Unified turbo/LDPC code decoder architecture for deep-space communications

A unified turbo/LDPC code decoder architecture for deep-space communications

Carlo Condo, Guido Masera, *Senior Member IEEE*
Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino, Italy

***Abstract*—Deep space communications are characterized by extremely critical conditions: current standards foresee the usage of both turbo and Low-Density-Parity-Check (LDPC) codes to ensure recovery from received errors, but each of them displays consistent drawbacks. Code concatenation is widely used in all kinds of communication to boost the error correction capabilities of single codes: serial concatenation of turbo and LDPC codes has been recently proven effective enough for deep space communications, being able to overcome the shortcomings of both code types. This work extends the performance analysis of this scheme, and proposes a novel hardware decoder architecture for concatenated turbo and LDPC codes based on the same decoding algorithm. This choice leads to a high degree of datapath and memory sharing: post-layout implementation results obtained with CMOS 90 nm technology show small area occupation (0.98 mm^2) and very low power consumption (2.1 mW).**

***Index Terms*—LDPC; turbo; concatenation; deep space**

I. Introduction

While in most kinds of communications a steady growth towards improved performance, higher throughput and low power consumption can be noticed, deep space communications are characterized by some particularities. The limited number of fully developed applications plays a major restraining role in the evolution of this field, that tends to be slower than that of on-Earth communications. Since spacecraft-to-Earth communications are supposedly very rare events **their throughput requirements are lower than those of on-Earth communications.** However, due to the limited amount of power available and the long transmission times, a failed reception and consequent retransmission are often unacceptable.

The Consultative Committee for Space Data Systems (CCSDS) has produced over the years a *de facto* standard for all space-related communication systems. Four channel coding schemes have been described in [1], and consequently assembled into application-wise Forward-Error-Correction (FEC) schemes in [2]. Both turbo [3] and Low-Density-Parity-Check (LDPC) [4] codes are currently contemplated for deep space communications [1]: while the suggested turbo codes target stricter Bit Error Rate (BER) constraints, LDPC codes have been recently included in the standard and have higher rate, and they are currently subject to CCSDS experimentation [5]. While CCSDS current requirements are not very demanding in terms of Frame Error Rate (FER), future standards are expected to require much stricter performance constraints. A FEC relying on the serial concatenation of turbo and LDPC codes has been proposed in [6]: thanks to its very good error correction capabilities, it has been deemed suitable for the extremely critical deep space communications.

To the best of our knowledge no implementation solution for the concatenated scheme has been proposed so far, but decoders for both turbo and LDPC codes are present in the state of the art, mainly targeting wireless communications. Multi-code and multi-standard decoders that make flexibility their primary concern have also been introduced recently [7]–[12]: they are characterized by different degrees of datapath and memory sharing.

This work proposes a decoder for concatenated turbo and LDPC codes targeting deep space communications. The usage of the same decoding algorithm for both codes greatly reduces the area overhead of the concatenated scheme decoder with respect to a single LDPC or turbo code decoder. In facts, it allows to exploit a high degree of datapath sharing and obtain very low power consumption and area occupation. In addition to deep-space communications, the proposed solution could be also useful in further applications where retransmission of lost packets is not allowed, such as for example broadcasting.

The rest of the paper is organized as follows: Section II introduces turbo and LDPC code decoding, while Section III describes the concatenated FEC schemes and its performance. The hardware structure of the proposed decoder is explained in Section IV, and Section V gives the results of the implementation. Finally, conclusions are drawn in Section VI.

II. Turbo and LDPC decoding

Turbo codes can be obtained by concatenating in parallel two convolutional code encoders. The dual encoding structure is reflected on the decoder, that is consequently made of two parts as well, known as Soft-In-Soft-Out (SISO) decoders. These are connected by an interleaver Π and a de-interleaver Π^{-1}. Each of them implements the BCJR algorithm [13], which produces extrinsic metrics from *a priori* information: each iteration of the algorithm can be divided into an in-order half-iteration and an interleaved-order half-iteration, due to the presence of two SISOs. The BCJR algorithm relies on the trellis representation of the constituent convolutional codes: let us define k as a trellis step and u as an uncoded symbol. Each SISO performs $\lambda_k[u] = \lambda_k^{apo}[u] - \lambda_k^{apr}[u] - \lambda_k[\mathbf{c}^u]$ where $\lambda_k^{apo}[u]$ is the a-posteriori information, $\lambda_k^{apr}[u]$ is the *a priori* information and $\lambda_k[\mathbf{c}^u]$ is the systematic component of the intrinsic information. The a-posteriori information can be obtained as follows:

$$\lambda_k^{apo}[u] = \max_{e:u(e)=u}{}^{*} \{b(e)\} - \max_{e:u(e)=\tilde{u}}{}^{*} \{b(e)\} \qquad (1)$$

where $\tilde{u} \in \mathcal{U}$ is an uncoded reference symbol (usually $\tilde{u} = \mathbf{0}$) and $u \in \mathcal{U} \setminus \{\tilde{u}\}$ with $\mathcal{U}$ the set of uncoded symbols; e is a trellis transition and $u(e)$ is the corresponding uncoded symbol. According to the Max-Log-MAP approximation [14], the $\overset{*}{\max}\{x_i\}$ function can be approximated to $\max\{x_i\}$ at the cost of a small BER degradation. The term $b(e)$ in (1) can consequently be defined as:

$$b(e) = \alpha_{k-1}[s^S(e)] + \gamma_k[e] + \beta_k[s^E(e)] \tag{2}$$

$$\alpha_k[s] = \max_{e:s^E(e)=s} \left\{\alpha_{k-1}[s^S(e)] + \gamma_k[e]\right\} \tag{3}$$

$$\beta_k[s] = \max_{e:s^S(e)=s} \left\{\beta_{k+1}[s^E(e)] + \gamma_k[e]\right\} \tag{4}$$

$$\gamma_k[e] = \lambda_k^{apr}[u(e)] + \lambda_k[\mathbf{c}(e)] \tag{5}$$

where $s^S(e)$ and $s^E(e)$ are the starting and the ending states of e, $\alpha_k[s^S(e)]$ and $\beta_k[s^E(e)]$ are the forward and backward metrics associated to $s^S(e)$ and $s^E(e)$ respectively, and $\lambda_k[\mathbf{c}(e)]$ is the channel intrinsic information. When large frames are involved, the BCJR algorithm is usually applied to a subset of the symbols to reduce latency, defining a window of w steps. The forward and backward recursions are applied on each window separately: between iterations, $\alpha_k[s]$ and $\beta_k[s]$ are exchanged between border symbols of adjacent windows. This technique is a particular version of the sliding window method, since windows are static and state metrics are exchanged only once per iteration, or can be seen as a classical sliding window with sliding step equal to w.

LDPC codes are identified by an $M \times N$ sparse parity check matrix $\mathbf{H}$, that represents all the parity checks a codeword must satisfy, i.e. $\mathbf{H} \cdot x' = 0$, where x is the codeword of length N. Various decoding approaches are possible, depending on the graph representation of $\mathbf{H}$, but the most performing one is the layered decoding approach [15]. It sees $\mathbf{H}$ as a multipartite graph composed of different layers of parity check constraints: multiple updates of the bit error probabilities within a single iteration allow for a fast convergence of the decoding algorithm. It is particularly advantageous in case of Quasi-Cyclic LDPC codes (QC-LDPC), where the parity check layers are inherent to the structure of $\mathbf{H}$. In fact, the parity check matrix is constituted of multiple instantiations of an $m \times m$ identity matrix circulated of a variable shift factor: each layer will consequently be constituted of m rows.

Let us define as $\lambda[c]$ the Logarithmic Likelihood Ratio (LLR) of symbol c. The bit LLR $\lambda_k[c]$ related to column k of $\mathbf{H}$ is initialized to the corresponding received soft value. For every parity constraint l in a given layer, the following operations are performed and reiterated up to the desired level of reliability:

$$Q_{lk}[c] = \lambda_k^{old}[c] - R_{lk}^{old} \tag{6}$$

$$\lambda_k^{new}[c] = Q_{lk}[c] + R_{lk}^{new} \tag{7}$$

where $\lambda_k^{new}[c]$ is the updated version of LLR $\lambda_k^{old}[c]$. R_{lk}^{new} is the updated version of R_{lk}^{old}, that is initialized to zero and stored for the next iteration: a different R_{lk} is identified for each $\mathbf{H}$ matrix non-zero entry at column k and row l. Several exact and approximated algorithms have been proposed to calculate R_{lk}^{new}: the most common algorithm used in LDPC decoding is the *Belief Propagation* (BP) algorithm, together with the *min-sum* approximation and its variations [16].

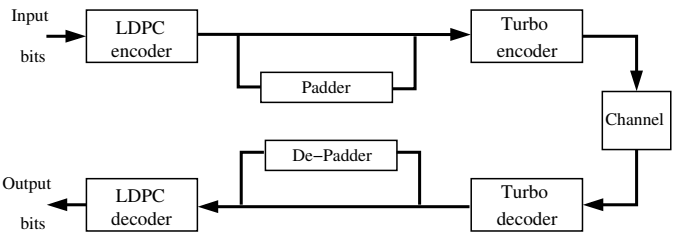

Figure 1. Serial concatenation of LDPC and turbo codes FEC scheme

It can be noticed how the LDPC and turbo decoding processes share many characteristics. Both of them are iterative, rely on soft information, are usually implemented in their logarithmic form, while commonly being represented through special kinds of graphs. A particularly interesting exploitation of these characteristics has been proposed in [17]. Every row of $\mathbf{H}$ is seen as a turbo code with trellis length equal to the row weight: a direct link between turbo and LDPC codes is consequently drawn, and turbo decoding algorithms like BCJR can be applied to LDPC codes with minor adjustments. The BCJR-based LDPC decoding relies on the fact that binary LDPC codes can be represented with a 2-state trellis: state metrics can consequently be expressed as differences $\Delta\alpha[c]$ and $\Delta\beta[c]$, reducing the quantization noise. Considering the Max-Log-MAP approximation [18], the calculation of R_{lk}^{new} becomes:

$$R_{lk}^{new} = \Phi(\Delta\alpha_k[c], \Delta\beta_k[c]) \tag{8}$$

where the operator $\Phi(\cdot)$ is defined as

$$\Phi(x, y) = \max(x, y) - \max(x + y, 0) \tag{9}$$

and $\Delta\alpha[c]$ and $\Delta\beta[c]$ can be computed as

$$\Delta\alpha_k = \Phi(\Delta\alpha_{k-1}[c], Q_{lk}[c]) \tag{10}$$

$$\Delta\beta_k = \Phi(\Delta\beta_{k+1}[c], Q_{lk}[c]) \tag{11}$$

$\Delta\alpha[c]$ and $\Delta\beta[c]$ at the edge of the trellis are initialized as the minimum value of the dynamic range.

III. Turbo and LDPC concatenated FEC scheme

The concatenation of different codes targets the improvement of performance via careful code selection. The concatenation of code A and B is in fact meaningful only when A+B performs better than both A and B. This means that the coupled codes must be in some way complementary, each one overcoming the shortcomings of the other. Outer Codes (OCs) are often chosen among those with guaranteed performance, like Reed-Solomon (RS) [19] or BCH [20] codes, thanks to their theoretically predictable error correction capabilities. They are associated to powerful Inner Codes (ICs) such as convolutional or LDPC codes, as used in WiMAX and DVB-S2, that greatly reduce the number of errors the OC has to correct. The RS+convolutional FEC scheme used by CCSDS allows these codes to rival with the more powerful LDPC and turbo codes. In [21] the performance of the common turbo+RS

FEC scheme is analyzed in relation to their interleaver. Good results are observed with complex interleavers and at very high Signal-to-Noise Ratio (SNR). However, this is not the only possible criterion of choice. In [22] LDPC and recursive systematic convolutional codes are concatenated in parallel, with good performance and little additional complexity with respect to a standard LDPC code. Block turbo codes and LDPC codes have been concatenated in [23], where a FEC scheme for 3D HDTV is devised. The scheme outperforms the DVB-T2 standard serial concatenation of BCH and LDPC codes. Satellite communications are handled through the concatenation of Luby Transform (LT) codes and non-binary LDPC (NB-LDPC) codes in [24]. Thanks to the high error correction capabilities of NB-LDPC and the intrinsic flexibility of LT codes, the resulting system is very versatile.

Serial concatenation of codes is based on the concept that the output bits of an encoder are used as input bits for another encoder. Turbo and LDPC codes in particular have been considered for concatenation in [6], where deep space communications were targeted: Fig. 1 shows the proposed idea. The performance of these two types of codes are somewhat complementary: while turbo codes guarantee much better performance than LDPC codes at low SNR, they suffer from higher error floors [25]. Consequently, the LDPC encoder is placed before the turbo encoder, while the decoders are in inverted order. The turbo code, working as an IC and being the first one to be decoded, can exploit its early waterfall region, while the outer LDPC code receives already refined error probabilities and can thus work at higher equivalent SNR. To prove the soundness of this choice, simulations were run also inverting the order of the encoders. With an LDPC IC and a turbo OC, BER results in the waterfall region improve with respect to the use of the LDPC code alone, but they are much worse than the proposed concatenated scheme. Moreover, error floors can be noticed at BER levels only slightly lower than that of turbo codes alone. The encoders are connected by an optional padding block, that adapts the respective block sizes in case they are different by adding zeros. This means that not all the IC input bits carry useful information, but experimentations with a wide variety of codes are possible. At the same time, a de-padding block is inserted between the decoders. The IC decoder receives an initial measure of the bit error probabilities from the channel estimator, and performs a fixed number of iterations Iter_{in}. Afterwards, the potential padding bits are removed, and the bit-level output error probabilities $\lambda_k[u]$ are passed to the OC decoder, that interprets them as input $\lambda_k[c]$. Having gone through the turbo decoding, the $\lambda_k[c]$ at the input of the LDPC decoder can not be considered channel-estimated LLRs. Those pertaining to correct bits in particular have diverged from zero, and are suitable for a hard decision on bits. This could lead to poor performance on the LDPC part, but the nature of the concatenated scheme prevents it. To exploit their low error floor, LDPC codes need to work at high SNR: the refined LLRs used as inputs guarantee the required general high level of reliableness and avoid undesired bit flipping. The correction of the errors that could not be corrected by the turbo code is helped by the inherent interleaving effect brought by the LDPC

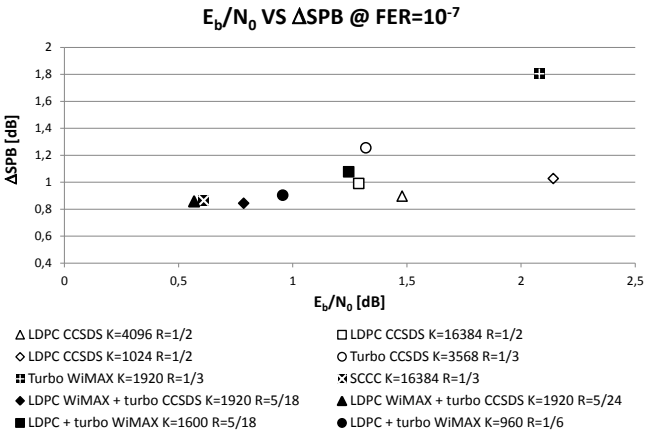

Figure 2. $\text{E}_\text{b}/\text{N}_0$ versus ΔSPB at FER=10^{-5}

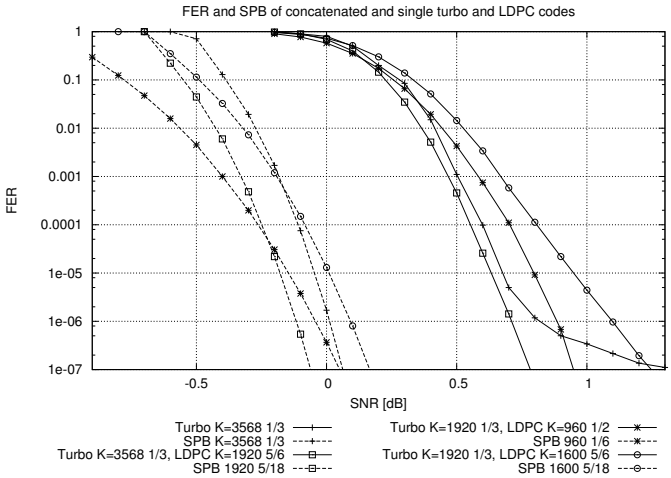

Figure 3. Concatenated LDPC and turbo FER, AWGN channel, BPSK modulation

H matrix structure [26].

Comparing the performance of codes differing in both block size and rate is a complex task. To help a fair evaluation of the effectiveness of the concatenation, the distance of each concatenated scheme from the Sphere Packing Bound (SPB) [27] has been considered. The SPB is an evolution of the channel capacity that binds the achievable performance of a code not only to its code rate, but also to the block size: different methods of calculation and refinements of this measure exist, but the authors refer to that proposed by Dolinar *et al.* in [27]. In particular, the asymptotic approximation devised for long blocks ($\geq$ 100 symbols) is used. In Fig. 2 the performance of various concatenations, along with that of codes currently used in different standards, is evaluated in terms of SPB and $\text{E}_\text{b}/\text{N}_0$. **The x axis represents the $\text{E}_\text{b}/\text{N}_0$ at which FER=**10^{-7} , while the distance of the code from its SPB at that particular FER is shown on the y axis (ΔSPB). These results, together with those shown in Fig. 3 have been obtained taking in account the performance degradation brought by the Max-Log-MAP approximation and the bit-level metric conversion addressed in Section IV-B. Together, they sum up to around 0.3 dB loss. Simulations have been run with 10 maximum iterations for both turbo and LDPC codes: R_{lk}^{new} and $\lambda_k^{apo}[u]$ values have been quantized with 10 bits, three of

which are assigned to the fractional part, while 9 bits are used for channel LLRs and state metrics. The selected quantization leads to negligible degradation with respect to floating point and guarantees a much higher level of precision than typical LDPC and turbo decoder quantizations, that can be as small as 4 bits [9]. **The white symbols are codes taken from the current CCSDS standard, while full black symbols in the Fig. 2 represent different choices of concatenations. Where padding bits have been used, the SPB has been computed by considering K=K$_{IC}\times$rate$_{OC}$. At FER=10^{-7}, it is possible to observe the effect of the error floor on turbo codes in both the high ΔSPB and the large E_b/N_0 (white circle and white cross in Fig. 2). CCSDS LDPC codes show good ΔSPB, especially for K=4096: however, they are characterized by quite large E_b/N_0. The results obtained with the largest CCSDS LDPC code are similar to those obtained by concatenated WiMAX LDPC and turbo codes. The CCSDS turbo codes, when used as IC in concatenation, give the best results, with the CCSDS turbo + WiMAX LDPC outperforming all the other solutions. The comparison with SCCCs [28], which are able to obtain lower error floors than parallel turbo code, shows very similar performance, with the concatenated scheme yielding slightly better results in terms of both ΔSPB and E_b/N_0.**

The advantage of the concatenated scheme is particularly evident at low FER, as can be observed in Fig. 3, where FER curves are plotted alongside SPB. While concatenated schemes exploit the very low error floors of LDPC codes and follow the behavior of SPB closely, the FER of turbo codes alone suffers from an early divergence from the theoretical achievable performance. For example, while the FER curve of the CCSDS turbo code plotted in Fig. 3 displays ΔSPB$= 0.72$ dB at FER$= 10^{-5}$, it rises to 1.26 dB at FER$= 10^{-7}$ due to error floor.

IV. Unified LDPC/turbo decoding architecture

Following the effectiveness of the concatenated FEC scheme presented in [6], the decoder architecture for turbo and LDPC codes concatenation shown in Fig. 4 has been designed. The gray blocks represent the duplicated datapath described in Section IV-A, while the structure of the memory banks, with their alternative usage according to half-iterations, is detailed in Section IV-B. The common turbo/LDPC decoding technique depicted in Section II paves the way for highly shared datapaths, in the wake of works like [7] and [8], as opposed to separate datapath turbo/LDPC decoders like [9]–[11]. The proposed decoder relies on an innovative smart memory structure that allows to increase the percentage of module reuse within the datapath and avoid complex interleaving mechanisms between the decoding modes.

A. Datapath

The structure of the designed LDPC/turbo datapath positions itself in between a completely shared approach and datapath separation. The turbo and LDPC datapaths have great disparities in terms of complexity, with the turbo datapath

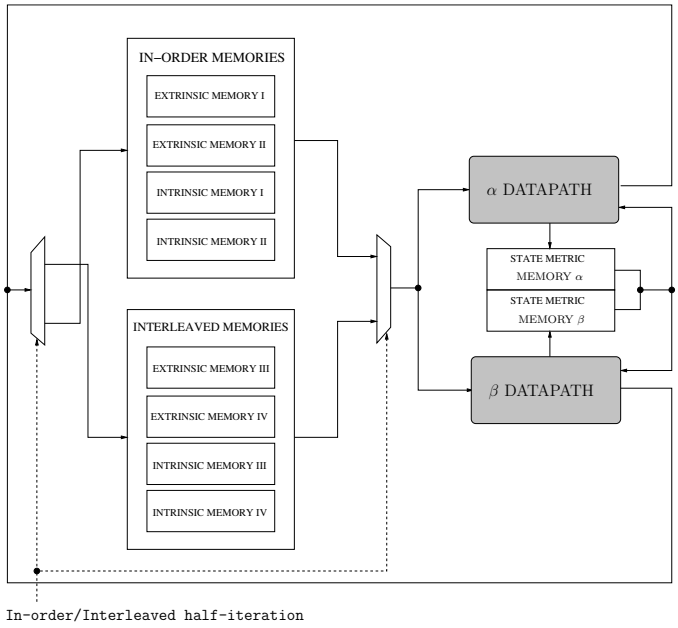

Figure 4. Unified LDPC/turbo decoder overall block diagram

requiring more resources. As shown in this section and in Section V, the LDPC datapath is included within the turbo datapath, while constituting a limited percentage of its overall logic. The concatenated scheme can consequently be decoded at little more than the logic cost of a turbo decoder.

Fig. 5 shows a block diagram of the designed datapath. It is characterized by a pipelined architecture, with registers represented by striped blocks. The turbo decoding process makes use of a *butterfly* structure: the datapath is duplicated in an α and β datapath, respectively entrusted with the concurrent forward and backward scanning of the trellis steps. They implement the modified sliding window technique described in Section II. Each half of the duplicated datapath receives as an input from the memories the $\lambda_k^{apr}[u(e)]$ and $\lambda_k[\mathbf{c}(e)]$ relative to a trellis step: these are used by the Branch Metric Units (BMUs) to perform (5) and obtain $\gamma_k[e]$. These are passed to the α and β units, that perform the computations of (3) and (4) respectively. The structure of the α and β units is similar. Along with the output of BMU, the α unit receives $\alpha_{k-1}[s^S(e)]$ either from the memory (when computing the first trellis step of a window) or from its own outputs (all other trellis steps), as shown by the feedback loop in Fig. 5. Together with the updated $\alpha_k[s]$, that are stored in the state metric memory α (Fig. 4), the α unit also produces the $\alpha_{k-1}[s^S(e)] + \gamma_k[e]$ partial sums needed by (2). These are passed to one of the extrinsic computation units (EXT-α in Fig. 5). EXT-α completes (2) by taking the $\beta_k[s^E(e)]$ stored in the state metric memory β by the β unit and finally performs (1). The same computation is concurrently carried out on another trellis step by EXT-β, to which are given $\alpha_{k-1}[s^S(e)]$ stored by the α unit in the state metric memory α and $\gamma_k[e] + \beta_k[s^E(e)]$ partial sums calculated in the β unit.

The LDPC decoding process makes mostly use of the turbo mode datapath. LDPC codes are characterized by 2-

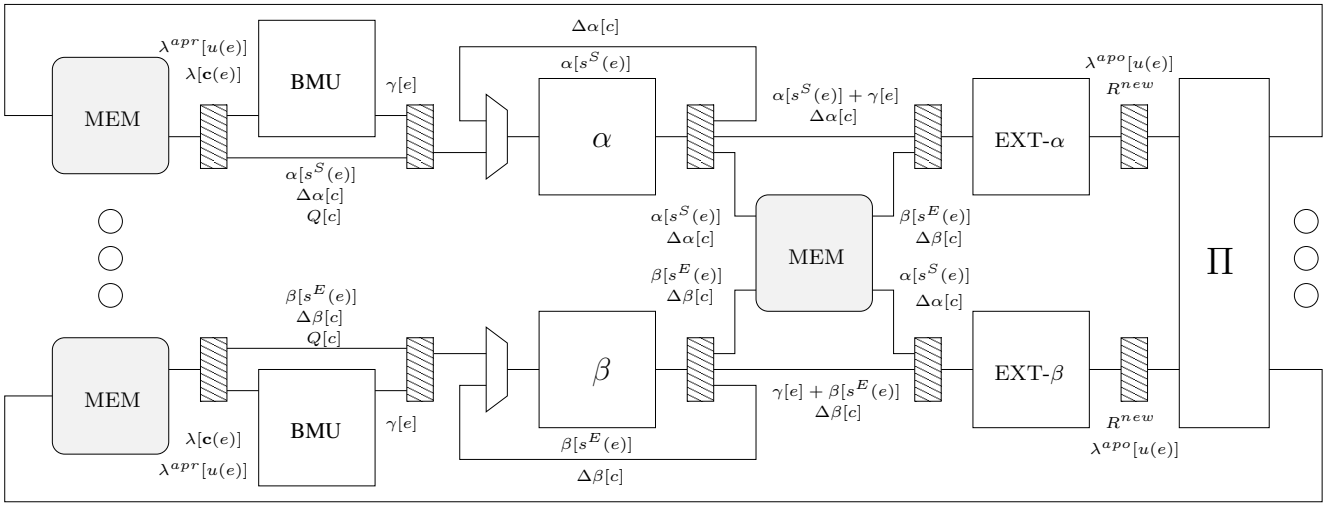

Figure 5. Unified LDPC/turbo decoder datapath block diagram

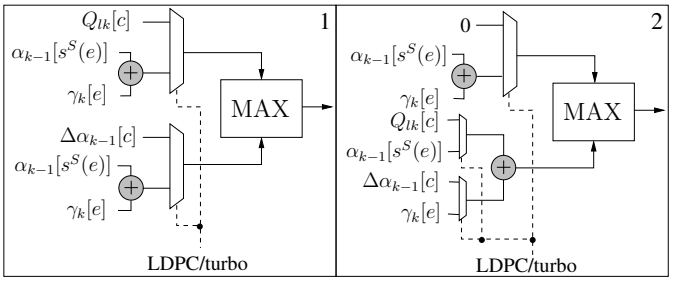

Figure 6. Unified LDPC/turbo comparator components, with inputs used in the α unit

state binary trellises: since the turbo codes considered are either 16-state Single-Binary (SBTC) or 8-state Duo-Binary (DBTC), LDPC codes can easily exploit an additional parallelism factor, **concurrently performing the computations associated to multiple parity checks**. The BMU is not used and consequently deactivated in LDPC mode, while both α and β units are shared with the turbo mode. In Fig. 6, the core components of the unified α unit are depicted. Adders and comparators are shared among the two operating modes. Both structures are equivalent when in turbo mode, while their operations differ in LDPC mode. Architecture '1' implements the $\max(x, y)$ operator of $\Phi(x, y)$, while architecture '2' implements $\max(x + y, 0)$. The EXT-α and EXT-β units perform (8), and rely on the same architectures used for α and β unit (Fig. 6). Also in this case they are shared with the turbo datapath.

B. Memory

As occupied area in both turbo and LDPC decoders is dominated by storage components, efficient memory sharing is very important: for example, a scheme suitable for serial PEs with disjoint turbo and LDPC datapaths has been used in [9], resulting in large memory saving. However, a different approach is needed with this work. Since the sizing of memories strongly depends on the supported codes, the following analysis is carried out supposing the concatenation of a rate 5/6, N=1920 WiMAX LDPC code with a rate 1/3, K=960 DBTC taken from the same standard, decoded considering a window size $w = 80$. As already shown in [6] and in Section III, this FEC scheme guarantees performance comparable to that of more powerful codes. No padding bits are necessary, since the size of the input frame for the DBTC (960 symbols, i.e. 1920 bits) is equal to the size of the LDPC codeword. However, the following discussion on memory requirements stands also in case of padding, as long as the padding bits are added at the end of the LDPC codeword. The memories necessary to support the designed decoder can be observed in Fig. 4: two sets of four memory banks serve the in-order and interleaved half-iterations respectively, storing extrinsic and intrinsic information, while two memories are dedicated to the storage of state metrics.

In turbo mode, the duplication of the datapath required by the *butterfly* structure rises the need for concurrent data reading and writing. For the correct computation of a trellis step the following metrics are necessary:

- $\lambda_k^{apr}[u(e)]$ and $\lambda_k[\mathbf{c}(e)]$ for the computation of (5). Since WiMAX codes are duo-binary, $\lambda_k^{apr}[u(e)]$ consists of three different metrics, while $\lambda_k[\mathbf{c}(e)]$ of four. However, as explained in [29], symbol-level information in duo-binary codes can be converted to bit-level information and vice versa, with a small performance degradation. This means that the memory requirements for $\lambda_k^{apr}[u(e)]$ can be reduced by approximately 1/3. Due to the *butterfly* structure, eight $\lambda_k[\mathbf{c}(e)]$ metrics and four $\lambda_k^{apr}[u(e)]$ are needed. While $\lambda_k[\mathbf{c}(e)]$ values are received by the decoder at the beginning of a frame and not updated anymore, the $\lambda_k^{apr}[u(e)]$ metric is updated at least once per iteration.
- $\alpha[s^E(e)]$ and $\beta[s^E(e)]$ for (2), (3) and (4). Every trellis

step computation requires a number of $\alpha[s^E(e)]$ and $\beta[s^E(e)]$ metrics equal to the number of states of the turbo code: in this case, eight of each. The loading and storing needs for these metrics vary during the decoding process. At the beginning of each trellis window, the $\alpha[s^E(e)]$ and $\beta[s^E(e)]$ values coming from adjacent windows must be read as initialization values. The updated $\alpha[s^E(e)]$ and $\beta[s^E(e)]$ must be stored during the first half of the window, and loaded again in the second half. Finally, the metrics belonging to trellis steps at the edge of a window must be stored for the adjacent windows.

In LDPC mode, for every trellis step computation a $\lambda_k[c]$ and R_{lk}^{old} pair must be loaded in both parts of the datapath to perform (6), along with $\Delta\alpha_k[c]$ and $\Delta\beta_k[c]$ for (8), (10) and (11). Similarly to the turbo case, only the $\Delta\alpha_k[c]$ and $\Delta\beta_k[c]$ metrics at the edge of the trellis need to be stored for further usage, while the $\lambda_k[c]$ and R_{lk}^{new} metrics involved in (7) are to be updated once per trellis.

Since the chosen turbo code is duo-binary and has an eight-state trellis, its decoding process needs a much larger number of metrics than the LDPC code, which decoding is similar to that of a single-binary, two-state turbo code. From the LDPC point of view, this translates in an internal level of parallelism in the datapath that is not, however, directly available. In fact, the structure of the **H** matrix and the layered scheduling require the same LLR to be read and updated multiple times during a single LDPC decoding iteration, resulting in complex load and store patterns not found in turbo decoding. Careful planning of the memory structure is consequently necessary to maximize the level of memory sharing and to concurrently allow the LDPC datapath to exploit the internal parallelism. Figure 7 shows an in-depth detail of one of the two sets of memory banks depicted in Fig. 4. Memories are sized to accommodate the considered codes in case two concurrent parity check computations are performed in LDPC decoding (parallelism factor $\times 2$). They are dual-port, and the usage percentage of each memory is portrayed for both turbo and LDPC codes, along with its depth and width.

In turbo mode, two $\lambda_k[\mathbf{c}(e)]$ metrics are stored at each address in the two 1920×16 bit intrinsic memories: both ports are always kept in read mode, except during initialization. In this way, four $\lambda_k[\mathbf{c}(e)]$ are concurrently available to the α datapath, and four to the β datapath. These same intrinsic memories are used to store the R_{lk}^{old} values in LDPC mode. At every clock cycle both the datapaths need a R_{lk}^{old} value: during the second half of the trellis, the α datapath will need the values fed to the β datapath during the first half in backward order, and vice versa. This means that by storing at the same memory address R_{lk}^{old} values in symmetrical positions with respect to the trellis half-point (e.g. R_{l1}^{old} with R_{l20}^{old}, R_{l2}^{old} with R_{l19}^{old} etc.) the storage requirements are reduced by 1/2 without decreasing the number of concurrently available metrics. The total number of memory locations required becomes 3200, resulting in a 83.3% usage of each intrinsic memory. Two 960×8 bit extrinsic memories hold the $\lambda_k^{apr}[u(e)]$ values. From the turbo decoding point of view, these two memories could be merged into a single 960×16 bit memory, since $\lambda_k^{apr}[u(e)]$ metrics must be paired to obtain the symbol-level

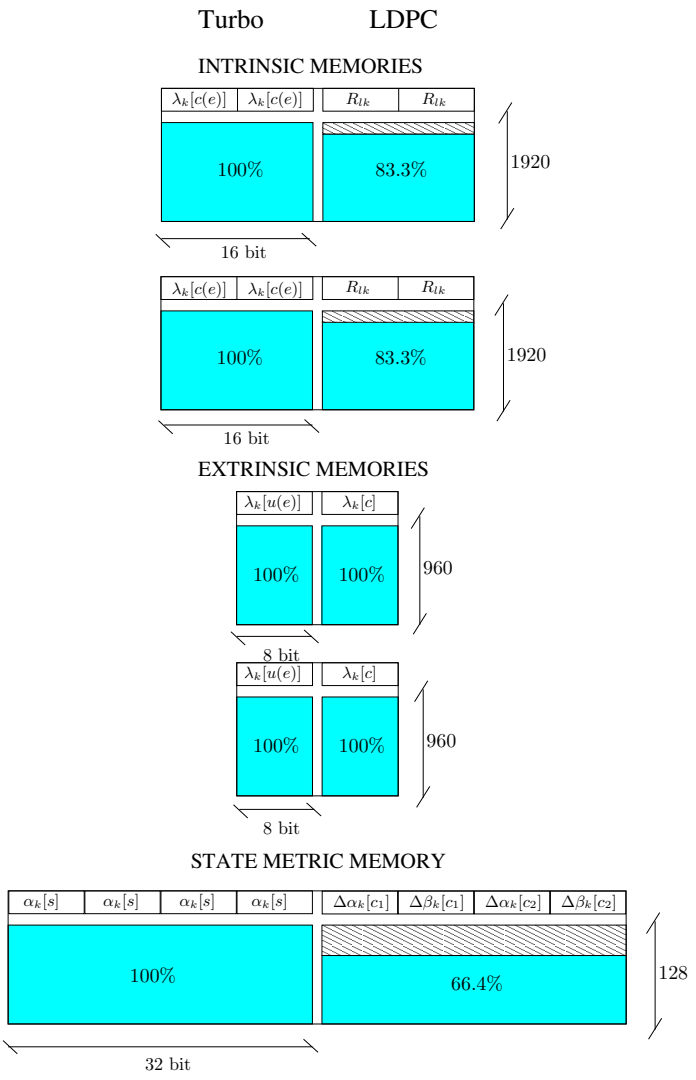

Figure 7. Unified turbo/LDPC decoder memory sharing scheme

metrics used in the BCJR algorithm. LDPC decoding must hold at least N $\lambda_k[c]$: four of them must be read concurrently. A single 960×16 bit memory would not suffice, since the coupling of values changes with every row of the **H** matrix. Both extrinsic memories are used to their full capacity in both decoding modes.

The memories portrayed so far compose the in-order half-iteration memory banks (Fig. 4) and need to be kept, during turbo decoding, with both ports in read mode. The in-order half-iteration makes use of two additional extrinsic memories to store the newly computed $\lambda_k^{apo}[u(e)]$, that are used in the interleaved-order half-iteration (Fig. 4). Also the $\lambda_k[\mathbf{c}(e)]$ needed by the interleaved half-iteration are stored in two additional intrinsic memories. **The same data could be retrieved by adding complexity to the address generation logic and reading in interleaved order the intrinsic memories used in the first half-iteration. However, the extra storage is useful for LDPC decoding. In fact, by having a total of four 1920×16 bit and four 960×8 bit memories, LDPC decoding can exploit a $\times 2$ internal parallelism factor.**

Table I
MEMORY REQUIREMENTS

	Memory Bits			
	Turbo	LDPC	Total	
Separate memories No smart allocation	176384	210240	386624	100%
Separate memories Smart allocation	161024	138560	299584	77.5%
Shared memories Smart allocation	161024	138560	161792	41.8%

Finally, two wider 128×32 state metric memories, as the one shown in Figure 7, are used to hold the $\alpha[s^E(e)]$ and $\beta[s^E(e)]$ values for both window initialization and intra-window state metrics in turbo mode. The same memories are used in LDPC mode to store $\Delta\alpha_k[c]$ and $\Delta\beta_k[c]$. Each address holds the values used by each datapath in both levels of parallelism.

Table I synthesizes the advantages of the devised memory structure. The first row gives the memory bits necessary for the decoder architecture described in Section IV-A to work in both turbo and LDPC mode, while considering separate memories. If smart metric allocation techniques are used (R_{lk}^{old} coupling and reusage, bit-level $\lambda_k^{apo}[u(e)]$), the required bits are reduced of 22.5%. Moreover, by sharing the memories between the two modes, only 41.8% of total bits is necessary, with LDPC mode being completely supported by the memories required by turbo mode.

C. Interleaving and addressing

Address generation for the described memory structure is in most cases straightforward. In turbo mode, all read operations are sequential, either in forward or backward order, and are consequently handled by simple counters. Write operations to the following half-iteration memories are based on the permutation law associated to the turbo code encoding, and the memory addresses can be obtained via simple operation on the current half-iteration read address. In the considered case study, the interleaving rules are those associated to the WiMAX standard turbo codes: the interleaved addresses are obtained on-the-fly by dedicated logic implementing the WiMAX permutation function. Address generation for the intrinsic memories is sequential in both read and write operations when in LDPC mode, and the counters used in the turbo mode can be reused, but problems arise when dealing with the extrinsic memories. While sequential addressing in intrinsic memories is possible thanks to the local nature of R_{lk}^{old} values, $\lambda_k[c]$ are read and updated multiple times and in variable order during an iteration. Address generation, however, can still exploit the regular structure of the **H** of QC-LDPC codes. By storing in a small memory the shift factors of the constituent $m \times m$ circulant identity matrices, together with the position of the nonzero entry of their first row, read and write addresses can be obtained with modulo-m counters and adders. A single 160×36 bit memory is sufficient to support also the $\times 2$ internal parallelism.

The devised memory structure is particularly advantageous when switching between turbo and LDPC decoding: after the last turbo iteration, the extrinsic memories relative to the in-order half-iteration contain the data needed by the LDPC decoding process in the correct order. The memories relative to the interleaved-order half-iteration in turbo mode, to be used in LDPC mode, will only need to have the read and write addresses pass through the permutation law circuit.

V. IMPLEMENTATION

The decoder architecture described in Section IV has been implemented in 90 nm CMOS technology: synthesis and power estimation have been carried out with Synopsys Design Compiler, while the switch activity has been analyzed with Mentor Graphics Modelsim.

Several design choices are related to the set of codes that is going to be implemented, in particular the sizing of the memories. The largest codes considered for the implementation are taken from the WiMAX standard: an LDPC code with block size 1920 and rate 5/6, and a DBTC with information block size 1920 and rate 1/3. Soft metrics have been quantized with nine and eight bits, with two bits of fractional part; the maximum number of iterations has been set as $\mathrm{It}_{OC} = 10$ for LDPC and $\mathrm{It}_{IC} = 6$ for turbo. **The CCSDS standard foresees in [30] a wide range of possible throughputs for spacecraft-to-Earth communications, depending on the modulation scheme, frequency band and type of mission. Downlink data rates supported by spacecrafts employed in current missions vary consistently between one another. For example, the Curiosity rover deployed on the surface of Mars can communicate directly with Earth at 32 Kb/s: however, it can exploit two different orbiters (the Mars Odyssey Orbiter and the Mars Reconnaissance Orbiter) to reach data rates of up to 110 Kb/s and 6 Mb/s respectively. The Cassini orbiter around Saturn has a maximum downlink data rate of 248.85 Kb/s, and the Kepler planet search spacecraft communicates at 4.33 Mb/s. Higher rates (up to 28 Mb/s) will be considered by the James Webb Space Telescope.**

The throughput of the Cassini orbiter can be achieved by the proposed decoder architecture at 12 MHz (252 Kb/s): with this target frequency, the total area occupation is 0.98 mm^2. Thanks to the shared datapath approach more than 90% of the LDPC datapath is included in the larger turbo datapath, with very few LDPC-exclusive components. This is also reflected on the power consumption estimate, resulting in 2.1 mW at 12 MHz. Memories occupy 82.6% of the decoder area, and account for 70.1% of the total power consumption. Pipeline stages contribute for 10.3% of the area and 13.4% of power consumption, with the remaining 7.1% area occupation and 16.5% power consumption being taken by processing, addressing and control logic. The implementation results show that this decoder has a smaller area and lower power consumption than most LDPC and turbo decoders [9], [31]–[33]. Obviously, due to the very reduced throughput target, the obtained throughput-to-area ratio is low. It

yields, however, an energy efficiency of 120 Mb/s per Watt, outperforming the majority of the state of the art.

The 6 Mb/s required by the Mars Reconnaissance Orbiter are obtained by targeting a frequency of 286 MHz, for which the occupied area results 1.03 mm^2, and the power consumption 56.6 mW. Whereas the energy efficiency is reduced to 106 Mb/s per Watt, this implementation of the decoder allows to comply with most current deep space downlink throughput requirements. The proposed architecture, however, can sustain even higher throughputs: 10.5 Mb/s have been obtained by synthesizing the presented decoder without any modifications targeting a frequency of 500 MHz. The implementation yields an area occupation of 1.06 mm^2 and 111.9 mW power consumption. To achieve even higher throughputs, it is possible to reduce the system critical path by adding a pipeline stage in the EXT-α, EXT-β modules and another in the α, β modules: with these straightforward modifications, up to 14.5 Mb/s can be obtained. Another possible approach can be incrementing the degree of parallelism of the decoder: by subdividing the current memory structure in a number of smaller banks, multiple instances of the datapath can work concurrently, virtually multiplying the achievable throughput.

The state of the art is currently lacking extensive information about decoders aimed at deep space communications, making the comparison between the concatenated FEC scheme implementation and alternative solutions unfeasible. The work in [34] presents an FPGA-based LDPC decoder for space communications: however, the considered near-Earth transmissions involve codes and specifications very different from deep-space links. Turbo codes are a more mature technology in the deep space field, and various CCSDS-compliant turbo decoders are available on the market [35], [36]. However, very few scientific papers have been written on the subject. The work in [37] discusses the implementation of a CCSDS-compliant turbo decoder, but it is based on multiple off-the-shelf Digital Signal Processors, lacking area occupation and power consumption details. Also evaluating the area, power and energy efficiency gain of the proposed solution with respect to similar architectures is problematic. Shared datapath LDPC and turbo decoders are present in the literature, for which complete implementation results are provided [7], [8]. Their target applications are wireless communication standards like 3GPP-LTE, WiMAX, WiFi and DVB, for which BER and throughput requirements are extremely different from deep-space communications. These decoder designs are often based on high levels of parallelism, favoring speed over performance, especially in video broadcasting. For example, the decoder presented in [7] relies on a completely shared datapath. Since the target throughput ranges between 450 and 600 Mb/s, the internal parallelism of each decoding core can be close to a hundred, while the frequency is set to 500 MHz. Moreover, to give full support to high-throughput communication standards, multiple instances of parallel cores are used. Consequently, while the concept of datapath sharing and turbo/LDPC code decoding behind the presented work and [7] is similar, the difference in throughput requirements results in diverging design choices, that lead to a more than three-fold area occupation and an estimated $\times 20$ factor in power consumption. While it is clear that a fair comparison with the state of the art cannot be performed, it is possible to get a sense of where the proposed decoder stands. The CCSDS-compliant RS decoder [38] and Viterbi decoder [39] yield a total area normalized to 90 nm CMOS technology of 0.63 mm^2. The RS+CC FEC schemes is consequently cheaper to implement than the proposed turbo/LDPC concatenation, but its performance are much worse. An additional evaluation can be made thanks to the resource utilization data given in Lattice Semiconductor FPGA-based CCSDS turbo decoder [35]. Approximately 8000 Look-Up Tables (LUTs) and 4000 registers are necessary for different Lattice devices. This work, implemented on a Xilinx Virtex 6 FPGA, requires 6000 LUTs and 1000 registers, having better performance while at the same time occupying a smaller area than [35].

VI. Conclusion

This work presents a unified turbo/LDPC decoder architecture for concatenated LDPC and turbo codes aimed at deep space communications. The performance of such FEC scheme is compared to that of FEC schemes currently used by the CCSDS standard, extending the evaluation of previous works: this solution greatly outperforms both CCSDS LDPC and turbo codes. The architecture of the joint turbo/LDPC decoder is described: it yields a high percentage of datapath sharing ($> 90\%$ in the LDPC case) and completely shared memories. The novelty of the solution and the lack of similar implementations in the state of the art make a fair comparison impossible. The proposed decoder has been implemented obtaining post-layout results, that show very small area occupation (1.01 mm^2) and power consumption (18.4 mW).

References

[1] *TM Synchronization and Channel Coding - Summary of Concept and Rationale*, Consulative Committee for Space Data Systems (CCSDS) Std. 130.1-G-1, Jun. 2006.

[2] *TM Channel Coding Profiles*, Consulative Committee for Space Data Systems (CCSDS) Std. 131.4-M-1, Jul. 2011.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding and decoding: Turbo codes," in *IEEE International Conference on Comm.*, 1993, pp. 1064–1070.

[4] R. G. Gallager, "Low density parity check codes," *IRE Transactions on Information Theory*, vol. IT-8, no. 1, pp. 21–28, Jan 1962.

[5] G. Liva, E. Paolini, T. De Cola, and M. Chiani, "Codes on high-order fields for the CCSDS next generation uplink," in *Advanced Satellite Multimedia Systems Conference and 12th Signal Processing for Space Communications Workshop*, 2012, pp. 44–48.

[6] C. Condo, "Concatenated turbo/LDPC codes for deep space communications: performance and implementation," in *International Conference on Advances in Satellite and Space Communications (SPACOMM)*, apr. 2013, pp. 1 –6.

[7] Y. Sun and J. R. Cavallaro, "A flexible LDPC/Turbo decoder architecture," *Jour. of Signal Processing Systems*, vol. 64, no. 1, pp. 1–16, 2010.

[8] F. Naessens, B. Bougard, S. Bressinck, L. Hollevoet, P. Raghavan, L. Van der Perre, and F. Catthoor, "A unified instruction set programmable architecture for multi-standard advanced forward error correction," in *Signal Processing Systems, 2008. SiPS 2008. IEEE Workshop on*, oct. 2008, pp. 31 –36.

[9] C. Condo, M. Martina, and G. Masera, "VLSI implementation of a multi-mode turbo/LDPC decoder architecture," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 6, pp. 1441–1454, 2013.

[10] M. Alles, T. Vogt, and N. Wehn, "FlexiChaP: A reconfigurable ASIP for convolutional, turbo, and LDPC code decoding," in *Turbo Codes and Related Topics, 2008 5th International Symposium on*, 2008, pp. 84 –89.

[11] P. Murugappa, R. Al-Khayat, A. Baghdadi, and M. Jezequel, "A flexible high throughput multi-ASIP architecture for LDPC and turbo decoding," in *Design, Automation and Test in Europe Conference and Exhibition*, 2011, pp. 1–6.

[12] G. Gentile, M. Rovini, and L. Fanucci, "A multi-standard flexible turbo/LDPC decoder via ASIC design," in *International Symposium on Turbo Codes & Iterative Information Processing*, 2010, pp. 294–298.

[13] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, no. 3, pp. 284–287, Mar 1974.

[14] S. Papaharalabos, P. T. Mathiopoulos, G. Masera, and M. Martina, "On optimal and near-optimal turbo decoding using generalized $\max^*$ operator," *IEEE Comm. Letters*, vol. 13, no. 7, pp. 522–524, Jul 2009.

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

[16] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. on Comm.*, vol. 47, no. 5, pp. 673 –680, May 1999.

[17] M. Mansour and N. Shanbhag, "Turbo decoder architectures for low-density parity-check codes," in *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, vol. 2, nov. 2002, pp. 1383 – 1388 vol.2.

[18] M. Martina, G. Masera, S. Papaharalabos, P. T. Mathiopoulos, and F. Gioulekas, "On practical implementation and generalizations of max* operator for turbo and LDPC decoders," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 4, pp. 888–895, Apr 2012.

[19] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, Jun. 1960.

[20] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, Mar. 1960.

[21] M. Ferrari, F. Osnato, M. Siti, S. Valle, and S. Bellini, "Performance of concatenated reed-solomon and turbo codes with non ideal interleaving," in *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, vol. 2, 2001, pp. 911–915 vol.2.

[22] S. Gounai, T. Ohtsuki, and T. Kaneko, "Performance of concatenated code with LDPC code and RSC code," in *Communications, 2006. ICC '06. IEEE International Conference on*, vol. 3, june 2006, pp. 1195 – 1199.

[23] K. Kwon, H. H. Im, and J. Heo, "An improved FEC system for next generation terrestrial 3D HDTV broadcasting," in *Consumer Electronics (ICCE), 2012 IEEE International Conference on*, jan. 2012, pp. 327 – 328.

[24] W. Lei, L. Jing, and W. J. bo, "Design of concatenation of fountain and non-binary LDPC codes for satellite communications," in *Information Engineering and Computer Science (ICIECS), 2010 2nd International Conference on*, dec. 2010, pp. 1 –4.

[25] *Low density parity check codes for use in near-earth and deep space applications*, Consulative Committee for Space Data Systems (CCSDS) Std. 131.1-O-2, Sep. 2007.

[26] S. H. Lee, J. A. Seok, and E. K. Joo, "Serial concatenation of LDPC and turbo code for the next generation mobile communications," in *Wireless and Optical Communications Networks, 2005. WOCN 2005. Second IFIP International Conference on*, march 2005, pp. 425 – 427.

[27] S. Dolinar, D. Divsalar, and F. Pollara. (1998, May) Code performance as a function of block size. Jet Propulsion Laboratory (JPL), TMO Progress Report 42-133. [Online]. Available: http://tmo.jpl.nasa.gov/progress_report/42-133/133K.pdf

[28] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding," *Information Theory, IEEE Transactions on*, vol. 44, no. 3, pp. 909–926, 1998.

[29] J.-H. Kim and I.-C. Park, "A 50Mbps double-binary turbo decoder for WiMAX based on bit-level extrinsic information exchange," in *Solid-State Circuits Conference, IEEE Asian*, nov. 2008, pp. 305 –308.

[30] *Radio Frequency and Modulation Systems Part 1: Earth Stations and Spacecraft*, Consulative Committee for Space Data Systems (CCSDS) Std. 401.0-B-23, Dec. 2013.

[31] T. Brack, M. Alles, T. Lehnigk-Emden, F. Kienle, N. Wehn, N. L'Insalata, F. Rossi, M. Rovini, and L. Fanucci, "Low complexity LDPC code decoders for next generation standards," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, 2007, pp. 1 –6.

[32] M. May, T. Ilnseher, N. Wehn, and W. Raab, "A 150mbit/s 3gpp lte turbo code decoder," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, 2010, pp. 1420–1425.

[33] X.-Y. Shih, C.-Z. Zhan, C.-H. Lin, and A.-Y. Wu, "An 8.29 mm^2 52 mw multi-mode LDPC decoder design for mobile WiMAX system in 0.13 μm CMOS process," *IEEE Jour. of Solid-State Circuits*, vol. 43, no. 3, pp. 672 –683, 2008.

[34] F. Demangel, N. Fau, N. Drabik, F. Charot, and C. Wolinski, "A generic architecture of CCSDS low density parity check decoder for near-earth applications," in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, 2009, pp. 1242–1245.

[35] *Lattice Turbo Decoder*, Lattice Semiconductor Corporation, 2008. [Online]. Available: http://www.latticesemi.com/lit/docs/ip/ipug14.pdf

[36] *TC6000 CCSDS turbo decoder*, Turbo Concept. [Online]. Available: http://www.turboconcept.com/prod_tc6000.php

[37] J. Berner and K. Andrews, "Deep space network turbo decoder implementation," in *Proc. of IEEE Aerospace Conference*, vol. 3, 2001, pp. 3/1149–3/1157 vol.3.

[38] Y.-K. Lu and M.-D. Shieh, "Efficient architecture for Reed-Solomon decoder," in *International Symposium on VLSI Design, Automation, and Test (VLSI-DAT)*, april 2012, pp. 1 –4.

[39] M. Kawokgy, C. Andre, and T. Salama, "Low-power asynchronous viterbi decoder for wireless applications," in *International Symposium on Low Power Electronics and Design*, 2004, pp. 286–289.