Arbitrarily Parallel Turbo Decoding for Ultra-Reliable Low Latency Communication in 3GPP LTE

Luping Xiang, Matthew Brejza, Robert G. Maunder, *Senior Member, IEEE*, Bashir M. Al-Hashimi, *Fellow, IEEE* and Lajos Hanzo, *Fellow, IEEE*

***Abstract*—In order to meet the latency requirements of the Ultra-Reliable Low Latency Communication (URLLC) mode of the 3GPP Long Term Evolution (LTE) mobile communication standard, this paper proposes a novel turbo decoding algorithm that supports an arbitrarily-high degree of parallel processing, facilitating significantly higher processing throughputs and substantially lower processing latencies than the state-of-the-art (SOTA) LTE turbo decoder. As in conventional turbo decoding algorithms, the proposed Arbitrarily Parallel Turbo Decoder (APTD) decomposes each frame of information bits into a sequence of windows, where the bits within different windows are processed simultaneously using forward and backward recursions in a serial manner. However, in contrast to conventional turbo decoding algorithms, the APTD does not require the different windows to be composed of an identical number of bits, which allows the use of an arbitrary number of windows and hence an arbitrary degree of parallelism, when decoding information bits of an arbitrary frame length. Furthermore, conventional turbo decoding algorithms alternate between simultaneously processing the windows in the upper decoder and those in the lower decoder. By contrast, the APTD processes the odd-indexed windows in the upper decoder at the same time as the even-indexed windows in the lower decoder and alternates between this and the reversed arrangement, hence further improving the decoding throughput and latency. Furthermore, the APTD achieves a reduced hardware resource requirement by calculating the extrinsic information based only on the outputs of the forward recursions, rather than based on both the forward and backward recursions of conventional turbo decoding algorithms. We demonstrate that the proposed APTD achieves superior latency, throughput and computational efficiency than the SOTA LTE turbo decoder at all frame lengths, but particularly at the short frame lengths that are typically used in URLLC approaches. For example, at a frame length of $N = 504$ bits, the proposed APTD achieves an FER of 10^{-5} at the same E_b/N_0 as $I = 8$ iterations of a conventional turbo decoder, but with a computational efficiency that is 6 times higher than that of the SOTA turbo decoder, while achieving a latency and throughput that are 0.7 and 1.4 times those of the SOTA decoder, respectively.**

***Index Terms*—Turbo decoding, FPTD, parallel algorithms, latency, throughput**

I. Introduction

As one of three pillars of the fifth generation (5G) of mobile communication, Ultra-Reliable Low Latency Communication (URLLC) targets a Frame Error Rate (FER) below 10^{-5} with an end-to-end latency within 1 ms. The initial roll-out of 5G will be based on the 3rd Generation Partnership Project's (3GPP) non-standalone New Radio (NR), which is built upon a foundation offered by 3GPP Long Term Evolution (LTE) [1]. Therefore, the 3GPP is currently standardising a URLLC mode for the LTE standard for mobile communication as part of the 5G development [2–5]. Several enhancements have been proposed in LTE Release 15, aiming for meeting the latency and reliability requirements [2]. To be more specific, LTE Release 15 has introduced the shortened Transmission Time Interval (sTTI) technique, which imposes a 7-fold reduction upon the signal processing time available at the User Equipment (UE) and evolved Node B (eNB) basestation, namely a reduction from 3 ms in Release 14, to 0.43 ms in Release 15 [6, 7]. Within this 0.43 ms, several signal processing tasks must be completed, including receive buffering, FFT, turbo decoding, IFFT and transmit power control [4, 5]. In order for an eNB to support the processing of multiple users' transmissions within this processing time without employing a unique, user-specific set of signal processing hardware per user, each of these tasks must be completed in much less than 0.43 ms. In the case of turbo decoding, it is reasonable to assume that a 7-fold reduction from the 52 μs latency of the commercial implementations of 3GPP Release 14 [8] is required, giving a specification of 7.4 μs for 3GPP Release 15. This significant reduction in the turbo decoding latency requirement motivates the design of new low-latency turbo decoding techniques.

The latency of a turbo decoder is dictated by the data dependencies imposed by the Logarithmic Bahl-Cocke-Jelinek-Raviv (Log-BCJR) algorithm [9], which is typically employed for the iterative decoding of turbo codes [10–12]. More specifically, a turbo decoder comprises an upper-branch and a lower-branch convolutional decoder, which may operate alternately using the Log-BCJR to generate extrinsic information exchanged via an interleaver. However, data dependencies of the Log-BCJR algorithm require the processing to be performed one step at a time, using forward and backward recursions alternately along the entire length of the interleaved information frame. This serial nature of the Log-BCJR algorithm imposes a bottleneck upon the throughput and results in processing latencies of hundreds of micro seconds of the turbo decoding process [9], unless parallel processing techniques are employed.

Several sophisticated approaches have been proposed for improving both the throughput and latency of the Log-BCJR

The authors are with Electronics and Computer Science, University of Southampton, SO17 1BJ, United Kingdom, e-mail: lx1g15@soton.ac.uk. The research data for this paper is available at https://github.com/lupingX/researchData.

turbo decoder. For example, the forward and backward recursions of the Log-BCJR can be completed simultaneously [13], rather than one after the other, hence doubling the throughput and halving the latency. Furthermore, the Radix-4 transform of [14, 15] allows two steps of the Log-BCJR algorithm's forward and backward recursions to be completed at a time, therefore further doubling the throughput and halving the latency, albeit at the cost of significantly increasing the hardware resource requirements. Meanwhile, the non-sliding window based technique of [16] decomposes each information frame of N bits into P number of windows, which can be processed simultaneously using separate parallel processors. However, the combination of these techniques requires as many as $4P$ extrinsic values to be interleaved at a time. The interleaver has to simultaneously read and write these extrinsic values from and to memories associated with the parallel processors, causing contentions when attempting to make more than one access to a particular memory at any instant. This contention problem is solved in the LTE standards by the employment of the quadratic permutation polynomial interleavers of [17, 18], which inherently avoid contention provided that the P number of windows in each of the upper and lower decoder is an integer factor of the information frame length N, hence ensuring that each window is composed of an equal number of N/P bits. Motivated by this, conventional implementations of the LTE turbo decoder [8] typically employ a parallelism of $P = 8$, since this is the Greatest Common Divisor (GCD) of the LTE turbo code's 188 supported frame lengths N, which are in the range 40 to 6144 bits. It is using the combination of techniques described above, that the commercial Field-Programmable Gate Array (FPGA) implementation of the LTE turbo decoder of [8] achieves processing latencies of up to 52 μs.

In order to achieve the 7-fold improvement to turbo decoding latency required for URLLC, the parallelism may be increased to $P = 64$, as in the state-of-the-art (SOTA) turbo decoder of [16]. However, this decoder is only able to fully exploit this parallelism for the longest frame lengths and disables up to 56 of the parallel processors at the short frame lengths that are common in URLLC applications, leading to poor hardware efficiency.

Recently, a Fully-Parallel Turbo Decoder (FPTD) has been proposed for significantly increasing the grade of parallelism in the LTE turbo decoder to $p = N$, hence reducing the latency [19]. The employment of odd-even interleavers [20, 21] in LTE allows the FPTD to alternate between processing all odd-indexed bits in the upper decoder at the same time as all even-indexed bits in the lower decoder, and vice-versa, hence completing each decoding iteration in two clock cycles. However, the hardware requirement of the FPTD is dictated by the longest supported frame length, leading to poor hardware efficiency at short frame lengths.

The low hardware efficiency of SOTA non-sliding window based techniques [16] and the FPTD [19] motivates the novel Arbitrarily Parallel Turbo Decoder (APTD) concept we propose in this paper for achieving both high processing throughputs and low processing latencies, while maintaining a high grade of hardware efficiency flexibility across all frame lengths.

- First, our APTD algorithm allows the parallel operation of an arbitrary number P of processors, which is not limited to an integer factor of N, nor to N itself. The first of two versions of the proposed APTD decomposes the frame into the highest possible number of windows that can support equal window lengths, where some processors are disabled as in the non-sliding window based and FPTD algorithms, respectively, when processing a frame length of N. For many frame lengths, this facilitates a grade of parallelism in excess of 64, which is the highest achieved previously in the literature owing to the requirement for P having to be a common divisor of many supported frame lengths N. In this first version of the APTD, the processors operate on the basis of windows of equal length, benefitting from the contention-free property of the LTE quadratic permutation polynomial interleaver. However, in a second version of the APTD, all processors are activated for all but the shortest of frame lengths. Here, the different processors operate on windows of slightly different lengths for frame lengths N. This breaks the contention-free property of the LTE quadratic permutation polynomial interleaver [17, 18], but we show that contention can be avoided by rescheduling the interleaver operation independently from the generation of extrinsic information, hence achieving even higher throughputs.
- Furthermore, rather than operating the upper and lower decoder alternately like in conventional turbo decoding algorithms, the arbitrarily parallel turbo decoding algorithm employs odd-even operation, in a similar fashion to the FPTD. More specifically, the APTD alternates between processing the odd-indexed windows of the upper decoder at the same time as the even-indexed windows of the lower decoder, and vice versa. We will show that for short frame lengths, this odd-even operation gives superior FER performance compared to the upper-lower operation.
- As a further contribution, we conceive and compare two techniques for providing systematic information to the APTD, namely the interleaved, and the non-interleaved systematic approaches. In the interleaved systematic approach, the systematic information is entered into the upper decoder, but additionally it is also interleaved and entered to the lower decoder. By contrast, in the non-interleaved systematic approach, the systematic information is only entered directly into the upper decoder, but it is also added into the upper decoder's extrinsic information, which is then interleaved and entered into the lower decoder. The benefit of the latter is that this reduces the number of interleavers that are required at the cost of a slightly degraded FER performance.
- Finally, while conventional turbo decoders compute their extrinsic information in both the forward and backward recursions of the Log-BCJR algorithm in the proposed design, the extrinsic information is calculated only in the forward recursions of the APTD. We will show that this further reduces the proposed APTD's decoding

complexity and its interleaving complexity, albeit at the cost of requiring slightly more decoding iterations to maintain the same FER performance.

When combining the proposed techniques described above, the proposed APTD achieves superior latency, throughput and computational efficiency than the SOTA LTE turbo decoder at all frame lengths, but particularly at the short frame lengths that are typically used in URLLC approaches. For example, at a frame length of $N = 504$ bits, the proposed APTD achieves an FER of 10^{-5} at the same E_b/N_0 as $I = 8$ iterations of a conventional turbo decoder, but with a computational efficiency that is 6 times higher than that of the SOTA turbo decoder, while achieving a latency and throughput that are 0.7 and 1.4 times those of the SOTA decoder, respectively. Note however that this is achieved at the cost of increasing the computational complexity by 2.3 times compared to the SOTA decoder of $N = 504$.

The proposed APTD algorithm offers particular benefits at short frame lengths. This is because the conventional approach to parallel processing is limited by the greatest common divisor of the QPP or ARP interleaver lengths, which tends to be low for short frame lengths. For example, the information frame lengths supported by the LTE turbo code in the range of 40 to 512 bits are all multiples of 8 bits, limiting the degree of parallel processing to 8 in conventional implementations. By contrast, the proposed APTD can usefully apply significantly higher degrees of parallel processing, leading to significant improvements in throughput and latency.

Furthermore, in [22], a 6144-bit FPTD was found to occupy a chip area of 109 mm^2, when using TSMC 65 nm process technology. However, this design has only limited flexibility to support shorter block lengths and it may be expected that a significantly greater chip area would be required to support all 188 block lengths of the LTE turbo code. Hence, the chip area requirements of the FPTD may be deemed excessive for LTE applications. In contrast to this, the proposed APTD allows the degree of parallelism to be arbitrarily flexible between fully-serial and fully-parallel turbo decoding. In this way, the parallelism of the proposed APTD may be carefully selected in order to meet the strict latency and throughput requirements of URLLC LTE, but with minimum chip area.

The rest of this paper is structured as follows. Section II gives a brief overview of LTE turbo decoding, including both the SOTA turbo decoder, also known as the non-sliding window decoders, and of the FPTD. Following this, both versions of the APTD are proposed in Section III. Section IV characterises the FER performance associated with the techniques employed by the APTD, both individually and in combination. Then the SOTA LTE turbo decoder, the FPTD and the proposed APTD are compared in terms of latency, throughput, complexity and hardware resource requirement in Section V. Finally, our conclusions and future work ideas are discussed in Section VI.

II. LTE TURBO DECODING OVERVIEW

This section provides an overview of turbo decoding and defines the notation employed in the following sections. Section II-A describes the decoding process of the SOTA LTE turbo decoder, while our previously proposed FTPD is highlighted in Section II-B.

A. State-of-the-art LTE turbo decoder

In an LTE transmitter, each frame of information bits has one of 188 legitimate lengths N in the range of 40 to 6144 bits, which are turbo encoded before being modulated and transmitted over the wireless channel. After demodulation in the receiver, soft information pertaining for the turbo encoded bits is provided to the turbo decoder in the form of Logarithmic Likelihood Ratios (LLRs). Here, we define the LLR $\bar{b}$ pertaining to a bit $b \in \{0, 1\}$ as

$$\bar{b} = \ln \frac{\Pr(b = 1|\mathbf{y})}{\Pr(b = 0|\mathbf{y})}, \tag{1}$$

where $\mathbf{y}$ is the received signal vector.

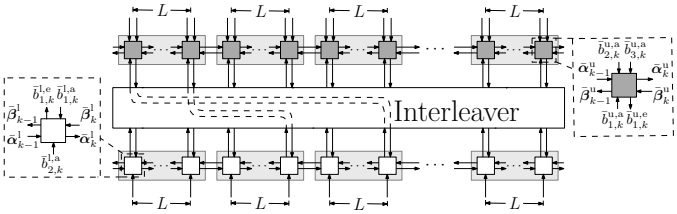


Fig. 1. Schematic of the SOTA turbo decoder.

As depicted in Figure 1, the upper and lower decoder each comprises a row of N algorithmic blocks. Each block in the upper decoder is associated with the corresponding one of the parity *a priori* LLRs $[\bar{b}^{\mathrm{u,a}}_{2,k}]^N_{k=1}$ and the corresponding one of the systematic *a priori* LLRs $[\bar{b}^{\mathrm{u,a}}_{3,k}]^N_{k=1}$, which are provided by the demodulator. Likewise, the demodulator provides each block in the lower decoder with the corresponding one of the parity *a priori* LLRs $[\bar{b}^{\mathrm{l,a}}_{2,k}]^N_{k=1}$. Note that the demodulator also provides LLRs pertaining to twelve termination bits [23], although we do not discuss these further for the sake of simplicity.

The upper and lower decoders operate alternately, where each algorithmic block generates the corresponding one of the extrinsic LLRs $[\bar{b}^{\mathrm{u,e}}_{1,k}]^N_{k=1}$ or $[\bar{b}^{\mathrm{l,e}}_{1,k}]^N_{k=1}$, which are interleaved to provide *a priori* LLRs $[\bar{b}^{\mathrm{u,a}}_{1,k}]^N_{k=1}$ or $[\bar{b}^{\mathrm{l,a}}_{1,k}]^N_{k=1}$ for the next operation of the other decoder. All algorithmic blocks in Figure 1 operate in an identical manner, as detailed in [19]. The SOTA turbo decoder performs this processing by activating P parallel processors, where each alternates between processing the corresponding window of $L = N/P$ algorithmic blocks in the upper and lower decoder, as shown in Figure 1. For example, the first processor performs the processing for the first L algorithmic blocks of the upper decoder, then performs the processing for the first L algorithmic blocks of the lower decoder, before repeating the process in each successive decoding iteration. Here, we define the time required for one of the algorithmic blocks as one clock cycle. Although the SOTA LTE turbo decoder comprises $p = 64$ parallel processors in

total, different numbers P of these p processors are activated for different frame lengths N, according to [16].

$$P = \begin{cases} 8, & N \in [40, 48, 56, ..., 504]; \\ 16, & N \in [512, 528, 544, ..., 1008]; \\ 32, & N \in [1024, 1056, 1088, ..., 2016]; \\ 64, & N \in [2048, 2112, 2176, ..., 6144]. \end{cases} \quad (2)$$

Note that some of the $p = 64$ parallel processors are unused when shorter frame lengths are decoded, in order to ensure that all windows have an equal length L. This is necessary for exploiting the contention-free property of the LTE interleaver [1, 17]. More specifically, if all windows have an equal length, their processing can then be scheduled so that none of the parallel processor is provided with more than one *a priori* LLR at a time by the LTE interleaver. This allows the processors to have simple input/output interfaces.

In addition to the LLRs described above, each processor is provided with a vector of forward state metrics $\bar{\boldsymbol{\alpha}}^{\mathrm{u}}_{k-1}$ or $\bar{\boldsymbol{\alpha}}^{\mathrm{l}}_{k-1}$ as well as a vector of backward state metrics $\bar{\boldsymbol{\beta}}^{\mathrm{u}}_{k}$ or $\bar{\boldsymbol{\beta}}^{\mathrm{l}}_{k}$, which are used for initiating forward and backward processing recursions, respectively. The scheduling of these recursions over several successive clock cycles is depicted in Figure 2 for the non-sliding window technique of [16]. During the forward recursion, each successive algorithmic block in a left-to-right ordering is processed in each successive clock cycle. Simultaneously, the backward recursion processes the successive algorithmic blocks in a right-to-left ordering in the successive clock cycles. Each successive algorithmic block passes a vector of state metrics $\bar{\boldsymbol{\alpha}}^{\mathrm{u}}_{k}$, $\bar{\boldsymbol{\alpha}}^{\mathrm{l}}_{k}$, $\bar{\boldsymbol{\beta}}^{\mathrm{u}}_{k-1}$ or $\bar{\boldsymbol{\beta}}^{\mathrm{l}}_{k-1}$ values to the next algorithmic block in the forward or backward recursion, respectively. The requirement to exchange these state metrics between successive algorithmic blocks imposes data dependencies, which requires the forward and backward recursions. Furthermore, once the two recursions have crossed over, as shown in Figure 2, the algorithmic blocks generate the extrinsic LLRs mentioned above, based on both the forward and the backward state metrics. Each iteration comprises a total of $2N/P$ clock cycles, with N/P clock cycles used by each of the upper and lower decoders. Here, 8 iterations may be required in order to achieve iterative decoding convergence towards the best possible FER, corresponding to hundreds or thousands of clock cycles.

The Radix-4 [14, 15] further improves both the throughput and latency of the SOTA LTE turbo decoder, by merging the trellis stages within each pair of adjacent algorithmic blocks and processing them at the same time, as shown in Figure 3 (b). Therefore, in contrast to the Radix-2 operation depicted in Figure 3 (a), where *two* extrinsic LLRs are calculated simultaneously once the forward and backward recursions have crossed over, the non-sliding window technique calculates *four* LLRs at a time once the forward and backward recursions have crossed over, when we employ Radix-4 technique. In this way, the Radix-4 technique further doubles the throughput and halves the latency, although at the cost of significantly increasing the hardware resource requirement.

As shown in Figure 4 (a), the extrinsic LLRs $\bar{b}^{\mathrm{u,e}}_{1,k}$ and $\bar{b}^{\mathrm{l,e}}_{1,k}$ may be scaled in order to improve the FER when employing the Max-Log-MAP algorithm [22, 24]. Here, typically a

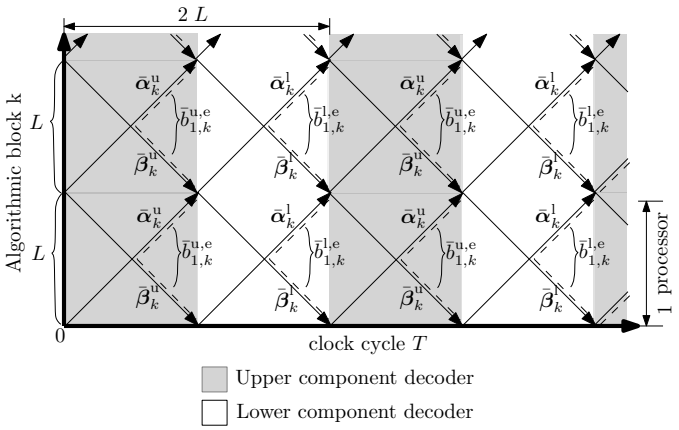


Fig. 2. Schedule of the forward and backward recursions for the non-sliding window technique, which uses equal window lengths, upper-lower processing Radix-4 operation, and calculates four extrinsic LLRs at a time once the forward and backward recursions have crossed over.

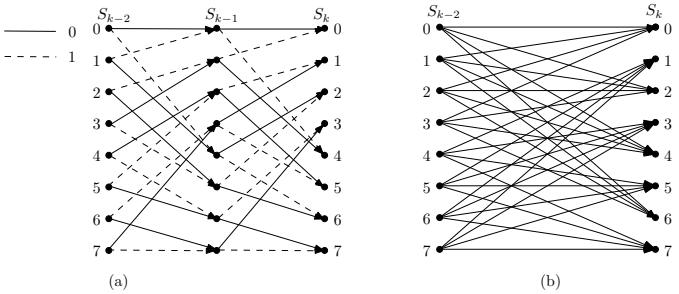


Fig. 3. Trellis diagram of (a) Radix-2 computation, and (b) Radix-4 computation.

scaling factor of $f = 0.75$ is selected, owing to its ease of implementation when using fixed point numbers to represent the LLRs. Figure 4 (a) also shows that the upper decoder can benefit from the systematic LLRs $\bar{b}^{\mathrm{u,a}}_{3,k}$ by adding them into the *a priori* LLRs $\bar{b}^{\mathrm{u,a}}_{1,k}$. A similar approach can be used for ensuring that the lower decoder can benefit from these systematic LLRs. However, since the upper decoder's operations are processed before the lower decoder in the SOTA turbo decoder, the systematic LLRs $\bar{b}^{\mathrm{u,a}}_{3,k}$ can be delivered to the lower decoder by adding them into the upper decoder's extrinsic LLRs $\bar{b}^{\mathrm{u,e}}_{1,k}$, which are interleaved to become the lower decoder's *a priori* LLRs $\bar{b}^{\mathrm{l,a}}_{1,k}$. This avoids the hardware requirement to employ a separate interleaver to interleave the systematic LLRs $\bar{b}^{\mathrm{u,e}}_{3,k}$, so that they can be added to the *a priori* LLRs of the lower decoder $\bar{b}^{\mathrm{l,a}}_{1,k}$, as shown in Figure 4 (b).

B. Fully-parallel turbo decoder

In the SOTA LTE turbo decoder of [16], the data dependencies of the forward and backward recursions require the turbo-encoded bits to be processed serially, spread over numerous consecutive clock cycles. As a result, hundreds or thousands of clock cycles are required for completing the iterative decoding process, hence limiting the achievable processing throughput and latency. In order to address this problem, we have previously proposed a FPTD algorithm [19], which dramatically increases the grade of parallelism in

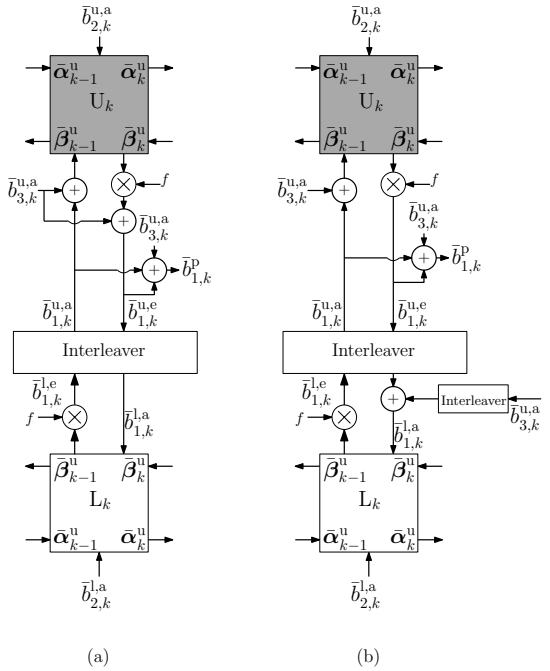


Fig. 4. (a) Non-interleaved systematic LLRs. (b) Interleaved systematic LLRs.

the decoding process, which is achieved by dispensing both with the recursions of the Log-BCJR algorithm and with the associated data dependencies, allowing a much higher number of parallel processors to be used. Indeed, the FPTD algorithm is capable of processing all LLRs corresponding to both the upper and lower decoders at the same time.

While a dedicated processor could be employed for each of the $2N$ algorithmic blocks, our previous work demonstrated that the same processing throughput and latency can be achieved using just N parallel processors. More specifically, in contrast to the alternated operation of the upper and lower decoders in the non-sliding window technique of the SOTA turbo decoder, the FPTD exploits the odd-even property of the LTE interleaver to enable an odd-even processing schedule for the algorithmic blocks. To be more specific, the LTE interleaver of all 188 supported frame lengths only connects the algorithmic blocks of the upper decoder having an odd index k to algorithmic blocks in the lower decoder that also have an odd index k. Likewise, even-indexed algorithmic blocks in the upper decoder are only connected to algorithmic blocks with even indices in the lower decoder. As a result, the algorithmic blocks can be grouped into two sets, where no two blocks in the same set have connections to each other. To be more explicit, the first set consists of all the odd-indexed blocks in the upper decoder, along with all even-indexed blocks in the lower decoder. Likewise, the second set is composed of the remaining blocks, namely of the even-indexed blocks in the upper decoder, together with the odd-indexed blocks in the lower decoder. Accordingly, the exchange of extrinsic LLRs and state metrics among the whole set of $2N$ algorithmic blocks in the iterative decoding process can now be considered as an iterative exchange process between the two sets. This allows the processing of the $2N$ algorithmic blocks to be mapped onto N processors, which alternate between the processing of the two sets. In this odd-even FPTD operation, each iteration requires only two clock cycles, but more iterations are needed than by the conventional Log-BCJR decoders in order to achieve the same FER. However, the total number of clock cycles required in order to achieve iterative decoding convergence decreases from the hundreds or thousands in the SOTA LTE turbo decoder to just tens of clock cycles.

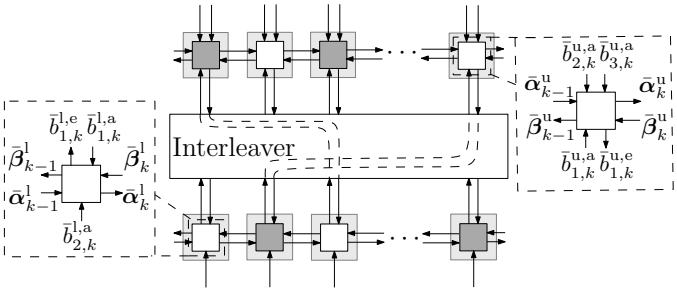


Fig. 5. Schematic of the FPTD.

III. Arbitrarily parallel Turbo Decoder

As discussed in Section II, both the SOTA turbo decoding algorithm and the FPTD algorithm have their own restrictions. To be more specific, a large number of processors remain idle when decoding short frame lengths N in the SOTA turbo decoder. Meanwhile, a frame having a length of N must be decoded using N numbers of FPTD processors, which limits flexibility. Motivated by this, we propose a novel APTD algorithm in this section, which can employ an arbitrary number p of processors and exploit them flexibly across all of the LTE interleaver lengths. Two versions of the APTD will be proposed separately in Section III-A and III-B, respectively. The first version of the APTD decomposes the frame into the highest number of windows that can support equal window lengths, exploiting the contention-free property of the LTE interleaver, as it will be detailed in Section III-A. However, some idle processors still remain for some frame lengths in this version. By contrast, the second version of Section III-B does not rely on the contention-free property of the LTE interleaver, allowing different windows to adopt different lengths. This enables all processors to be exploited for all but the shortest frame lengths, which is achieved by decomposing the frame into the same number of windows as there are processors.

A. APTD employing equal window lengths

The first version of the APTD employs an arbitrary number p of processors in total, but activates only a subset P of these depending on the frame length N. When decoding frames of length N, the number of activated processors P is given by the *greatest integer factor (gif)* of N that is also smaller than the number of processors p, which can be expressed as $P = gif(N, p)$. Figure 6 (a) illustrates the relationship between the number of activated processors P and the frame lengths N when employing a total number of

$p = 64$ or $p = 128$ processors, as compared to the SOTA turbo decoders and FPTD discussed above. Each processor alternates between processing a window from the upper decoder and the corresponding window from the lower decoder. Likewise, Figure 6 (b) shows the relationship between the window length $L = N/P = N/gif(N, p)$ and the frame length N, for $p = 64$ and $p = 128$. Note that for shorter frame lengths $N \leq p$, the APTD operation becomes identical to that of the FPTD, where each window effectively has a length L of one bit.

Like the FPTD, the APTD relies on odd-even operation, as depicted in Figure 7. However, rather than alternating the operation of odd- and even-indexed algorithmic blocks, the odd-even operation is performed at the window level in the APTD algorithm. To be more specific, the odd-indexed windows in the upper decoder and the even-indexed windows in the lower decoder operate at the same time. This alternates with operating the even-indexed windows in the upper decoder at the same time as the odd-indexed windows in the lower decoder. A total number of $2L$ clock cycles have to be completed in one iteration. The forward state metrics $\bar{\boldsymbol{\alpha}}_k^{\mathrm{u}}$, $\bar{\boldsymbol{\alpha}}_k^{\mathrm{l}}$ and the backward state metrics $\bar{\boldsymbol{\beta}}_{k-1}^{\mathrm{u}}$, $\bar{\boldsymbol{\beta}}_{k-1}^{\mathrm{l}}$ are calculated according to the forward and backward recursions that are synchronised among all windows processed at the same time. Note that the odd-even operation ensures that the end of the recursions of the odd-indexed windows seamlessly leads to the beginning of the adjacent even-indexed windows, and vice versa. This allows information to more quickly propagate along the upper and lower decoder than in the upper-lower operation. Radix-4 operation [14, 15] is also employed in the proposed APTD, which further doubles the decoding throughput. However, since the Radix-4 operation processes a pair of trellis stages at a time, a special solution is required when the window length L is not an even integer. More specifically, the final algorithmic block in each window may be processed using Radix-2 operation.

The APTD calculates two extrinsic LLRs at a time alongside the forward recursion, when using Radix-4 operation. During the first half of each forward recursion, the $\bar{\boldsymbol{\beta}}$ values calculated during the previous iteration are recalled and are used for calculating the extrinsic LLRs. Once the forward and backward recursions have crossed over in the second half of the recursions, the $\bar{\boldsymbol{\beta}}$ values calculated during the first half of the backward recursion are used. This is in contrast to the SOTA LTE turbo decoder, which calculates four LLRs at a time, during the second half of the forward and backward recursions once they have crossed over. Our approach requires only two extrinsic LLR calculators and two interleavers that are used all the time, rather than four LLR calculators and four interleavers that are only used in the second half of the recursions, which is less efficient.

Note that a particular extrinsic LLR in an odd-indexed window of one component decoder may become an *a priori* LLR for an even-indexed window of the other component decoder or vice versa in the proposed FPTD. Since these windows are processed at the same time in the FPTD, some interesting interactions may arise. In some cases, the extrinsic LLR may be generated and interleaved before it is used as an *a priori* LLR in the same iteration, depending on the position of these LLRs in the windows. This is advantageous compared to the classic upper-lower operation, which must always wait until the next half iteration before an interleaved LLR can be exploited. Other times, however, the extrinsic LLR may be delivered too late to be used in the same iteration, but can be saved for use in the next iteration. This is a disadvantage compared to the upper-lower operation, since the delay before the interleaved LLR can be exploited is extracted in this case. Overall, the advantage of sometimes being able to exploit an interleaved LLR immediately and the disadvantage of sometimes having to wait until the next iteration may be expected to cancel out. However, the advantage of quicker $\bar{\boldsymbol{\alpha}}$ and $\bar{\boldsymbol{\beta}}$ propagation can be exploited in the APTD versions, as we will demonstrate for short frames in Section IV.

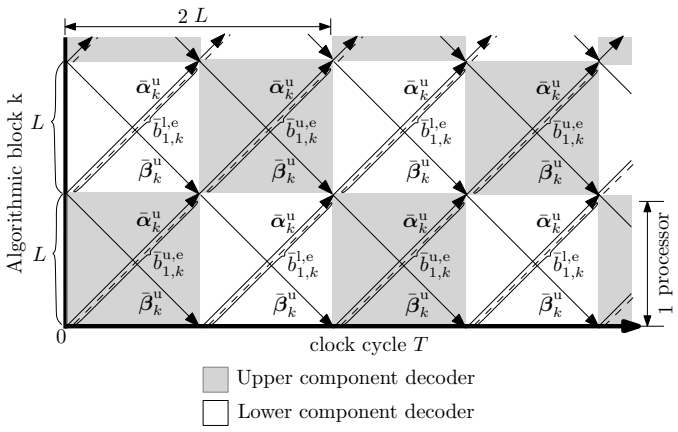


Fig. 7. Schedule of the forward and backward recursions for the first version of the APTD, which uses equal windows lengths, Radix-4 operation, odd-even processing and calculates two extrinsic LLRs at a time alongside the forward recursion.

In addition to the non-interleaved systematic approach discussed in Section II-A, we also consider an interleaved systematic approach, as shown in Figure 4 (b). As described in Section II-A, the non-interleaved systematic approach has the advantage of avoiding the requirement for a separate interleaver for the systematic LLRs $\bar{b}_{3,k}^{\mathrm{u,a}}$. In the upper-lower schedule, there is no disadvantage compared to the non-interleaved systematic selection, since the lower decoder is not activated until after the systematic LLRs $\bar{b}_{3,k}^{\mathrm{u,a}}$ are combined with the extrinsic LLRs $\bar{b}_{1,k}^{\mathrm{u,e}}$ and passed through the interleaver by the upper decoder. However, in the case of our odd-even schedule, some blocks in the lower decoder are activated in the first half of the first iteration, before the systematic LLRs can be passed through the interleaver by the upper decoder. This motivates the consideration of the interleaved systematic approach, which improves the FER by providing systematic LLRs for these blocks in the lower decoder at the start of the iterative decoding process, albeit at the cost of requiring an additional interleaver.

B. APTD employing unequal window lengths

In a second version of the APTD algorithm, frame lengths N that are not divisible by the number of processors p are

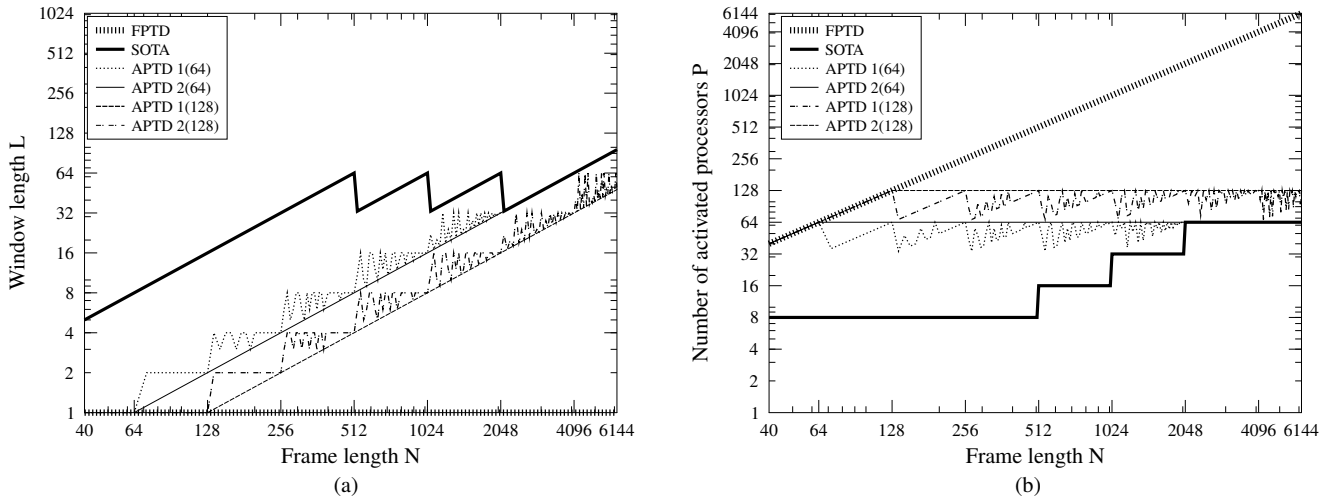


Fig. 6. (a) The relationship between the number of activated processors P and the frame length N for various turbo decoders; (b) The relationship between window length L and frame length N, for various turbo decoders. Note that in the SOTA turbo decoder, FPTD and the first version of the APTD algorithms, all windows have the same length, since the number of activated processors P is chosen as an integer factor of N in these schemes. By contrast, some windows have a length $\lceil \frac{N}{P} \rceil$ and others have the length $\lfloor \frac{N}{P} \rfloor$ in the second version of APTD algorithm. In this case, the average window lengths is plotted.

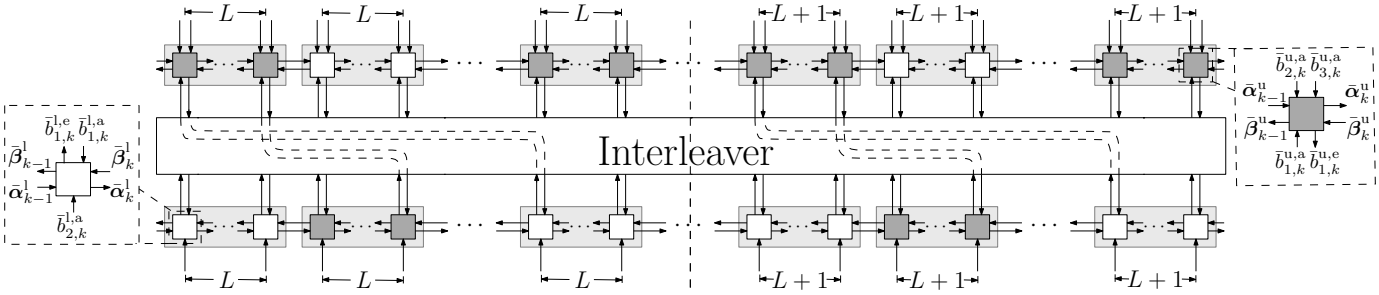


Fig. 8. Schematic of the second version of the proposed APTD with unequal window lengths.

handled in a different way. Rather than activating only a subset P of the processors, so that each process has an equal window length, this second version activates all processors for frame lengths $N \geq p$, where some process windows having a slightly shorter length, while others process windows have a slightly longer window. More specifically, the first $[P - \mathrm{mod}(N, P)]$ windows have a length of $L = \lfloor N/P \rfloor$ trellis stages, while the remaining $\mathrm{mod}(N, P)$ windows have a length of $L+1 = \lceil N/P \rceil$, as shown in Figure 8. Indeed, for frame lengths N lower than the number of processors p, this version of the APTD also operates identically to the FPTD, as discussed in Section III-A.

Figure 9 illustrates the scheduling of the forward and backward recursions in the second version of the APTD. Each processor alternates between processing the corresponding window in the upper decoder and the corresponding window in the lower decoder. As in the first version of the APTD, the extrinsic LLRs are calculated alongside the forward recursions. Note that the windows having length L are grouped together and are scheduled independently of the windows having lengths $L+1$, because they require different numbers of clock cycles to complete each iteration. As a result, the two groups of windows become de-synchronised as the decoding process proceeds. More specifically, the forward recursion of the last processor for the shorter windows does not seamlessly lead into the forward recursion of the first processor for the longer windows, since a one-clock-cycle delay exists between the two groups. Likewise, the backward recursion does not seamlessly flow across the boundary between the processors of the shorter windows and those of the longer windows. Instead, the boundary state metrics may be written into memory when generated by a processor on one side of the boundary, and read from that memory later upon initialising a recursion on the other side of the boundary.

Since not all window lengths are identical in the second version of the APTD algorithm, interleaving the extrinsic LLRs $\bar{b}^{\mathrm{u,e}}_{1,k}$, $\bar{b}^{\mathrm{l,e}}_{1,k}$ according to the schedule at which they are generated does not benefit from the LTE interleaver's contention free property. More specifically, some processors would be provided with more than one *a priori* LLR in some clock cycles by the LTE interleaver, hence requiring a complex input/output interface. However, we note that when interleaving the LLRs according to this schedule, many of the resultant *a priori* LLRs $\bar{b}^{\mathrm{u,a}}_{1,k}$, $\bar{b}^{\mathrm{l,a}}_{1,k}$ do not get used by the other decoder for a number of clock cycles later, when they are reached by the forward or backward recursions. This illustrates

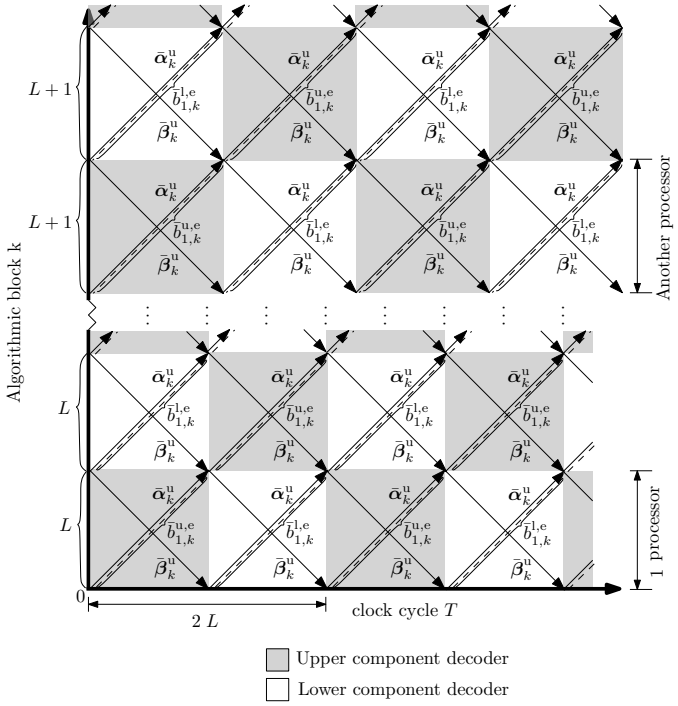


Fig. 9. Schedule of the forward and backward recursions for the second version of the APTD, which uses unequal window lengths, odd-even processing and calculates two extrinsic LLRs at a time alongside the forward recursion.

that the interleaving of some extrinsic LLRs can be delayed by a number of clock cycles without having any effect on the operation of the turbo decoder. Note however that the operation is adversely affected if the interleaving of an extrinsic LLR is delayed beyond the time when the resultant *a priori* LLRs are first used by the other decoder, requiring it to instead use an out-of-date LLR from a previous iteration. Motivated by this, our future work will address the contention problem in the second version of the APTD algorithm by designing schedules for the interleaving of the extrinsic LLRs, so that only one extrinsic LLR is delivered to each processor at a time, enabling the employment of a simple input/output interface. In the meantime, Section IV will investigate the effect of delaying the interleaving of the extrinsic LLRs in the second version of the APTD.

IV. Performance analysis

In this section, we characterise the error correction performance associated with the different techniques of Section III individually, as well as in combination with the techniques employed by the two versions of the APTD, which include odd-even operation, non-interleaved systematic LLRs and the calculation of the extrinsic values alongside the forward recursion.

The error correction performance achieved is characterised in Figures 10 and 11, which present a range of capacity bounds that provide references for the attainable performance of turbo codes. More specifically, given a code rate of $R = 1/3$ and QPSK modulation, upon communicating over an AWGN channel, the Continuous-input Continuous-output Memoryless Channel (CCMC) capacity of $E_b/N_0 = -0.49$ dB is obtained according to [25]. The corresponding modulation-specific Discrete input Continuous-output Memoryless Channel (DCMC) capacity of $E_b/N_0 = -0.51$ dB was derived in [26]. Additionally, the EXtrinsic Information Transfer (EXIT) chart capacity band of -0.07 dB derived in [27] is the lowest E_b/N_0 value for which an open tunnel can be created between the two mutual information curves of the pair of convolutional decoders in the EXIT chart. We note that in all cases considered, the various turbo decoders do not present any error floors above an FER of 10^{-5}, indicating that the quality of service requirements of URLLC LTE are indeed met.

Figure 10 (a) compares the FER performance of the APTD for the LTE turbo code employing $P = 64$ processors, $I = 8$ iterations, Radix-4 operation, odd-even (O-E) and the conventional upper-lower (U-L) operation, in cases of short, medium and long frame lengths of $N = 64, 512$ and 6144 bits, where the non-interleaved systematic LLRs (*NIn*) are employed and the extrinsic LLRs are calculated four at a time, once the forward and backward recursions have crossed over (Ext on Both). Note that the combination of U-L, *NIn* and Ext on Both is as employed in the SOTA approach. For the short frame length of $N = 64$ bits, Figure 10 (a) shows that a 2dB gain is achieved at a FER of 10^{-5} by employing O-E, rather than U-L. However, this gain decreases as the window length $L = N/P$ is increased. This is because the advantage offered by having the recursions of one window seamlessly leading into those of its neighboring window becomes less significant, when the windows are longer. Furthermore, Figure 10 (a) shows that *NIn* degrades the FER performance by 0.2 dB associated with O-E compared to *In* in the case of short frame lengths N, but no degradation is observed for longer frame lengths. This represents a small price to pay for the benefit of eliminating the requirement for an additional interleaver. Note that *In* and *NIn* attain an identical FER performance for U-L, as discussed in Section III.

Recall from Section III, that the proposed APTD algorithm calculates the extrinsic LLRs two at a time alongside the forward recursions (Ext on F), hence resulting in a significant reduction in hardware resources, compared to schemes that calculate the extrinsic LLRs four at a time, once the forward and backward recursions have crossed over (Ext on Both). However, this hardware reduction is achieved at the cost of a slight FER degradation for medium and long frame lengths, as shown in Figure 10 (b). Note that the FER degradation associated with Ext on F is higher when employing U-L, demonstrating a further benefit of O-E.

Based on Figure 10, we may conclude that the combination of O-E, *Nln* and Ext on F results in an attractive APTD algorithm, since it achieves significantly reduced complexity at the cost of only slight FER performance degradation, compared to the SOTA LTE turbo decoder. In the following discussions, we will employ this combination for investigating the FER performance of the two versions of the APTD algorithms.

In order to better demonstrate the difference in operation between the two versions of the proposed APTD algorithm,

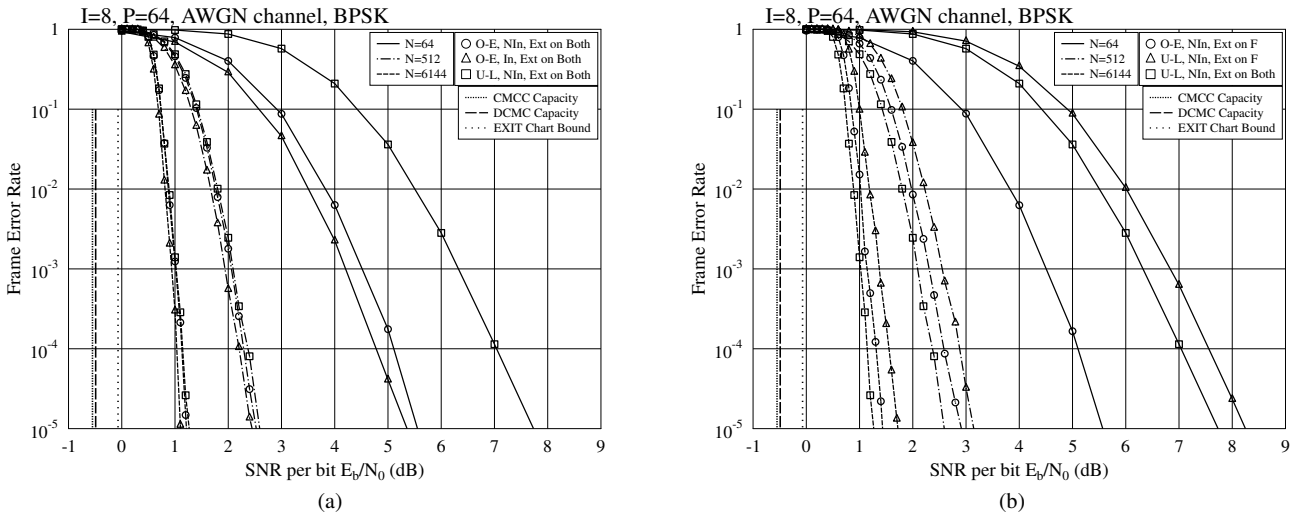


Fig. 10. FER performance of the APTD algorithm for the $N = 64, 512$ and 6144-bit LTE turbo code, employing $P = 64$ processors, $I = 8$ iterations, Radix-4 operation, (a) interleaved (*In*) or non-interleaved (*NIn*) systematic, upper-lower (U-L) or odd-even (O-E) operation, where the extrinsic LLRs are calculated four at a time once the forward and backward recursions have crossed over (Ext on Both); (b) *NIn*, U-L or O-E operation, where the extrinsic LLRs are either obtained two at a time alongside the forward recursions (Ext on F), or four at a time once the forward and backward recursions have crossed over (Ext on Both).

we employ $p = 56$ processors in the following discussions, rather than $p = 64$ as in our discussions above. Since 56 is not divisible by the frame lengths of $N = 64, 512$ or 6144, this choice will result in different schedules between the two versions of the APTD. More specifically, the first version will activate $P = 32, 32$ and 48 of the $p = 56$ processors, when $N = 64, 512$ and 6144, respectively, in order to ensure that each of the P windows has the same length L. By contrast, the second version will activate all $p = 56$ processors for all frame lengths N by employing windows having different lengths. Figure 11 (a) shows that the second version achieves the best FER performance among the four different turbo decoders at the same number of clock cycles T, especially for the short frame lengths. We can see that while the conventional and SOTA decoder suffer from bad reliability, which may result in poor quality of service (QoS), the second version is capable of achieving the URLLC FER requirement of 10^{-5}. Note that when a frame length of $N = 6144$ is employed, the SOTA turbo decoder slightly outperforms both versions of the APTD. However, this gain is negligible compared to the gap between the FER performance and the capacity bounds.

In order to facilitate the practical implementation of the second version of the APTD, it is necessary to solve the contention problem, which occurs when more than one processor attempts to pass LLRs through the interleaver to the same processor at the same time. A measure that may be taken to avoid contention in the second version of the APTD is to reschedule the operation of the interleaver, by delaying the interleaving of some LLRs that would otherwise cause contention. This approach is motivated since many *a priori* LLRs are not used in O-E until a number of clock cycles after they are generated. However, in some cases, the contention may only be eliminated by delaying the interleaving of some LLRs until after they would have otherwise been used. In this case, the LLRs generated in the most recent previous iteration may be used instead, albeit at the cost of degraded FER. In our investigations, we found that $1/3$ of the LLRs must be rescheduled, in order to avoid contention in the second version of the APTD. However, in order to investigate the worst-case FER degradation that may be imposed, we quantify the impact of delaying the interleaving of *all* LLRs, rather than just that of the subset which causes contention. Figure 11 (b) characterises the FER when ω clock cycles of delay is imposed on the interleaving of the LLRs. By comparing Figures 11 (a) and 11 (b), it may be seen that the second version of the APTD offers superior FER over the first version, even if 1 or 2 clock cycles of delay are applied to all LLRs. Since contention can be eliminated by delaying only a subset of the LLRs, we may conclude that the second version is superior.

V. Complexity Analysis

This section compares the proposed APTD employing $p = 64$ processors to the conventional LTE turbo decoder of [8] employing $p = 8$ processors, to the SOTA LTE turbo decoder of [16] employing $p = 64$ processors, and to the FPTD of [19] employing $p = 6144$ processors, both in terms of the number of clock cycles required for achieving a low FER and in terms of the complexity of each processor. These are used for characterising the latency, throughput and hardware resource requirements that may be expected by these turbo decoder algorithms.

Table I summarises our comparisons of the proposed APTD with the conventional LTE turbo decoder, with the FPTD, and with the SOTA LTE turbo decoder. In contrast to the Radix-2 operation of the FPTD, the conventional turbo decoder, the SOTA LTE turbo decoder and the APTD use Radix-4 processing [14, 15], allowing two algorithmic blocks to be

[1]The number of activated processors P for different frame lengths N of the SOTA LTE turbo decoder is given in (2).

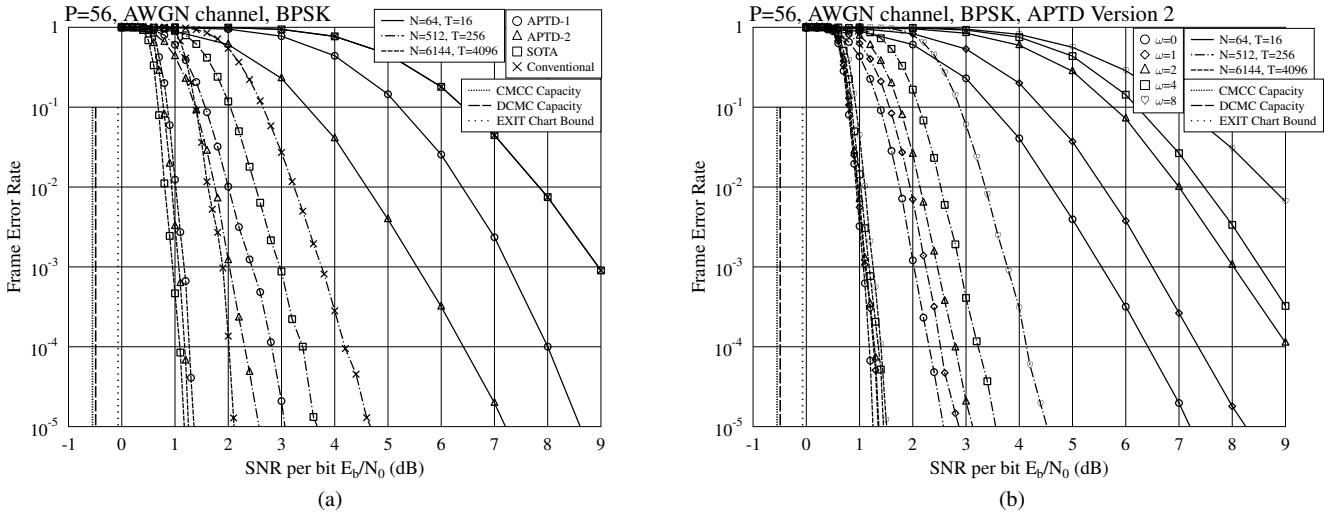


Fig. 11. (a) FER performance of the conventional and SOTA LTE turbo decoder and both versions of the proposed APTD for the $N = 64, 512$ and 6144 bits LTE turbo code that employs $T = 16, 256$ and 4096 clock cycles, respectively. The proposed APTD employs $P = 56$ processors, Radix-4 operation, *NIn* systematic approach, O-E operation, and the extrinsic values are obtained from forward recursions only (Ext on F); (b) FER performance for second version of the APTD algorithm for $N = 64, 512, 6144$, with $T = 16, 256$ and 4096 clock cycles, respectively, activating $P = 56$ processors when different delays are imposed. Here, *NIn* systematic and O-E operation are employed and the extrinsic values are obtained from forward recursions only (Ext on F).

TABLE I
COMPLEXITY ANALYSIS OF DIFFERENT TURBO DECODERS

	Conventional $N \in [40, 6144]$	SOTA $N \in [40, 6144]$	FPTD $N \in [40, 6144]$	APTD $N \in [40, 6144]$	
				Version 1	Version 2
Window length (L)	$N/8$	N/P[1]	2	$\frac{N}{gif(N,p)}$	$\lfloor N/P \rfloor$ or $\lceil N/P \rceil$
Overall latency (T)	T_{C}	T_{S}	T_{F}	T_{A1}	T_{A2}
Overall throughput ($1/T$)	$\frac{1}{T_{\mathrm{C}}}$	$\frac{1}{T_{\mathrm{S}}}$	$\frac{1}{T_{\mathrm{F}}}$	$\frac{1}{T_{\mathrm{A1}}}$	$\frac{1}{T_{\mathrm{A2}}}$
Complexity per processor per clock cycle (C)	320	320	155	320	
Interleaved LLRs by each PE at a time	4	4	1	2	
Overall complexity (TCP)	$2560T_{\mathrm{C}}$	$320T_{\mathrm{S}}P$	$155T_{\mathrm{F}}N$	$320T_{\mathrm{A1}}P$	$320T_{\mathrm{A2}}P$
Computational efficiency ($1/TCp$)	$\frac{1}{320T_{\mathrm{C}}p}$	$\frac{1}{320T_{\mathrm{S}}p}$	$\frac{1}{155T_{\mathrm{F}}p}$	$\frac{1}{320T_{\mathrm{A1}}p}$	$\frac{1}{320T_{\mathrm{A2}}p}$

processed per clock cycle. For example, in the case of a frame length of $N = 504$ bits, the conventional and the SOTA LTE turbo decoder decomposes the frame into $P = 8$ windows and processes them using 8 processors. In the case of the SOTA decoder, its other 56 processors remain disabled. Here, each window in the conventional or the SOTA LTE turbo decoder comprises $L = 63$ bits, requiring 63 clock cycles to complete the processing of the whole frame length, when using Radix-4 processing. By contrast, when $N = 504$, the first and the second version of our APTD activate $P = 63$ and 64 processors, respectively. In both schemes, 8 clock cycles are used for processing the entire 504 bits once, which is an eight-fold delay reduction. Indeed, the second version of the APTD activates all processors for all frame lengths that satisfy $N \geq P$, hence avoiding wasted hardware, as discussed in Section III-B. Since the FPTD processes the entire frame simultaneously, alternating between the odd- and even-indexed algorithmic blocks, its window length L is consistently 2, regardless of the frame length N.

The overall latency may be compared in terms of the number of clock cycles required for each turbo decoder to perform sufficient iterations to achieve a FER of 10^{-5} at the same E_b/N_0 as the conventional turbo decoder using $I = 8$ iterations. Meanwhile, the overall throughput is proportional to the reciprocal of the decoding latency. Figures 12 (a) and (b) characterise the latency and the throughput as functions of frame length $N \in [40, 6144]$ for both versions of the proposed APTD and compare them to those of the conventional, FPTD and the SOTA turbo decoders. In the case of $N = 504$ bits, a total of $T_{\mathrm{C}} = T_{\mathrm{S}} = 1008$ clock cycles are required for the conventional and SOTA LTE turbo decoder to complete $I = 8$ iterations. By contrast, the first and second version of the APTD require 320 and 296 clock cycles, respectively. The APTD achieves this latency improvement of at least 2.23 times by activating all of its 64 processors, while the conventional and SOTA decoder activate only 8 processors at $N = 504$. In a hardware implementation operating at the same clock frequency of 250 MHz used in the commercial LTE turbo decoder implementation of [8], the corresponding processing latency of the APTD would become around 3.6 μs, achieving a 14-times improvement compared to the 52 μs demonstrated in [8]. Hence, the proposed APTD is capable of meeting the

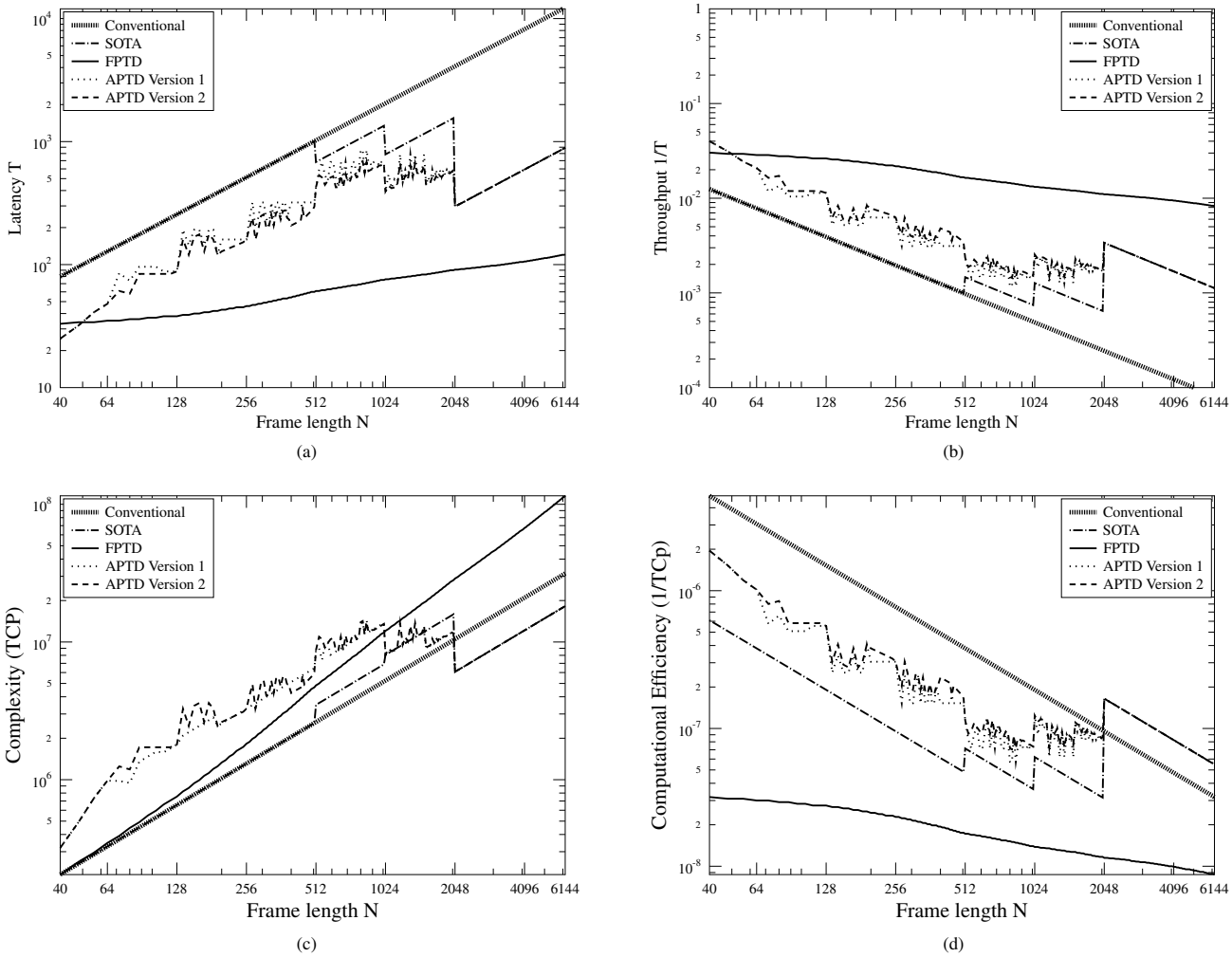


Fig. 12. (a) The number of clock cycles required for all frame lengths $N \in [40, 6144]$ to achieve a FER of 10^{-5} at the same E_b/N_0 as the conventional turbo decoder using $I = 8$ iterations in different turbo decoders; (b) The reciprocal of the number of clock cycles required for all frame lengths $N \in [40, 6144]$ in different turbo decoders; (c) The overall complexity in different turbo decoders as a function of the frame length N; (d) The overall computational efficiency in different turbo decoders as a function of the frame length N.

7.4 μs latency requirement of LTE URLLC, even in the case of the longest $N = 6144$-bit frames. Meanwhile, at the longest frame length of $N = 6144$ bits a total of 12288 clock cycles are required for the conventional turbo decoder to complete $I = 8$ iterations, while the proposed APTD requires only 890 clock cycles to achieve the same FER performance.

The computational complexity may also be used for characterizing both the energy consumption and the hardware resource requirement of a practical hardware implementation. In [19], the complexity C per processor per clock cycle imposed by the SOTA LTE turbo decoder and by the FPTD was given as 320 and 155 Add Compare Select (ACS) operations, respectively. For the conventional LTE turbo decoder and both versions of the APTD algorithm, the number of ACS operations performed per processor per clock cycle is also $C = 320$, as in the SOTA turbo decoder, since they also employ Radix-4 processing.

As discussed in Section II, the maximum number of LLRs that must be interleaved at a time by each PE varies among the decoding algorithms compared in Table I. Since the proposed APTD algorithm obtains extrinsic LLRs alongside the forward recursions, its PEs generate two extrinsic LLRs for interleaving in each clock cycle when using Radix-4 processing. Similarly, the PEs of the FPTD algorithm process windows comprising only a single trellis stage, and so they generate only one extrinsic LLR for interleaving at a time. By contrast, the conventional and the SOTA LTE turbo decoder generates four extrinsic LLRs at a time once the forward and backward recursions have crossed over, when using Radix-4 processing. The requirement for the conventional and the SOTA turbo decoder to interleave four extrinsic LLRs at a time represents a significant hardware overhead, causing the interleaver to occupy 15.3% of the chip area in the turbo decoder implementation [28].

Combining the above-mentioned design considerations, the energy consumption of a turbo decoder implementation may be characterised by its overall computational complexity, as shown in Table I. The overall complexity is defined as the product of the number T of clock cycles required to achieve a FER of 10^{-5} at the same E_b/N_0 as the conventional LTE

turbo decoder using $I = 8$ iterations, the computational complexity C per processor per clock cycle and the number of activated processors P. Figure 12 (c) shows the overall complexity associated with all frame lengths $N \in [40, 6144]$ for the various turbo decoders considered in this section. In the example of $N = 504$ bits, the overall complexity of the second version of the proposed APTD employing $P = 64$ PEs is 2.3 times that of the conventional and the SOTA LTE turbo decoders, which employ 8 processors. However, its complexity is 1.3 times higher than that of the FPTD in this case. Note that for frame lengths in excess of $N = 2048$, the number of activated processors and the overall latency are the same in the SOTA LTE turbo decoder and both versions of the APTD, so the overall computational complexity remains identical.

In addition to the computational complexity TCP, we also compare the decoders in terms of their hardware efficiency $1/(TCp)$. Note that while the computational complexity depends on the number P of activated processors, the hardware efficiency considers the total number p of processors employed in the decoder, as summarised in Table I. This is because all processors occupy a certain chip-area, regardless whether they are activated or not. Figure 12 (d) characterizes the hardware efficiency of the three turbo decoders discussed in this paper, as functions of the frame length $N \in [40, 6144]$. In the example of $N = 504$ bits, the second version of the APTD employing $p = 64$ processors achieves a 7-times efficiency improvement compared to the SOTA turbo decoder employing $p = 64$ processors. Note that for frame lengths in excess of $N = 2048$, the computational efficiency is similar in both versions of the APTD and the SOTA turbo decoder. In the case of the FPTD, it is assumed that $p = 6144$ processors are employed and that $P = N$ processors are activated when decoding frames of length N. Owing to this, the hardware efficiency of the FPTD is poor, especially when the frame length N is small compared to p. In the case of $N = 504$ bits, the second version of the APTD employing $P = 64$ processors achieves a 22-times efficiency improvement compared to the FPTD decoder.

VI. Conclusions

This paper proposed a novel APTD algorithm, which facilitates an arbitrarily high degree of turbo decoding parallelism for the first time, enabling significantly improved throughput, latency, and computational efficiency in comparison to the SOTA turbo decoder while meeting the requirements of LTE URLLC. More specifically, conventional commercial implementations of the LTE turbo decoder have latencies of up to 52 μs, which are not able to meet the 7.4 μs requirements of LTE URLLC. By contrast, the proposed APTD can achieve the same error correction performance as the conventional decoder down to FERs of 10^{-5}, but with latencies of no more than 3.6 μs, meeting the requirements of LTE URLLC. Furthermore, the APTD achieves a significant reduction in complexity in long frame lengths, compared to FPTD. In particular, none of the processors in our proposed algorithm remain idle for any frame length $N \geq p$, leading to better FER performance than the SOTA turbo decoder in cases of short frame lengths. For instance, when $p = 56$ processors and 16 clock cycles are employed for decoding a frame length $N = 64$, our APTD achieves a coding gain of 3.5 dB compared to the SOTA LTE turbo decoder at a FER level of 10^{-3}, as shown in Figure 11, whereas only slight improvements can be observed when decoding a frame length of $N = 6144$ employing the same number of $p = 56$ processors with 4096 clock cycles. We have proposed an odd-even processing of windows in the upper and lower decoder, which achieves better FER performance for short frame lengths, compared to conventional upper-lower processing. Furthermore, we reduce the interleaving complexity by generating extrinsic LLRs alongside the forward recursion, at the cost of slightly degraded FER performance. Like the FPTD, the proposed APTD is capable of achieving the same error correction performance as a conventional LTE turbo decoder, at all frame lengths. However, our APTD achieves this using significantly fewer decoding iterations and hence a lower complexity at long frame lengths. As shown in Figure 12 (a), the proposed APTD achieves superior latency, throughput and computational efficiency than the SOTA LTE turbo decoder at all frame lengths, but particularly at the short frame lengths that are typically used in URLLC approaches. For example, at a frame length of $N = 504$ bits, the proposed APTD achieves an FER of 10^{-5} at the same E_b/N_0 as $I = 8$ iterations of a conventional turbo decoder, but with a computational efficiency that is 6 times higher than that of the SOTA turbo decoder, while achieving a latency and throughput that are 0.7 and 1.4 times those of the SOTA decoder, respectively. Note however that this is achieved at the cost of increasing the computational complexity by 2.3 times compared to the SOTA decoder of $N = 504$.

Our future work will address the contention-free problem in the second version of the APTD, which arises when activating the number of PEs P that is not an integer factor of the frame length N. We will achieve this by designing schedules that delay the interleaving of some extrinsic LLRs relative to the forward recursions in which they are generated. Our results seen in Figure 11 (b) demonstrate that delaying the interleaving of extrinsic LLRs in this way has only a negligible impact upon the FER performance. Furthermore, our future work will consider the practical hardware implementation of the proposed APTD algorithm in order to determine the throughput, latency, energy efficiency and hardware efficiency that can be achieved in practice.

References

[1] E-UTRA, "Multiplexing and channel coding," *3rd Generation Partnership Project Std. 3GPP TS*, vol. 36, p. V8, 2008.

[2] 3GPP RP-171489, "Work item on ultra reliable low latency communication for LTE," June 2017.

[3] T. Fehrenbach, R. Datta, B. Göktepe, T. Wirth, and C. Helge, "URLLC services in 5G-low latency enhancements for LTE," *Accepted for publication at IEEE Vehicular Technology Conference (VTC)*, Fall 2018.

[4] X. Zhang, "Latency reduction with short processing time and short TTI length," in *Intelligent Signal Processing and Communication Systems (ISPACS), 2017 International Symposium on*, pp. 545–549, IEEE, 2017.

[5] H. Ji, S. Park, J. Yeo, Y. Kim, J. Lee, and B. Shim, "Introduction to ultra reliable and low latency communications in 5G," *Computing Research Repository (CoRR) abs/1704.05565*, 2017.

[6] J. C. S. Arenas, T. Dudda, and L. Falconetti, "Ultra-low latency in next generation lte radio access," in *SCC 2017; 11th International ITG Conference on Systems, Communications and Coding; Proceedings of*, pp. 1–6, VDE, 2017.

[7] 3GPP RP-161299, "Work item on shortened TTI and processing time for LTE," June 2016.

[8] Altera, "3GPP LTE turbo reference design," Jan 2011.

[9] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *IEEE International Conference on Communications, ICC'95 Seattle,'Gateway to Globalization',*, vol. 2, pp. 1009–1013, IEEE, 1995.

[10] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1261–1271, 1996.

[11] C. Berrou and A. Glavieux, "Turbo codes," *Encyclopedia of Telecommunications*, 2003.

[12] L. Hanzo, T. Liew, B. Yeap, R. Tee, and S. X. Ng, *Turbo coding, turbo equalisation and space-time coding: EXIT-chart-aided near-capacity designs for wireless channels*. John Wiley & Sons, 2011.

[13] S. Yoon and Y. Bar-Ness, "A parallel MAP algorithm for low latency turbo decoding," *IEEE Communications Letters*, vol. 6, no. 7, pp. 288–290, 2002.

[14] M. Bickerstaff, L. Davis, C. Thomas, D. Garrett, and C. Nicol, *A 24Mb/s radix-4 logMAP turbo decoder for 3GPP-HSDPA mobile wireless*. PhD thesis, IEEE, 2003.

[15] Y. Zhang and K. K. Parhi, "High-throughput radix-4 logMAP turbo decoder architecture," in *2006 Fortieth Asilomar Conference on Signals, Systems and Computers*, 2006.

[16] T. Ilnseher, F. Kienle, C. Weis, and N. Wehn, "A 2.15 GBit/s turbo code decoder for LTE Advanced base station applications," in *7th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, pp. 21–25, IEEE, 2012.

[17] J. Sun and O. Y. Takeshita, "Interleavers for turbo codes using permutation polynomials over integer rings," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 101–119, 2005.

[18] O. Y. Takeshita, "On maximum contention-free interleavers and permutation polynomials over integer rings," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 1249–1253, 2006.

[19] R. G. Maunder, "A fully-parallel turbo decoding algorithm," *IEEE Transactions on Communications*, vol. 63, no. 8, pp. 2762–2775, 2015.

[20] A. S. Barbulescu and S. S. Pietrobon, "Interleaver design for turbo codes," *Electronics Letters*, vol. 30, no. 25, pp. 2107–2108, 1994.

[21] L. Hanzo, S. X. Ng, W. Webb, and T. Keller, *Quadrature amplitude modulation: From basics to adaptive trellis-coded, turbo-equalised and space-time coded OFDM, CDMA and MC-CDMA systems*. IEEE Press-John Wiley, 2004.

[22] 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*, vol. 4, pp. 323–346, 2016.

[23] 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, 2003.

[24] J. Vogt and A. Finger, "Improving the Max-Log-MAP turbo decoder," *Electronics Letters*, vol. 36, no. 23, pp. 1937–1939, 2000.

[25] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

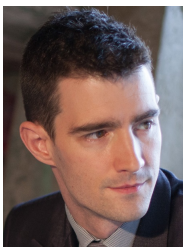
[26] L. Kong, S. X. Ng, R. G. Maunder, and L. Hanzo, "Maximum-throughput irregular distributed space-time code for near-capacity cooperative communications," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 3, pp. 1511–1517, 2010.

[27] J. Hagenauer, "The exit chart-introduction to extrinsic information transfer in iterative processing," in *Signal Processing Conference, 2004 12th European*, pp. 1541–1548, IEEE, 2004.

[28] R. Shrestha and R. P. Paily, "High-throughput turbo decoder with parallel architecture for LTE wireless communication standards," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 9, pp. 2699–2710, 2014.

Luping Xiang received the B.Eng. (Hons.) degree from Xiamen University, China, in 2015. He is currently pursuing the Ph.D. degree with the Southampton Wireless Group, University of Southampton. His research interests include channel coding and ultra low latency scheme.

Matthew F. Brejza recieved a first class honors BEng in Electronic Engeering from the University of Southampton, UK, in July 2012, where he is currently working toward the Ph.D. degree with the Communications Research Group, School of Electronics and Computer Science. His research interests include flexible hardware implementation, source and channel coding and their applications in low power data communications.



Robert G. Maunder has studied with the School of Electronics and Computer Science, University of Southampton, UK, since October 2000. He was awarded a first class honours BEng in Electronic Engineering in July 2003, as well as a PhD in Telecommunications in December 2007. He began a lectureship in November 2007 and was promoted to Associate Professor in March 2013 and to Professor in August 2017. He was awarded Senior Member status of the IEEE in December 2012, Chartered Engineer status of the IET in November 2013 and Fellow status of the IET in January 2017. Rob's research interests include joint source/channel coding and the holistic design of algorithms and hardware implementations for wireless communications. He has published a number of IEEE papers in these areas. He is the founder and CTO of AccelerComm Ltd, which is commercialising his research as soft-IP.

Bashir M. Al-Hashimi i is an ARM Professor of Computer Engineering and Dean of the Faculty of Physical Sciences and Engineering, University of Southampton. In 2009, he was elected fellow of the IEEE for significant contributions to the design and test of low-power circuits and systems. He holds a Royal Society Wolfson Research Merit Award (2014-2019). He has published over 300 technical papers, authored or co-authored 5 books and has graduated 31 PhD students.

Lajos Hanzo FREng, FIEEE, FIET, Fellow of EURASIP, DSc received his degree in electronics in 1976 and his doctorate in 1983. In 2009 he was awarded an honorary doctorate by the Technical University of Budapest and in 2015 by the University of Edinburgh. In 2016 he was admitted to the Hungarian Academy of Science. During his 40-year career in telecommunications he has held various research and academic posts in Hungary, Germany and the UK. Since 1986 he has been with the School of Electronics and Computer Science, University of Southampton, UK, where he holds the chair in telecommunications. He has successfully supervised 111 PhD students, co-authored 18 John Wiley/IEEE Press books on mobile radio communications totalling in excess of 10 000 pages, published 1700+ research contributions at IEEE Xplore, acted both as TPC and General Chair of IEEE conferences, presented keynote lectures and has been awarded a number of distinctions. Currently he is directing a 60-strong academic research team, working on a range of research projects in the field of wireless multimedia communications sponsored by industry, the Engineering and Physical Sciences Research Council (EPSRC) UK, the European Research Council's Advanced Fellow Grant and the Royal Society's Wolfson Research Merit Award. He is an enthusiastic supporter of industrial and academic liaison and he offers a range of industrial courses. He is also a Governor of the IEEE VTS. During 2008 - 2012 he was the Editor-in-Chief of the IEEE Press and a Chaired Professor also at Tsinghua University, Beijing. For further information on research in progress and associated publications please refer to http://www-mobile.ecs.soton.ac.uk Lajos has 30 000+ citations and an H-index of 71.