State metric compression techniques for turbo decoder architectures

State Metric Compression Techniques for Turbo Decoder Architectures

Maurizio Martina, *Member IEEE*, Guido Masera, *Senior Member IEEE*

***Abstract*—This work proposes to compress state metrics in turbo decoder architectures to reduce the decoder area. Two techniques are proposed: the first is based on non-uniform quantization and the second on the Walsh-Hadamard transform followed by non-uniform quantization. The non-uniform quantization technique reduces state metric memory area of about 50% compared with architectures where state metric compression is not performed, at the expense of slightly increasing the error correcting performance floor. On the other hand, the Walsh-Hadamard transform based solution offers a good trade-off between performance loss and memory complexity reduction, which reaches in the best case 20% of gain with respect to other approaches. Both solutions show lower power consumption than architectures previously proposed to compress state metrics.**

***Index Terms*—Turbo Decoder, data compression, VLSI**

I. Introduction

Turbo codes [1] are known as a class of channel codes with excellent error correction capabilities. Due to this relevant characteristic they are employed in several standards for wireless communications as UMTS, CDMA2000, WiMax and LTE (see Table I in [2]). However, these standards impose to support throughputs that range from tens to hundreds of Mb/s. Moreover, being the decoding algorithm iterative, the design of high performance turbo decoders is a challenging task that involves the search for efficient solutions to handle data dependencies and potential parallelism. Since turbo codes are based on the concatenation of two convolutional codes (CC), an iteration at the decoder side is made of two half iterations, each of which is devoted to perform the BCJR algorithm [3] on one of the constituent CCs.

Thus, a widely adopted solution to achieve high throughput relies on parallel architectures [4], [5], where the computation is split on P processing elements, usually referred to as SISO or MAP processors [6]. Similarly, also the memories used to store input and output data are divided into P separated components: Fig. 1 gives a general view of such a parallel turbo decoder architecture (details of the architecture and adopted notation will be introduced in Section II). As highlighted in [7] to achieve the throughput required by modern standards for wireless communications, as WiMax or LTE, at least $P = 8$ is required with a 140 MHz clock frequency. Even larger P values are necessary to support the higher throughputs of future standards [8], [9], [10]. However, parallel architectures lead to a significant area increase and in particular the percentage of area occupied by SISO memories becomes more relevant as long as P increases, as will be show in section II (Table I). To that purpose, some recent works [11], [12], [13], [14], [15], [16] propose different techniques to reduce the amount of memory required in turbo decoders.

The authors are with Dipartimento di Elettronica - Politecnico di Torino - Italy.

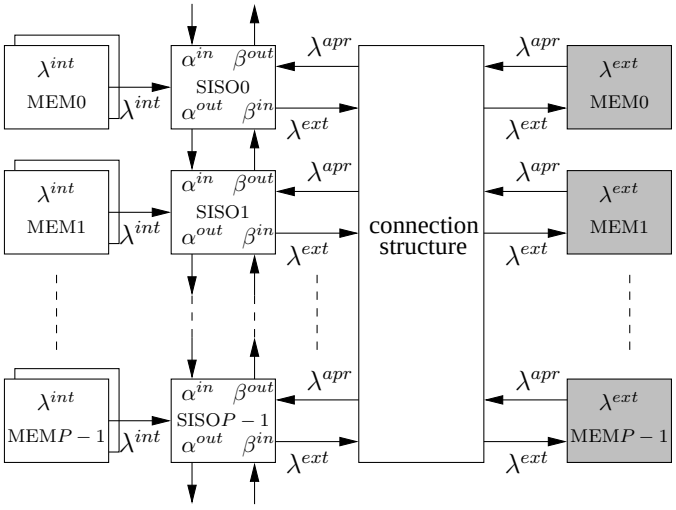

Figure 1. General parallel architecture for turbo decoding

As detailed in section II, the memory required by a turbo decoder can be coarsely classified in parallelism dependent and parallelism independent memory. While the works detailed in [11], [14], [15] and [16] deal with parallelism independent memories, this work, as [12] and [13], concerns parallelism dependent memories. In particular, in section III we analyze the statistical characteristics of state metrics and in section IV we present two techniques to reduce the amount of memory to store state metrics: i) the first one is derived from the non-uniform quantization of border state metrics described in [13]; ii) the second technique employs a compression method based on the Walsh-Hadamard transform [17] followed by non-uniform quantization. Section V shows that the non-uniform quantization technique reduces the state metric memory area of about 50%, compared with architectures where state metric compression is not performed, at the expense of slightly increasing the error correcting performance floor. On the other hand, the Walsh-Hadamard transform based solution, features negligible error correcting performance degradation and in the best case offers a complexity reduction of more than 20%. Finally, both solutions show lower power consumption than architectures previously proposed to compress state metrics.

II. Reference architecture

In order to detail the proposed technique we introduce the notation shown in Fig. 2 (a), where in a generic trellis section

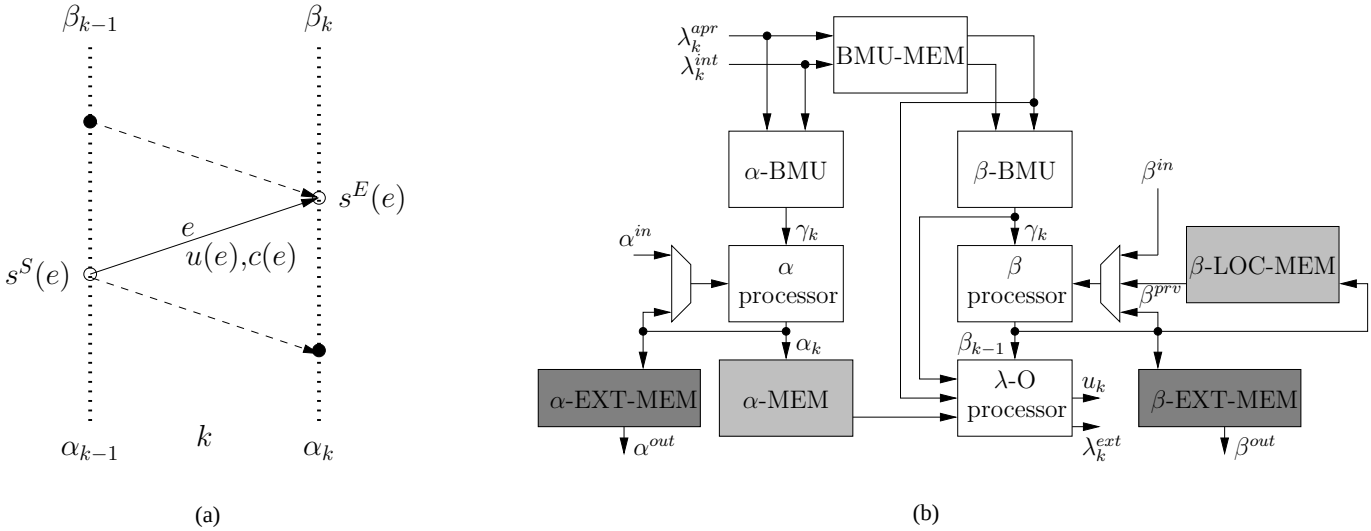

Figure 2. Notation for the trellis section in the SISO (a), SISO reference architecture (b)

k for each transition e on the trellis we define $u(e)$ ($c(e)$) as the uncoded (coded) symbol associated to e and $s^S(e)$ ($s^E(e)$) as the starting (ending) state of e. Besides, $\alpha_k[s^S(e)]$ and $\beta_k[s^E(e)]$ are the forward and backward state metrics (SMs) associated to $s^S(e)$ and $s^E(e)$ respectively. Moreover, stemming from the sliding-window decoding algorithm [18] and initializing each window with the metric-inheritance technique proposed in [19] and [20], each SISO processor can be implemented as in Fig. 2 (b) where the meaning of each block will be detailed in the following paragraphs.

A. SISO equations

During each half iteration the decoder reads N intrinsic information values λ_i^{int} received from the channel and $N \cdot R$ a-priori information values λ_i^{apr} in the form of logarithmic-likelihood-ratios (LLRs), where R is the rate of the constituent CC. The a-priori information is a proper permutation of the extrinsic information produced by the decoder during the previous half iteration

$$\lambda_k^{ext} = \lambda_k^{apo} - \lambda_k^{apr} \tag{1}$$

where

$$\lambda_k^{apo} = \max_{e:u(e)=u}^{*} \{b(e)\} - \max_{e:u(e)=\tilde{u}}^{*} \{b(e)\} \tag{2}$$

and $\tilde{u}$ is an input symbol taken as a reference (usually $\tilde{u} = \mathbf{0}$). The $b(e)$ terms in (2) are defined as

$$b(e) = \alpha_{k-1}[s^S(e)] + \gamma_k[e] + \beta_k[s^E(e)] \tag{3}$$

with

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

$$\beta_k[s] = \max_{e:s^S(e)=s}^{*} \left\{\beta_{k+1}[s^E(e)] + \gamma_k[e]\right\} \tag{5}$$

$$\gamma_k[e] = \pi_k[u(e)] + \pi_k[c(e)] \tag{6}$$

The $\max^{*}\{x_i\}$ function [21] can be implemented with several techniques [22]. The most common solution is based on a maximum selection followed by a correction term stored in a small Look-Up-Table (LUT) [23]. The correction term, usually adopted when decoding binary codes, can be omitted for double-binary turbo codes with minor performance loss [24].

The $\pi_k[c(e)]$ term in (6) is computed as a weighted sum of the λ_k^{int} produced by the soft demodulator.

$$\pi_k[c(e)] = \sum_{i}^{n_c} c_i(e)\lambda_k^{int}[c_i(e)] \tag{7}$$

where $c_i(e)$ is the i-th bit of the coded symbol associated to e and n_c is the number of bits forming the coded symbol. Thus, in (7) we assume that, even if symbols are not binary, bit-level LLRs are available to the decoder. Under this hypothesis, (7) can be used for double binary codes as well as for binary ones; otherwise, if symbol level LLRs are available at the decoder (7) should be slightly modified [4].

On the other hand, we can write $\pi_k[u(e)] = u(e)\lambda_k^{apr}[u(e)]$ for a binary turbo code, whereas for a double-binary turbo code the $\pi_k[u(e)]$ terms are piece wise functions:

$$\pi_k[u(e)] = \begin{cases} 0 & \text{if } u(e) = (\text{'0'}, \text{'0'}) \\ \lambda_k^{apr}[01] & \text{if } u(e) = (\text{'0'}, \text{'1'}) \\ \lambda_k^{apr}[10] & \text{if } u(e) = (\text{'1'}, \text{'0'}) \\ \lambda_k^{apr}[11] & \text{if } u(e) = (\text{'1'}, \text{'1'}) \end{cases} \tag{8}$$

For further details on the decoding algorithm, the reader can refer to [6].

B. SISO architecture

According to Fig. 2 (b) each branch-metric-unit (BMU) computes the branch metrics (γ) at the k-th trellis step as in (6). The outputs of α-BMU and β-BMU are used by the α and β processors to compute the forward and backward state metrics respectively, see (4) and (5). Finally, the λ-O processor computes the extrinsic information λ_k^{ext} by implementing (1) and (2); furthermore, it generates the decoded symbols u_k.

The architecture shown in Fig 2 (b) assumes that the forward recursion is computed first with the sliding window approach.

The set of windows is processed sequentially in natural order. As a consequence, λ^{int} and λ^{apr} values belonging to a window are processed by the α-BMU and concurrently stored in a buffer (BMU-MEM) that acts as a Last-In-First-Out (LIFO) buffer. When the buffer contains a window of data the β-BMU starts the computation of γ. Thus, when the β processor works on the i-th window, the α processor works on the $i+1$-th one. In order to align the forward and backward metrics a one-window-size LIFO buffer (α-MEM) is employed. When border metric inheritance is used a buffer to store border metrics (β^{prv}) is required (β-LOC-MEM). Moreover, in a parallel decoder the SISOs process concurrently different slices of the trellis, so that proper trellis initialization is required. As highlighted in Fig. 1 and 2 we consider that the i-th SISO exchanges the boundary metrics of its trellis slice with the $i-1$-th and the $i+1$-th SISO respectively. These metrics are stored in the α-EXT-MEM and β-EXT-MEM buffers.

As highlighted in Fig. 1 buffers are required also to store intrinsic and a-priori/extrinsic information. Intrinsic information memory is duplicated to accommodate in-order and interleaved LLRs. It is worth pointing out that in Fig. 1 and 2 (b) we depicted as white those buffers whose size is assumed to be minimized with well known methods [23]. On the other hand, dark-grey-shaded buffers are the ones required for boundary metric exchange among neighboring SISOs (α^{in}, α^{out}, β^{in} and β^{out}). These buffers can be implemented as two position shift registers where each position stores the boundary metrics computed during one half iteration. As a consequence, the minimization of these memories leads to a minor improvement in the decoder area reduction. Light-grey-shaded buffers in Fig. 1 and 2 (b) are the ones studied in [11], [12], [13], [14], [15], [16]. In particular in [12] the α-MEM footprint is reduced by applying saturation to the metrics, in [13] the β-LOC-MEM is minimized by applying to border backward metrics an encoding technique based on non-uniform quantization. The works proposed in [11], [14], [15], [16] are all aimed at reducing the footprint of the λ^{ext}-MEM buffers at expense of reducing the error correcting capability of about 0.1 or 0.2 dB. In [11] a heuristically-determined nonlinear quantizer is proposed to reduce the bit-width of the extrinsic information. On the other hand, in [14] the same goal is achieved by using a pseudo-floating point representation, whereas in [15] a technique based on most significant bit (MSB) clipping combined with least significant bit (LSB) drop (at transmitter) and append (at receiver) is proposed. Finally, the work in [16] is aimed at reducing the bit width of the extrinsic information in double binary turbo decoders by converting symbol-level extrinsic information to bit-level information and vice-versa. As highlighted in the third row of Table I the area of parallelism dependent memories, increases linearly with P. On the other hand, according to [25], the throughput of a double binary turbo decoder architecture can be estimated as

$$T = \frac{N_b \cdot f_{clk}}{2I\left(\frac{N_T}{P} + W + \Delta\right)} \tag{9}$$

where N_b is the number of decoded bits, f_{clk} is the clock frequency, I is the number of iterations, N_T is the number of trellis steps ($N_T = N_c$ in this case), W is the window size and Δ is the pipeline depth of the λ-O processor. A similar expression can be written for binary codes as well, nevertheless, the throughput of the decoder grows sub-linearly with P due to the latency of the SISO processor ($W + \Delta$). As a consequence, by increasing P we increase more the area of parallelism dependent memories than the throughput of the decoder. In order to better highlight the contribution of each buffer to the total amount of memory in the turbo decoder we summarize in the fourth row of Table I the worst case values used in [26] and [25] for the implementation of the eight state WiMax double-binary turbo decoder: $N = 4 \cdot N_c$, $N_c = 2400$, $R = 0.5$ (that corresponds to an uncoded frame size $K = 2 \cdot N_c = 4800$ bits), window size $W = 40$ and $P = 4$. The other rows refer to $P = 8$ and $P = 16$ respectively with $W = 30$ so that $N_c/(P \cdot W) \in \mathbb{N}$ [26]. Since the complexity of the output buffer, which stores the decoded bits u_k, is negligible, it is not considered in this analysis. Furthermore, we do not consider memories that might be required to store the permutations for interleaving the extrinsic information. The following quantization scheme has been used in [25] for the representation of the LLRs and the SMs where n_x is the number of bits used to represent x, namely $n_{\lambda^{int}} = 6$, $n_{\lambda^{ext}} = 8$ and $n_{SM} = n_\alpha = n_\beta = 12$. In Fig. 1 and 2 (b), we identify two contributions to the total amount of memory bits in the decoder architecture: i) buffers whose contribution to the total memory bits does not depend on the decoder parallelism (parallelism independent buffers), as λ^{int}-MEM and λ^{ext}-MEM in Fig. 1; ii) buffers whose contribution to the total memory bits increases with P (parallelsim depended buffers), as BMU-MEM, α-MEM, β-LOC-MEM, α-EXT-MEM and β-EXT-MEM in Fig. 2 (b).

As shown in Table I the α-MEM is the most relevant memory among the parallelsim dependent buffers. Furthermore, as long as P increases, its relative cost becomes comparable with the λ^{ext}-MEM. As a consequence, reducing the α-MEM footprint in highly parallel turbo decoders has a significant impact on the whole decoder area and power consumption. Similarly to [11], [12], [13], [14], [15], [16], the memory reduction achieved in this work comes at the expense of a limited degradation of the error correcting performance as detailed in the next sections.

III. State metric compression

According to the standard data compression terminology, state metrics can be compressed resorting to either lossless or lossy techniques. It is known that lossless compression techniques do not alter performace, but the compression ratio that can be achieved is usually limited. On the other hand, lossy compression techniques achieve higher compression ratios at the expense of performance degradation. As an example, the non-uniform quantizations used in [11] and [13] to compress extrinsic information and border backward metrics respectively are lossy techniques, but they introduce a limited performance degradation. A generic data compression system is usually composed of two stages: a transform stage, which exploites data correlation, and an encoding stage,

Table I
WIMAX DOUBLE-BINARY TURBO DECODER MEMORY BREAKDOWN, $N_c = 2400$, $n_{\lambda^{int}} = 6$, $n_{\lambda^{ext}} = 8$, $n_{SM} = 12$

	parallelism independent		parallelism dependent			
	λ^{int}-MEM [bit]	λ^{ext}-MEM [bit]	BMU-MEM [bit]	α-MEM [bit]	β-LOC-MEM [bit]	α/β-EXT-MEM [bit]
P/W	$6 \cdot N_c \cdot n_{\lambda^{int}}$	$3 \cdot N_c \cdot n_{\lambda^{ext}}$	$W \cdot (3 \cdot n_{\lambda^{ext}} + 4 \cdot n_{\lambda^{int}}) \cdot P$	$8 \cdot W \cdot n_{SM} \cdot P$	$8 \cdot \left(\frac{N_c}{W \cdot P} - 1\right) \cdot n_{SM} \cdot P$	$2 \cdot (16 \cdot n_{SM} \cdot P)$
4/40	86400 (49.67%)	57600 (33.11%)	7680 (4.42%)	15360 (8.83 %)	5376 (3.09%)	1536 (0.88%)
8/30	86400 (45.82%)	57600 (30.55%)	11520 (6.11%)	23040 (12.22%)	6912 (3.67%)	3072 (1.63%)
16/30	86400 (38.33%)	57600 (25.55%)	23040 (10.22%)	46080 (20.44%)	6144 (2.73%)	6144 (2.73%)

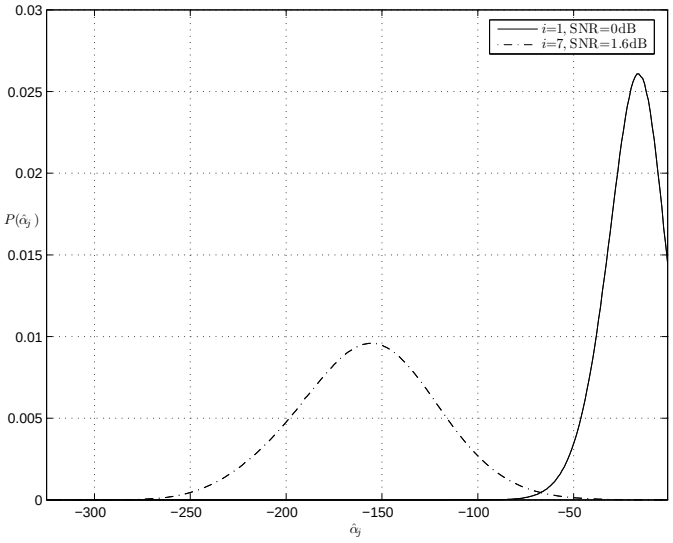

Figure 3. Distribution of one element of $\hat{\boldsymbol{\alpha}}$ (all the n_s elements have the same distribution)

which actually compresses the information. In the case of state metric compression, given a step on the trellis, we should exploit correlation among metrics. Thus, said n_s the number of states of the code and k a step in the trellis, we have $\boldsymbol{\alpha}_k = \{\alpha_{0,k}, \alpha_{1,k}, \ldots, \alpha_{n_s-1,k}\}$. The wrapping metric technique [27], [28] is usually employed to reduce the critical path in SM processors. However, it requires a normalization stage before computing the extrinsic information to minimize the memory requirement and to reduce the bitwidth of the λ-O processor data-path. On the other hand, as detailed in section V, this stage increases the length of the critical path. To minimize the number of bits required to represent normalized metrics, the normalization is usually performed by calculating $\hat{\boldsymbol{\alpha}}_k = \boldsymbol{\alpha}_k - M_k$ where $M_k = \max_j\{\alpha_{j,k}\}$[1]. During the first iteration, and particularly at low signal to noise ratios (SNR), all the metrics are likely to show small differences with respect to each other. Thus, they can be interpreted as a signal with low frequency components. On the other hand, during the last iterations, and particularly at medium or high SNR values, most of the SMs are far from the maximum M_k and just one or two of them are clearly higher than the other ones. In other words, the spread of values in $\hat{\boldsymbol{\alpha}}_k$ tends to increase with both SNR and iterations.

[1]With a slight abuse of notation we mean that each element of the $\hat{\boldsymbol{\alpha}}_k$ array is obtained by subtracting the scalar M_k from each element of the $\boldsymbol{\alpha}_k$ array

A. SM distribution analysis

To verify these conjectures we consider the WiMax double-binary turbo decoder settings detailed in section II. Then, we simulated 2×10^5 frames of 4800 ($2 \times N_c$) bits sent over an AWGN channel with a BPSK modulation at the first iteration ($i = 1$) with SNR = 0 dB and at the seventh iteration ($i = 7$) with SNR = 1.6 dB respectively. Finally, we collected the values of the n_s normalized forward state metrics and the corresponding Discrete-Fourier-Transform (DFT) values at each trellis step to estimate the occurrence probability of each value $\hat{\alpha}_j$ along the trellis. To that purpose, in Fig. 3 we show the statistical frequency $P(\hat{\alpha}_j)$ of one normalized forward state metric $\hat{\alpha}_j$ with $0 \leq j \leq n_s - 1$ (all the n_s elements have the same distribution along the trellis). Since $P(\hat{\alpha}_j = 0) \geq 1/8$ then $P(\hat{\alpha}_j = 0)$ is significantly higher than $P(\hat{\alpha}_j \neq 0)$; thus, in Fig. 3 $P(\hat{\alpha}_j = 0)$ is not shown for the sake of clarity. The corresponding values for $P(\hat{\alpha}_j = 0)$ are $P(\hat{\alpha}_j = 0) = 1.32 \times 10^{-1}$ with $i = 1$, SNR = 0 dB and $P(\hat{\alpha}_j = 0) = 1.25 \times 10^{-1}$ with $i = 7$, SNR = 1.6 dB respectively. Said $\boldsymbol{\phi}$ the DFT of $\hat{\boldsymbol{\alpha}}$, in Fig. 4 and 5 we show $P(\phi_j)$, the distribution of the j-th sample of $\boldsymbol{\phi}$ at the first iteration ($i = 1$) with SNR = 0 dB and at the seventh iteration ($i = 7$) with SNR = 1.6 dB. As it can be observed, in both cases the mean value of the DC component ($j = 0$) is the highest value.

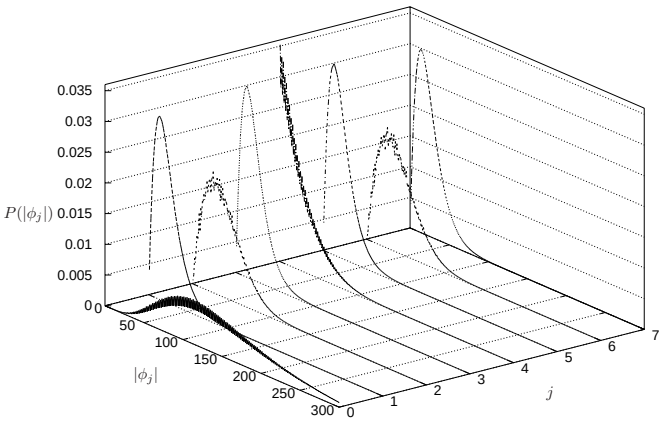

Figure 4. Distribution of ϕ_j at the first iteration ($i = 1$) with SNR = 0 dB

B. SM distance analysis

However, we need to study also the contribution of components at higher frequencies to properly represent $\hat{\boldsymbol{\alpha}}$ values. As a consequence, it makes sense to study the distance among metrics to achieve compression. Thus, depending on the SNR

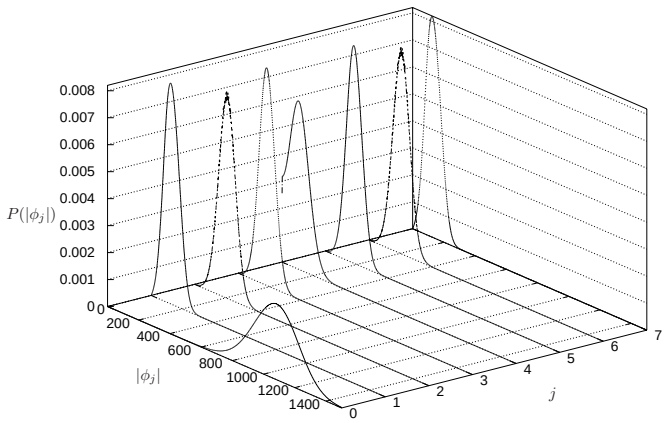

Figure 5. Distribution of ϕ_j at the seventh iteration ($i = 7$) with SNR = 1.6 dB

and the current iteration, for each trellis step we can build a set $\mathcal{A}_k$ where $\alpha_{j,k} \in \mathcal{A}_k$, $0 \leq j \leq n_s - 1$ if $\alpha_{j,k} < M_k$. Now we can define h_k as the number of elements belonging to $\mathcal{A}_k$ ($0 \leq h_k < n_s$) and $l_k = n_s - h_k$. Similarly, we can define $\hat{\mathcal{A}}_k$, where $\hat{\alpha}_{j,k} \in \hat{\mathcal{A}}_k$, $0 \leq j \leq n_s - 1$ if $\hat{\alpha}_{j,k} < 0$. From the definition of $\hat{\mathcal{A}}_k$, we can infer that h_k is also the number of elements in $\hat{\mathcal{A}}_k$. The introduced l_k amount evolves along the trellis according to the values assumed by $\alpha_{j,k}$. This evolution shows no regularity and appears as a random process. We then define a random variable, referred to as l, to indicate values assumed by l_k across trellis steps. Probability $P(l)$ for the defined random variable l indicates the probability of having l metrics equal to M_k in the same trellis step. $P(l)$ is lower for higher values of l and tends to decrease with both iterations and SNR. This can be seen in Fig. 6, where we show the

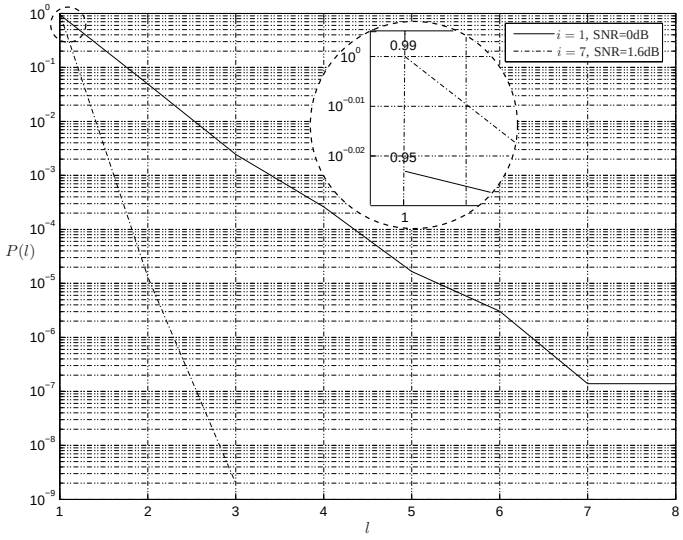

Figure 6. Probability of having l metrics equal to M_k: distribution of l at the first iteration ($i = 1$) with SNR = 0 dB (solid line) and at the seventh iteration ($i = 7$) with SNR = 1.6 dB (dotted line)

distribution of l at the first iteration ($i = 1$) with SNR = 0 dB and at the seventh iteration ($i = 7$) with SNR = 1.6 dB. As it can be observed, in both cases the probability of having only one metric equal to M_k ($l = 1$) at step k is maximum and very close to 1 (0.95 and 0.99 respectively). On the other hand, the value of $P(l > 1)$ is significantly higher when $i = 1$, SNR = 0 than when $i = 7$, SNR = 1.6 dB.

Then, we expect that for every couple $\hat{\alpha}_{p,k}, \hat{\alpha}_{q,k} \in \hat{\mathcal{A}}_k$ with $p \neq q$ and $0 \leq p, q \leq n_s - 1$, the difference $d_{p,q,k} = |\hat{\alpha}_{p,k} - \hat{\alpha}_{q,k}|$ is very small. Also amounts $d_{p,q,k}$ show a random-like behavior in the trellis evolution, thus, we define a second random variable d. However, the distribution of d values at trellis steps where l is large is highly different from distribution at trellis steps where l is small. Therefore we introduce n_s sets $\mathcal{D}_l$ defined as follows: $d_{p,q,k} \in \mathcal{D}_l$ if $l_k = l$. $P(d \in \mathcal{D}_l)$ gives the distribution of d values in each set $\mathcal{D}_l$. It is worth pointing out that, since $\mathcal{D}_{n_s} = \emptyset$ by construction, $P(d \in \mathcal{D}_{n_s}) = 0$. Distributions of d values in $\mathcal{D}_l$ sets are given in Fig. 7 and 8, where $P(d \in \mathcal{D}_l)$ are plotted at the first iteration ($i = 1$) with SNR = 0 dB and at the seventh iteration ($i = 7$) with SNR = 1.6 dB respectively.

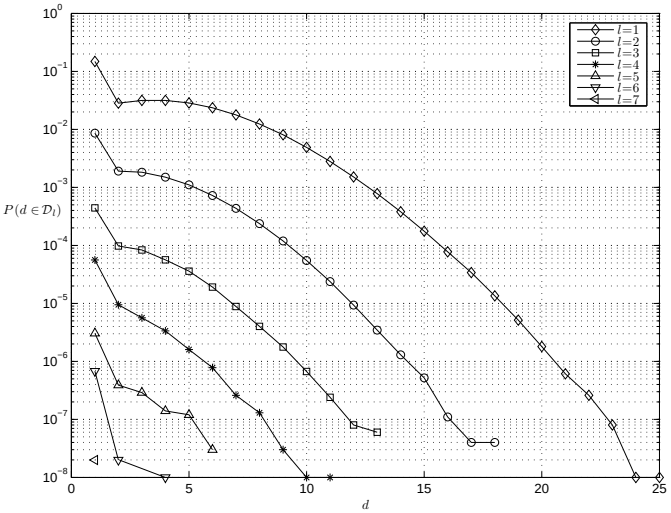

Figure 7. Distribution of d at the first iteration ($i = 1$) with SNR = 0 dB

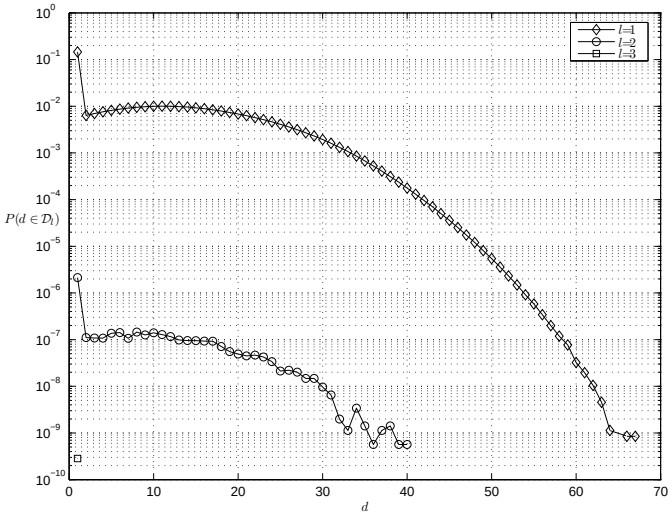

Figure 8. Distribution of d at the seventh iteration ($i = 7$) with SNR = 1.6 dB

As it can be observed in Fig. 7 and 8 the maximum d (d_M)

is 25 and 67 respectively. Compared with the mean value of $\hat{\alpha}_j$ ($\mu_{\hat{\alpha}_j}$) in Fig. 3, that are -16 and -155 respectively, we observe that in the first case ($i = 1$ and SNR = 0 dB) $d_M > |\mu_{\hat{\alpha}_j}|$, whereas, in the second case, $d_M < |\mu_{\hat{\alpha}_j}|$. However, these two events have a probability that is less than 10^{-8}. Moreover, in the case $i = 1$ and SNR = 0 dB, $l = 1$ collects about the 95% of the distribution of d and more that the 99% is obtained for $l = 1, 2$. Furthermore, for the case $i = 7$ and SNR = 1.6 dB $l = 1$ represents more that the 99% of the distribution of d.

From the analysis presented in the previous paragraphs we can infer that:

- At each trellis step there is a high probability of having few metrics higher than the other ones (almost only one metric is equal to M_k, $l = 1$).
- The remaining ones differ one from each other of few tens and the larger is the difference value, the smaller is its probability.

These results show that correlation exists and can be exploited to compress forward state metrics. As a consequence, a proper transform stage should be employed. This stage should be able to extract the DC component of $\hat{\boldsymbol{\alpha}}_k$ and to effectively represent d. However, the complexity overhead induced by the compression/decompression technique must be as limited as possible. Unfortunately, several transform stages able to separate the frequency components of a signal require multiplications [29]. Thus, multiplierless transform stages are interesting solutions to extract the existing correlation among state metrics with a limited complexity overhead.

IV. Proposed state metric compression scheme

The optimal transform stage to extract the correlation of a random process is the Karhunen-Loéve Transform (KLT) [29]. Unfortunately, its prohibitive complexity makes the KLT use for state metric compression not practical. Depending on the amount of correlation among data Discrete-Sine-Transform (DST) and Discrete-Cosine-Transform (DCT) are usually used instead of the KLT [29]. However, both the DST and the DCT require multiplications. In this scenario the Walsh-Hadamard-Transform is a particularly simple solution. Even if it is known that the Walsh-Hadamard-Transform has lower energy compaction capability than other transforms, it can be implemented by resorting to only additions and subtractions. This reduced complexity figure makes it an attractive candidate to compress state metrics.

A. Walsh-Hadamard-Transform

The Walsh-Hadamard-Transform (WHT) [17] is an orthogonal transform where only additions are required. It can be represented as matrix $\mathbf{H}$ containing only +1 and -1. The smallest orthonormal Hadamard matrix is the 2×2 matrix defined as

$$\mathbf{H}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{10}$$

In general, the $2^n \times 2^n$ Hadamard matrix is obtained as

$$\mathbf{H}_n = \begin{pmatrix} \mathbf{H}_{n-1} & \mathbf{H}_{n-1} \\ \mathbf{H}_{n-1} & -\mathbf{H}_{n-1} \end{pmatrix} \tag{11}$$

Moreover, since the $\mathbf{H}_n$ is symmetric and orthogonal $(\mathbf{H}_n)^{-1} = \mathbf{H}_n$. Thus, for a constituent CC with n_s states we can perform the WHT on $\hat{\boldsymbol{\alpha}}_k$ resorting to the $n_s \times n_s$ Hadamard matrix (11). As an example for the WiMax turbo code ($n_s = 8$) we have $\mathbf{H}_3 = K_3 \cdot \hat{\mathbf{H}}_3$ with $K_3 = 1/(\sqrt{2})^3$ and

$$\hat{\mathbf{H}}_3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix} \tag{12}$$

To ease hardware implementation, we can neglect K_3 at the direct transform side by computing $\boldsymbol{\xi}_k = \hat{\mathbf{H}}_3 \cdot \hat{\boldsymbol{\alpha}}_k$. Then, at the inverse transform side we implement $\hat{\boldsymbol{\alpha}}_k = (K_3)^2 \cdot \hat{\mathbf{H}}_3 \cdot \boldsymbol{\xi}_k$. Since $(K_n)^2$ is a power of two its implementation is trivial. It is worth pointing out that the WHT can be effectively implemented in a butterfly fashion with $n_s \cdot \log_2 n_s$ adders, as shown in Fig. 9 (a) and (b) for $n_s = 8$.

B. Quantization

A reduced complexity quantization scheme should be employed. To that purpose the non-uniform quantization scheme used in [13] to encode border backward metrics is a suitable solution. In the following we will discuss the quantizer applied to the WHT outputs, even if in section V we will show the results obtained by quantizing either $\xi_{j,k}$ or $\hat{\alpha}_{j,k}$. This quantization scheme floors the original metric value to the closest power-of-two. Since $\xi_{j,k}$ can be either positive or negative, we first check $|\xi_{j,k}| \neq 0$ and compute $|\xi_{j,k}|$, then with a leading-one-detector (LOD) and an encoder we obtain $\lfloor \log_2(|\xi_{j,k}|) \rfloor$ [30]. However, in order to reduce the quantization error we pose

$$\zeta_{j,k} = \text{sign}(\xi_{j,k}) \cdot \lfloor \log_2(|\xi_{j,k}|) + 0.5 \rfloor \tag{13}$$

Since $y = \log_2 x = y_i + y_f$ where y_i and y_f are the integer and the fractional part of y respectively, and $y_i = \lfloor \log_2 x \rfloor$ we have

$$|\zeta_{j,k}| = \begin{cases} \lfloor \log_2(|\xi_{j,k}|) \rfloor & \text{if } f_{j,k} < 0.5 \\ \lfloor \log_2(|\xi_{j,k}|) \rfloor + 1 & \text{if } f_{j,k} \geq 0.5 \end{cases} \tag{14}$$

where $f_{j,k}$ is the fractional part of $\log_2(|\xi_{j,k}|)$. Then, exploiting the monotonicity of the function $y = 2^x$ we obtain

$$|\zeta_{j,k}| = \begin{cases} \lfloor \log_2(|\xi_{j,k}|) \rfloor & \text{if } 2^{f_{j,k}} < \sqrt{2} \\ \lfloor \log_2(|\xi_{j,k}|) \rfloor + 1 & \text{if } 2^{f_{j,k}} \geq \sqrt{2} \end{cases} \tag{15}$$

Since $2^{f_{j,k}} = |\xi_{j,k}|/2^{\lfloor \log_2(|\xi_{j,k}|) \rfloor}$ we can infer that $2^{f_{j,k}}$ binary representation is equal to the binary representation of $|\xi_{j,k}|$ except for the binary point position. As a consequence, we can compute $2^{f_{j,k}} \geq \sqrt{2}$ in (15) by considering $|\xi_{j,k}|$ and $\sqrt{2}$ binary representations, aligning the leading '1' of $|\xi_{j,k}|$ to the leading '1' of $\sqrt{2}$ and comparing these values. The alignment is performed by a small left-shifter with the shift-amount command driven by $\lfloor \log_2(|\xi_{j,k}|) \rfloor$. The complete block scheme of the quantizer is show in Fig. 9 (c) and (d)

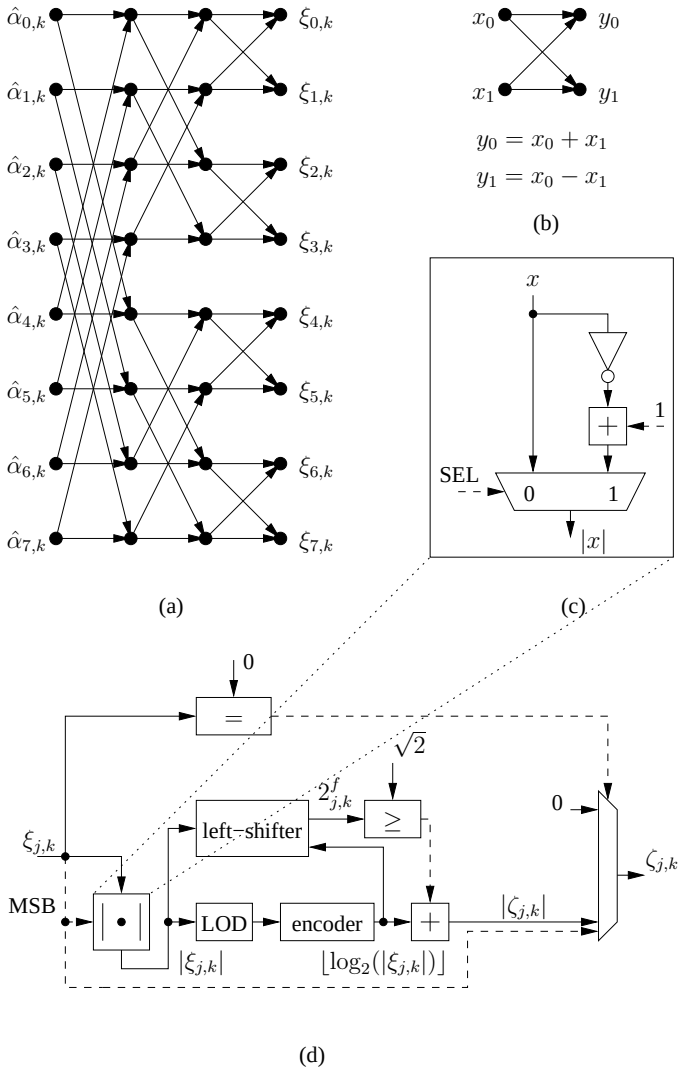

Figure 9. Butterfly based 8-point WHT data flow graph (a), (b) and quantizer block scheme (c), (d)

where MSB stands for most-significant-bit. On the other hand, the dequantizer computes $\bar{\xi}_{j,k} = \text{sign}(\zeta_{j,k}) \cdot 2^{|\zeta_{j,k}|}$ by the means of a shifter and few logic.

Due to the presence of the quantizer/dequantizer at the inverse transform side we obtain $\bar{\boldsymbol{\alpha}}_k = (K_3)^2 \cdot \hat{\mathbf{H}}_3 \cdot \bar{\boldsymbol{\xi}}_k$ instead of $\hat{\boldsymbol{\alpha}}_k$. It is worth pointing out that the implementation of $(K_3)^2$ at the inverse transform side increases the dynamic range of $\xi_{j,k}$. However, as it will be detailed in section V, this has no effect on the dynamic range of $\zeta_{j,k}$ in the considered cases.

V. Experimental results

The proposed techniques have been compared in terms of bit-error-rate (BER) performance and complexity with other techniques in two significant cases: i) the WiMax turbo decoder architecture with the settings summarized in sections II and III ii) the serial concatenation turbo decoder (SCCC) proposed in the MHOMS system [31] and implemented as a parallel architecture in [32].

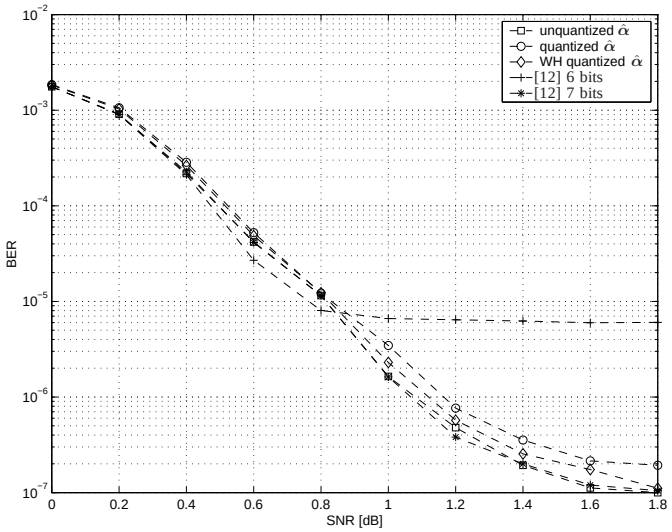

Figure 10. WiMax turbo decoder BER performance comparison

A. WiMax turbo decoder

As highlighted in Fig. 3 given $n_{\lambda^{int}} = 6$ and $n_{\lambda^{ext}} = 8$ as in [25], we obtain that $\hat{\alpha}_{j,k}$ magnitude is represented on 9 bits and as a two complement value on 10 bits. Simulations show that $\xi_{j,k}$ requires no more than 11 bits and, as a consequence, $\zeta_{j,k}$ is represented on 5 bits as a sign and module value.

1) Performance: In Fig. 10 we show the performance obtained for the WiMax turbo decoder configured as detailed in sections II and III after seven iterations. The square-marked curve represents performance obtained with unquantized metrics (UM). With the circle-marked curve we depict the performance obtained by directly applying the quantizer described in section IV-B to $\hat{\alpha}_{j,k}$ (QM). Since $\hat{\alpha}_{j,k} \leq 0$ the corresponding encoded value ($\chi_{j,k} = \lfloor \log_2(|\hat{\alpha}_{j,k}|) + 0.5 \rfloor$) is represented on 4 bits. As it can be observed, the curve of this solution is extremely closed to the unquantized curve at the beginning of the waterfall region. However, as long as the SNR becomes higher than 0.8 dB the distance between the two curves increases and the circle-marked curve floors to 2×10^{-7}. The diamond-marked curve shows the performance obtained with the proposed state metric compression system (WHT and quantizer, WM). As show in Fig. 10 the performance of the proposed solution falls in between the unquantized and the circle-marked curve with a floor of about 10^{-7} as for the UM square-marked curve. On the other hand, the cross-marked and asterisc-marked curves show the performance of SM saturation applied outside the metric update loop (OM) as proposed in [12]. Since applying saturation on 4 bits leads to excessive performance degradation we impose to saturate $\hat{\alpha}_{j,k}$ on 6 and 7 bits respectively. In the following we will refer to the saturated $\hat{\boldsymbol{\alpha}}_k$ values as $\hat{\boldsymbol{\alpha}}^s_k$. As it can be observed the OM technique with SM saturation on 7 bits shows nearly the same performance of the proposed WM technique.

2) Complexity: In Fig. 11 UM, QM, WM and OM architectures are shown to highlight the blocks employed in each architecture. In order to save memory we perform the forward metric normalization at the input of the α-MEM buffer, instead

Table II
COMPARISON OF UM, QM, WM AND 7 BITS OM SOLUTIONS $W = 40$ (WIMAX TURBO DECODER $n_s = 8$): AREA (A), CRITICAL PATH (CP) AND POWER CONSUMPTION (PC)

Arch.	Data	word width	Mem. SP [bit]/[μm^2]	Mem. DP [bit]/[μm^2]	LO [EG]/[μm^2]	CP [ns]	Mem. SP + LO A [μm^2]	Mem. SP + LO PC [mW]	Mem. DP + LO A [μm^2]	Mem. DP + LO PC [mW]
UM	$\hat{\boldsymbol{\alpha}}_k$	$9n_s$	5760/118530	2880/84909	-/-	1.8	118530 (100%)	41.26	84909 (100%)	24.01
QM	$\boldsymbol{\chi}_k$	$4n_s$	2560/56760	1280/43409	820/4922	2.2	61682 (52.0%)	23.19	48331 (56.9%)	13.99
WM	$\boldsymbol{\zeta}_k$	$5n_s$	3200/69115	1600/51709	3931/23585	3.2	92700 (78.2%)	29.84	75294 (88.7%)	18.34
OM	$\hat{\boldsymbol{\alpha}}^s_k$	$7n_s$	4480/93825	2240/68309	89/533	2.0	94358 (79.6 %)	34.26	68842 (81.1%)	19.31

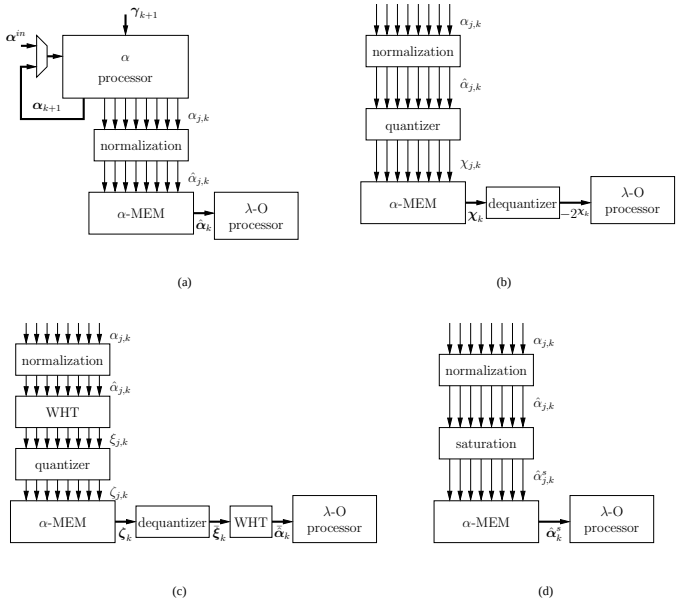

Figure 11. UM (a), QM (b), WM (c) and OM (d) block schemes

of into the λ-O processor as in [25] (see Fig. 11 (a)). To compare the complexity of hardware implementation of the UM, QM, WM and OM solutions, see Fig. 11 (a), (b), (c) and (d), we implemented them in VHDL and synthesize them on a 130 nm standard cell technology with Synopsys Design Compiler imposing a clock frequency of 200 MHz. Moreover, we generate the corresponding memories with a 130 nm RAM generator both as single port (SP) and double port (DP) RAMs. In fact the α-MEM memory (as the other memories in the decoder architecture) can be implemented either as one DP RAM or as a double buffer with two SP RAMs. In Table II we compare the complexity in terms of area (A), giving both the equivalent gates (EG) and the μm^2, the critical path (CP) and the power consumption (PC) of the UM, QM, WM and OM architectures. As it can be inferred from Table II and Fig. 10 the QM solution leads to a complexity reduction of about 50%, with a moderate BER performance degradation. This memory reduction leads also to a significant reduction of the power consumption, with a small increase of the critical path. On the other hand, both OM on 7 bits and the proposed WM solutions achieve nearly the BER performance of the UM architecture with a complexity reduction between about 10% and 20%. However, the WM solution has higher logic overhead (LO) than the OM one, besides WM has a longer critical path than OM. For a 200 MHz target clock frequency, the critical path of WM leads to no more than a one cycle pipeline delay. The throughput of an UM turbo decoder architecture can be estimated with (9); since the WM technique adds at most one clock cycle we have

$$T_{WM} = \frac{N_b \cdot f_{clk}}{2I\left(\frac{N_T}{P} + W + \Delta + 1\right)} \tag{16}$$

As a consequence, we obtain a throughput reduction with respect to T_{UM} of $(N_T/P+W+\Delta)/(N_T/P+W+\Delta+1)$. With $N_T = 2400$, $P = 4$, $W = 40$ and $\Delta = 5$ (as in [25]) leads to T_{WM} about 0.16% of T_{UM}. It is worth pointing out that the reduced memory footprint achieved with the WM solution leads to a lower power consumption than the OM architecture. Finally, we observe that the OM architecture reduces also the hardware complexity and the power consumption of the λ-O processor, as it produces forward state metrics on a reduced number of bits with respect to UM, QM and WM solutions. Post synthesis results show that the λ-O processor for the 7 bits OM architecture occupies 70277 μm^2 and consumes 12.3 mW, whereas it occupies 71128 μm^2 and consumes 12.6 mW in the case of UM, QM and WM solutions. These results, with the ones shown in table II, confirm the interesting power consumption figure of the WM architecture and that WM and OM solutions have comparable complexity.

B. MHOMS turbo decoder

The MHOMS SCCC turbo decoder is based on a four state ($n_s = 4$), rate 1/2, recursive systematic CC which is used both as inner and outer constituent code. In this work we set the uncoded frame size to $K = 1022$ and the coded frame size to 3076. Since the concatenation is serial this leads in the worst case (inner CC) to $N = 3076$. The quantization scheme for the LLRs is $n_{\lambda^{int}} = 6$, $n_{\lambda^{ext}} = 8$ and $n_{SM} = n_\alpha = n_\beta = 10$ [32]. The $\overset{*}{\max}\{x_i\}$ function has been implemented as a $\max$ followed by a 3 bit correction term stored in a 22 position LUT. The decoder parallelism degree is $P = 16$ and the window size, that is different for inner (I) and outer (O) SISOs, is $W_I = 48$ and $W_O = 32$. Experimental results show that the required bitwidth for $\hat{\alpha}_{j,k}$, $\chi_{j,k}$ and $\zeta_{j,k}$ is the same obtained for the WiMax turbo decoder.

1) Performance: In Fig. 12 we show the performance obtained for the MHOMS SCCC turbo decoder configured as detailed in section V-B after ten iterations using 4PSK modulation and AWGN channel.

As it can be observed, the obtained BER performance is very close to what shown for the WiMax turbo decoder, namely the BER performance of the proposed WM solution is in between UM and QM; the OM technique performs nearly as the WM one.

Table III
COMPARISON OF UM, QM, WM AND 7 BITS OM SOLUTIONS $W = 48$ (MHOMS SCCC TURBO DECODER $n_s = 4$): AREA (A), CRITICAL PATH (CP) AND POWER CONSUMPTION (PC)

Arch.	Data	word width	Mem. SP [bit]/[μm^2]	Mem. DP [bit]/[μm^2]	LO [EG]/[μm^2]	CP ns	Mem. SP + LO A [μm^2]	Mem. SP + LO PC [mW]	Mem. DP + LO A [μm^2]	Mem. DP + LO PC [mW]
UM	$\hat{\boldsymbol{\alpha}}_k$	$9n_s$	3456/64400	1720/49753	-/-	1.4	64400 (100%)	24.12	49753 (100%)	13.77
QM	$\boldsymbol{\chi}_k$	$4n_s$	1536/32839	768/28135	411/2461	1.9	35300 (54.8%)	14.99	31225 (61.5%)	8.09
WM	$\boldsymbol{\zeta}_k$	$5n_s$	1920/39151	960/32459	1507/9043	2.4	48194 (74.8%)	17.85	41502 (83.4%)	9.80
OM	$\hat{\boldsymbol{\alpha}}_k^s$	$7n_s$	2688/51775	1344/41106	356/2132	1.5	53907 (83.7 %)	19.44	43238 (86.9 %)	11.39

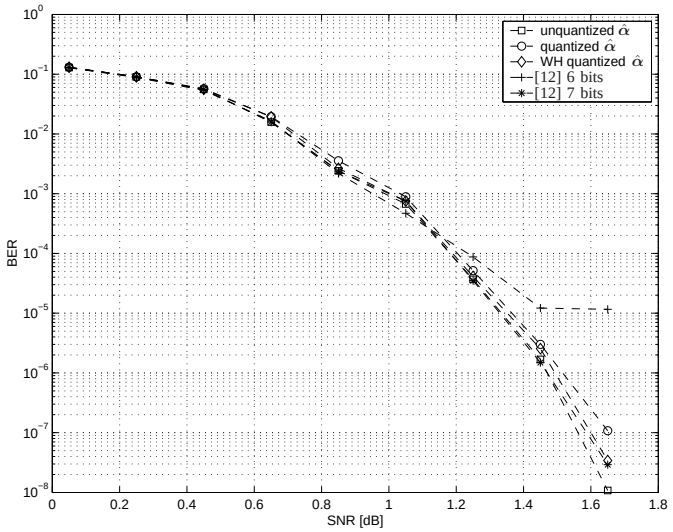

Figure 12. MHOMS SCCC turbo decoder performance comparison

2) Complexity: Similarly to the WiMax case, we perform the forward metric normalization at the input of the α-MEM buffer, instead of into the λ-O processor as in [32]. From the implementation point of view, in the following, we consider the worst case window size: $W = \max\{W_I, W_O\} = 48$. In table III we compare the complexity in terms of area (A), giving both the equivalent gates (EG) and the μm^2, the critical path (CP) and the power consumption (PC) of the UM, QM, WM and OM architectures. These results are obtained as post synthesis values with Synopsys Design Compiler on a 130 nm standard cell technology for a 200 MHz clock frequency. As it can be inferred from Table III and Fig. 12 the QM solution leads to a complexity reduction of about 50%, with a moderate performance degradation and a significant reduction of the power consumption. The OM solution with 7 bits and the proposed WM architecture have nearly the same BER performance as the UM implementation; besides they achieve a complexity reduction between about 15% and 25%. As for the WiMax case, the WM solution presents a longer critical path than the OM one. On the contrary, the WM technique has better power consumption figures than the OM one. Considering the complexity and power consumption of the λ-O processor we obtain an area of 43119 μm^2 and a power consumption of 4.7 mW for UM, QM and WM architectures and an area of 42849 μm^2 and a power consumption of 4.6 mW for OM.

VI. CONCLUSIONS

In this work two techniques to compress state metrics to reduce the memory in turbo decoder architectures have been presented. The first technique, based on non-uniform quantization, reduces the SM memory of about 50%, compared with architectures where state metric compression is not performed, at the expense of slightly increasing the error correcting performance floor. Thus, it can be employed with codes that exhibit verly low error floor, as the MHOMS SCCC, to obtain a significant complexity reduction. The second technique, based on the Walsh-Hadamard transform and non-uniform quantization, shows excellent error correcting performance. Moreover, its complexity overhead is moderate and compared with a decoder where SM are not compressed allows for a SM memory reduction of more that 20% in the best case. As a consequence, this solution is well suited to reduce the decoder area when the code error floor should be preserved, as for the WiMax turbo code. Finally, both solutions show lower power consumption than architectures previously proposed to compress state metrics.

ACKNOWLEDGMENT

This work is partially supported by the NEWCOM++ network of excellence, funded by the European Community, and by the WIMAGIC project, funded by the European Community.

REFERENCES

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

[2] T. Vogt and N. Wehn, "Reconfigurable ASIP for convolutional and turbo decoding in an SDR environment," *IEEE Transactions on VLSI*, vol. 16, no. 10, pp. 1309–1320, Oct 2008.

[3] L. 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.

[4] E. Boutillon, C. Douillard, and G. Montorsi, "Iterative decoding of concatenated convolutional codes: Implementation issues," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1201–1227, Jun 2007.

[5] M. Martina and G. Masera, "Turbo NOC: A framework for the design of network-on-chip-based turbo decoder architectures," *IEEE Transactions on Circuits and Systems I*, vol. 57, no. 10, pp. 2776–2789, Oct 2010.

[6] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-input soft-output modules for the construction and distributed iterative decoding of code networks," *European Transactions on Telecommunications*, vol. 9, no. 2, pp. 155–172, Mar/Apr 1998.

[7] J. H. Kim and I. C. Park, "A unified parallel radix-4 turbo decoder for mobile WiMAX and 3GPP-LTE," in *IEEE Custom Integrated Circuits Conference*, 2009, pp. 487–490.

[8] O. Muller, A. Baghdadi, and M. Jezequel, "From parallelism levels to a multi-ASIP architecture for turbo decoding," *IEEE Transactions on VLSI*, vol. 17, no. 1, pp. 92–102, Jan 2009.

[9] Y. Sun, Y. Zhu, M. Goel, and J. R. Cavallaro, "Configurable and scalable high throughput turbo decoder architecture for multiple 4G wireless standards," in *IEEE International Conference on Application-Specific Systems, Architectures and Processors*, 2008, pp. 209–214.

[10] 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*, 2010, pp. 1420–1425.

[11] J. Vogt, J. Ertel, and A. Finger, "Reducing bit width of extrinsic memory in turbo decoder realisations," *IEE Electronics Letters*, vol. 36, no. 20, pp. 1714–1716, Sep 2000.

[12] H. Liu, J. P. Diguet, C. Jego, M. Jezequel, and E. Boutillon, "Energy efficient turbo decoder with reduced state metric quantization," in *IEEE Workshop on Signal Processing and Systems*, 2007, pp. 237–242.

[13] J. H. Kim and I. C. Park, "Double-binary circular turbo decoding based on border metric encoding," *IEEE Transactions on Circuits and Systems II*, vol. 55, no. 1, pp. 79–83, Jan 2008.

[14] S. M. Park, J. Kwak, and K. Lee, "Extrinsic information memory reduced architecture for non-binary turbo decoder implementation," in *IEEE Vehicular Technology Conference*, 2008, pp. 539–543.

[15] A. Singh, E. Boutillon, and G. Masera, "Bit-width optimization of extrinsic information in turbo decoder," in *International Symposium on Turbo Codes & Related Topics*, 2008, pp. 134–138.

[16] J. H. Kim and I. C. Park, "Bit-level extrinsic information exchange method for double-binary turbo codes," *IEEE Transactions on Circuits and Systems II*, vol. 56, no. 1, pp. 81–85, Jan 2009.

[17] C. R. Gonzalez, E. R. Woods, and S. L. Eddins, *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., 2006.

[18] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Algorithm for continuous decoding of turbo codes," *IET Electronics Letters*, vol. 32, no. 4, pp. 314–315, Feb 1996.

[19] A. Abbasfar and K. Yao, "An efficient and practical architecture for high speed turbo decoders," in *IEEE Vehicular Technology Conference*, 2003, pp. 337–341.

[20] C. Zhan, T. Arslan, A. T. Erdogan, and S. MacDougall, "An efficient decoder scheme for double binary circular turbo codes," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2006, pp. 229–232.

[21] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the Log domain," in *IEEE ICC*, 1995, pp. 1009–1013.

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

[23] G. Montorsi and S. Benedetto, "Design of fixed-point iterative decoders for concatenated codes with interleavers," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 5, pp. 871–882, May 2001.

[24] C. Berrou, M. Jezequel, C. Douillard, and S. Kerouedan, "The advantages of non-binary turbo codes," in *IEEE Information Theory Workshop*, 2001, pp. 61–63.

[25] M. Martina, M. Nicola, and G. Masera, "VLSI implementation of WiMax convolutional turbo code encoder and decoder," *Journal of Circuits, Systems and Computers*, vol. 18, no. 3, pp. 534–564, May 2009.

[26] ——, "Hardware design of a parallel, collision-free interleaver for WiMax duo-binary turbo decoding," *IEEE Communications Letters*, vol. 12, no. 11, pp. 846–848, Nov 2008.

[27] A. P. Hekstra, "An alternative to metric rescaling in Viterbi decoders," *IEEE Transactions on Communications*, vol. 37, no. 11, pp. 1220–1222, Nov 1989.

[28] E. Boutillon, W. J. Gross, and P. G. Gulak, "VLSI architectures for the MAP algorithm," *IEEE Transactions on Communications*, vol. 51, no. 2, pp. 175–185, Feb 2003.

[29] K. Sayood, *Introduction to Data Compression (3rd Edition)*. Morgan Kaufmann, 2005.

[30] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Transactions on Electronic Computers*, vol. 11, no. 4, pp. 512–517, Aug 1962.

[31] S. Benedetto, R. Garello, G. Montorsi, C. Berrou, C. Douillard, D. Giancristofaro, A. Ginesi, L. Giugno, and M. Luise, "MHOMS: High-speed ACM modem for satellite applications," *IEEE Wireless Communications*, vol. 12, no. 2, pp. 66–77, Apr 2005.

[32] M. Martina, A. Molino, F. Vacca, G. Masera, and G. Montorsi, "High throughput implementation of an adaptive serial concatenation turbo decoder," *Journal of Communications, Software and Systems*, vol. 2, no. 3, pp. 252–261, Sep 2006.

Maurizio Martina was born in Pinerolo, Italy, in 1975. He received the M.Sc. and Ph.D. in electrical engineering from Politecnico di Torino, Italy, in 2000 and 2004 respectively. He is currently a Post-doctoral Researcher at the VLSI Lab, Politecnico di Torino. His research activities include VLSI design and implementation of architectures for digital signal processing and comunications.

Guido Masera received the Dr.Eng. degree (summa cum laude) in 1986, and the Ph.D. degree in electrical engineering from Politecnico di Torino, Italy, in 1992. Since 1986 to 1988 he was with CSELT (Centro Studi e Laboratori in Telecomunicazioni, Torino, Italy) as a researcher involved in the standardization activities for the GSM system. Since 1992 he has been Assistant Professor and then Associate Professor at the Electronic Department, where he is a member of the VLSI-Lab group. His research interests include several aspects in the design of digital integrated circuits and systems, with special emphasis on high-performance architecture development (especially for wireless communications and multimedia applications) and on-chip interconnect modeling and optimization. He has coauthored more than 160 journal and conference papers in the areas of ASIC-SoC development, architectural synthesis, VLSI circuit modeling and optimization. In the frame of National and European research projects, he has been co-designer of several ASIC and FPGA implementations in the fields of Artificial Intelligence, Computer Networks, Digital Signal Processing, Transmission and Coding.