Iteration Overlap and Dual-Sided LSOVA for Efficient Parallel Turbo Decoding Architectures

Stefan Weithoffer, Charbel Abdel Nour

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

Iteration Overlap and Dual-Sided LSOVA for Efficient Parallel Turbo Decoding Architectures

STEFAN WEITHOFFER[1], (Member, IEEE), CHARBEL ABDEL NOUR[2] (Senior Member, IEEE)
[1]IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France (e-mail: stefan.weithoffer@imt-atlantique.fr)
[2]IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France (e-mail: charbel.abdelnour@imt-atlantique.fr)

Corresponding author: Stefan Weithoffer (e-mail: stefan.weithoffer@imt-atlantique.fr).

This work was partially funded by the French National Research Agency TurboLEAP project (ANR-20-CE25-0007)

ABSTRACT

An efficient turbo decoder architecture is proposed by jointly exploring advancements in code design, decoding algorithms, and hardware implementation to achieve high-throughput and low-latency performance. At the algorithmic level, we introduce Dual-Sided Local-SOVA (DS-LSOVA), an enhanced variant of Local-SOVA that significantly lowers the complexity of extrinsic information computation while preserving decoding accuracy. To further improve implementation efficiency, iteration overlap interleavers are developed specifically for fully pipelined, iteration-unrolled UXMAP decoders, enabling full iteration overlap. This design reduces decoding latency and boosts hardware area efficiency by up to 22%. Importantly, these gains are achieved without sacrificing error-correcting performance: DS-LSOVA matches the Frame Error Rate (FER) of conventional max-Log-MAP decoding, while the proposed interleavers deliver FER results comparable to the Quadratic Permutation Polynomial (QPP) interleavers used in LTE.

INDEX TERMS Forward Error Correction, Fully Parallel, High-throughput, Local-SOVA, Turbo decoder.

I. INTRODUCTION

THE *Long Term Evolution Advanced Pro* (LTE-A Pro) standard [1] has played a pivotal role in the transition toward 5G networks, supporting both the progressive evolution and seamless coexistence of LTE-A Pro systems with *5G New Radio* (5G-NR) [2]. Beyond mobile broadband, LTE-A and LTE-A Pro are also being adapted to emerging use cases such as Machine-to-Machine (M2M) communication [3] and Vehicle-to-Everything (V2X) applications [4]. This ongoing evolution imposes increasingly stringent requirements on throughput and latency—constraints that directly impact the physical layer, where efficient and robust robust decoding mechanisms for error correction play a crucial role.

Turbo codes [5], widely employed for error correction in LTE standards, are renowned for their inherent rate flexibility and low-complexity encoding. However, when targeting ultra-high-throughput decoding, they face strong competition from Low-Density Parity-Check (LDPC) codes [6], which leverage a decoding algorithm that lends itself more naturally to parallelization.

As a result, highly parallel, ultra-high-throughput Turbo Decoding (TD) has garnered relatively limited attention from the research community, despite notable advancements in area efficiency (measured in throughput per mm^2) [7]–[9]. Nevertheless, recent progress in code design [10], decoding algorithms [11], and hardware architectures [7], [12] demonstrates that substantial performance gains continue to be achieved. These developments also highlight a critical limitation in prior research: the predominant reliance on variants of the Bahl, Cocke, Jelinek, and Raviv (BCJR) algorithm [13], which, due to its recursive nature, poses significant challenges for efficient hardware parallelization.

For Beyond 5G applications, for which channel coding schemes are discussed in [14], [15], decoding latency has emerged as a critical performance metric [16]. To address this, shuffled decoding—which extends parallelism to the iteration level through immediate extrinsic information exchange [17], [18]—has been proposed as a technique to reduce the latency of iterative decoders. However, the use of incomplete extrinsic information in this approach leads to a degradation in error-correcting performance, necessitating additional decoding iterations to compensate. This not only undermines the intended latency benefits but also results in increased area overhead for fully pipelined, iteration-unrolled decoders designed for ultra-high-throughput scenarios [7], [8].

While efficient scheduling of layer-wise decoding has been extensively explored for LDPC decoders—primarily to address the limitations of immediate or partial extrinsic infor-

mation exchange [19]–[21]—the concept has only recently regained attention in the context of Turbo decoding in the form of iteration overlap [22], [23]. Building on our recent contributions in code and interleaver design, decoding algorithms, and hardware architectures, this work takes a significant step forward by integrating these advances into a fully realized decoder implementation for the first time.

The main contributions of this work are as follows:

- The introduction of a dual-sided computation scheme for extrinsic information within the Local-SOVA (LSOVA) algorithm, resulting in the Dual-Sided Local-SOVA (DS-LSOVA) variant (Section II-B);
- The design of iteration overlap interleavers that enable full iteration overlap between the half-iteration pipeline stages in fully pipelined, iteration-unrolled decoders (Section III);
- The development of the first hardware architecture and implementation results for fully iteration-unrolled decoders based on both LSOVA and DS-LSOVA (Sections IV and V).

The remainder of this paper is organized as follows: Section II introduces the three decoding algorithms considered in this work, including the proposed DS-LSOVA. Section III details the iteration-overlap scheduling strategy and the corresponding interleaver design. The focus then shifts to hardware architectures in Section IV, with implementation results presented in Section V. Finally, Section VI concludes the paper.

II. DECODING ALGORITHMS

Fig. 1 shows the structure of a turbo decoder that consists of two component decoders, which are connected by an interleaver Π and a de-interleaver Π^{-1}. The two decoders exchange extrinsic information Λ^e in an iterative loop where we refer to an execution of one component decoder as a *Half Iteration* (HI) and the execution of both decoders as a *Full Iteration* (FI). Most classical hardware architectures

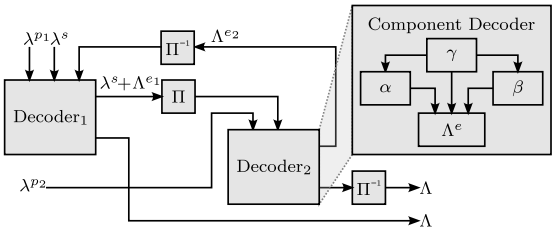

FIGURE 1. General Turbo Decoder structure.

for TD implement one decoder instance based on the max-Log-MAP algorithm (MLM) which alternately operates as Decoder$_1$ and Decoder$_2$ [5]. Similarly, the Fully Pipelined Iteration Unrolled decoders from [7]–[9] are based on the MLM as well (see also Sec. IV). However, in [11] the LSOVA algorithm was proposed as an alternative to the MLM that offers significant complexity reductions for the computation of the extrinsic information. In subsequent work, we confirmed the predicted complexity reductions with hardware implementations of its computational units [12].

A. THE MAX-LOG-MAP ALGORITHM

In the following, we will briefly recall the MLM algorithm [24]. For each step in the code trellis (see Fig. 2 (a)), the MLM algorithm computes a *forward state metric* α_i^s, *backward state metric* $\beta_{i+1}^{s'}$ and a *branch metric* $\gamma_{i,i+1}^{s,s'}$ for each state s, s' and each branch (s, s') respectively. The state metrics α and β are calculated recursively:

$$\alpha_{s'} = \max_{\forall s} (\alpha_s + \gamma_{s,s'}) \qquad \beta_s = \max_{\forall s'} (\beta_{s'} + \gamma_{s,s'}) . \tag{1}$$

From the state and branch metrics, an *a Posteriori Probability* (APP) *Logarithmic Likelihood Ratio* (LLR) Λ_i is calculated by considering the "best" branch carrying the hard decision $u = 0$ and the best branch carrying the hard decision $u = 1$:

$$\Lambda_i = \max_{\forall (s,s')} \left(\gamma_{s,s'}^{u=0} + \alpha_s + \beta_{s'}\right) - \max_{\forall (s,s')} \left(\gamma_{s,s'}^{u=1} + \alpha_s + \beta_{s'}\right) ..$$

Processing r trellis steps at the same time, referred to as *radix-*2^r decoding [25], reduces the state metric memory and the decoding latency. However, due to the increase in complexity and critical path, typically a radix $\leq$ 16 will be used for hardware implementations[1].

B. THE LOCAL-SOVA ALGORITHM

The LSOVA algorithm, proposed in [11] offers a significant reduction in complexity over the classical MLM by reformulating and re-ordering of computations. This is particularly true for the case of processing r trellis steps at the same time, referred to as *radix-*2^r decoding.

Where the MLM operates on branches in the trellis diagram, the LSOVA operates on paths stretching along k trellis steps $i, i+1, \ldots i+(k-1)$ for a radix order $R = 2^r$. A path in the trellis is defined as a 3-tuple consisting of a path metric M, a set of binary hard decisions $\mathbf{u} = \{u_i, u_{i+1}, \ldots, u_{i+(k-1)}\}$ and a set of positive reliability values $\mathbf{L} = \{L_i, L_{i+1}, \ldots, L_{i+(k-1)}\}$:

$$P = \{M, \mathbf{u}, \mathbf{L}\} \in \mathbb{R} \times \{0,1\}^k \times (\mathbb{R}^+)^k. \tag{2}$$

The path metric M along a branch (s, s') in a radix-2^r trellis, i.e. traversing r subsequent branches carrying the hard decisions $\mathbf{u}$, is composed of α and β state metrics which themselves are calculated following Eq. 1 as well as the *branch* metrics. For radix-2 we get:

$$M = \alpha_s + \gamma_{s,s'} + \beta_{s'}. \tag{3}$$

For the path structure, the Local-SOVA defines a *merge* operation, which takes as input two paths $P_a = \{M_a, \mathbf{u^a}, \mathbf{L^a}\}$ and

[1]For the hardware implementations discussed in Sec. V, we use radix-4.

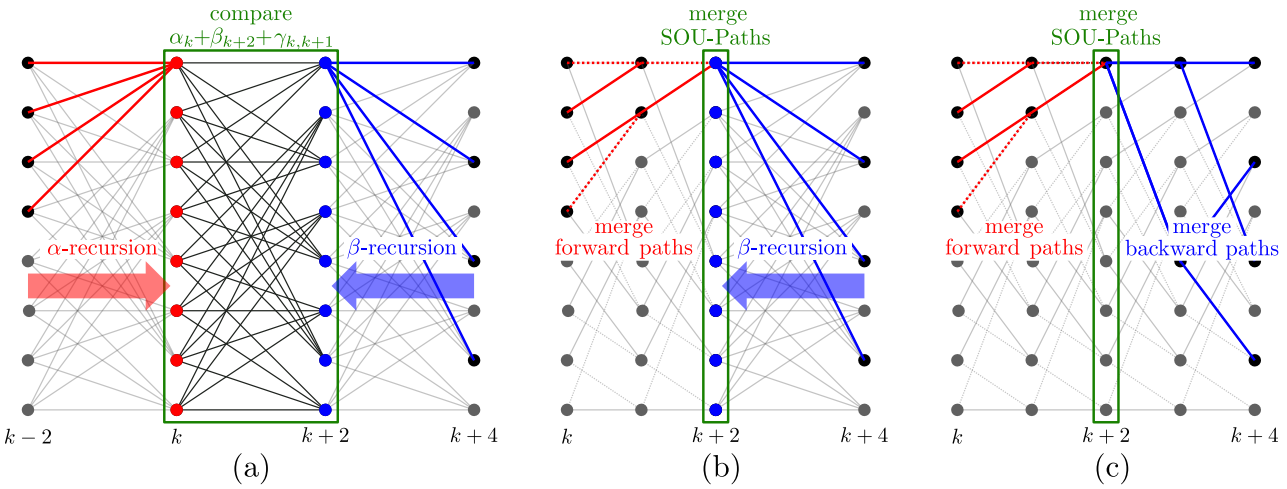

FIGURE 2. Algorithmic steps on the radix-4 code trellis for (a) MLM (b) LSOVA and (c) the proposed Dual-Sided LSOVA.

$P_b = \{M_b, \mathbf{u^b}, \mathbf{L^b}\}$ and outputs one path $P_c = \{M_c, \mathbf{u^c}, \mathbf{L^c}\}$. It relies on three functions:

$$M_c = f_0(M_a, M_b) = \max(M_a, M_b) \tag{4}$$

$$\mathbf{u^c} = f_1(\mathbf{u^a}, \mathbf{u^b}) = \begin{cases} \mathbf{u^a} & \text{, if } f_0(M_a, M_b) = M_a \\ \mathbf{u^b} & \text{, if } f_0(M_a, M_b) = M_b \end{cases} \tag{5}$$

$$\mathbf{L^c} = \left\{ L_i^c | L_i^c = f_2(L_i^a, L_i^b) \right\}. \tag{6}$$

The update of the reliability values $\mathbf{L}$ through f_2 is performed on the basis of the Hagenauer rule (HR) [26] and the Battail rule (BR) [27], [28]. Let P_a and P_b be two paths to be merged and M_p and $M_{p'}$ be defined as $M_p = \max(M_a, M_b)$ and $M_{p'} = \min(M_a, M_b)$. The metric difference between P_a and P_b is then defined as $\Delta_{p,p'} = M_p - M_{p'}$ and is always positive. Then the updates on L_i^c are performed as follows:

$$\begin{aligned} L_i^c =& f_2(L_i^a, L_i^b) \\ =& \begin{cases} \min\left(L_{p'}^i, \Delta_{p,p'}\right) & \text{, if } u_i^a \neq u_i^b \text{ (HR)} \\ \min\left(L_{p'}^i, \Delta_{p,p'} + L_p^i\right) & \text{, if } u_i^a = u_i^b \text{ (BR).} \end{cases} \end{aligned} \tag{7}$$

If all paths at a trellis section are merged, the resulting path P_c is the *maximum likelihood* (ML) path and it carries the hard decision provided by the decoder for this section. The merge operation was proven in [11] to be commutative and associative which allows merging in an arbitrary order. In particular, the merges are rearranged so that paths P_a and P_b with the same ending state s' are merged first to yield the path metric

$$\begin{aligned} M_c =& f_0\left(M_a, M_b\right) && (8) \\ =& \max\left(\alpha_{s_0} + \gamma_{s_0,s'} + \beta_{s'}, \alpha_{s_1} + \gamma_{s_1,s'} + \beta_{s'}\right) \\ =& \max\left(\alpha^{s_0} + \gamma_{s_0,s'}, \alpha_{s_1} + \gamma_{s_1,s'}\right) + \beta_{k+1}^{s'} && (9) \\ =& \alpha_{s'} + \beta_{s'}, && (10) \end{aligned}$$

The careful reordering of merges can allow for the implicit computation of the forward state metrics, as can be seen from (9). To achieve this, the computations are divided into two distinct phases. The first is the *Add-Compare-Select (ACS) phase*, which merges paths that terminate in the same trellis state. This is followed by the *Soft Output Unit (SOU) phase*, where the remaining paths, ending in different states, are further merged. It is important to note that the number of layers in the merge tree during the SOU phase depends solely on the memory length ν of the underlying convolutional code. This structure is independent of the radix order, as exactly one path per trellis state is passed forward from the ACS phase. Thus, for a memory length ν, the SOU merges require a binary tree of ν layers, leading to a significant reduction in complexity of the SOU in comparison to the MLM.

C. PROPOSED DUAL-SIDED LOCAL-SOVA

Building on the use of paths and path-merging during the computation of the backward metric, we propose reusing the same comparison tree in the SOU phase to compute both extrinsic information and hard decisions for the forward and backward paths simultaneously—effectively performing two radix-2^r steps in parallel. This approach further reduces the number of comparison operations required in the SOU phase, as only a single comparison tree needs to be traversed. We get a slightly different merge operation in forward and backward direction in the ACS phase. For even k, uptates on reliabilities and hard decisions are only performed in forward direction and for odd k in the backward direction. For the forward direction, we get:

$$M_c = f_0(M_a, M_b) \tag{11}$$

$$\mathbf{u^c} = \begin{cases} \text{forward: } f_1(\mathbf{u^a}, \mathbf{u^b}), \text{ if } k \bmod 2 = 0 \\ 0 \text{ else} \end{cases} \tag{12}$$

$$\mathbf{L^c} = \begin{cases} \text{forward: } f_2(\mathbf{L^a}, \mathbf{L^b}), \text{ if } k \bmod 2 = 0 \\ 0 \text{ else} \end{cases} \tag{13}$$

Where k refers to the *k-th* trellis step. Note, that the overall computational complexity in the ACS phase remains the same since half of the reliability values $\mathbf{L}$ and hard decisions $\mathbf{u}$ are

computed in the even steps of the forward ACS and the other half in the odd steps of the backward ACS. For the backward ACS operations, we obtain:

$$M_c = f_0(M_a, M_b) \tag{14}$$

$$\mathbf{u^c} = \begin{cases} \text{backward: } f_1(\mathbf{u^a}, \mathbf{u^b}), \text{ if } k \bmod 2 = 1 \\ 0 \text{ else} \end{cases} \tag{15}$$

$$\mathbf{L^c} = \begin{cases} \text{backward: } f_2(\mathbf{L^a}, \mathbf{L^b}), \text{ if } k \bmod 2 = 1 \\ 0 \text{ else} \end{cases} \tag{16}$$

In the SOU phase, the path metrics M of the winning forward and backward paths are first summed, and the corresponding hard decision vectors $\mathbf{u^{fw}}$ and $\mathbf{u^{bw}}$, along with the preliminary reliability values $\mathbf{L^{fw}}$ and $\mathbf{L^{bw}}$ from the forward and backward merges, are concatenated (see Fig. 2(c)). The merging of these combined paths—each spanning two radix-2^r trellis steps—is then carried out as in the LSOVA algorithm: The merges are performed in a tree of ν layers. However, in difference to the LSOVA, the amount of updates is doubled. This extended version of LSOVA is referred to as the DS-LSOVA.

III. ITERATION OVERLAP INTERLEAVERS

From a code design perspective, the interleaver plays a crucial role in determining the error-correcting performance of turbo-based forward error correction (FEC) schemes. However, the interleaving and de-interleaving operations—responsible for data reordering between the two component decoders—also introduce significant implementation challenges, particularly in terms of hardware complexity.

In architectures with a single decoder instance, the degree of parallelism must be carefully managed to prevent memory access conflicts [29]–[31]. For fully pipelined, iteration-unrolled architectures—where complete frames are processed in parallel—the interleaver imposes *precedence constraints* that directly influence the scheduling of successive HI computations.

In our prior work [22], we demonstrated that with proper interleaver design, *full computational overlap* between half-iterations is achievable: extrinsic information generated in HI i can be immediately consumed in HI $i+1$.

To enable this, we focus on Almost Regular Permutation (ARP) interleavers [31], and adopt a protograph-based design method from [10], which explicitly accounts for the structural characteristics of the UXMAP decoder.

A. PROPOSED PROTOGRAPH ARP INTERLEAVER DESIGN

An ARP interleaver is defined by a value P, which is relatively prime to the frame size K, a shift vector $\mathbf{S}$, and a disorder degree Q [31]. Given an interleaved frame index i, the corresponding index in the natural (non-interleaved) order is given by $\Pi_{\text{ARP}}(i)$, computed as:

$$\Pi_{\text{ARP}}(i) = \left(P \cdot i + S_{(i \bmod Q)}\right) \bmod K. \tag{17}$$

For a layered construction [10], the indices $\Pi_{\text{ARP}}(i)$ are divided into Q groups of K/Q bits called *layers* such that

$$\Pi_{\text{ARP}}(i+Q) \mod Q = \Pi_{\text{ARP}}(i) \mod Q. \tag{18}$$

Each layer $l = i \mod Q$ in the interleaved order is linked to a layer $l' = \Pi_{\text{ARP}}(i) \mod Q$ in the natural order. Fig. 3 (a) illustrates that for the example of $K = 16$, $P = 3$, $Q = 4$, $S = \{3, 11, 15, 4\}$. The link between the layers of the non-interleaved and the interleaved order can be expressed by a protograph as shown for the same example in Fig. 3 (b).

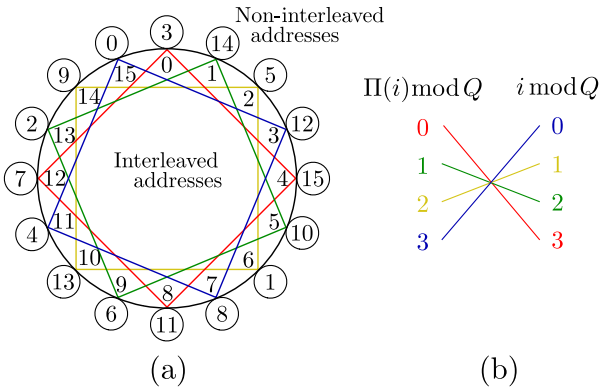

FIGURE 3. Layer connections in an ARP interleaver with $K = 16$, $P = 3$, $Q = 4$, $S = \{3, 11, 15, 4\}$.

In [10], the protograph structure was used to impose specific constraints on the interleaver design, aiming to mitigate the adverse effects introduced by puncturing. The resulting *puncture-constrained* interleavers demonstrated significantly improved performance compared to LTE turbo codes of the same frame length K.

B. FULLY UNROLLED WINDOWED SCHEDULE & OVERLAP

Figure 4 depicts the decoding schedule for a fully pipelined, iteration-unrolled decoder. In this example, two decoding windows of size $W_S = K/2$ are processed in parallel as they progress through the pipeline. Generally, the interleaving between HIs j and $j+1$ can only occur after the complete processing of HI j, due to precedence constraints imposed by the interleaver on extrinsic information exchange [22]. As a result, delay registers must be introduced to maintain pipeline alignment between successive HIs, as shown in Fig. 4(a).

To enable immediate extrinsic information exchange and fully overlap the processing of consecutive HIs j and $j+1$, we propose explicitly designing the interleaver to align the precedence constraints with the pipeline's processing schedule (Fig. 4(b)). To achieve the maximum *Overlap Depth* (OD), the extrinsic information corresponding to bit index i, denoted as Λ_i^e and produced at time slot $G_{\text{UX}}(i)$ during HI j, must be available for consumption in HI $j+1$ at or after time slot $C_{\text{UX}}(i)$:

$$G_{\text{UX}}(i) = \left\lfloor \left| \bmod(i, W_S) - (W_S - 1)/2 \right| \right\rfloor \tag{19}$$

$$C_{\text{UX}}(i) = \left\lfloor (W_S - 1)/2 \right\rfloor - G_{\text{UX}}(i), \tag{20}$$

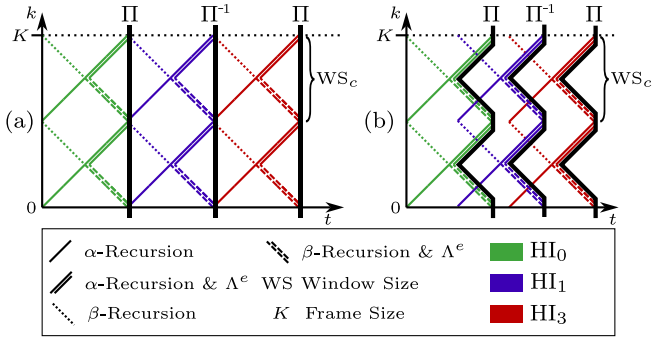

FIGURE 4. (a) Non overlapped UXMAP decoding schedule (b) UXMAP decoding schedule with full overlap.

Here, $\lfloor \cdot \rfloor$ denotes the floor function. Note that $G_{\mathrm{UX}}(i)$ and $C_{\mathrm{UX}}(i)$ are defined relative to the start of HI j and HI $j+1$, respectively. Given the values of $G_{\mathrm{UX}}(i)$ and $C_{\mathrm{UX}}(i)$, the corresponding precedence constraint can be expressed as:

$$0 \leq C_{\mathrm{UX}}(i) - G_{\mathrm{UX}}(i). \tag{21}$$

This constraint inherently depends on both the structure of the interleaver and the window size W_S.

To ensure that the precedence constraints are satisfied for all bit positions under full overlap, we propose embedding these constraints directly into the interleaver structure during code design. In particular, the ARP interleaver can be constructed to inherently satisfy these constraints when the number of layers is set equal to the window size of the UXMAP decoder, i.e., $Q = W_S$. Under this condition, for each layer $l \in \{0, \ldots, Q-1\}$ (i.e. each position within a window), the following constraint must hold:

$$0 \leq (\Pi(l) \bmod Q - l \bmod Q) \tag{22}$$

This condition ensures that the precedence constraints are respected, i.e. the extrinsic information at window position (layer) l is generated in the non-interleaved HI before it is consumed in the interleaved HI (see Eq. 21). Thus, Eq. 22 enables immediate extrinsic information exchange and full overlap between successive HIs.

As a proof of concept, we designed a set of interleavers that enable full overlap for various values of W_S by adapting the methodology proposed in [10] to comply with the constraint in (22). The corresponding parameters are summarized in Table 1. Figure 5 presents the error-rate performance, measured in terms of FER, over an AWGN channel using binary phase-shift keying (BPSK) modulation and four full decoding iterations. Results for the corresponding LTE interleavers are also included for DS-LSOVA decoding with identical configuration of frame- and window size to allow a fair comparison. To maintain clarity, only the results for MLM and DS-LSOVA decoding are shown, since the FER performance of the LSOVA for the same K/W_S configuration is comparable to the MLM [11], [12]. MLM decoding of the LTE Turbo Codes serves as baseline reference. The results in Fig. 5 are based on bit-true models of the hardware implementations described in Section V, corresponding to the various codes and decoding algorithms under consideration.

Notably, the DS-LSOVA algorithm demonstrates performance comparable to that of the MLM algorithm across all simulated scenarios. While the overlap interleavers with a window size of $W_S = 16$ (WS16) achieve error-rate performance identical to their respective LTE interleavers, the WS32 configuration exhibits a slight performance degradation of approximately 0.25 dB at a FER of 10^{-4}. This degradation is not attributed to the DS-LSOVA, but it instead arises from the additional design constraints imposed on the interleaver to satisfy the precedence conditions required for full overlap. In particular, for a relatively small interleaver size of $K = 128$ bits and a larger window size of 32, the increased regularity introduced by these constraints adversely impacts the error-correcting performance.

K	Q	P	S
128	16	79	[8, 90, 28, 126, 87, 119, 68, 39, 103, 106, 119, 87, 112, 82, 116, 70]
128	32	95	[15, 79, 15, 79, 15, 79, 15, 79, 15, 79, 15, 79, 8, 79, 15, 78, 16, 79, 15, 86, 15, 79, 15, 79, 15, 79, 15, 79, 15, 79, 15, 79]
256	16	111	[7, 117, 120, 104, 8, 120, 107, 104, 120, 117, 104, 88, 120, 104, 107, 89]

TABLE 1. Designed full overlap ARP interleaver parameters.

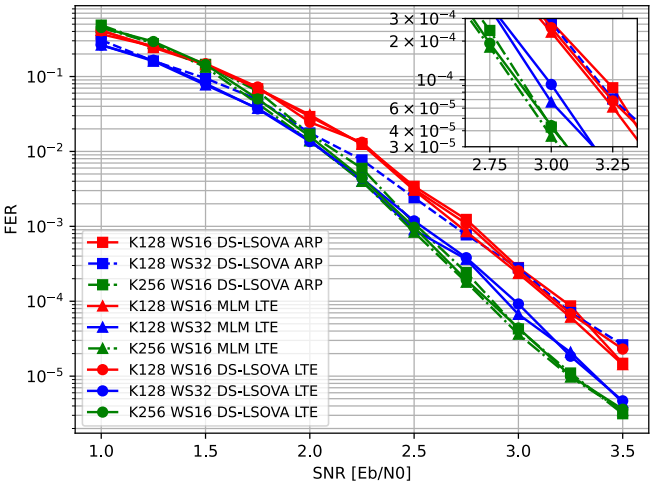

FIGURE 5. FER comparison of full iteration overlap with LTE interleavers under 4 full iterations MLM and DS-LSOVA decoding.

IV. DECODER HARDWARE ARCHITECTURES

In turbo decoder hardware architectures, both spatial and functional parallelism can be leveraged at multiple levels, giving rise to a variety of architectural archetypes [7]. The *parallel MAP* (PMAP) [32]–[35] and *pipelined MAP* (XMAP) [36]–[38] architectures implement a single decoder instance that alternates between operating as $\mathrm{Decoder}_1$ and $\mathrm{Decoder}_2$ (as illustrated in Fig. 1). In contrast, the *fully parallel MAP* (FPMAP) [39] and *fully pipelined iteration-unrolled* (UXMAP) [7] architectures represent the extreme

forms of the PMAP and XMAP approaches, respectively, offering maximum parallelism and throughput.

The FPMAP architecture further employs a shuffled decoding scheme, allowing it to process both component codes—namely, the non-interleaved and interleaved HIs—simultaneously. For each trellis step and for both component decoders, a dedicated processing element is instantiated. These elements exchange state metrics and extrinsic information with neighboring elements within the same component decoder and across to the other component decoder. FPMAP implementations have demonstrated throughput rates of up to 15 Gb/s, albeit at the cost of requiring a higher number of decoding iterations to achieve comparable error-rate performance [40], [41]. This trade-off becomes particularly pronounced for short frame sizes and highly punctured codes, where the shuffled decoding approach leads to a notable degradation in decoding performance—even when additional iterations are applied [7].

This work focuses exclusively on the Fully Pipelined Iteration-Unrolled MAP (UXMAP) decoder architecture. For detailed descriptions of the PMAP, XMAP, and FPMAP architectures, the reader is referred to [32]–[35], [36]–[38], and [39]–[41], respectively. For a detailed comparison of Turbo Decoder implementations with decoders for other code families, we refer to [42].

The fundamental UXMAP architecture is composed of multiple *Half-Iteration* pipeline stages (HI-stages), interconnected through hardwired or configurable interleaving and de-interleaving links [7], [8], as shown in Fig. 6(a). Each HI-stage may contain several *X-windows* to reduce area overhead by efficiently pipelining the channel LLRs, extrinsic information, and state metrics [9]. Figures 6(b), (c), and (d) illustrate the different X-window architectures implemented in this work, namely the MLM-based, LSOVA-based, and DS-LSOVA-based configurations.

A. NON-OVERLAPPING DECODER ARCHITECTURES

1) MLM

An MLM-based X-window without iteration overlap is used as the reference architecture, as illustrated in Fig. 6(b). The scheduling of operations—including branch metric computation, forward recursion, and backward recursion—is unrolled across the pipeline following the scheme depicted in Fig. 4(a).

Each trellis step includes dedicated forward and backward recursion units, along with a soft-output computation unit. Additionally, two branch metric units are instantiated per trellis step. This duplication allows recomputation of branch metrics from the channel values for the right half of the X-window, eliminating the need to pipeline branch metrics across the window.

The architecture is further complemented by delay pipelines to ensure proper synchronization. Specifically, the pipeline section labeled (2) in Fig. 6(b) aligns the arrival of channel LLRs and extrinsic information prior to interleaving and de-interleaving. The pipeline segment labeled (1) provides additional delay elements for the channel LLRs, extrinsic information, and state metrics, ensuring coherent data flow throughout the pipeline.

2) LSOVA

The LSOVA-based X-window, shown in Fig. 6(c), is derived from the MLM-based X-window by modifying the recursion units in the right half of the window and incorporating LSOVA-specific soft-output units. In previous work [12], we demonstrated that these modifications lead to significant area savings at the level of the computational units. Consequently, for the implementations presented in Section V, we adopt the LSOVA computational unit architectures introduced in [12].

3) DS-LSOVA

The hardware implementation of the algorithm proposed in Section IIC builds upon the computational unit architectures from [12], extending them to support the modified merge operations required by the DS-LSOVA. It is important to note that the DS-LSOVA variant proposed in this work differs from the LSOVA variants introduced in [43], which would necessitate more substantial modifications to the decoding algorithm.

The DS-LSOVA X-window architecture features an alternating structure in its forward and backward recursions, switching between LSOVA and MLM recursion units. Additionally, due to the dual-sided merge requirement—which involves combining paths from both forward and backward recursions—the pipelines labeled (1) in Fig. 6(d) carry full path information $(M, \mathbf{L}, \mathbf{u})$, as opposed to state metrics, which are used in the MLM and LSOVA architectures. Finally, the number of soft-output unit (SOU) instances is reduced by half compared to the MLM and LSOVA cases, contributing to overall area savings.

B. OVERLAPPING DECODER ARCHITECTURES

When implementing the decoder architectures shown in Figure 6 (b)-(d) for the non-overlapping case, it is necessary to introduce delays in the inputs to and outputs from the X-window. These delays ensure that the channel values and extrinsic information remain aligned with the forward and backward recursion computations. As discussed in Section III-B, such delays are generally required to account for precedence constraints arising from interleaving and de-interleaving between the half-iteration stages.

When the decoder is implemented with a carefully designed interleaver—such as those listed in Table 1—that aligns the interleaver constraints with the decoder's window size, full iteration overlap becomes possible. As a result, overlap architectures only require input delay registers for the first half iteration and output delay registers for the hard decision outputs generated in the final half iteration. For all X-windows in subsequent half iterations, these delay pipelines can be eliminated ((2) in Fig. 6 (b)-(d)). Instead, extrinsic and channel value outputs are directly interleaved or de-interleaved into the inputs of the X-windows for the next half iteration. However, the delay pipelines that store state/path

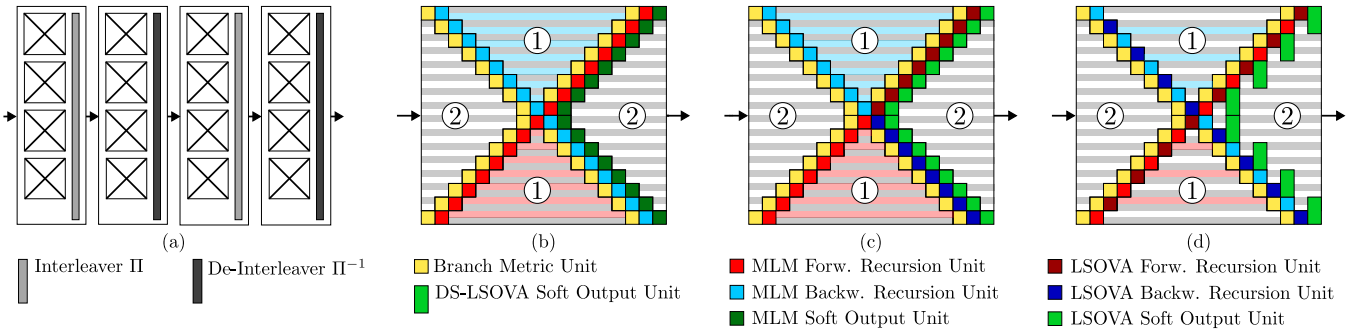

FIGURE 6. (a) Basic UXMAP architecture with Half Iteration (HI) pipeline stages and X-windows, (b) MLM-based X-window, (c) LSOVA-based X-window, (d) DS-LSOVA-based X-window.

metrics as well as channel and extrinsic values within the X-windows ((1) in Fig. 6 (b)-(d)) remain in place.

V. IMPLEMENTATION RESULTS

The hardware architectures discussed in the previous section were implemented in VHDL and synthesized using *Synopsys Design Compiler* for a 22 nm process.

	This Work	[8]
Frame Size K	128,256	128
Window Size WS	16,32	32
Spacial Parallelism	K/WS	4
Radix	4	
Algorithm	MLM, LSOVA, DS-LSOVA	MLM
Extrinsic Scaling	0.75	
State Metric Initialization	Next Iteration Initialization [44]	
Input Quantization	6 bit	
Termination	Tailbiting	

TABLE 2. Algorithmic parameters for the implementations.

The algorithmic parameters are detailed in Table 2, while the interleaver parameters are provided in Table 1. To ensure a fair comparison, the benchmark configuration architecture commonly used in previous work (frame size $K = 128$, window size $WS = 32$, [8]) has also been re-implemented. The synthesis results, in terms of core area, are presented in Table 3 for all implemented configurations. This section primarily focuses on comparing Non-overlapping and overlapping architectures, as well as evaluating the differences among MLM, LSOVA, and DS-LSOVA implementations. Additionally, Table 3 includes scaled results from [8] for reference. As shown in Table 3, the DS-LSOVA implementations outperform their MLM counterparts by up to 17%, with the most area-efficient configurations in terms of $Gb/s/mm^2$ being DS-LSOVA with $K = 128, WS = 16$ and $K = 256, WS = 16$. Furthermore, implementations with overlap achieve better performance than their non-overlap counterparts, with gains ranging from 6% ($K = 128, WS = 16$, MLM) to 11.49% ($K = 128, WS = 16$, LSOVA). Notably, while DS-LSOVA outperforms LSOVA by approximately 3% (both with and without overlap) for $WS = 16$, its performance for $WS = 32$ is slightly lower (around -0.5%), even when overlap is applied. Lastly, comparing DS-LSOVA with

		K128 / WS16		K256 / WS16		K128 / WS32	
		no ov.	ov.	no ov.	ov.	no ov.	ov.
MLM	Area [mm^2]	8.55	8.03	17.10	16.07	9.79	8.80
	TP [Gb/s]	102.4		204.8		102.4	
	Area eff. [*]	11.97	12.75	11.97	12.74	10.45	11.63
LSOVA	Area [mm^2]	7.39	6.86	14.78	13.73	8.53	7.55
	TP [Gb/s]	102.4		204.8		102.4	
	Area eff. [*]	13.85	14.94	13.85	14.91	12.00	13.56
DS-LSOVA	Area [mm^2]	7.16	6.65	14.32	13.31	8.57	7.60
	TP [Gb/s]	102.4		204.8		102.4	
	Area eff. [*]	14.30	15.39	14.30	15.38	11.94	13.47

		K128 / WS32	
		no ov.	ov
[8] (MLM)	Area [mm^2]	10.40♯	N/A
	TP [Gb/s]	102.4	
	Area eff. [*]	9.84	N/A

TABLE 3. Comparison of 22 nm synthesis results.
Area scaling to 22 nm according to [45]: ♯: 1.591

overlap to the widely used MLM without overlap reveals an overall improvement of approximately 22%.

VI. CONCLUSION

In this paper, we have presented a comprehensive synthesis of our recent advancements in code and interleaver design, decoding algorithms, and hardware architectures aimed at achieving efficient high-throughput turbo decoding. The proposed Dual-Sided Local-SOVA (DS-LSOVA) variant of the LSOVA algorithm demonstrated comparable FER performance to the MLM while offering notable architectural simplifications. Additionally, we extended the concept of interleaver design for iteration overlap decoding architectures, showing that for smaller window sizes, the proposed interleavers can achieve performance on par with the QPP interleavers of the LTE standard while significantly reducing decoding latency. Finally, we introduced hardware architectures optimized to leverage all proposed improvements, both for full iteration overlap and conventional non-overlap decoding. Synthesis results for a 22 nm technology confirm the anticipated complexity reductions, with area savings of up to 22% compared to state-of-the-art MLM decoding architectures.

While the presented work addressed the DS-LSOVA algorithm and iteration overlap interleavers in the context of fully pipelined architectures with a focus on high throughput

requirements in 5G/6G systems [14], [15], the demonstrated improvements in area efficiency are applicable to classical PMAP and XMAP decoder architectures as well. This makes the proposed decoding algorithm and interleavers well suited in the context of M2M and V2X scenarios, where frame size and code rate flexibilities may be more important than raw throughput.

ACKNOWLEDGMENT

We acknowledge the kind support by Prof. Dr.-Ing. Norbert Wehn of RPTU Kaiserslautern-Landau.

REFERENCES

[1] Third Generation Partnership Project. *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (3GPP TS 36.212 version 17.1.0 Release 17)*, April 2022.
[2] J. Lee, Y. Kim, Y. Kwak, J. Zhang, A. Papasakellariou, T. Novlan, C. Sun, and Y. Li. Lte-advanced in 3gpp rel -13/14: an evolution toward 5g. *IEEE Comm. Magazine*, 54(3):36–42, 2016.
[3] U. Singh, A. Dua, S. Tanwar, N. Kumar, and M. Alazab. A survey on lte/lte-a radio resource allocation techniques for machine-to-machine communication for b5g networks. *IEEE Access*, 9:107976–107997, 2021.
[4] H. Ullah, N. G. Nair, A. Moore, C. Nugent, P. Muschamp, and M. Cuevas. 5g communication: An overview of vehicle-to-everything, drones, and healthcare use-cases. *IEEE Access*, 7:37251–37268, 2019.
[5] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *Proc. of ICC '93 - IEEE Int. Conf. on Comm.*, volume 2, pages 1064–1070 vol.2, 1993.
[6] D. J. C. MacKay and R. M. Neal. Near shannon limit performance of low density parity check codes. *Electronics Lett.*, 33(6):457–458, March 1997.
[7] S. Weithoffer, C. Abdel Nour, N. Wehn, C. Douillard, and C. Berrou. 25 years of turbo codes: From mb/s to beyond 100 gb/s. In *2018 IEEE 10th Int. Symp. on Turbo Codes & Iterative Inf. Process. (ISTC)*, pages 1–6, 2018.
[8] S.Weithoffer, O. Griebel, R. Klaimi, C. Abdel Nour, and N. Wehn. Advanced hardware architectures for turbo code decoding beyond 100 gb/s. In *2020 IEEE Wireless Comm. and Netw. Conf. (WCNC)*, pages 1–6, 2020.
[9] S. Weithoffer, R. Klaimi, C. Abdel Nour, N. Wehn, and C. Douillard. Fully pipelined iteration unrolled decoders the road to tb/s turbo decoding. In *ICASSP 2020 - 2020 IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*, pages 5115–5119, 2020.
[10] R. Garzón-Bohórquez, C. Abdel Nour, and C. Douillard. Protograph-based interleavers for punctured turbo codes. *IEEE Trans. on Comm.*, 66(5):1833–1844, 2018.
[11] V.H.S. Le, C. Abdel Nour, E. Boutillon, and C. Douillard. Revisiting the max-log-map algorithm with sova update rules: New simplifications for high-radix siso decoders. *IEEE Trans. on Comm.*, 68(4):1991–2004, 2020.
[12] S. Weithoffer, R. Klaimi, C. Abdel Nour, N. Wehn, and C. Douillard. Low-complexity computational units for the local-sova decoding algorithm. In *2020 IEEE 31st Annual Int. Symp. on Personal, Indoor and Mobile Radio Comm.*, pages 1–6, 2020.
[13] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate (corresp.). *IEEE Trans. on Inf. Theory*, 20(2):284–287, 1974.
[14] Sisi Miao, Claus Kestel, Lucas Johannsen, Marvin Geiselhart, Laurent Schmalen, Alexios Balatsoukas-Stimming, Gianluigi Liva, Norbert Wehn, and Stephan Ten Brink. Trends in channel coding for 6g. *Proceedings of the IEEE*, 112(7):653–675, 2024.
[15] Mohammad Rowshan, Min Qiu, Yixuan Xie, Xinyi Gu, and Jinhong Yuan. Channel coding toward 6g: Technical overview and outlook. *IEEE Open Journal of the Communications Society*, 5:2585–2685, 2024.
[16] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Muller, T. Elste, and M. Windisch. Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture. *IEEE Comm. Magazine*, 55(2):70–78, 2017.
[17] J. Zhang and M. P. C. Fossorier. Shuffled iterative decoding. *IEEE Trans. on Comm.*, 53(2):209–213, 2005.
[18] O. Muller, A. Baghdadi, and M. Jezequel. Exploring parallel process. levels for convolutional turbo decoding. In *2006 2nd Int. Conf. on Inf. & Commun. Techn.*, volume 2, pages 2353–2358, 2006.
[19] P. Radosavljevic, A. de Baynast, and J. R. Cavallaro. Optimized message passing schedules for ldpc decoding. In *Conf. Record of the Thirty-Ninth Asilomar Conf. on Signals, Syst. and Computers, 2005.*, pages 591–595, 2005.
[20] E. Sharon, S. Litsyn, and J. Goldberger. An efficient message-passing schedule for ldpc decoding. In *2004 23rd IEEE Convention of Electrical and Electronics Engineers in Israel*, pages 223–226, 2004.
[21] V. Petrović L., M. M. Marković, D. M. El Mezeni, L. V. Saranovac, and A. Radošević. Flexible high throughput qc-ldpc decoder with perfect pipeline conflicts resolution and efficient hardware utilization. *IEEE Trans. on Circuits and Syst. I: Regular Papers*, 67(12):5454–5467, 2020.
[22] S. Weithoffer, G. Aousaji, J. Nadal, and C. Abdel Nour. Iteration overlap for low-latency turbo decoding. In *2023 12th Int. Symp. on Topics in Coding (ISTC)*, pages 1–5, 2023.
[23] J. Nadal, S. Weithoffer, C. Abdel Nour, and C. Douillard. Advanced iteration overlap for low-latency turbo decoding. In *2024 19th Int. Symp. on Wireless Commun. Syst. (ISWCS)*, pages 1–6, 2024.
[24] P. Robertson, E. Villebrun, and P. Hoeher. A comparison of optimal and sub-optimal map decoding algorithms operating in the log domain. In *Proc. IEEE Int. Conf. on Comm. ICC '95*, volume 2, pages 1009–1013 vol.2, 1995.
[25] G. Fettweis and H. Meyr. Parallel viterbi algorithm implementation: breaking the acs-bottleneck. *IEEE Trans. on Comm.*, 37(8):785–790, 1989.
[26] J. Hagenauer and P. Hoeher. A viterbi algorithm with soft-decision outputs and its applications. In *1989 IEEE Global Telecommunications Conf. and Exhibition 'Communications Technology for the 1990s and Beyond'*, pages 1680–1686 vol.3, 1989.
[27] G. Battail. Pondération des symboles décodés par l'algorithme de viterbi. *Annales des telecommunications*, 42(1-2):31–38, 1987.
[28] L. Lin and R. S. Cheng. Improvements in sova-based decoding for turbo codes. In *Proc. of ICC'97 - Int. Conf. on Comm.*, volume 3, pages 1473–1478 vol.3, 1997.
[29] J. Sun and O. Y. Takeshita. Interleavers for turbo codes using permutation polynomials over integer rings. *IEEE Trans. on Inf. Theory*, 51(1):101–119, 2005.
[30] S. Crozier and P. Guinand. Distance upper bounds and true minimum distance results for turbo-codes designed with drp interleavers. In *Proc. 3rd Int. Symp. on Turbo Codes & Related Topics*, Brest, France, 2003.
[31] C. Berrou, Y. Saouter, C. Douillard, S. Kerouedan, and M. Jezequel. Designing good permutations for turbo codes: towards a single model. In *2004 IEEE Int. Conf. on Comm. (IEEE Cat. No.04CH37577)*, volume 1, pages 341–345, 2004.
[32] E. Boutillon, W. J. Gross, and P. G. Gulak. Vlsi architectures for the map algorithm. *IEEE Trans. on Comm.*, 51(2):175–185, 2003.
[33] T. Ilnseher, F. Kienle, C. Weis, and N. Wehn. A 2.15gbit/s turbo code decoder for lte advanced base station applications. In *2012 7th Int. Symp. on Turbo Codes and Iterative Inf. Process. (ISTC)*, pages 21–25, 2012.
[34] C. Roth, S. Belfanti, C. Benkeser, and Q. Huang. Efficient parallel turbo-decoding for high-throughput wireless systems. *IEEE Trans. on Circuits and Syst. I: Regular Papers*, 61(6):1824–1835, 2014.
[35] S. Favero, M. Martina, and G. Masera. Architectural comparison model for area-efficient pmap turbo-decoders. *IEEE Trans. on Circuits and Syst. II: Express Briefs*, 70(1):131–135, 2023.
[36] A. Worm, H. Lamm, and N. Wehn. Design of low-power high-speed maximum a priori decoder architectures. In *Proc. Design, Automation and Test in Europe. Conf. and Exhibition 2001*, pages 258–265, 2001.
[37] S. Weithoffer, F. Pohl, and N. Wehn. On the applicability of trellis compression to turbo-code decoder hardware architectures. In *2016 9th Int. Symp. on Turbo Codes and Iterative Inf. Process. (ISTC)*, pages 61–65, 2016.
[38] M. May, T. Ilnseher, N. Wehn, and W. Raab. A 150mbit/s 3gpp lte turbo code decoder. In *2010 Design, Automation & Test in Europe Conf. & Exhibition (DATE 2010)*, pages 1420–1425, 2010.
[39] R. G. Maunder. A fully-parallel turbo decoding algorithm. *IEEE Trans. on Comm.*, 63(8):2762–2775, 2015.
[40] A. Li, P. Hailes, Robert G. Maunder, Bashir M. Al-Hashimi, and L. Hanzo. 1.5 gbit/s fpga implementation of a fully-parallel turbo decoder designed for mission-critical machine-type communication applications. *IEEE Access*, 4:5452–5473, 2016.
[41] A. Li, L. Xiang, T. Chen, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo. Vlsi implementation of fully parallel lte turbo decoders. *IEEE Access*, 4:323–346, 2016.
[42] Shuai Shao, Peter Hailes, Tsang-Yi Wang, Jwo-Yuh Wu, Robert G. Maunder, Bashir M. Al-Hashimi, and Lajos Hanzo. Survey of turbo, ldpc,

and polar decoder asic implementations. *IEEE Communications Surveys Tutorials*, 21(3):2309–2333, 2019.

[43] R. Klaimi, S. Weithoffer, C. Abdel Nour, and C. Douillard. Simplified recursion units for max-log-map: New trade-offs through variants of local-sova. In *2021 11th Int. Symp. on Topics in Coding (ISTC)*, pages 1–5, 2021.

[44] J. Dielissen and J. Huiskens. State vector reduction for initialization of sliding windows map. In *Int. Symp. on Turbo codes and Iter. Process. (ISTC)*, pages 387–390, Brest, France, September 2000.

[45] S. Sarangi and B. Baas. Deepscaletool: A tool for the accurate estimation of technology scaling in the deep-submicron era. In *2021 IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, pages 1–5, 2021.

STEFAN WEITHOFFER (Member, IEEE) received the Ph.D. degree from the University of Kaiserslautern, Germany, in 2018, where he worked on implementation issues of high-throughput Turbo Decoding. He has designed several turbo decoder IP in the course of collaborations with academia as well as industry which have been successfully integrated and manufactured in systems-on-chip (SoC) on 28 nm.

Since September 2019, he has been an Associate Professor with the MEE Department, IMT Atlantique.

CHARBEL ABDEL NOUR (Senior Member, IEEE) received the computer and communications engineering degree from Lebanese University, Beirut, Lebanon, in 2002, the master's degree in digital communications from the University of Valenciennes, Valenciennes, France, in 2003, the Ph.D. degree in digital communications from Telecom Bretagne, Brest, France, in 2008, and the Habilitation thesis from the University of Southern Brittany, Lorient, France, in 2020. From November 2011 to November 2022, he was an Associate Professor with the Electronics Department, IMT Atlantique, Brest. He was involved in several research projects related to broadcasting and satellite communications and was active in the Digital Video Broadcasting DVB consortium where he had important contributions.

Since December 2022, he is currently a Professor with the Mathematical and Electrical Engineering Department, IMT Atlantique. His interests concern the radio mobile communications systems, broadcasting systems, coded modulations, error correcting codes, resource and power allocation for NOMA, waveform design, MIMO, and iterative receivers. He presented several contributions to the H2020 METIS, FANTASTIC5G, and EPIC projects and to the 3GPP consortium related to coding solutions for 5G.

In 2023 he served as co-chair of the technical program committee of International Symposium on Topics in Coding (ISTC).