RTL implementation of turbodecoder block.

beginning of each iteration of the algorithm and not be lost in the following clock edges, and in green the circuits corresponding to the implementation of the max-log MAP algorithm have been marked. You can also see other blocks of the architecture such as the de-interleaver or the de-puncturer and see the connection between them, i.e., how the blocks are connected and how the signals enter according to the workflow shown in Figure 17. It should be noted that without making cuts and zooming the images, a large number of primitives are generated, which, as mentioned above, occupy the entire canvas. This case is shown in Figure 19, where it can be seen how you get very little information from the full RTL implementation. This does not matter at the hardware level of the FPGA since its ability to work in parallel allows using different areas of the board to perform operations and modules at the same time. That's why, to facilitate the reading of the paper and not to introduce too much information about the implementation that would complicate its reading, only some details in the implementation are indicated to give a more precise vision.

C. IMPLEMENTATION RESULTS

Finally, the results of implementing the architecture described during this work are shown. In the case of end resource usage, it is the same as shown in Table 2 except that the implemented design uses 6807 LUTs instead. For the case when timing constrains are evaluated, results are shown in Table 3. It should be remembered that implementation has been carried out for a clock frequency of 100 MHz. These results reveal that timing requirements are very comfortably met in the WNS and WPWS measurements, so higher frequencies could be used. The results on the WHS measure are a little tighter, but this is to be expected since this measurement occurs according to the worst case FPGA resources hold up, and more taking into account that many BRAMs components are used where there have to be a lot of connections between LUTs, nets and these blocks,

TABLE 4. Summary of results of the power test carried out on the proposed implemented architecture.

Power test parameter	Value
Total On-Chip Power	1.386 W
Junction Temperature	26,2 ºC
Thermal Margin	73,8 ºC (84,7 W)
Effective ΘJA	0,8 ºC/W
Power supplied to off-chips devices	0 W

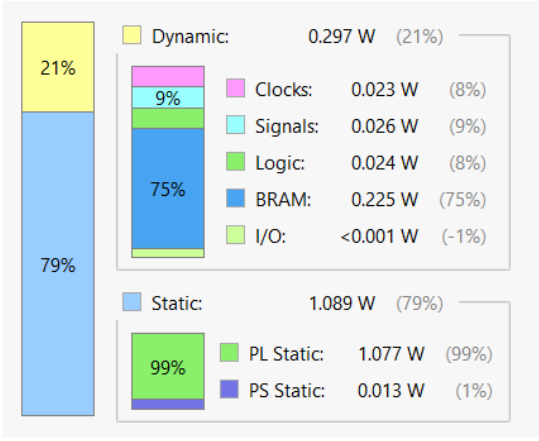

FIGURE 21. Summary of power expended by each on-chip component.

which affects the performance, but as the results show it is not a problem in the implementation. In the case of power consumption, the results are quite good as there is still a lot of thermal margin. These results are shown in Table 4.

In addition, as the resources used in the implemented design have been analysed, a breakdown of the individual power consumed by each component on these on-chip resources is added in Fig. 21. As expected, the BRAM blocks represent the highest consumption of the FPGA due to the fact that they represent the largest number of components used in the proposed design. All these implementation results are due to the routing positioning shown in Fig. 22, where a schematic of the FPGA is shown and the primitives that have been assigned to each of the processes describing the proposed architecture are marked in blue. It can be seen how the implementation tool has decided to put the primitives in the northern part of the device, i.e. those areas with a higher value for the Y coordinates with respect to the different clock sections.

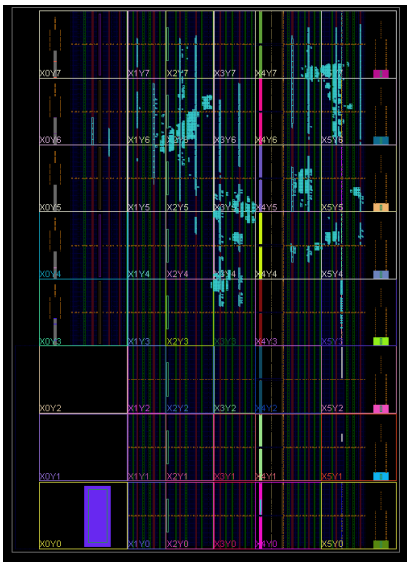

FIGURE 22. Implemented design on the FPGA device.

V. CONCLUSION

The work presented in this paper proposed a simple architecture for the implementation in the Xilinx Zynq UltraScale+RFSoC ZCU28DR evaluation board a decoding scheme valid for the SCC Turbo Coding Scheme block suggested in the CCSDS 131.2-B-1 standard. The algorithms to be implemented in the design have been detailed as well as complete and explained schemes on the data flow treatment and the nets connections that would be necessary to be able to be implemented in hardware. In addition, a structure consisting of independent components has been presented that favours a pipeline architecture, separating as much as possible the resources to be used by the FPGA, which is an advantage for the efficiency of the design. Results have been presented on a hardware simulation based on integer arithmetic to simplify the validation of results, showing that the results match the software simulation results and proving the effectiveness of the proposed architecture. Finally, it has been verified that the design is synthesizable and passes the time and temperature tests running the device at 100 MHz, as well as presenting the resources consumed by the FPGA, which are used in a very small percentage with respect to those provided by this board and a scheme of the implemented design on the device. Although the proposed scheme in this paper is for the standard CCSDS 131.2-B-1, the ideas and architecture can be easily extrapolated to other serial-based turbodecoder scheme what makes the contribution of this paper more valuable.

REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul. 1948.
[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 2, May 1993, pp. 1064–1070.
[3] C. Berrou and A. Glavieux, *Turbo Codes*. Hoboken, NJ, USA: Wiley, 2003.
[4] *BTM Synchronization and Channel Coding: Blue Book*, Consultative Committee for Space Data Systems, Standard CCSDS 131.2-B-0, Mar. 2013.
[5] *Low-density Parity-Check Codes for use in Near-Earth and Deep Space Applications. Experimental Specification*, Standard CCSDS 131.1-O-0.4, Consultative Committee for Space Data Systems, Mar. 2006.
[6] *Use of DVB-S2 ETSI Standard in High Data Rate Telemetry for Near Space-Earth Transmissions. Experimental Specification*, Standard CCSDS 131.1-O-0.4, Consultative Committee for Space Data Systems, Mar. 2006.
[7] G. Battail, "Weighting of the symbols decoded by the Viterbi algorithm (in French)," *Ann. Télécommun.*, vol. 42, pp. 31–38, Jan. 1987.
[8] J. Heller and I. Jacobs, "Viterbi decoding for satellite and space communication," *IEEE Trans. Commun. Technol.*, vol. CT-19, no. 5, pp. 835–848, Oct. 1971.
[9] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "A soft-input soft-output maximum a posteriori (MAP) module to decode parallel and serial concatenated codes," *Telecommun. Data Acquisition Prog. Rep.*, vol. 42, pp. 1–20, Nov. 1996.
[10] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 2, pp. 260–264, Feb. 1998.
[11] S. Benedetto and G. Montorsi, "Performance of continuous and blockwise decoded turbo codes," *IEEE Commun. Lett.*, vol. 1, no. 3, pp. 77–79, May 1997.
[12] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (Corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 2, pp. 284–287, Mar. 1974.
[13] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding," *Eur. Trans. Telecommun.*, vol. 8, no. 2, pp. 119–125, Mar. 1997.
[14] M. Jezequel, C. Berrou, C. Douillard, and P. PENARD, "Characteristics of a sixteen-state turbo-encoder/decoder (Turbo4)," in *Proc. Int. Symp. Turbo Codes Rel. Topics*, Brest, France, Sep. 1997, pp. 280–283.
[15] S. A. Barbulescu, "Turbo codes on satellite communications," in *Turbo Code Applications: A Journey from a Paper to Realization*, K. Sripimanwat, Ed. Dordrecht, The Netherlands: Springer, 2005, pp. 257–299, doi: 10.1007/1-4020-3685-X_11.
[16] M. Rice, P. Gray, S. A. Barbulescu, and W. N. Farrell, "Bandwidth efficient turbo coding for high speed mobile satellite communications," 1997.
[17] S. S. Pietrobon, "Implementation and performance of a turbo/MAP decoder," *Int. J. Satell. Commun.*, vol. 16, no. 1, pp. 23–46, Jan. 1998.
[18] *Introduction to CDMA2000 Standards for Spread Spectrum Systems*, 3rd Generation Partnership Project Jul. 1999.
[19] *Physical Layer Standard for CDMA2000 Spread Spectrum Systems*, 3rd Generation Partnership Project, Jul. 1999.
[20] D. Wisdom, E. Ajayi, U. Arinze, O. Aladesote, A. Ganya, H. Idris, and D. Wisdom, "IEEE compter society–Nigeria—Technical paper series a comprehensive survey on power saving schemes (CSPSS) in IEEE 802.16e/m networks," Jul. 2021.
[21] H. A. Latchman, S. Katar, L. Yonge, and S. Gavette, "The HomePlug AV network architecture," in *Homeplug AV and IEEE 1901: A Handbook for PLC Designers and Users*, 2013, pp. 12–17, doi: 10.1002/9781118527535.ch2.
[22] S. Li, B. Bai, J. Zhou, P. Chen, and Z. Yu, "Reduced-complexity equalization for faster-than-Nyquist signaling: New methods based on Ungerboeck observation model," *IEEE Trans. Commun.*, vol. 66, no. 3, pp. 1190–1204, Mar. 2018.
[23] F.-L. Luo and C. Zhang, *Faster-than-Nyquist Signaling for 5G Communication*. Chichester, U.K.: Springer, 2016, pp. 24–46.
[24] J. Fan, S. Guo, X. Zhou, Y. Ren, G. Y. Li, and X. Chen, "Faster-than-Nyquist signaling: An overview," *IEEE Access*, vol. 5, pp. 1925–1940, 2017.
[25] V. K. Veludandi, "BCJR vs SOVA for a practical coherent turbo coded OFDM system," in *Proc. 10th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2019, pp. 1–5.
[26] K. Vasudevan, "Coherent detection of turbo-coded OFDM signals transmitted through frequency selective Rayleigh fading channels with receiver diversity and increased throughput," *Wireless Pers. Commun.*, vol. 82, no. 3, pp. 1623–1642, Jun. 2015.

[27] J. H. Kang and W. E. Stark, "Turbo codes for noncoherent FH-SS with partial band interference," *IEEE Trans. Commun.*, vol. 46, no. 11, pp. 1451–1458, Nov. 1998.
[28] H. El Gamal and E. Geraniotis, "Turbo codes with channel estimation and dynamic power allocation for anti-jam FH/SSMA," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, vol. 1, Oct. 1998, pp. 170–175.
[29] J. H. Gass, P. J. Curry, and C. J. Langford, "An application of turbo trellis-coded modulation to tactical communications," in *Proc. MILCOM. IEEE Mil. Commun. Conf.*, Oct. 1999, pp. 530–533.
[30] S. Jiang, P. W. Zhang, F. C. M. Lau, C.-W. Sham, and K. Huang, "A turbo-Hadamard encoder/decoder system with hundreds of Mbps throughput," in *Proc. IEEE 10th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, Dec. 2018, pp. 1–5.
[31] A. Louliej, Y. Jabrane, V. P. Gil Jiménez, and A. García Armada, "Practical guidelines for approaching the implementation of neural networks on FPGA for PAPR reduction in vehicular networks," *Sensors*, vol. 19, no. 1, p. 116, Dec. 2018.
[32] *Flexible Advanced Coding and Modulation Scheme for High Rate Telemetry Applications: Blue Book*, Standard CCSDS 131.2-B-1, Consultative Committee for Space Data Systems, Mar. 2012.
[33] A. Lamoral Coines and V. P. G. Jiménez, "CCSDS 131.2-B-1 transmitter design on FPGA with adaptive coding and modulation schemes for satellite communications," *Electronics*, vol. 10, no. 20, p. 2476, Oct. 2021.
[34] E. Boutillon, C. Douillard, and G. Montorsi, "Iterative decoding of concatenated convolutional codes: Implementation issues," *Proc. IEEE*, vol. 95, no. 6, pp. 1201–1227, Jun. 2007.
[35] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes," in *Proc. Commun., Global Bridge (GLOBECOM)*, Dec. 1994, pp. 1298–1303.
[36] F. Dekking, C. Kraaikamp, H. Lopuhaä, and L. Meester, *A Modern Introduction to Probability and Statistics: Understanding Why and How*. London, U.K.: Springer, 2005.
[37] J. Erfanian, S. Pasupathy, and G. Gulak, "Reduced complexity symbol detectors with parallel structure for ISI channels," *IEEE Trans. Commun.*, vol. 42, no. 234, pp. 1661–1671, Apr. 1994.
[38] W. Koch and A. Baier, "Optimum and sub-optimum detection of coded data disturbed by time-varying intersymbol interference (applicable to digital mobile radio receivers)," in *Proc. IEEE Global Telecommun. Conf. Exhib.*, vol. 3, Dec. 1990, pp. 1679–1684.
[39] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE Int. Conf. Commun.*, vol. 2, Nov. 1995, pp. 1009–1013.

MIGUEL ÁNGEL PÉREZ NARANJO received the B.S. and M.S. degrees in telecommunication from the University Carlos III of Madrid, in 2019 and 2021, respectively. In addition to working in the private sector in Spain, most of his research career has been focused as a Senior Research Assistant at the University Carlos III of Madrid, from 2020 to 2022, where he has also participated in international projects. His research interests include advanced beamforming techniques and hardware implementation of hybrid algorithms applied to satellite communications.

VÍCTOR P. GIL JIMÉNEZ (Senior Member, IEEE) received the B.S. degree (Hons.) in telecommunication from the University of Alcalá, in 1998, and the M.S. and Ph.D. degrees (Hons.) in telecommunication from the Universidad Carlos III de Madrid, in 2001 and 2005, respectively. He was a Communications Staff with the Spanish Antarctica Base, in 1999. He visited the University of Leeds, U.K., in 2003, Chalmers Technical University, Sweden, in 2004, and the Instituto de Telecommunicaçoes, Portugal, from 2008 to 2010. He is currently an Associate Professor with the Department of Signal Theory and Communications, Universidad Carlos III de Madrid. He has also led several private and national Spanish projects and participated in several European and International projects. He holds one patent. He has published more than 80 journal articles/conference papers and nine book chapters. His research interests include advanced multicarrier systems for wireless radio, satellite, and visible light communications. He received the Master's Thesis and the Ph.D. Thesis Award from the Professional Association of Telecommunication Engineers of Spain, in 1998 and 2006, respectively. He held the IEEE Spanish Communications and Signal Processing Joint Chapter Chair, from 2015 to 2023.