IEEE Access
The journal for rapid open access publishing

Received October 29, 2015, accepted November 20, 2015, date of publication November 23, 2015,
date of current version December 22, 2015.

Digital Object Identifier 10.1109/ACCESS.2015.2503266

Fully Parallel Turbo Equalization for Wireless Communications

HOANG ANH NGO, ROBERT G. MAUNDER, AND LAJOS HANZO
Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K.

Corresponding author: L. Hanzo (lh@ecs.soton.ac.uk)

The financial support of the EPSRC, Swindon UK under the grants EP/J015520/1 and EP/L010550/1, as well as that of the TSB, Swindon UK under the auspices of grant TS/L009390/1 is gratefully acknowledged.

ABSTRACT Iterative turbo equalization is capable of achieving impressive performance gains over the conventional non-iterative equalization having the same complexity, when communicating over channels that suffer from intersymbol interference (ISI). The state-of-the-art turbo equalizers employ the logarithmic Bahl–Cocke–Jelinek–Raviv (Log-BCJR) algorithm. However, due to the specific nature of serial data processing, the Log-BCJR algorithm introduces significant processing delays at the receiver. Therefore, in low-latency applications having a high throughput, the turbo equalizer might be deemed less attractive than its conventional counterparts. In order to circumvent this problem, in this paper, we conceived a novel fully parallel turbo equalization algorithm, which is capable of significantly reducing the data processing delay and, hence, improving both the processing latency and the attainable throughput at the receiver. The fully parallel equalizer is then combined with the fully parallel turbo decoder for improving the system performance achieved in terms of the bit error ratio. Furthermore, we propose a novel odd–even interleaver design for employment between the fully parallel equalizer and the fully parallel turbo decoder in order to reduce complexity by 50% in fully parallel turbo equalization arrangements, while retaining a comparable performance. Finally, we compare the computational complexity, latency, throughput, hardware resource requirements, and the bit error ratio of the proposed fully parallel scheme to those of a Log-BCJR-based turbo equalizer benchmarker.

INDEX TERMS Fully-parallel turbo equalization, iterative equalization and decoding.

NOMENCLATURE

ACRONYMS

ARP	Almost regular Permutation
AWGN	Additive White Gaussian Noise
BER	Bit Error Ratio
BPSK	Binary Phase Shift Keying
Log-BCJR	Logarithmic Bahl-Cocke-Jelinek-Raviv
FPTD	Fully-Parallel Turbo Decoder
FPTDS	Fully-Parallel Turbo Detection Scheme
FPE	Fully-Parallel Equalizer
LLR	Logarithmic Likelihood Ratio
LTE	Long Term Evolution
NSW	Non-Slide Windows
QPP	Quadratic Polynomial Permutation
RAM	Random Access Memory
WCDMA	Wideband Code Division Multiple Access

LIST OF SYMBOLS

$\mathbf{b}$	The multiplexed bit vector at the transmitter
$\mathbf{b}_1$	The message bit vector at the transmitter
$\mathbf{b}_2$	The parity bit vector at the transmitter
$\mathbf{b}_3$	The systematic bit vector at the transmitter
$\bar{\mathbf{b}}^a$	The a priori multiplexed LLR vector at the receiver
$\bar{\mathbf{b}}^e$	The extrinsic multiplexed LLR vector at the receiver
$\bar{\mathbf{b}}_1^a$	The a priori message LLR vector at the turbo decoder
$\bar{\mathbf{b}}_1^e$	The extrinsic message LLR vector at the turbo decoder
$\bar{\mathbf{b}}_2^a$	The a priori parity LLR vector at the turbo decoder
$\bar{\mathbf{b}}_2^e$	The extrinsic parity LLR vector at the turbo decoder
$\bar{\mathbf{b}}_3^a$	The a priori systematic LLR vector at the turbo decoder
$\bar{\mathbf{b}}_3^e$	The extrinsic systematic LLR vector at the turbo decoder
$\bar{\mathbf{b}}_3^p$	The *a posteriori* systematic LLR vector at the turbo decoder
$\mathbf{c}$	The interleaved bit vector at the transmitter
$\mathbf{c}^c$	The transmitted symbol vector at the transmitter
$\bar{\mathbf{c}}^a$	The *a priori* LLR vector of the equalizer
$\bar{\mathbf{c}}^c$	The received symbol vector of the equalizer
$\bar{\mathbf{c}}^e$	The extrinsic LLR vector of the equalizer
$\mathbf{n}$	The additive white Gaussian noise

$\bar{\boldsymbol{\alpha}}^{\mathrm{a}}$	The *a priori* forward state metric of the turbo decoder
$\bar{\boldsymbol{\alpha}}^{\mathrm{e}}$	The extrinsic forward state metric of the turbo decoder
$\bar{\boldsymbol{\beta}}^{\mathrm{a}}$	The *a priori* backward state metric of the turbo decoder
$\bar{\boldsymbol{\beta}}^{\mathrm{e}}$	The extrinsic backward state metric of the turbo decoder
$\bar{\boldsymbol{\alpha}}^{\mathrm{E,a}}$	The *a priori* forward state metric of the equalizer
$\bar{\boldsymbol{\alpha}}^{\mathrm{E,e}}$	The extrinsic forward state metric of the equalizer
$\bar{\boldsymbol{\beta}}^{\mathrm{E,a}}$	The *a priori* backward state metric of the equalizer
$\bar{\boldsymbol{\beta}}^{\mathrm{E,e}}$	The extrinsic backward state metric of the equalizer
C	The computational complexity
D	The time period duration
I	The number of decoding iterations
I^I	The number of equalizer-to-turbo-decoder iterations
I^O	The number of turbo-decoder iterations
L	The number of states in the equalizer trellis
M	The number of states in the turbo code trellis
N	Frame length
S_k	The k^{th} state of the trellis
T	The number of time periods per decoding iteration
X	The computational resource requirement
Y	The register resource requirement
Z	The RAM resource requirement
h	The fading coefficient
k	Bit/symbol/state index
l	Tap index
l	Lower encoder/decoder
u	Upper encoder/decoder

I. INTRODUCTION

Berrou and his team [1] proposed the first turbo equalisation scheme, where the equalizer and the channel decoder exchange their soft-decision based information by performing iterative detection in order to gradually eliminate the channel-induced Inter-Symbol Interference (ISI). Inspired by this contribution, this problem was further investigated by a large number of researches [2], [3]. As shown in [4] and [5], the turbo equalizers offer a substantially improved performance over the family of non-iterative linear equalizers [6], [7]. The closely-related family of turbo codes [8], [9] has been adopted for providing error correction in a number of advanced communication systems, such as the 3rd-Generation Wideband Code Division Multiple Access (3G WCDMA) [10], [11] and the 4th-Generation Long Term Evolution (4G LTE) systems [12]. A turbo detection scheme [13], [14] may comprise a serial concatenation of an equalizer with a turbo decoder, which comprises a parallel concatenation of two component convolutional decoders. By iteratively exchanging soft information in the form of Logarithmic Likelihood Ratios (LLRs) [8] between the equalizer and the pair of constitute convolutional decoders of the turbo code, the resultant turbo detection scheme is capable of facilitating reliable communications at transmission throughputs that approach the channel capacity [3], [15]. Classic turbo detection schemes typically employ the Logarithmic Bahl-Cocke-Jelinek-Raviv (Log-BCJR) algorithm [16]. This is successively applied to the equalizer and to the two convolutional decoders, until an error-free decoded frame is obtained or until the maximum number of decoding iterations is reached. However, the Log-BCJR algorithm has an inherently serial processing nature, owing to the data dependencies within its forward and backward recursions as detailed in [3]. This limits both the achievable processing throughput and the latency of conventional turbo detection schemes, which imposes a bottleneck both on the transmission throughput and on the end-to-end latency in real-time communication systems.

A number of techniques have been proposed for increasing the grade of parallelism and hence for improving both the processing throughput and latency of Log-BCJR turbo decoders although these techniques have only found limited application to turbo equalizers. These solutions include shuffled iterative decoding [17], sub-block parallelism [18], [19], the Radix-4 transform [20] and the Non-Sliding Window (NSW) technique [20]. These techniques allow both recursions of both convolutional decoders to be performed simultaneously, as well as allowing the recursions to consider several turbo-encoded bits per time period. However, in each case, the data dependencies of the forward and backward recursions require the turbo encoded bits of each convolutional decoder to be processed serially, spread over numerous consecutive time periods. As a result, each turbo decoding iteration requires hundreds or even thousands of processing time periods, hence limiting the attainable processing throughput of the state-of-art turbo decoder [20] to 2.15 Gbit/s, which is far below the 10 Gbit/s target of the emerging 5G systems [21].

Against this background, we previously proposed the Fully-Parallel Turbo Decoder (FPTD) algorithm [22], where all turbo-encoded bits in the frame may be decoded in parallel, allowing each turbo decoder iteration to be completed using just one or two time periods. This offers a more than six-fold processing throughput and latency improvement over the state-of-the-art Log-BCJR turbo decoder, when employed for the LTE turbo code [22]. As a result, the FPTD facilitates both processing throughputs exceeding 10 Gbit/s and ultra-low processing latencies, hence satisfying the challenging requirements of 5G for the first time. The milestones of the development of the iterative turbo decoding and turbo equalization are shown in Table 1.

Against this background, in this paper we propose a novel Fully-Parallel Turbo Detection Scheme (FPTDS) for high-throughput and low-latency applications. Our novel contributions are detailed as follows:

1) We propose a novel Fully-Parallel Equalizer (FPE), as well as FPTDS, where the FPE is operated in parallel with the FPTD conceived in [22] and [26].
2) We propose a novel odd-even interleaver for the proposed FPTDS in order to reduce the complexity of the

TABLE 1. Development of turbo decoding and turbo equalization.

1991	Turbo codes were invented by Berrou's team.
1993	Turbo codes were publicly unveiled by Berrou, Glavieux, and Thitimajshima [8].
1995	Turbo equalization was introduced by Douillard, Jezequel, and Berrou [1].
1997	Glavieux, Laot, and Labat demonstrated that a linear equalizer may be used in a turbo equalizer framework [23]. The combination of equalization and turbo decoding was investigated by Raphaeli and Zarai [13]. Turbo equalization was further investigated by Wang and Poor [24].
1999	EXIT charts were proposed by ten Brink [25] for characterising the iterative decoding convergence of turbo codes.
2002	Turbo equalization was further investigated by Tuchler [2].
2006	The partial parallel processing concept was proposed for convolutional turbo decoding [18].
2012	The Radix-4 transform and the Non-Slide Window techniques were proposed for improving the throughput and latency of state-of-the-art turbo decoding [20].
2015	Maunder [22] proposed FPTD, which is capable of satisfying the challenging 10Gbit/s requirement of 5G for the first time.
2016	EXIT charts were proposed by Ngo, Maunder and Hanzo [26] for characterising the iterative decoding convergence of FPTD.

system by 50%, while maintaining a comparable Bit Error Ratio (BER).

3) We quantified the computational complexity, latency, throughput, hardware resource requirements as well as BER of the proposed FPTDS and compared them to those of the conventional Log-BCJR turbo detection benchmarkers.

The outline of the paper is as follows. Section II describes our novel FPTDS, where the novel FPE and the FPTD are operated in parallel. Our novel odd-even interleaver Conceived for reducing the computational complexity of the FPTDS is proposed in Section III. Section IV investigates the computational complexity, throughput, hardware resource requirements of the FPTDS and compare them to those of the Log-BCJR benchmarkers. The BER performance of the proposed FPTDS is quantified and compared to the benchmarkers in Section V. Finally, our concluding remarks are offered in Section VI.

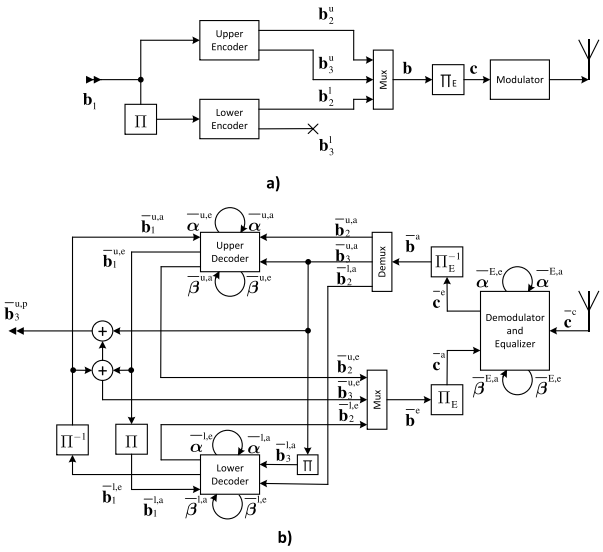


FIGURE 1. The iterative equalization and decoding system. (a) Transmitter. (b) Receiver.

II. SYSTEM ARCHITECTURE

The architecture of the proposed FPTDS is shown in Fig.1. In the turbo encoder [8] of the transmitter, a message bit vector $\mathbf{b}_1^{\mathrm{u}} = [b_{1,k}^{\mathrm{u}}]_{k=1}^{N}$ comprising N number of bits is encoded by the upper convolutional encoder, generating the parity bit vector $\mathbf{b}_2^{\mathrm{u}} = [b_{2,k}^{\mathrm{u}}]_{k=1}^{N}$ and the systematic bit vector $\mathbf{b}_3^{\mathrm{u}} = [b_{3,k}^{\mathrm{u}}]_{k=1}^{N} = \mathbf{b}_1^{\mathrm{u}}$. Meanwhile, the message bit vector $\mathbf{b}_1^{\mathrm{u}}$ is interleaved by the block Π in order to obtain the interleaved message bit vector $\mathbf{b}_1^{\mathrm{l}} = [b_{1,k}^{\mathrm{l}}]_{k=1}^{N}$ and then it is encoded by the lower convolutional encoder to produce the parity bit vector $\mathbf{b}_2^{\mathrm{l}} = [b_{2,k}^{\mathrm{l}}]_{k=1}^{N}$. Following this, the systematic bit vector $\mathbf{b}_3^{\mathrm{u}}$ and the parity bit vectors $\mathbf{b}_2^{\mathrm{u}}$ and $\mathbf{b}_2^{\mathrm{l}}$ are multiplexed in order to form the bit vector $\mathbf{b} = [b_k]_{k=1}^{3N}$ comprising $3N$ bits, which is then interleaved by the block Π_E of Fig. 1 into the bit vector $\mathbf{c} = [c_k]_{k=1}^{3N}$. Finally, the bit vector $\mathbf{c}$ is modulated, resulting in the symbol vector $\mathbf{c}^c = [c_k^{\mathrm{c}}]_{k=1}^{3N}$. For simplicity, Binary Phase Shift Keying (BPSK) modulation is assumed, according to $c_k^c = 2c_k - 1$.

The symbol vector $\mathbf{c}^c$ is assumed to travel through a wireless channel which is contaminated by ISI caused by

multipath fading having ℓ taps and then by the Additive Gaussian White Noise (AWGN) $\mathbf{n} = [n_k]_{k=1}^{3N}$ having a noise variance of σ^2. The fading coefficients $[h_l]_{l=0}^{\ell}$ obey the Rayleigh distribution and are normalized to a unity power. The received signal may be modelled as:

$$\bar{c}_k^c = \sum_{l=0}^{\ell} h_l c_{k-l}^c + n_k, \quad k = 1, 2, \ldots, 3N. \tag{1}$$

In the following sections, we will describe the conventional Log-BCJR turbo detection scheme and the novel FPTDS. In each section, we will detail the equalizer, the turbo decoder and the iterative turbo equalization and decoding operations exchanging soft-information between them.

A. CONVENTIONAL LOG-BCJR TURBO DETECTION

The conventional equalization and decoding may rely on I^I equalizer-to-turbo-decoder and I^O turbo-decoder iterations. The equalizer-to-turbo-decoder iterations are carried out between the equalizer and the turbo decoder, while the turbo-decoder iterations are performed between the two component decoders of the turbo decoder. The equalizer-to-turbo-decoder iterations between the equalizer and the turbo decoder are continued, until no more errors are detected or until reaching the maximum affordable number of equalizer-to-turbo-decoder iterations. The presence of errors may be detected using classic error detection codes, such as Cyclic Redundancy Check (CRC) codes.

1) LOG-BCJR EQUALIZER

The received signal vector of $3N$ symbol $\bar{\mathbf{c}}^c$ is first equalized by the equalizer, where the Log-BCJR equalization algorithm [4] is employed. As seen in Fig. 1, the inputs of the turbo equaliser comprise the symbol vector $\bar{\mathbf{c}}^{\mathrm{c}} = [\bar{c}_k^{\mathrm{c}}]_{k=1}^{3N}$ received from the channel and the *a priori* LLR vectors $\bar{\mathbf{c}}^{\mathrm{a}} = [\bar{c}_k^{\mathrm{a}}]_{k=1}^{3N}$ gleaned from the turbo decoders.

In response, the turbo equaliser forwards the extrinsic LLR vector $\bar{\mathbf{c}}^{\mathrm{e}} = [\bar{c}_k^{\mathrm{e}}]_{k=1}^{3N}$ to the turbo decoder. Before being forwarded to the turbo decoder, the extrinsic LLR vector $\bar{\mathbf{c}}^{\mathrm{e}}$ is deinterleaved by the block Π_E^{-1} of Fig. 1 into the vector of turbo-encoded LLRs $\bar{\mathbf{b}}^{\mathrm{a}} = [\bar{b}_k^{\mathrm{a}}]_{k=1}^{3N}$ and then demultiplexed into three *a priori* LLR vectors $\bar{\mathbf{b}}_2^{\mathrm{u,a}}$, $\bar{\mathbf{b}}_2^{\mathrm{l,a}}$ and $\bar{\mathbf{b}}_3^{\mathrm{u,a}}$, where the latter is deinterleaved by the block Π^{-1} of Fig. 1 to obtain $\bar{\mathbf{b}}_3^{\mathrm{l,a}}$.

In each iteration, the equalizer will sequentially compute the $3N$ extrinsic LLRs of $\bar{\mathbf{c}}^e$ based on (2)-(6). More specifically, the equalizer uses (2) to combine the $3N$ LLRs $\bar{c}_k^{\mathrm{c}}$ and the $3N$ LLRs $\bar{c}_k^{\mathrm{a}}$ gleaned from the channel and from the FPTD, respectively, in order to to produce an *a priori* transition metric $\bar{\gamma}_k^{\mathrm{E}}(S_{k-1}, S_k)$ for each transition in the $L = \log_2(\ell-1)$-state trellis [4], namely for each pair of states S_{k-1} and S_k, for which it is possible for the equalizer to transition between, as indicated using the notation $b(S_{k-1}, S_k) = 1$. Then the $3N$ extrinsic forward state metric vectors $\bar{\boldsymbol{\alpha}}^{\mathrm{E}} = \left[\bar{\boldsymbol{\alpha}}_k^{\mathrm{E}} = [\bar{\alpha}_k^{\mathrm{E}}(S_k)]_{S_k=0}^{M-1}\right]_{k=1}^{3N}$ and the $3N$ extrinsic backward state metric vectors $\bar{\boldsymbol{\beta}}^{\mathrm{E}} = \left[\bar{\boldsymbol{\beta}}_{k-1}^{\mathrm{E}} = [\bar{\beta}_{k-1}^{\mathrm{E}}(S_{k-1})]_{S_{k-1}=0}^{M-1}\right]_{k=1}^{3N}$ are computed by (3) and (4), respectively. As shown in (3), the k^{th} forward metric $\bar{\alpha}_k^{\mathrm{E}}(S_k)$ depends on the $(k-1)^{\mathrm{th}}$ forward metric $\bar{\alpha}_{k-1}^{\mathrm{E}}(S_k)$. Therefore, the $3N^{\mathrm{th}}$ forward metric $\bar{\alpha}_{3N}^{\mathrm{E}}(S_k)$ depends on the $(3N-1)^{\mathrm{th}}$ previous forward metrics. Consequently, the forward recursion is spread over $3N$ time periods, resulting in a slow processing. This is similar in the backward recursion.

$$\bar{\delta}^{\mathrm{E}}(S_{k-1}, S_k) = \Bigg[b_0(S_{k-1}, S_k) \cdot \bar{c}_k^{\mathrm{a}} + \left| \bar{c}_k^{\mathrm{c}} - \sum_{l=0}^{\ell} (2b_l(S_{k-1}, S_k) - 1) \cdot h_l \right|^2 / (2\sigma^2) \Bigg], \tag{2}$$

$$\bar{\alpha}_k^{\mathrm{E}}(S_k) = \max_{\{S_{k-1} | c(S_{k-1}, S_k)=1\}}{}^{*} \left[\gamma_k^{\mathrm{E}}(S_{k-1}, S_k) + \bar{\alpha}_{k-1}^{\mathrm{E}}(S_{k-1}) \right], \tag{3}$$

$$\bar{\beta}_{k-1}^{\mathrm{E}}(S_{k-1}) = \max_{\{S_k | c(S_{k-1}, S_k)=1\}}{}^{*} \left[\gamma_k^{\mathrm{E}}(S_{k-1}, S_k) + \bar{\beta}_k^{\mathrm{E}}(S_k) \right], \tag{4}$$

$$\bar{\delta}^{\mathrm{E}}(S_{k-1}, S_k) = \bar{\gamma}_k^{\mathrm{E}}(S_{k-1}, S_k) + \bar{\alpha}_{k-1}^{\mathrm{E}}(S_{k-1}) + \bar{\beta}_k^{\mathrm{E}}(S_k), \tag{5}$$

$$\bar{c}_k^{\mathrm{e}} = \left[\max_{\{(S_{k-1}, S_k) | b_1(S_{k-1}, S_k)=1\}}{}^{*} [\bar{\delta}^{\mathrm{E}}(S_{k-1}, S_k)] \right] - \left[\max_{\{(S_{k-1}, S_k) | b_1(S_{k-1}, S_k)=0\}}{}^{*} [\bar{\delta}^{\mathrm{E}}(S_{k-1}, S_k)] \right] - \bar{c}_k^{\mathrm{a}}. \tag{6}$$

Equations (3) and (4) employ the Jacobian logarithm, which is defined for two operands as [8]

$$\max^*(\bar{\delta}_1, \bar{\delta}_2) = \max(\bar{\delta}_1, \bar{\delta}_2) + \ln\left(1 + e^{-|(\bar{\delta}_1 - \bar{\delta}_2)|}\right), \tag{7}$$

and may be extended to more operands by exploiting its associative property. Alternatively, the exact max* of (7) may be approximated by [27]

$$\max^*(\bar{\delta}_1, \bar{\delta}_2) = \max(\bar{\delta}_1, \bar{\delta}_2) \tag{8}$$

Thereafter, an *a posteriori* transition metrics $\bar{\delta}_k^{\mathrm{E}}(S_{k-1}, S_k)$ is produced by (12) for each transition between the state S_{k-1} and S_k in the trellis. Finally, (13) is employed to generate the vector of $3N$ extrinsic LLRs $\bar{\mathbf{c}}^{\mathrm{e}} = [\bar{c}_k^{\mathrm{e}}]_{k=1}^{3N}$, which will be forwarded to the channel decoder.

2) LOG-BCJR TURBO DECODER

Again, the classic turbo decoder includes a pair of convolutional component decoders, where both rely on the Log-BCJR decoding algorithm [8], [22]. As illustrated in Fig. 1, the inputs of each component decoder comprise the *a priori* systematic LLR vector $\bar{\mathbf{b}}_3^{\mathrm{a}} = [\bar{b}_{3,k}^{\mathrm{a}}]_{k=1}^{N}$ and the *a priori* parity LLR vector $\bar{\mathbf{b}}_2^{\mathrm{a}} = [\bar{b}_{2,k}^{\mathrm{a}}]_{k=1}^{N}$ from the equalizer, as well as the *a priori* message LLR vector $\bar{\mathbf{b}}_1^{\mathrm{a}} = [\bar{b}_{1,k}^{\mathrm{a}}]_{k=1}^{N}$ from the other component decoder. Meanwhile, the outputs comprise the extrinsic message LLR vector $\bar{\mathbf{b}}_1^{\mathrm{e}} = [\bar{b}_{1,k}^{\mathrm{e}}]_{k=1}^{N}$ for the other decoder and the encoded extrinsic LLR vector $\bar{\mathbf{b}}_2^{\mathrm{e}} = [\bar{b}_{2,k}^{\mathrm{e}}]_{k=1}^{N}$ for the equalizer. For convenience, the superscripts ^{u} and ^{l}

are omitted in this section and thereafter, wherever our discussions are equivalent for the upper and lower convolutional decoders.

Similar to the equalizer, the decoding operations of the component decoders employ the Log-BCJR algorithm based on (9)-(14). More specifically, each component uses (9) to combine the *a priori* LLRs $\bar{b}^{\mathrm{a}}_{1,k}$, $\bar{b}^{\mathrm{a}}_{2,k}$ and $\bar{b}^{\mathrm{a}}_{3,k}$ to produce an *a priori* transition metric $\bar{\gamma}_k(S_{k-1}, S_k)$ for each pair of transition states S_{k1} and S_k, for which it is possible for the convolutional encoder to traverse between, as indicated using the notation $c(S_{k1}, S_k) = 1$. Here, $b_j(S_{k1}, S_k)$ is the value that is implied for the bit $b_{j,k}$ by the transition between the state S_{k1} and S_k, according to the state transition diagram [22]. These vectors of transition metrics are then combined according to (10)-(11), in order to produce the vector of N extrinsic forward state metric vectors $\bar{\boldsymbol{\alpha}} = \left[\bar{\boldsymbol{\alpha}}^{\mathrm{e}}_k = [\bar{\alpha}^{\mathrm{e}}_k(S_k)]^{M-1}_{S_k=0}\right]^N_{k=1}$, and the vector of N extrinsic backward state metric vectors $\bar{\boldsymbol{\beta}} = \left[\bar{\boldsymbol{\beta}}^{\mathrm{e}}_{k-1} = [\bar{\beta}^{\mathrm{e}}_{k-1}(S_{k-1})]^{M-1}_{S_{k-1}=0}\right]^N_{k=1}$, respectively. Like the Log-BCJR equalizer, the forward and backward recursions in the Log-BCJR turbo decoder are also spread over N periods, hence resulting in a slow serial processing.

$$\bar{\gamma}_k(S_{k-1},S_k) = \sum_{j=1}^{3}[b_j(S_{k-1}, S_k) \cdot \bar{b}^{a}_{j,k}], \tag{9}$$

$$\bar{\alpha}_k(S_k) = \max^*_{\{S_{k-1}|b_1(S_{k-1},S_k)=1\}}\left[\bar{\gamma}_k(S_{k-1}, S_k)+\bar{\alpha}_{k-1}(S_{k-1})\right], \tag{10}$$

$$\bar{\beta}_{k-1}(S_{k-1}) = \max^*_{\{S_k|b_1(S_{k-1},S_k)=1\}}\left[\bar{\gamma}_k(S_{k-1}, S_k) + \bar{\beta}_k(S_k)\right], \tag{11}$$

$$\bar{\delta}(S_{k-1}, S_k) = \bar{\gamma}_k(S_{k-1}, S_k) + \bar{\alpha}_{k-1}(S_{k-1}) + \bar{\beta}_k(S_k), \tag{12}$$

$$\bar{b}^{e}_{1,k} = \left[\max^*_{\{(S_{k-1},S_k)|b_1(S_{k-1},S_k)=1\}}[\bar{\delta}(S_{k-1}, S_k)]\right] - \bar{b}^{a}_{1,k} - \left[\max^*_{\{(S_{k-1},S_k)|b_1(S_{k-1},S_k)=0\}}[\bar{\delta}(S_{k-1}, S_k)]\right] - \bar{b}^{a}_{3,k}, \tag{13}$$

$$\bar{b}^{e}_{2,k} = \left[\max^*_{\{(S_{k-1},S_k)|b_2(S_{k-1},S_k)=1\}}[\bar{\delta}(S_{k-1}, S_k)]\right] - \left[\max^*_{\{(S_{k-1},S_k)|b_2(S_{k-1},S_k)=0\}}[\bar{\delta}(S_{k-1}, S_k)]\right] - \bar{b}^{a}_{2,k}. \tag{14}$$

Thereafter, an *a posteriori* transition metric $\delta(S_{k-1}, S_k)$ is computed by (12) for each transition between the states S_{k-1} and S_k in the trellis, which is then substituted into (13) and (14) for generating the uncoded and encoded extrinsic LLR vector $\bar{\boldsymbol{b}}^{\mathrm{e}}_{1,k} = \left[\bar{b}^{\mathrm{e}}_{1,k}\right]^N_{k=1}$ and $\bar{\boldsymbol{b}}^{\mathrm{e}}_{2,k} = \left[\bar{b}^{\mathrm{e}}_{2,k}\right]^N_{k=1}$, respectively. Again, these equations rely on the Jacobian logarithm of (7). Following the final turbo-decoder iteration between the two component decoders, an *a posteriori* LLR pertaining to the k^{th} message bit $b^{\mathrm{u}}_{1,k}$ may be obtained as

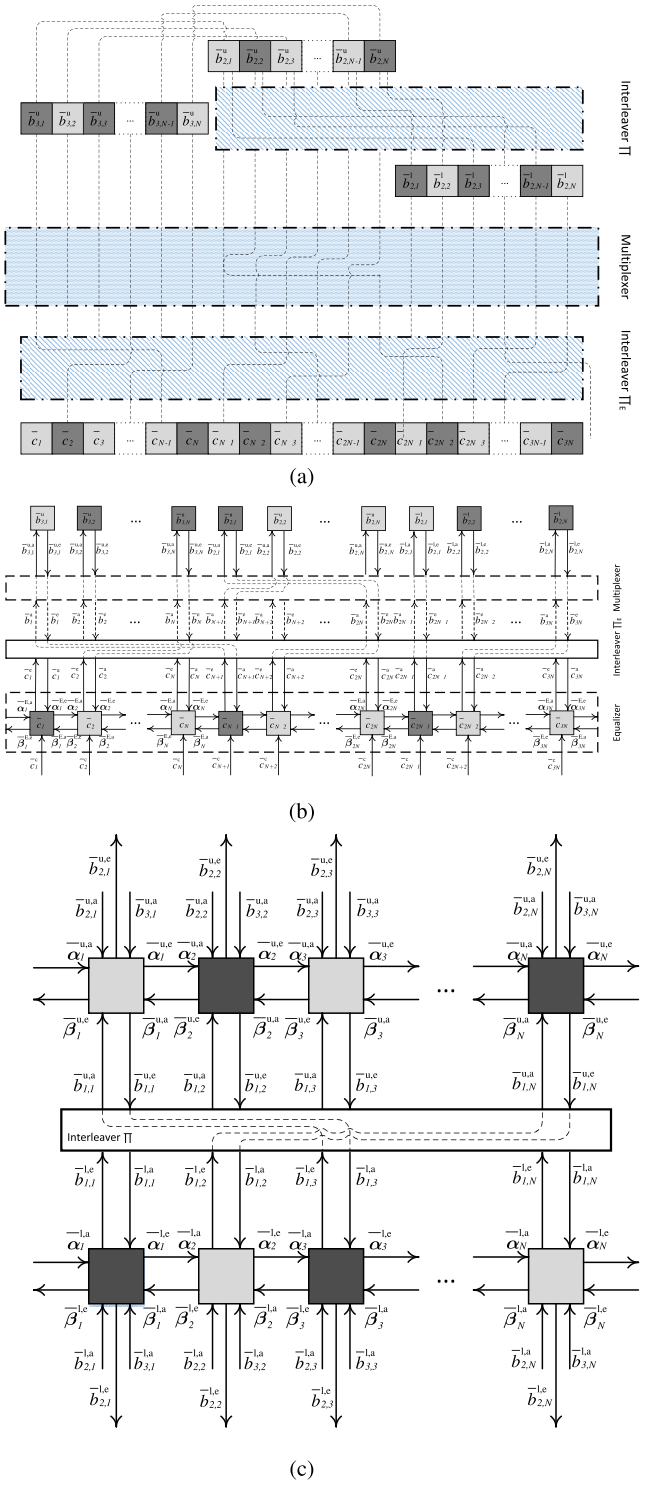


FIGURE 2. 2D EXIT charts of the FPTD at different E_b/N_0 values. A LTE $M = 8$-state turbo code [12] having a coding rate of 1/3 is employed along with BPSK modulation for communication over a Rayleigh fading channel. (a) Block connection of the FPTDS. (b) Block diagram of the FPE. (c) Block diagram of the FPTD.

$\bar{b}^{\mathrm{u,p}}_{3,k} = \bar{b}^{\mathrm{u,e}}_{1,k} + \bar{b}^{\mathrm{u,a}}_{1,k} + \bar{b}^{\mathrm{u,a}}_{3,k}$. A hard decision for the message bit $b^{\mathrm{u}}_{3,k}$ may then be obtained as the result of the binary test $\bar{b}^{\mathrm{u,p}}_{1,k} > 0$.

The extrinsic message LLR vectors $\bar{\mathbf{b}}_1^{\mathrm{u,e}}$ and $\bar{\mathbf{b}}_1^{\mathrm{l,e}}$ are iteratively exchanged between the upper and lower component decoders for I^O iterations. Following this, $\bar{\mathbf{b}}_1^{\mathrm{a,e}}$ and $\bar{\mathbf{b}}_1^{\mathrm{a,e}}$ are summed to provide $\bar{\mathbf{b}}_3^{\mathrm{u,e}}$. Then $\bar{\mathbf{b}}_3^{\mathrm{u,e}}$, $\bar{\mathbf{b}}_2^{\mathrm{u,e}}$ and $\bar{\mathbf{b}}_2^{\mathrm{l,e}}$ are multiplexed to obtain $\bar{\mathbf{b}}\mathrm{e}$ and interleaved to obtain $\bar{\mathbf{c}}^{\mathrm{a}}$.

B. FULLY-PARALLEL TURBO DETECTION

In contrast to the conventional turbo detection scheme discussed in Section II-A, all of the symbols in the received symbol vector $\bar{\mathbf{c}}^c$ may be simultaneously equalized by a FPE and all of the corresponding LLRs may be simultaneously decoded by a FPTD [22] in the FPTDS of Fig. 2, eliminating the requirement for equalizer-to-turbo-decoder and turbo-decoder iterations in the system. Instead, in each iteration of the proposed FPTDS, the extrinsic LLR vector is passed from the FPE to the FPTD through the deinterleaver Π_E^{-1} and the demultiplexer of Fig. 1, while that of the FPTD is forwarded to the FPE through the multiplexer and the interleaver Π_E of Fig. 1 and Fig. 2a.

1) FPE

The FPE comprises $3N$ algorithmic decoding blocks, as detailed in Fig. 2b. Observe in Fig. 2b that the inputs of the FPE comprise the vector of $3N$ symbols $\bar{\mathbf{c}}^{\mathrm{c}} = [\bar{c}_k^{\mathrm{c}}]_{k=1}^{3N}$ received from the channel, the *a priori* message LLR vector $\bar{\mathbf{c}}^{\mathrm{a}} = [\bar{c}_k^{\mathrm{a}}]_{k=1}^{3N}$ received from the FPTD during the previous time period, the *a priori* forward state metric vectors $\bar{\boldsymbol{\alpha}}^{\mathrm{E,a}} = [\bar{\boldsymbol{\alpha}}_k^{\mathrm{E,a}}]_{k=0}^{3N-1}$ and the *a priori* backward state metric vectors $\bar{\boldsymbol{\beta}}^{\mathrm{E,a}} = [\bar{\boldsymbol{\beta}}_k^{\mathrm{E,a}}]_{k=1}^{3N}$, which are fed back from the FPE during the previous time period. Meanwhile, the output of the FPE includes the extrinsic LLR vector $\bar{\mathbf{c}}^{e} = [\bar{c}_k^{\mathrm{e}}]_{k=1}^{3N}$, the forward state metric vectors $\bar{\boldsymbol{\alpha}}^{E,e} = [\bar{\boldsymbol{\alpha}}_k^{\mathrm{E,e}}]_{k=0}^{N}$ and the backward state metric vectors $\bar{\boldsymbol{\beta}}^{\mathrm{E,e}} = [\bar{\boldsymbol{\beta}}_k^{\mathrm{E,e}}]_{k=0}^{N}$, which will be fed forward to the FPE for use during the next time period. Before being forwarded to the FPTD, the extrinsic LLR vector $\bar{\mathbf{c}}^{\mathrm{e}}$ is deinterleaved into the vector $\bar{\mathbf{b}}^{\mathrm{a}} = [\bar{b}_k^{\mathrm{a}}]_{k=1}^{3N}$ and then demultiplexed into three *a priori* LLR vectors $\bar{\mathbf{b}}_2^{\mathrm{u,a}}$, $\bar{\mathbf{b}}_2^{\mathrm{l,a}}$ and $\bar{\mathbf{b}}_3^{\mathrm{u,a}}$, where the latter is interleaved in the block Π to obtain the *a priori* LLR vector $\bar{\mathbf{b}}_3^{\mathrm{l,a}}$ of Fig. 1.

$$\begin{aligned}\bar{\delta}^{\mathrm{E}}(S_{k-1},S_k) = {} & \bar{\alpha}_{k-1}^{\mathrm{E,a}}(S_{k-1}) + \bar{\beta}_k^{\mathrm{E,a}}(S_k) \\ & + \Bigg[b_0(S_{k-1},S_k)\cdot \bar{c}_k^{\mathrm{a}} \\ & + \Big| \bar{c}_k^{\mathrm{c}} - \sum_{l=0}^{\ell} (2b_l(S_{k-1},S_k)-1)\cdot h_l \Big|^2 / (2\sigma^2) \Bigg],\end{aligned} \tag{15}$$

$$\bar{\alpha}_k^{\mathrm{E,e}}(S_k) = \left[\max_{\{S_{k-1}|c(S_{k-1},S_k)=1\}}{}^{*} [\bar{\delta}^{\mathrm{E}}(S_{k-1},S_k)] \right] - \bar{\beta}_k^{\mathrm{E,a}}(S_k), \tag{16}$$

$$\bar{\beta}_{k-1}^{\mathrm{E,e}}(S_{k-1}) = \left[\max_{\{S_k|c(S_{k-1},S_k)=1\}}{}^{*} [\bar{\delta}^{\mathrm{E}}(S_{k-1},S_k)] \right] - \bar{\alpha}_{k-1}^{\mathrm{E,a}}(S_{k-1}), \tag{17}$$

$$\begin{aligned}\bar{c}_k^{\mathrm{e}} = {} & \left[\max_{\{(S_{k-1},S_k)|b_1(S_{k-1},S_k)=1\}}{}^{*} [\bar{\delta}^{\mathrm{E}}(S_{k-1},S_k)] \right] \\ & - \left[\max_{\{(S_{k-1},S_k)|b_1(S_{k-1},S_k)=0\}}{}^{*} [\bar{\delta}^{\mathrm{E}}(S_{k-1},S_k)] \right] - \bar{c}_k^{\mathrm{a}}.\end{aligned} \tag{18}$$

In each time period, some or all of the $3N$ algorithmic blocks of the FPE will compute the outputs based on (15)-(18) at the same time. More specifically, the algorithmic block having the index k uses (15) to combine the received symbol $\bar{c}_k^{\mathrm{c}}$ and the *a priori* LLR $\bar{c}_k^{\mathrm{a}}$ gleaned from the channel and the FPTD, respectively, as well as the *a priori* state metric vectors $\bar{\boldsymbol{\alpha}}_{k-1}^{\mathrm{E,a}} = [\bar{\alpha}_k^{\mathrm{E,a}}(S_k)]_{S_k=0}^{L-1}$ and $\bar{\boldsymbol{\beta}}_k^{\mathrm{E,a}} = [\bar{\beta}_k^{\mathrm{E,e}}(S_k)]_{S_k=0}^{L-1}$ in order to produce an *a posteriori* state metric $\bar{\delta}^{\mathrm{E}}(S_{k-1},S_k)$ for each transition in the state transition diagram [22], namely for each pair of states S_{k-1} and S_k for which it is possible for the convolutional encoder to transition between, as indicated using the notation $c(S_{k-1},S_k) = 1$. Note that $b_l(S_{k-1},S_k)$ is the value that is implied for the bits c_{k-l} by the transition between $S_{k-1} \in [0, L-1]$ and $S_k \in [0, L-1]$, where $L = 2^\ell$. These *a posteriori* transition metrics are then combined with the aid of (16)-(18), in order to produce the extrinsic forward state metric vector $\bar{\boldsymbol{\alpha}}_k^{\mathrm{E,e}} = [\bar{\alpha}_k^{\mathrm{E,e}}(S_k)]_{S_k=0}^{M-1}$, the extrinsic backward state metric vector $\bar{\boldsymbol{\beta}}_{k-1}^{\mathrm{E,e}} = [\bar{\beta}_{k-1}^{\mathrm{E,e}}(S_{k-1})]_{S_{k-1}=0}^{M-1}$ and the extrinsic LLR $\bar{c}_k^{\mathrm{e}}$, respectively. Again, (16) and (17) employ the Jacobian logarithm of (7).

In contrast to the classic Log-BCJR equalizer, the forward and backward state metrics $\bar{\boldsymbol{\alpha}}_k^{\mathrm{E,e}}$ and $\bar{\boldsymbol{\beta}}_{k-1}^{\mathrm{E,e}}$ at a given period only depend on the forward and backward state metrics fed back from the previous time period. Therefore, the data dependencies of the forward and backward recursions are broken, allowing fully-parallel operation. Hence, this speeds up the processing by a factor, of which is up to $3N$.

2) FPTD

The FPTD is described and analysed in great detail in [22] and [26]. Briefly, a FPTD includes two convolutional component decoders, each of which has N algorithmic blocks. As illustrated in Fig. 2c, the inputs of each component decoder comprise the *a priori* systematic LLR vector $\bar{\mathbf{b}}_3^{\mathrm{a}} = [\bar{b}_{3,k}^{\mathrm{a}}]_{k=1}^{N}$ and the *a priori* parity LLR vector $\bar{\mathbf{b}}_2^{\mathrm{a}} = [\bar{b}_{2,k}^{\mathrm{a}}]_{k=1}^{N}$ contributed by the FPE during the previous time period, the *a priori* message LLR vector $\bar{\mathbf{b}}_1^{\mathrm{a}} = [\bar{b}_{1,k}^{\mathrm{a}}]_{k=1}^{N}$ gleaned from the other component decoder in the previous time period, the *a priori* forward state metric vectors $\bar{\boldsymbol{\alpha}}^{\mathrm{a}} = [\bar{\boldsymbol{\alpha}}_k^{\mathrm{a}}]_{k=0}^{N-1}$ and the backward state metric vectors $\bar{\boldsymbol{\beta}}^{\mathrm{a}} = [\bar{\boldsymbol{\beta}}_k^{\mathrm{a}}]_{k=1}^{N}$ fed back from the component decoder in the previous time period, where we have $\bar{\boldsymbol{\alpha}}_k^{\mathrm{a}} = [\bar{\alpha}_k^{\mathrm{a}}(S_k)]_{S_k=0}^{M-1}$, $\bar{\boldsymbol{\beta}}_{k-1}^{\mathrm{a}} = [\bar{\beta}_{k-1}^{\mathrm{a}}(S_{k-1})]_{S_{k-1}=0}^{M-1}$ and M is the number of states in the corresponding state transition diagram [22]. Meanwhile, the outputs comprise the extrinsic message LLR vector $\bar{\mathbf{b}}_1^{\mathrm{e}} = [\bar{b}_{1,k}^{\mathrm{e}}]_{k=1}^{N}$ for the other decoder, the forward state metric vectors $\bar{\boldsymbol{\alpha}}^{\mathrm{e}} = [\bar{\boldsymbol{\alpha}}_k^{\mathrm{e}}]_{k=1}^{N}$ and the backward state metric vectors $\bar{\boldsymbol{\beta}}^{\mathrm{e}} = [\bar{\boldsymbol{\beta}}_k^{\mathrm{e}}]_{k=0}^{N-1}$ which will be fed forward for processing in the next time period,

where $\bar{\boldsymbol{\alpha}}_k^{\mathrm{e}} = [\bar{\alpha}_k^{\mathrm{e}}(S_k)]_{S_k=0}^{M-1}$, $\bar{\boldsymbol{\beta}}_{k-1}^{\mathrm{e}} = [\bar{\beta}_{k-1}^{\mathrm{e}}(S_{k-1})]_{S_{k-1}=0}^{M-1}$. Again for convenience, the superscripts $^{\mathrm{u}}$ and $^{\mathrm{l}}$ are omitted in this section and thereafter, wherever our discussions are equivalent for the upper and lower convolutional decoders.

In contrast to the FPTD of [22] and [26], the FPTD here also outputs the extrinsic encoded LLR vector $\bar{\mathbf{b}}_2^{\mathrm{u,e}} = [\bar{b}_{2,k}^{\mathrm{u,e}}]_{k=1}^{N}$ and $\bar{\mathbf{b}}_2^{\mathrm{l,e}} = [\bar{b}_{2,k}^{\mathrm{l,e}}]_{k=1}^{N}$ from the upper and lower component decoder, respectively. These extrinsic parity LLR vectors $\bar{\mathbf{b}}_2^{\mathrm{u,e}}$ and $\bar{\mathbf{b}}_2^{\mathrm{l,e}}$ along with the extrinsic systematic LLR vector $\bar{\mathbf{b}}_3^{\mathrm{u,e}} = \bar{\mathbf{b}}_1^{\mathrm{u,a}} + \bar{\mathbf{b}}_1^{\mathrm{u,e}}$ are multiplexed into $\bar{\mathbf{b}}^{\mathrm{e}}$ and then they are interleaved in the block Π of Fig. 1, forming the *a priori* LLR vector $\bar{\mathbf{c}}^{\mathrm{a}}$ for the equalizer to use during the next time period.

Simultaneously with the FPE, some or possibly all of the N algorithmic blocks in each component decoder of the FPTD are operated in parallel. Each of these block performs the operation of (19)-(23). More specifically, the algorithmic block having the index k uses (19) in order to combine the *a priori* LLRs $\bar{b}_{1,k}^{\mathrm{a}}$, $\bar{b}_{2,k}^{\mathrm{a}}$ and $\bar{b}_{3,k}^{\mathrm{a}}$, as well as the *a priori* state metric vectors $\bar{\boldsymbol{\alpha}}_{k-1}^{\mathrm{a}}$ and $\bar{\boldsymbol{\beta}}_k^{\mathrm{a}}$ for producing an *a posteriori* state metric $\bar{\delta}(S_{k-1}, S_k)$ for each transition in the state transition diagram [22], namely for each pair of states S_{k-1} and S_k for which it is possible for the convolutional encoder to transition between. These *a posteriori* transition metrics are then combined by (20)-(21), in order to produce the extrinsic forward state metric vector $\bar{\boldsymbol{\alpha}}_k^{\mathrm{e}} = [\bar{\alpha}_k^{\mathrm{e}}(S_k)]_{S_k=0}^{M-1}$ and the extrinsic backward state metric vector $\bar{\boldsymbol{\beta}}_{k-1}^{\mathrm{e}} = [\bar{\beta}_{k-1}^{\mathrm{e}}(S_{k-1})]_{S_{k-1}=0}^{M-1}$, respectively. Similar to the FPE, the forward and backward state metrics $\bar{\boldsymbol{\alpha}}_k^{\mathrm{e}}$ and $\bar{\boldsymbol{\beta}}_{k-1}^{\mathrm{e}}$ at a given period only depend on the forward and backward state metrics fed back from the previous time period. Therefore, the data dependencies of the forward and backward recursions are broken, therefore allowing fully-parallel operation. Hence, the processing is sped up by a factor of up to $2N$, compared to the classic serial Log-BCJR turbo decoder.

Furthermore, the *a posteriori* transition metrics are also employed in (13) for computing the uncoded extrinsic LLR $\bar{b}_{1,k}^{\mathrm{e}}$ while the encoded extrinsic LLR $\bar{b}_{2,k}^{\mathrm{e}}$ is achieved using (23). Again, these equations employ the Jacobian logarithm of (7). Following the final decoding iteration, an *a posteriori* LLR pertaining to the k^{th} message bit $b_{1,k}^{\mathrm{u}}$ may be obtained as $\bar{b}_{1,k}^{\mathrm{u,p}} = \bar{b}_{1,k}^{\mathrm{u,e}} + \bar{b}_{1,k}^{\mathrm{u,a}} + \bar{b}_{3,k}^{\mathrm{u,a}}$. A hard decision for the message bit $b_{1,k}^{\mathrm{u}}$ may then be obtained as the result of the binary test $\bar{b}_{1,k}^{\mathrm{u,p}} > 0$.

$$\bar{\delta}(S_{k-1}, S_k) = \left[\sum_{j=1}^{3}[b_j(S_{k-1}, S_k)\cdot \bar{b}_{j,k}^{a}]\right] + \bar{\alpha}_{k-1}^{a}(S_{k-1}) + \bar{\beta}_k^{a}(S_k), \tag{19}$$

$$\bar{\alpha}_k^{e}(S_k) = \left[\max^*_{\{S_{k-1}|b_1(S_{k-1},S_k)=1\}}[\bar{\delta}(S_{k-1}, S_k)]\right] - \bar{\beta}_k^{a}(S_k), \tag{20}$$

$$\bar{\beta}_{k-1}^{e}(S_{k-1}) = \left[\max^*_{\{S_k|b_1(S_{k-1},S_k)=1\}}[\bar{\delta}(S_{k-1}, S_k)]\right] - \bar{\alpha}_{k-1}^{a}(S_{k-1}), \tag{21}$$

$$\bar{b}_{1,k}^{e} = \left[\max^*_{\{(S_{k-1},S_k)|b_1(S_{k-1},S_k)=1\}}[\bar{\delta}(S_{k-1}, S_k)]\right] - \bar{b}_{1,k}^{a} - \left[\max^*_{\{(S_{k-1},S_k)|b_1(S_{k-1},S_k)=0\}}[\bar{\delta}(S_{k-1}, S_k)]\right] - \bar{b}_{3,k}^{a}, \tag{22}$$

$$\bar{b}_{2,k}^{e} = \left[\max^*_{\{(S_{k-1},S_k)|b_2(S_{k-1},S_k)=1\}}[\bar{\delta}(S_{k-1}, S_k)]\right] - \left[\max^*_{\{(S_{k-1},S_k)|b_2(S_{k-1},S_k)=0\}}[\bar{\delta}(S_{k-1}, S_k)]\right] - \bar{b}_{2,k}^{a}. \tag{23}$$

III. INTERLEAVER DESIGN FOR THE FPTDS

By employing the odd-even interleaver [28] like that of the LTE turbo code, an odd-even operation of the algorithmic blocks may be employed in the FPTD of [22], hence reducing its complexity by 50%. More explicitly, an odd-even interleaver only connects algorithmic blocks from the upper row having an odd index to blocks from the lower row that also have an odd index. Similarly, blocks from the upper row of the FPTD having an even index are only connected to those from the lower row also having an even index. This arrangement allows the $2N$ decoding blocks of the FPTD to be grouped into two sets. The first set includes the odd-indexed blocks in the upper row and the even-indexed blocks in the lower row, which are indicated by the light grey shading in Fig. 2c. Meanwhile, the second set comprises the even-indexed blocks in the upper row and the odd-indexed blocks in the lower row, which are highlighted by the dark grey shading in Fig. 2c. Given this arrangement, the FPTD may operate only the first set in odd indexed time periods and only the second set in even indexed time periods. This reduces the computational complexity of the FPTD by 50% without increasing the number of time periods required for completing the decoding process [22]. This is because in the odd-even arrangement, operating both sets in all time periods leads to redundancy, which can be eliminated without impairing the attainable performance.

Inspired by this idea, in this section we propose a novel odd-even design of the multiplexer and interleaver Π_E of Fig. 1 between the equalizer and the channel decoder. The design is illustrated in Fig. 2a. First, the LLR vectors $\bar{\mathbf{b}}_3^{\mathrm{u,e}}$, $\bar{\mathbf{b}}_2^{\mathrm{u,e}}$ and $\bar{\mathbf{b}}_2^{\mathrm{l,e}}$ are arranged into the vector $\bar{\mathbf{b}}^{\mathrm{e}} = [\bar{b}_k^{\mathrm{e}}]_{k=1}^{3N}$ of the multiplexer. More explicitly, the vector $\bar{\mathbf{b}}_3^{\mathrm{u,e}} = [\bar{b}_{3,k}^{\mathrm{u,e}}]_{k=1}^{N}$ is placed into $[\bar{b}_k^{\mathrm{e}}]_{k=1}^{N}$. Next, the first element $\bar{b}_{2,1}^{\mathrm{u,e}}$ of the vector $\bar{\mathbf{b}}_2^{\mathrm{u,e}}$ is placed at the position $\bar{b}_{2N}^{\mathrm{e}}$ of the vector $\bar{\mathbf{b}}^{\mathrm{e}}$ while the remaining elements $[\bar{b}_{2,k}^{\mathrm{u,e}}]_{k=2}^{N}$ of vector $\bar{\mathbf{b}}_2^{\mathrm{u,e}}$ are placed from the position $\bar{b}_{N+1}^{\mathrm{e}}$ to the position $\bar{b}_{2N-1}^{\mathrm{e}}$. Finally, the vector $\bar{\mathbf{b}}_2^{\mathrm{l,e}} = [\bar{b}_{2,k}^{\mathrm{l,e}}]_{k=1}^{N}$ is placed into the remaining positions $[\bar{b}_k^{\mathrm{e}}]_{k=2N+1}^{3N}$ of the vector $\bar{\mathbf{b}}^{\mathrm{e}}$. Thereafter, an odd-even interleaver is employed for connecting the vector $\bar{\mathbf{b}}^{\mathrm{e}}$ of the multiplexer with the vector $\bar{\mathbf{c}}^{\mathrm{a}}$ of the equalizer in the same manner as between the upper and lower decoder of the FPTD [22]. The odd-even connections may employ either random or structured designs, such as the

TABLE 2. The number of operations of equalizers and decoders per decoding iteration.

Equalizer					
Equation	Log-BCJR		Equation	Fully-parallel	
	+ or -	max*		+ or -	max*
(2)	1	0			
(5)	$3.5L$	0	(15)	$3.5L+1$	0
(3)	$1.5L$	L	(16)	L	L
(4)	$1.5L$	L	(17)	L	L
(6)	2	$2L-2$	(18)	2	$2L-2$
Total/block	$C^{+}_{BE}=6.5L+3$	$C^{*}_{BE}=4L-2$	Total/block	$C^{+}_{FE}=5.5L+3$	$C^{*}_{FE}=4L-2$
Total/equalizer	$C_{BE}=(C^{+}_{BE}+C^{*}_{BE})\cdot 3N$ $=(10.5L+1)\cdot 3N$		Total/equalizer	$C_{FE}=(C^{+}_{FE}+C^{*}_{FE})\cdot 3N$ $=(9.5L+1)\cdot 3N$	
Decoder					
Equation	Log-BCJR		Equation	Fully-parallel	
	+ or -	max*		+ or -	max*
(9)	2	0			
(12)	$3.5M$	0	(19)	$3.5M+2$	0
(10)	$1.5M$	M	(20)	M	M
(11)	$1.5M$	M	(21)	M	M
(13)	3	$2M-2$	(22)	3	$2M-2$
(14)	2	$2M-2$	(23)	2	$2M-2$
Total/block	$C^{+}_{BD}=6.5M+7$	$C^{*}_{BD}=6M-4$	Total/block	$C^{+}_{FD}=5.5M+7$	$C^{*}_{FD}=6M-4$
Total/decoder	$C_{BD}=(C^{+}_{BD}+C^{*}_{BD})\cdot 2N$ $=(12.5M+3)\cdot 2N$		Total/decoder	$C_{FD}=(C^{+}_{FD}+C^{*}_{FD})\cdot 2N$ $=(11.5M+3)\cdot 2N$	

Almost Regular Permutation (ARP) and Quadratic Polynomial Permutation (QPP) interleavers [28].

By contrast, the vector $\bar{\mathbf{c}}^{\mathrm{e}}$ of the equalizer is connected to the vector $\bar{\mathbf{b}}^{\mathrm{a}}$ of the multiplexer using the same order of the odd-even interleaver. The vector $\bar{\mathbf{b}}^{\mathrm{a}}$ is further demultiplexed into three LLR vectors $\bar{\mathbf{b}}_3^{\mathrm{u,a}}$, $\bar{\mathbf{b}}_2^{\mathrm{u,a}}$ and $\bar{\mathbf{b}}_2^{\mathrm{l,a}}$ with the same order of the multiplexer.

As illustrated in Fig. 2a, the odd blocks of the vector $\bar{\mathbf{c}}$ marked by the light grey colour are connected to the odd blocks of the vector $\bar{\mathbf{b}}$ in the dark grey zone, which is further connected to the dark grey blocks of the vector $\bar{\mathbf{b}}_3^{\mathrm{u}}$, $\bar{\mathbf{b}}_2^{\mathrm{u}}$ and $\bar{\mathbf{b}}_2^{\mathrm{l}}$. Meanwhile, the even blocks of the vector $\bar{\mathbf{c}}$ in the dark grey zones are connected to the even blocks of the vector $\bar{\mathbf{b}}$ in the light grey zones, which is further connected to the light grey zones of the vector $\bar{\mathbf{b}}_3^{\mathrm{u}}$, $\bar{\mathbf{b}}_2^{\mathrm{u}}$ and $\bar{\mathbf{b}}_2^{\mathrm{l}}$. Consequently, the FPTDS are divided into the pair of sets: the dark gray set and the light grey set. In this way, the iterative exchange of the extrinsic information within the FPTDS can be instead thought of as an iterative exchange of extrinsic information between the two sets. When fully parallel equalization and decoding is employed, the operation of FPTDS relying on the odd-even interleaver corresponds to two independent processes, which have no influence on each other. Therefore, one of the two iterative processes is redundant. This can be achieved by activating the algorithmic blocks of only one set in each time period, with two consecutive time periods alternating between the two sets. By doing this, each detection is spread into $T = 2$ time periods. However, in order to achieve the same BER performance, the number of iterations required can be halved. Therefore, compared to the FPTDS where all blocks are activated in $T = 1$ time period, the FPTDS associated with the odd-even interleaver is capable of reducing the complexity by 50%, while retaining the same processing throughput.

IV. SYSTEM CHARACTERISTICS

In [22], the characteristics of the FPTD, of the Log-BCJR turbo decoder as well as of the state-of-the-art turbo decoder [20] were compared in the context of the LTE and WiMAX turbo codes. However, the NSW, radix-4 and pipelining techniques of the state-of-the-art turbo decoder have not been proposed and investigated for the equalizer. Therefore, in this section, we will compare the characteristics of the FPTDS and of the classic Log-BCJR detection scheme described in Section II. These characteristics include the computational complexity, the throughput and latency, as well as the hardware resource requirements of the iterative equalization and decoding operation.

In the FPTDS employing an odd-even interleaver, each iteration requires two time periods as described in Section III. However, it is not straightforward to define the iterations of the Log-BCJR turbo detection scheme, since it contains I^I equalizer-to-turbo-decoder iterations and each equalizer-to-turbo-decoder iteration has further I^O turbo-decoder iterations. For convenience, it is assumed that the classic Log-BCJR detection system has the number of iterations as the number of equalizer-to-turbo-decoder iterations I^I of the FPTDS. More specifically, each iteration of the Log-BCJR system contains one equalization and I^O turbo decoding operations.

The characteristics of both the FPTDS and of the Log-BCJR system are summarized in Table 3. Note that the FPTDS of Table 2 is assumed to employ the odd-even interleaver of Section III.

A. COMPUTATIONAL COMPLEXITY

The computational complexity of each trellis stage of the conventional Log-BCJR and each algorithmic block (which process one trellis stage) of the FPTDS is quantified in Table 2. The computational complexity is quantified in terms of the number of addition, subtraction and max* evaluation operations. The number of operations of the Log-BCJR equalizer is based on evaluating (2) - (6) while that of the FPE equalizer is based on (15) - (18).

In the Log-BCJR equalizer, (2) requires 1 addition operation for adding $\bar{c}_k^c$ and $\bar{c}_k^a$. Meanwhile, as $M/2\ \gamma_k$ values equal

to zero, (3) or (4) requires $3M/2$ additions of α_k/β_{k-1} and γ_k, and a further M max* evaluation operations. Similarly, (5) requires $2M$ additions between α_{k-1} as well as β_k, and a further $3M/2$ additions with γ_k. Finally, (6) needs $(2M-2)$ max* evaluations and 2 subtraction operations.

In the FPE, (15) requires $2M$ additions between α_{k-1} as well as β_k, and a further $3M/2$ additions, as $M/2$ of $2M$ transitions have both the uncoded and encoded bits equal to zero. Meanwhile, (16) and (17) require 8 max* and 8 subtractions of α_{k-1} or β_k for each equation. Since (15) is identical with (6), they have the same complexity.

Likewise, the number of operations of the Log-BCJR turbo decoder are based on (9) - (14), while that of the FPTD is based on (19) - (23). Note that (9) and (19) require one additional addition for adding a systematic LLR $\bar{b}^{\mathrm{a}}_{3,k}$, while (13) and (22) require one additional subtraction for removing a systematic LLR $\bar{b}^{\mathrm{a}}_{3,k}$.

As shown in [29], the complexity of the approximate max* operation of (7) equals to that of an addition. Therefore, in Table 2 the overall complexity of the classic Log-BCJR equalizer and of the FPE are denoted by C_{BE} and C_{FE}, respectively. Observe that in Table 2, the overall complexity of the Log-BCJR turbo decoder and of the FPTD are denoted by C_{BD} and C_{FD}.

Furthermore, the complexity of each iteration of the FPTDS C_F is equal to the summation of the complexity of both the FPTD and the FPE ($C_F = C_{FE} + C_{FD}$). By contrast, the complexity of each iteration of the classic Log-BCJR scheme equals to those of the Log-BCJR equalizer and I^O times the complexity of the Log-BCJR turbo decoder ($C_B = C_{BE} + I^O \cdot C_{BD}$). Clearly, the complexity of the Log-BCJR turbo decoder in each iteration depends on the number of turbo-decoder iterations I^O set up. Therefore, a careful considered configuration of the turbo detection is required in order to have a fair comparison between the classic Log-BCJR turbo detection and the FPTDS, which will be detailed in Section V.

B. TIME PERIODS PER DECODING

As described in Section III, a FPTDS using an odd-even interleaver requires two time periods for the dark grey and light grey groups to complete one iteration. By contrast, each component of the Log-BCJR turbo decoder requires N time periods for the computation of the forward recursion and N time periods for the backward recursion. Therefore, the Log-BCJR turbo decoder requires $4N$ time periods. Similar to each component of the Log-BCJR turbo decoder, the Log-BCJR equalizer requires $3N$ time periods for each forward and backward recursion computation. With I^O turbo-decoder iterations of the Log-BCJR turbo decoder, each iteration of the Log-BCJR turbo detection scheme requires a total of $T = 4N \cdot I^O + 6N = (2I^O + 3) \cdot 2N$ time periods. Hence, in order to complete one detection iteration, the classic Log-BCJR turbo detection scheme needs $(2I^O + 3)N$ time periods more than the FPTDS.

C. TIME PERIOD DURATION

The time period duration here is defined as the longest time for an algorithmic block to complete all computations. It depends on the dependencies between the additions, subtractions and max* operations and it is quantified by the length of the critical path containing most operations. In practical hardware implementations, this dictates the highest clock frequency that can be used. For the FPTDS, the time period duration is the longer one of the pair of durations that one block of the FPTD completes (19)-(23) and the duration that one block of the FPE completes (15)-(18). As analysed in [22], the computation of (19)-(23) has a critical path comprising five additions plus $\log_2(M)$ max* evaluation operations, where M is the number of states in the turbo code trellis. As described in [29], the times required to compute an addition and the approximation of the max* are equal, giving a time period duration $D_{FD} = 5 + \log_2(M)]$ operations for the FPTD. Similarly, it may be inferred from (15)-(18) that each algorithmic block of the FPE has a critical path comprising five additions and $\log_2(L)$ max* operations, where L is the number of states in the equalizer trellis. Therefore, the time period duration of the FPE is $D_{FD} = 5 + \log_2(L)]$. Finally, the time period duration of the FPTDS D_F is the longer one between the two durations D_{FD} and D_{FE}.

Meanwhile, the time period duration of the Log-BCJR system is the longer one of the duration that one trellis stage of the Log-BCJR decoder completes (9)-(14) and the duration that one trellis stage of the Log-BCJR equalizer completes (2)-(6). In contrast to the FPTD [22], the Log-BCJR turbo decoder requires one max* evaluation of (10) and (11) to be completed before (12)-(14). As a result, the time period duration of the Log-BCJR turbo decoder is $D_{BD} = 6 + \log_2(M)]$ operations. Similarly, the time period duration of the Log-BCJR equalizer is $D_{BE} = 6 + \log_2(L)]$ operations. Hence, the duration of the Log-BCJR scheme D_B is the longer one of the pair of durations D_{FD} and D_{FE}.

Again, all of the time durations of the FPTDS and of the classic Log-BCJR turbo detection scheme are provided in Table 3. It is noted that the time period of the FPTDS given by $D_F = 5 + \log_2[\max(M, L)]$ is lower than that of the conventional Log-BCJR turbo detection formulated as $D_B = 6 + \log_2[\max(M, L)]$, albeit only by the time of one operation.

D. THROUGHPUT AND LATENCY

The detection latency is defined as the time duration in which a turbo detection scheme requires to completes its iterative equalization and decoding operations. Hence, it is given by the product of the time period D, the number of time periods T per decoding iteration and the required number of decoding iterations I, where the latter will be determined in Section V. The latency and throughput of the schemes are detailed in Table 3.

TABLE 3. The characteristics of the FPTDS and the Log-BCJR turbo detection when communication over a multipath fading channel.

Characteristic	FPTD	FPE	FPTDS	Log-BCJR turbo decoder	Log-BCJR equalizer	Log-BCJR system
Decoding iterations required I	I^E	I^E	I^E	$I^O \cdot I^I$	I^I	I^I
Time periods per decoding iteration T	2	2	2	$4N$	$6N$	$(2I^O+3)\cdot 2N$
Time periods duration D	$D_{FD}=5+\log_2(M)$	$D_{FE}=5+\log_2(L)$	$D_F=5+\log_2[\max(M,L)]$	$D_{BD}=6+\log_2(M)$	$D_{BE}=6+\log_2(L)$	$D_B=6+\log_2[\max(M,L)]$
Complexity per decoding iteration C	C_{FD}	C_{FE}	$C_F=C_{FD}+C_{FE}$	C_{BD}	C_{BE}	$C_B=C_{BD}\cdot I^O+C_{BE}$
Overall throughput $\frac{1}{T\times D\times I}$	$\frac{1}{2D_{FD}\cdot I^E}$	$\frac{1}{2D_{FE}\cdot I^E}$	$\frac{1}{2D_F\cdot I^E}$	$\frac{1}{4N\cdot D_{BD}\cdot I^O}$	$\frac{1}{6N\cdot D_{BE}\cdot I^I}$	$\frac{1}{(2I^O+3)\cdot 2N\cdot D_B\cdot I^I}$
Overall latency $T\times D\times I$	$2D_{FD}\cdot I^E$	$2D_{FE}\cdot I^E$	$2D_F\cdot I^E$	$2N\cdot D_{BD}\cdot I^O$	$3N\cdot D_{BE}\cdot I^I$	$(2I^O+3)\cdot 2N\cdot D_B\cdot I^I$
Overall complexity $C\times I$	$C_{FD}\cdot I^E$	$C_{FE}\cdot I^E$	$C_F\cdot I^E$	$C_{BD}\cdot I^O\cdot I^I$	$C_{BE}\cdot I^I$	$C_B\cdot I^I$
Comp. resource requirement X	$X_{FD}=C_{FD}/2+N$	$X_{FE}=C_{FD}/2$	$X_F=C_F/2+N$	$X_{BD}=C_{BD}/2N+1$	$X_{BE}=C_{BE}/3N$	$X_B=\max(X_{BD},X_{BE})$
Register resource requirement Y	$Y_{FD}=2N\cdot(2M+4)$	$Y_{FE}=3N/2\cdot(2L+3)$	$Y_F=N\cdot(2M+3L+8.5)$	$Y_{BD}=M$	$Y_{BE}=L$	$Y_B=\max(M,L)$
RAM resource requirement Z	$Z_{FD}=0$	$Z_{FE}=0$	$Z_F=0$	$Z_{BD}=N\cdot(3M+4)$	$Z_{BE}=3N\cdot(3L+3)$	$Z_B=max(Z_{BD},Z_{BE})$
Overall resource requirement $9X+5Y+Z$	$9X_{FD}+5Y_{FD}$	$9X_{FD}+5Y_{FE}$	$9X_F+5Y_F$	$9X_{BD}+5Y_{BD}+Z_{BD}$	$9X_{BE}+5Y_{BE}+Z_{BE}$	$9X_B+5Y_B+Z_{BD}$

E. RESOURCE REQUIREMENTS

In practical hardware implementations, the chip area or hardware resource requirement depends both on the computational requirement X as well as on the memory requirement, which can be separated into the register and Random Access Memory (RAM) resources. The register resource requirement Y quantifies the amount of memory that is arranged into registers, which store values that can be accessed all at once, in every time period. By contrast, the RAM resource requirement Z quantifies the amount of storage that is arranged into RAM, which store different values that can be accessed in different time periods.

As analysed in [22], the FPTDS having an odd-even interleaver can share hardware in alternate time periods. Thus, the computational resource required by the FPTD having an odd-even interleaver equals to half of the complexity plus N additional resources for adding the systematic *a priori* bits $\bar{\mathbf{b}}_3^{\mathrm{a}}$, hence resulting in a total computational resource X_{FD} of $C_{FD}/2+N$. Since the FPE does not require the addition of the systematic *a priori* information, the computational resource X_{FE} is reduced to $C_{FE}/2$. The total computational resource required by the FPTDS is given by the summation of those of the FPE and the FPTD or quantified by $X_F = C_F/2+N = C_{FD}/2+C_{FE}/2+N$. By contrast, the Log-BCJR system is capable of reusing the same hardware for processing successive trellis stages in successive time periods for computation within the equalizer and both within the decoder as well as between the equalizer and the decoder. Therefore, the computational resource required by the classic Log-BCJR system is reduced to the higher number of resources between the conventional Log-BCJR equalizer and the Log-BCJR decoder, which is formulated as $X_B = \max(C_{BD}/2N+1, C_{BE}/3N)$.

In the FPTD, memory resources are required for storing the forward state metrics, backward state metrics and the extrinsic LLRs of (20), (21), (22) and (23), respectively. These outputs are produced, whenever an algorithmic block is operated and they must be stored for the next time period, where they are employed by the connected blocks. However, by using the odd-even interleaver described in Section III, only half of the blocks operated, while the other half remain idle. This allows the memory resources to be shared and physically positioned between two group of algorithmic blocks. Therefore, the memory resources have to store MN forward state metrics, MN backward state metrics and $4N$ extrinsic LLRs, resulting in a total requirement of $Y_{FD}=(2M+4)N$ memory resources. Similarly, the FPE requires $Y_{FE} = (2L+3)3N/2$ memory resources. Finally, the total memory resources required by the FPTDS may be expressed as $Y_F = Y_F+Y_F=(2M+3L+8.5)N$.

As quantified in [22], the classic Log-BCJR decoder only requires M memory resources due to the reuse of the same hardware for processing successive trellis stages in successive time periods. Additionally, it requires $(3M+4)N$ RAM resources for storing the state metrics and the extrinsic LLRs. Similarly, the Log-BCJR equalizer requires L memory resources and $(3L+3)3N$ RAM resources. Consequently, the memory resources required by the Log-BCJR system obey $Y_B = \max(M,L)$, while the RAM requirement is $Z_B = N\times\max[(3M+4),(9L+9)]$.

The final resource requirements depend on the specific system configuration, namely on the number of states L and M in the trellis as well as on the frame length N. Therefore, our comparison between the classic Log-BCJR turbo detection and the FPTDS will be detailed in Section V, where the specific system configurations will be defined.

TABLE 4. The characteristics of the FPTDS and the Log-BCJR turbo detection when employing LTE codes for communications over a 3-tap fading channel.

Characteristic	FPTD	FPE	FPTDS	Log-BCJR turbo decoder	Log-BCJR equalizer	Log-BCJR system
Decoding iterations required I	32	32	32	8	2	2
Time periods per decoding iteration T	2	2	2	2048	3072	19456
Time periods duration D	8	8	8	9	9	9
Complexity per decoding iteration C	194,560	236,544	431,104	210,944	261,120	1,948,672
Overall throughput $\frac{1}{T \times D \times I}$	1.95×10^{-3}	1.95×10^{-3}	1.95×10^{-3}	7.62×10^{-6}	30.35×10^{-6}	3.21×10^{-6}
Overall latency $T \times D \times I$	512	512	512	131,070	196,608	311,296
Overall complexity $C \times I$	6.23×10^{6}	7.57×10^{6}	13.79×10^{6}	1.69×10^{6}	0.52×10^{6}	3.90×10^{6}
Comp. resource requirement X	98,304	118,272	216,576	104	85	104
Register resource requirement Y	20,480	29,184	49,664	8	8	8
RAM resource requirement Z	0	0	0	28,672	82,944	82,944
Overall resource requirement $9X + 5Y + Z$	987.14×10^{3}	1210.37×10^{3}	2197.50×10^{3}	29.65×10^{3}	83.75×10^{3}	83.92×10^{3}

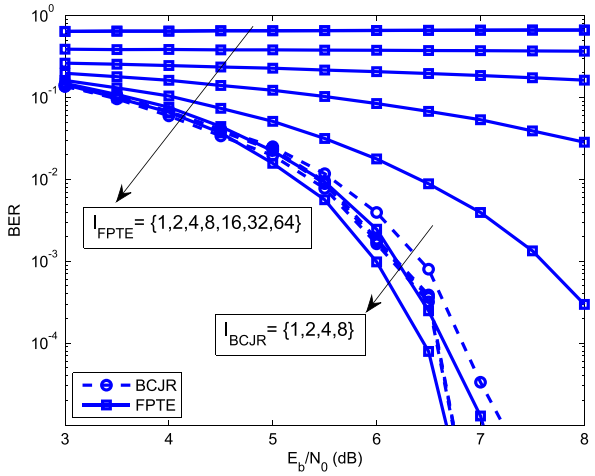


FIGURE 3. The BER performance of the FPTDS and the Log-BCJR turbo detection when employing LTE turbo codes for communication over a 3-tap fading channel for various number of iterations.

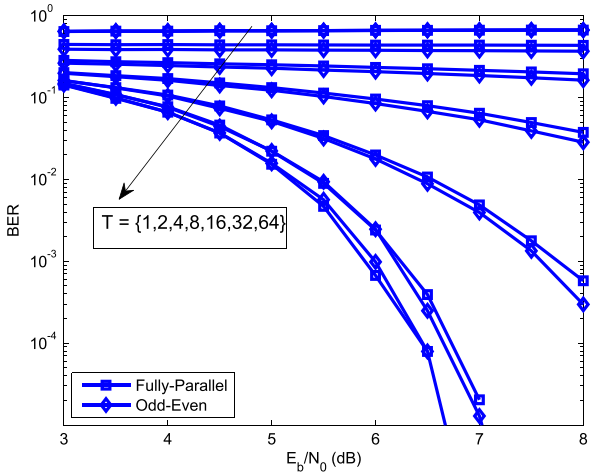


FIGURE 4. The BER performance of the FPTDS when employing fully-parallel and odd-even arrangements for communication over a 3-tap fading channel.

V. PERFORMANCE STUDY

In the simulations of this section, we employ an $M = 8$-state LTE turbo code [12] having a coding rate of 1/3, a frame length of $N = 1024$ bits relying on a 3-bit trellis termination, as described in [22], Furthermore, BPSK modulation is used. The channel imposes 3-tap multipath fading plus AWGN. The fading between transmission frames is assumed to be independent.

Fig. 3 shows the performance of the systems, where both the Log-BCJR and the fully-parallel algorithms are characterized. The classic Log-BCJR system is used both for iterative equalization and decoding. In each of the I_{BCJR} equalizer-to-turbo-decoder iterations, the Log-BCJR system performs Log-BCJR equalization followed by $I^O = 8$ iterations of Log-BCJR turbo decoding. By contrast, the FPE system carries out fully-parallel equalization and decoding simultaneously. In Fig. 3, the performance of the BCJR system is represented by the dashed curves, while that of the FPE system is shown by the continuous ones. Observe that the FPE system exhibits a high BER, when the number of iterations is below 16. By contrast, when the number of iterations is increased to 32 and 64, the FPE system achieves a comparable performance to that of the Log-BCJR system employing $I_{BCJR} = 2$ equalizer-to-turbo-decoder iterations

Fig. 4 shows the performance of the FPE systems, where the fully-parallel and odd-even mechanisms are employed. The number of time periods of $T = \{1, 2, 4, 8, 16, 32, 64\}$ are characterized in this figure. Recall that the fully parallel arrangement employs one time period for each equalization and decoding iteration, while the odd-even arrangement employs two periods for each equalization and decoding iter-

ation. In Fig. 4, the performance of the fully-parallel system is shown by the square-marked curves, while that of the odd-even system is represented by the diamond-marked curves. The results of Fig. 4 showed that the performance of both systems are comparable, regardless of the number of time periods observed. Hence, the FPTDS employing an odd-even interleaver achieves the same performance in conjunction with the same number of time periods, while reducing the complexity by 50% compared to the FPTDS using fully-parallel detection.

Table 4 summaries the characteristics of both the Log-BCJR turbo detection and of the FPTDS when communicating over a 3-tap multipath fading channel. The complexity of both schemes is quantified in the operating region, where a BER below 10^{-6} is achieved. The results showed that upon aiming for such a low BER, the FPTDS is capable of improving the latency and throughput by a factor of 600 over the conventional Log-BCJR scheme, which is achieved at the modest cost of increasing the computational complexity by a factor of 3.5 as well as the computational and memory resource requirements by a factor of 26.

VI. CONCLUSIONS

In this paper we proposed a novel FPTDS, where all the algorithmic decoding blocks of both the equalizer and of the turbo decoders are being operated in parallel. The odd-even interleaver between the equalizer and the channel decoder was designed for reducing the computational complexity. Our simulations demonstrated that when the LTE turbo code is employed for communication over a 3-tap fading channel, at the same near-error-free performance the FPTDS increases the complexity by a modest factor of 3.5 and the hardware resources by a factor of 26, while improving the processing latency and throughput by a factor of 600, making it an attractive candidate for high-throughput and low-latency applications. In our future research, the hardware implementation will be considered and the scope for potential complexity reduction will be further investigated. Finally, comparison with benchmarkers employing radix-4, Non-Slide Window and pipelining techniques will also be studied.

ACKNOWLEDGEMENT

The research data for this paper is available at http://dx.doi.org/10.5258/SOTON/384898.

REFERENCES

[1] C. Douillard, M. Jézéquel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, "Iterative correction of intersymbol interference: Turbo-equalization," *Eur. Trans. Telecommun.*, vol. 6, no. 5, pp. 507–511, 1995.

[2] M. Tüchler, R. Koetter, and A. C. Singer, "Turbo equalization: Principles and new results," *IEEE Trans. Commun.*, vol. 50, no. 5, pp. 754–767, May 2002.

[3] L. Hanzo, T. H. Liew, B. L. Yeap, R. Y. S. Tee, and S. X. Ng, *Turbo Coding, Turbo Equalisation and Space-Time Coding: EXIT-Chart-Aided Near-Capacity Designs for Wireless Channels*, 2nd ed. New York, NY, USA: Wiley, 2010.

[4] R. Koetter, A. C. Singer, and M. Tüchler, "Turbo equalization," *IEEE Signal Process. Mag.*, vol. 21, no. 1, pp. 67–80, Jan. 2004.

[5] M. Tüchler and A. C. Singer, "Turbo equalization: An overview," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 920–952, Feb. 2011.

[6] J. G. Proakis, *Digital Communications*, 4th ed. New York, NY, USA: McGraw-Hill, 2001.

[7] H. V. Poor, *An Introduction to Signal Detection and Estimation*, 4th ed. New York, NY, USA: Springer-Verlag, 1994.

[8] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error—Correcting coding and decoding: Turbo-codes (1)," in *Proc. IEEE Int. Conf. Commun. (ICC Geneva), Tech. Program, Conf. Rec.*, vol. 2. May 1993, pp. 1064–1070.

[9] M. F. Brejza, L. Li, R. G. Maunder, B. M. Al-Hashimi, C. Berrou, and L. Hanzo, "20 years of turbo coding and energy-aware design guidelines for energy-constrained wireless applications," *IEEE Commun. Surveys Tuts.*, to be published. [Online]. Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=7131434

[10] (2001). *3GPP Specifications—Series 25: Radio Aspects of 3G, Including UMTS*. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/25-series.htm

[11] L.-N. Lee, A. R. Hammons, Jr., F.-W. Sun, and M. Eroz, "Application and standardization of turbo codes in third-generation high-speed wireless data services," *IEEE Trans. Veh. Technol.*, vol. 49, no. 6, pp. 2198–2207, Nov. 2000.

[12] *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding*, document ETSI TS 136 212, 2013.

[13] D. Raphaeli and Y. Zarai, "Combined turbo equalization and turbo decoding," in *Proc. Global Telecommun. Conf. (GLOBECOM)*, vol. 2. Nov. 1997, pp. 639–643.

[14] D. Raphaeli and Y. Zarai, "Combined turbo equalization and turbo decoding," *IEEE Commun. Lett.*, vol. 2, no. 4, pp. 107–109, Apr. 1998.

[15] J. P. Woodard and L. Hanzo, "Comparative study of turbo decoding techniques: An overview," *IEEE Trans. Veh. Technol.*, vol. 49, no. 6, pp. 2208–2233, Nov. 2000.

[16] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 2. Seattle, WA, USA, Jun. 1995, pp. 1009–1013.

[17] J. Zhang and M. P. C. Fossorier, "Shuffled iterative decoding," *IEEE Trans. Commun.*, vol. 53, no. 2, pp. 209–213, Feb. 2005.

[18] O. Muller, A. Baghdadi, and M. Jézéquel, "Exploring parallel processing levels for convolutional turbo decoding," in *Proc. 2nd Int. Conf. Inf. Commun. Technol. (ICTTA)*, vol. 2. 2006, pp. 2353–2358.

[19] O. Muller, A. Baghdadi, and M. Jézéquel, "From parallelism levels to a multi-ASIP architecture for turbo decoding," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 1, pp. 92–102, Jan. 2009.

[20] T. Ilnseher, F. Kienle, C. Weis, and N. Wehn, "A 2.15 Gbit/s turbo code decoder for LTE advanced base station applications," in *Proc. 7th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, Gothenburg, Sweden, Aug. 2012, pp. 21–25.

[21] *5G White Paper*, 1st ed., Next Generat. Mobile Netw. Alliance, Frankfurt, Germany, Feb. 2015.

[22] R. G. Maunder, "A fully-parallel turbo decoding algorithm," *IEEE Trans. Commun.*, vol. 63, no. 8, pp. 2762–2775, Aug. 2015.

[23] A. Glavieux, C. Laot, and J. Labat, "Turbo equalization over a frequency selective channel," in *Proc. Int. Symp. Turbo Codes Rel. Topics*, Brest, France, Sep. 1997, pp. 96–102.

[24] X. Wang and H. V. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Trans. Commun.*, vol. 47, no. 7, pp. 1046–1061, Jul. 1999.

[25] S. ten Brink, "Convergence of iterative decoding," *Electron. Lett.*, vol. 35, no. 10, pp. 806–808, May 1999.

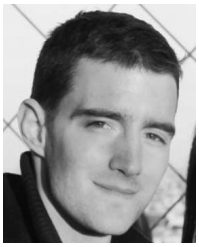
[26] H. A. Ngo, R. G. Maunder, and L. Hanzo, "Extrinsic information transfer charts for characterizing the iterative decoding convergence of fully parallel turbo decoders," *IEEE Access*, vol. 3, pp. 2100–2110, Nov. 2015.

[27] L. Fanucci, C. Pasquale, and G. Colavolpe, "VLSI design of a fully-parallel high-throughput decoder for turbo Gallager codes," *IEICE Trans. Fundam.*, vol. E89-A, no. 7, pp. 1976–1986, 2006.

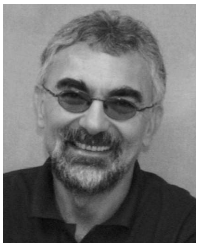
[28] A. Nimbalker, Y. Blankenship, B. Classon, and T. K. Blankenship, "ARP and QPP interleavers for LTE turbo coding," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar./Apr. 2008, pp. 1032–1037.

[29] L. Li, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "A low-complexity turbo decoder architecture for energy-efficient wireless sensor networks," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 1, pp. 14–22, Jan. 2013.

HOANG ANH NGO received the B.Eng. (Hons.) degree in electronics engineering from the Hanoi University of Science and Technology (HUST), Vietnam, in 2007, and the M.Sc. and Ph.D. degrees in wireless communications from the University of Southampton, U.K., in 2008 and 2012, respectively. From 2012 to 2014, he was with the Viettel Institute of Research and Development (R&D), Vietnam. Since 2014, he has been a Research Fellow with the University of Southampton. Currently, he is also with the R&D center, VTTEK, Viettel. His research interests include colocated and distributed MultipleInputMultipleOutput (MIMO) communications, spacetime coding and modulation, and channel coding and modeling. He is a recipient of several academic awards from HUST, the University of Southampton, and the Engineering and Physical Sciences Research Council, U.K.



ROBERT G. MAUNDER has been with the department of Electronics and Computer Science at the University of Southampton, UK, since October 2000. He was awarded the B.Eng. (Hons.) degree in electronic engineering in 2003, as well as a Ph.D. degree in wireless communications in 2007. He became a lecturer in 2007 and an Associated Professor in 2013. His research interests include joint source/channel coding, iterative decoding, irregular coding, and modulation techniques.



LAJOS HANZO received the degree in electronics in 1976, the Ph.D. degree in 1983, and the Doctor Honoris Causa degree from the Technical University of Budapest, in 2009. During his 38-year career in telecommunications, he has held various research and academic positions in Hungary, Germany, and the U.K. Since 1986, he has been with the School of Electronics and Computer Science, University of Southampton, U.K., as the Chair in Telecommunications. He has successfully supervised 100 Ph.D. students, co-authored 20 John Wiley/IEEE Press books in mobile radio communications totaling in excess of 10 000 pages, authored over 1500 research entries at the IEEE Xplore, acted as the TPC Chair and General Chair of the IEEE conferences, presented keynote lectures, and received a number of distinctions. He is directing 100 strong academic research teams, working on a range of research projects in the field of wireless multimedia communications sponsored by the industry, the Engineering and Physical Sciences Research Council, U.K., 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 offers a range of industrial courses.

He is a fellow of the Royal Academy of Engineering, the Institution of Engineering and Technology, and the European Association for Signal Processing. He is also a Governor of the IEEE VTS. From 2008 to 2012, he was the Editor-in-Chief of the *IEEE Press* and a Chaired Professor with Tsinghua University, Beijing. He has over 22 000 citations.