20 Years of Turbo Coding and Energy-Aware Design Guidelines for Energy-Constrained Wireless Applications

Matthew F. Brejza*, Liang Li*‡, Robert G. Maunder*, Bashir M. Al-Hashimi*, Claude Berrou† and Lajos Hanzo*
*School of ECS, University of Southampton, SO17 1BJ, UK
†Technople Brest-Iroise, CS 83818, 29238 Brest Cedex 3, France
Email: *{mfb2g09, rm, bmah, lh}@ecs.soton.ac.uk, ‡liang.li@cirrus.com, †claude.berrou@telecom-bretagne.eu

***Abstract*—During the last two decades, wireless communication has been revolutionized by near-capacity Error-Correcting Codes (ECCs), such as Turbo Codes (TCs), which offer a lower Bit Error Ratio (BER) than their predecessors, without requiring an increased transmission Energy Consumption (EC). Hence, TCs have found widespread employment in spectrum-constrained wireless communication applications, such as cellular telephony, Wireless Local Area Network (WLAN) and broadcast systems. Recently however, TCs have also been considered for energy-constrained wireless communication applications, such as Wireless Sensor Networks (WSNs) and the 'Internet of Things' (IoT). In these applications, TCs may also be employed for reducing the required transmission EC, instead of improving the BER. However, TCs have relatively high computational complexities and hence the associated signal-processing-related ECs are not insignificant. Therefore, when parameterizing TCs for employment in energy-constrained applications, both the processing EC and the transmission EC must be jointly considered. In this tutorial, we investigate holistic design methodologies conceived for this purpose. We commence by introducing turbo coding in detail, highlighting the various parameters of TCs and characterizing their impact on the encoded bit rate, on the Radio Frequency (RF) bandwidth requirement, on the transmission EC and on the BER. Following this, energy-efficient TC decoder Application-Specific Integrated Circuit (ASIC) architecture designs are exemplified and the processing EC is characterized as a function of the TC parameters. Finally, the TC parameters are selected in order to minimize the sum of the processing EC and the transmission EC.**

***Index Terms*—Turbo code, BCJR algorithm, energy efficiency, holistic design, optimization, wireless sensor network**

I. Introduction

Wireless communication holds the promise of ubiquity, featuring in almost all electronic devices employed for a wide variety of applications. These applications may be classified by the particular constraints that they impose both upon the design of the wireless communication schemes and on the electronic devices. For example, cellular telephony, Wireless Local Area Networks (WLANs) and broadcast systems [1]–[3] may be considered to be spectrum-constrained, since the ever-increasing demand for faster data rates creates a correspondingly increased demand for the limited Radio Frequency (RF) resources. Therefore, successive generations of cellular telephony, as well as WLAN and broadcast systems have been designed to make increasingly efficient use of the RF spectrum. In parallel to this trend, there has been a significant amount of recent interest in energy-constrained wireless communication applications [4]–[7], such as in Wireless Sensor Networks (WSNs) and in the 'Internet of Things' (IoT) [8]–[12]. These applications are characterized by the requirement of maintaining sporadic, but reliable data transmissions for extended periods of time. Typically, the communication devices employed in this scenario are required to be mobile, preventing them from relying on access to fixed energy supplies, such as mains electricity. The devices are often required to be shirt-pocket-sized, light-weight and low-cost, preventing the employment of high-capacity batteries. Furthermore, the communication devices may be expected to operate without human interaction, preventing the regular replacement or recharging of batteries. For these reasons, the communication devices are required to make efficient use of all available energy resources, which may include low-capacity batteries and energy harvesters, such as solar cells. In this paper, we focus our attention on the employment of Turbo Codes (TCs) [13], [14] in energy-constrained wireless communication applications, considering the joint design of both the communications and the hardware architecture. In this paper, TCs are invoked for energy-constrained wireless communication applications due to their widespread employment in operational communication standards, such as LTE [1] and WiMAX [15].

Wireless communication has been revolutionized by the invention of TCs [13], [14] and other sophisticated Error-Correcting Codes (ECCs). These codes provide resilience to the transmission errors that are caused by noise, interference and fading during wireless transmission. This is achieved by using a turbo encoder to process all information before transmitting it, then employing a corresponding turbo decoder in the receiver to detect and correct any transmission errors. Compared to previous ECCs, TCs facilitate signifi-

The financial support of the EPSRC, Swindon UK under the grants EP/J015520/1 and EP/L010550/1 is gratefully acknowledged.

cantly higher information bit rates and/or significantly lower RF bandwidth requirements, without requiring an increased transmission Energy Consumption (EC) or imposing an increased transmission error probability. In other words, TCs facilitate significantly improved *spectral efficiencies* η, without requiring an increased *Signal-to-Noise Ratio (SNR) per bit* $E_\mathrm{b}^\mathrm{tx}/N_0$ or imposing an increased *Bit Error Ratio (BER)*. Here, the *spectral efficiency* η has units of bit/s/Hz and is given by the ratio of the information bit rate to the required bandwidth. Meanwhile, the transmission EC E_b^tx has units of J/bit and is expressed by the *SNR per bit* $E_\mathrm{b}^\mathrm{tx}/N_0$, where it is normalized by the noise power spectral density N_0. Finally, the *BER* quantifies the transmission error probability by expressing the number of information bits that are erroneously decoded as a ratio to the total number of information bits. Figure 1 plots the *capacity* of a particular wireless channel, which quantifies the maximum spectral efficiency η for which it is theoretically possible to achieve a vanishingly low BER [16], as a function of the SNR per bit $E_\mathrm{b}^\mathrm{tx}/N_0$. The crosses in Figure 1 show that at an $E_\mathrm{b}^\mathrm{tx}/N_0$ of about 11 dB, a low BER can be achieved by a particular repetition code having a spectral efficiency of $\eta = 1/3$ bits/s/Hz, assuming a Nyquist roll-off-factor of $\alpha = 0$. By contrast, a particular punctured TC is capable of achieving this BER, while using a significantly higher spectral efficiency of $\eta = 0.81$ bits/s/Hz, which is much nearer to the channel capacity. Owing to this benefit, TCs are often referred to as near-capacity ECCs and have found widespread employment in spectrum-constrained wireless communication applications, such as cellular telephony, WLAN and broadcast systems. However, Figure 1 also illustrates an alternative application for TCs in energy-constrained wireless communication systems, where the attainable energy efficiency of $1/E_\mathrm{b}^\mathrm{tx}$ is of more grave concern than the spectral efficiency η. The crosses in Figure 1 show that when no puncturing is used, the TC considered achieves the same low BER and the same spectral efficiency of $\eta = 1/3$ bits/s/Hz as the repetition code, albeit at a significantly lower $E_\mathrm{b}^\mathrm{tx}/N_0$ value of 1.6 dB. This corresponds to a 9.4 dB reduction in transmission EC E_b^tx, which is nearly an order of magnitude. This demonstrates why TCs have found application not only in spectrum-constrained wireless communication scenarios, but also recently in energy-constrained scenarios, such as WSNs and the IoT.

TCs most commonly take advantage of the Bahl-Cocke-Jelinek-Raviv (BCJR) decoding algorithm and its variants with the objective of mitigating the transmission errors corrupting the received information. When used in TCs, the BCJR decoder, also known as the Maximum A Posteriori (MAP) decoder, is activated in an iterative manner. In a similar fashion to the classic Low-Density Parity-Check (LDPC) decoders [17] operating on the basis of the min-sum and sum product algorithm, the iterative operation of the BCJR algorithm approximates the capacity-approaching performance of a Maximum Likelihood Detector (MLD), with the appealing benefit of imposing a fraction of the complexity [18]. The BCJR algorithm operates on the basis of a trellis in a similar manner to the Viterbi Algorithm (VA) [19], which has a lower complexity but does not facilitate iterative decoding and hence has a reduced error correction capability. The complexity

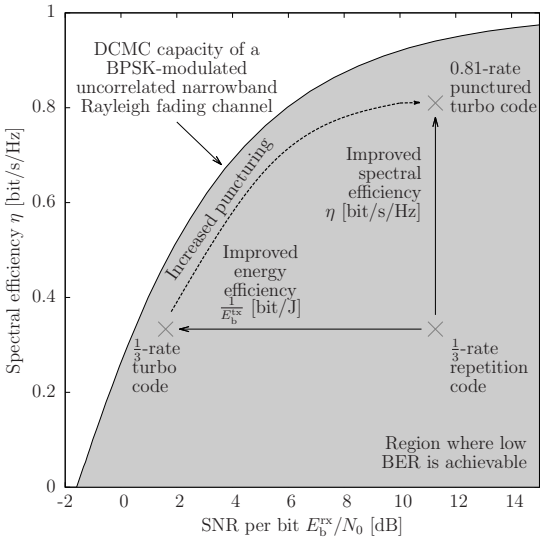


Fig. 1. Combinations of spectral efficiency η and SNR per bit $E_\mathrm{b}^\mathrm{rx}/N_0$ for which BERs of 10^{-3} can be achieved by three ECCs, namely (a) a 1/3-rate repetition code employing soft decoding, (b) the 1/3-rate LTE TC [1] employing a message length of 2048 bits and 6 decoding iterations, as well as (c) the same LTE TC but punctured to give a coding rate of 0.81.

of a decoding algorithm is often quantified in terms of the number of operations required for decoding, which can be expressed in terms of the number of states or trellis-transitions. However, this paper will demonstrate that the complexity of the algorithm does not necessarily determine the complexity and the EC of its Application-Specific Integrated Circuit (ASIC) implementation, hence motivating the holistic design methodologies investigated in this paper.

Despite having a complexity significantly less than the optimal MLD, when employing TCs for the sake of reducing the *transmission* EC E_b^tx, consideration should also be given to the TC's *processing* EC E_b^pr dissipated by its iterative decoder. While turbo encoders have relatively low complexity and EC [20], the EC E_b^pr of turbo decoders is not insignificant [21], even when implemented using an ASIC. This may be attributed to the relatively high complexity of turbo decoding algorithms, such as that of the BCJR algorithm [22]. Indeed, the authors of [23] considered the power consumption of the various components of a transceiver, finding that for the range of LTE base-stations which were considered, the turbo code consumes approximately the same power as the baseband radio components. Additionally, it was found for the smallest 'femto' base-stations that the turbo code also consumes approximately the same power as the Power Amplifier (PA) components. Conventionally, it has been a challenge to jointly optimize both the transmission EC E_b^tx and the processing EC E_b^pr during the design of TCs for energy-constrained wireless communication applications. While the transmission EC E_b^tx can be characterized at an early design stage using BER simulations, it has not previously been possible to characterize the processing EC E_b^pr until after the turbo decoder ASIC has been designed, which is a much later design stage. If at this time, it is discovered that the processing EC E_b^pr is unacceptably high, then it becomes necessary to revert to an

earlier design stage and try again. This motivates the holistic TC design methodologies that we demonstrate in this tutorial. These methodologies model the processing EC $E_{\mathrm{b}}^{\mathrm{pr}}$ of an energy-efficient TC decoder ASIC architecture as a function of the TC design parameters, allowing joint optimization at an early design stage.

Typically, the open literature on wireless communication algorithms [24]–[26] considers them independently of the hardware implementation, despite the dependence on each other. Instead, often a simplistic approach is pursued, when considering the implementation aspects. For example, it is typical for a paper in wireless communications to quantify the computational complexity of an algorithm using the number of computational operations which have to be undertaken [24]. This gives a reasonable metric for comparing similar schemes, however this method typically does not offer a fair comparison between dissimilar schemes [27]. Typically the parameters which are important are the energy consumption and hardware resources of a scheme, as this is what ultimately determines the cost and battery life of the system. Furthermore, without considering the hardware implementation, it is not possible to consider metrics such as processing latency and throughput, which can impose bottlenecks upon the overall latency and throughput, particularly in applications such as Machine-to-Machine (M2M) communications for next generation devices [28]. As explored in this paper, considering the algorithm and its implementation jointly allows for holistic optimization of the overall energy consumption, cost, latency and throughput as functions of all algorithmic and implementation design decisions.

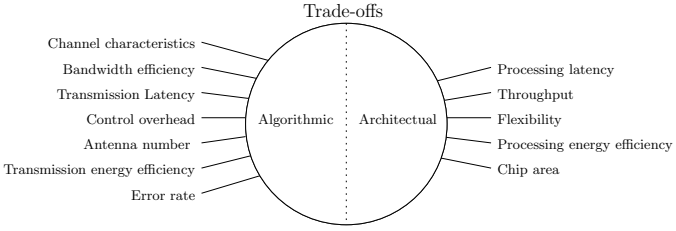


Fig. 2. Design trade-offs in a communication system.

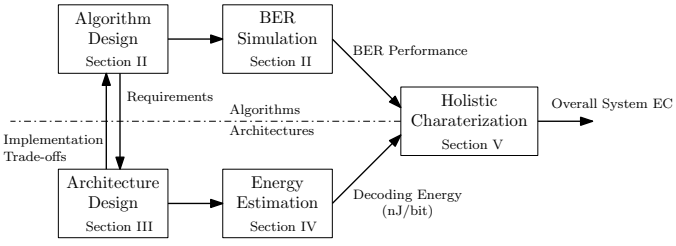


Fig. 3. The design methodology explored by this paper

Against this background, Figure 2 summarizes the trade-offs the designer of a TC decoder has to consider. These have been split into the categories of algorithmic trade-offs and architectural trade-offs, since these have previously been considered separately. Building on this, Figure 3 illustrates the structure of this paper and the holistic design and optimization approach of this tutorial. This facilitates a system-wide EC optimization, while considering how the trade-offs on different sides of Figure 2 influence each other. We commence in Section II by introducing in detail the TC, and its BER performance. Section III considers the implementation of the TC, with particular consideration of the computationally intensive Logarithmic BCJR (Log-BCJR) algorithm. The requirements of the Log-BCJR algorithm affect the design decisions made for the architecture, while conversely, architectural trade-offs have to be made which may modify the operation and performance of the Log-BCJR algorithm. This reciprocal relationship is shown in Figure 3, where the algorithmic design and architectural design are closely linked. We focus our attention on the three main areas of the architectural design, namely on the datapath, on the controller and on the memory, exploring different methods which have been developed for reducing the corresponding EC. The remainder of this tutorial then focuses on the joint optimization of the algorithm and architecture parameterization, with consideration of the possible options developed during the design stage. To achieve this, Section IV discusses a range of different approaches conceived for estimating the processing EC $E_{\mathrm{b}}^{\mathrm{pr}}$ for the different algorithm parameterizations. Although typically extensive simulations are required for estimating the EC of a circuit, this section discusses methods of significantly reducing the required simulation complexity, which is achieved by characterizing the processing EC $E_{\mathrm{b}}^{\mathrm{pr}}$ of a turbo decoder as a function of its parameters. Finally, we holistically consider the performance and energy consumption of the candidate algorithm and architecture trade-offs in Section V. The techniques gleaned from the literature and explored in this section facilitate all of the factors seen in Figure 2 to be jointly considered, allowing the selection of carefully optimized TC parameters that minimize the sum of the processing EC $E_{\mathrm{b}}^{\mathrm{pr}}$ and of the transmission EC $E_{\mathrm{b}}^{\mathrm{tx}}$. The tutorial concludes with our recommended design guidelines in Section VI.

II. Turbo coding

In this section, we introduce the TC scheme of Figure 4. We begin in Section II-A by describing the convolutional encoders, which are concatenated in parallel in order to form the turbo encoder of Figure 4. The integration of the turbo encoder into a BPSK transmitter is discussed in Section II-B. Following this, Section II-C describes the modeling of transmission over an Additive White Gaussian Noise (AWGN) channel, subject to a certain path loss. Section II-D discusses the operation of the turbo-coded BPSK receiver of Figure 4. This operates on the basis of the most frequently used variant of the BCJR decoder, namely the Log-BCJR decoder, which is detailed in Section II-E. Modifications of the Log-BCJR algorithm are conceived for the practical implementations, which are discussed in Section II-G, before the TC's error correction performance is characterized in Section II-F.

A. Convolutional encoder

The convolutional encoder [29] is a widely adopted component in sophisticated error correcting schemes, forming

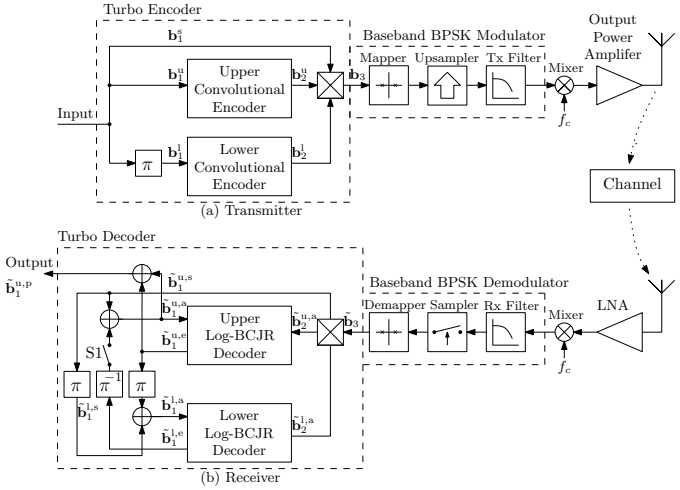


Fig. 4. A BPSK-modulated $R = 1/3$ TC scheme.

the basis of the turbo encoder, as shown in Figure 4. In this application, the input of the convolutional encoder is a message frame $\mathbf{b}_1$ comprising N bits, while the output is an N-bit encoded frame $\mathbf{b}_2$. The parametrization of a convolutional encoder may be specified by a trellis, which graphically illustrates the relationship between the frames $\mathbf{b}_1$ and $\mathbf{b}_2$. The example trellis of Figure 5 corresponds to a simple convolutional encoder, which may be used for encoding a message frame $\mathbf{b}_1$ comprising $N = 5$ bits. This encoder adopts one of two possible states following the encoding of each bit in the frame $\mathbf{b}_1$, as represented by the dots in Figure 5. Depending on the value of this bit, the encoder state is selected by following one of two possible transitions from the previous state, as represented by the lines in Figure 5. As shown in Figure 5, the convolutional encoder is initialized in state 1 before encoding the first bit in the message frame $\mathbf{b}_1$. Each selected transition identifies a bit value for the encoded frame $\mathbf{b}_2$. For example, the message frame $\mathbf{b}_1 = [1, 1, 0, 0, 1]$ corresponds to the sequence of transitions that is highlighted in bold in Figure 5. In turn, this sequence identifies the encoded frame $\mathbf{b}_2 = [1, 0, 0, 0, 1]$.

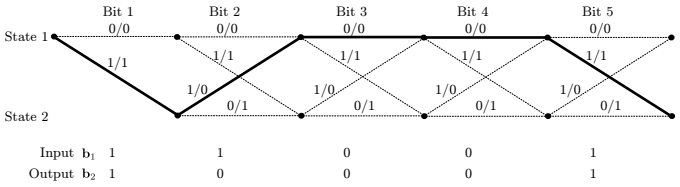


Fig. 5. An example convolutional code trellis having two possible states. Each transition T is labeled with the notation $a_1(T)/a_2(T)$. A particular transition T from the current state will be selected if the corresponding bit in the message frame $\mathbf{b}_1$ has the value $a_1(T)$, while $a_2(T)$ is the value that will be output for the corresponding bit in the encoded frame $\mathbf{b}_2$.

Note that the convolutional code's trellis of Figure 5 has $2^m = 2$ states, which corresponds to a shift register having $m = 1$ memory element. Furthermore, each transition between states is selected based on the value of $k = 1$ message bit, resulting in the generation of $n = 1$ encoded bit. This results in a coding rate for this convolutional encoder of $R = k/n = 1$, and an overall coding rate for the turbo code of Figure 4 of $R = 1/3$. However, the convolutional codes of generalized TCs may employ a shift register having any number m of memory elements. Furthermore, transitions may be selected based on any number k of message bits, resulting in the generation of any number n of encoded bits. While the TC of the LTE standard in cellular telephony [1] also employs $k = 1$ and $n = 1$, its shift register has $m = 3$ memory elements, resulting in a trellis having $2^m = 8$ states. The mapping of message and encoded bit values to each transition in the LTE TC trellis is specified by its *generator polynomials*. Furthermore, the LTE TC appends three additional termination bits to each message frame $\mathbf{b}_1$, in order to guarantee that the convolutional encoder always reaches the same particular state at the end of the encoding process.

B. Turbo coded transmitter

As shown in Figure 4, the turbo encoder comprises a parallel concatenation of two convolutional encoders, which we refer to as the upper and lower encoders. The upper encoder processes the frame of message bits $\mathbf{b}_1^{\mathrm{u}}$ in their original order, while the lower encoder processes the same bits, but in a different order. This reordering is performed by the interleaver π of Figure 4, which outputs the interleaved message frame $\mathbf{b}_1^{\mathrm{l}}$. The upper and lower convolutional encoders produce the N-bit encoded frames $\mathbf{b}_2^{\mathrm{u}}$ and $\mathbf{b}_2^{\mathrm{l}}$, respectively. These encoded frames provide $2N$ parity bits, which are multiplexed in the crossed block of Figure 4 with N systematic bits, which are provided by the N-bit message frame $\mathbf{b}_1^{\mathrm{u}}$. The resultant transmission frame $\mathbf{b}_3$ comprises $3N$ bits, corresponding to a coding rate of $R = N/(3N) = 1/3$.

Following turbo encoding, the transmitter of Figure 4 employs BPSK modulation, upsampling, pulse shaping, RF mixing and power amplification. These are employed in order to transmit the frame $\mathbf{b}_3$ using the desired carrier frequency f_c at a desired transmission energy per bit $E_{\mathrm{b}}^{\mathrm{tx}}$. Note that the power amplifier may have an efficiency of only around 33%, which corresponds to a power amplifier efficiency loss A of 4.8 dB [4]. Here, $E_{\mathrm{b}}^{\mathrm{tx}}$ is related to the energy $E_{\mathrm{s}}^{\mathrm{tx}}$ dissipated per modulated symbol according to $E_{\mathrm{b}}^{\mathrm{tx}}[\mathrm{dBJ}] = E_{\mathrm{s}}^{\mathrm{tx}} - 10\log_{10}(\eta)$, where $\eta = R\log_2(M)$, R is the coding rate and M is the modulation order of the modulation scheme, with $M = 2$ in the case of BPSK. Note that the employment of $E_{\mathrm{b}}^{\mathrm{tx}}$ is typically preferable to $E_{\mathrm{s}}^{\mathrm{tx}}$, since this allows a fair comparison amongst schemes having different coding rates R and modulation orders M in terms of their transmission energy consumption.

C. Channel

The wireless channel of Figure 4 conveys the BPSK-modulated signal between the transmitter and receiver antennas, but imposes degradation. These antennas can be characterized by their gain (G_{tx} and G_{rx}) for the intended direction of propagation. In the scenario where there is a dominant line-of-sight (LOS) path between these antennas, the degradation may be modeled by the inverse-second-power free space path loss and AWGN. Here, the path loss is imposed by the attenuation

of the BPSK-modulated signal as it propagates through free space. This depends on the distance between the transmit and receive antennas d (in m) and the carrier frequency f_c (in Hz) [30], according to

$$P_{\mathrm{l}}(d)[\mathrm{dB}] = 20\log_{10}(d) + 20\log_{10}(f_c) + 20\log_{10}\left(\frac{4\pi}{c}\right), \quad (1)$$

where $c = 2.998 \times 10^8$ m/s is the speed of light, resulting in the last term of (1) having a constant value of -147.55 dB. However, the free space path loss model may be optimistic, since often there are multiple paths between the transmitter and receiver but the LOS path might be absent. In order to account for this, the path loss equation can be generalized by parameterizing the path loss exponent p [4], [5], according to

$$P_{\mathrm{l}}(d)[\mathrm{dB}] = 10p\log_{10}(d) + 20\log_{10}(f) - 147.55. \quad (2)$$

Path loss exponents between $p = 2$ and $p = 4$ can be expected in the diverse environments encountered. The AWGN is imposed by the Brownian motion of electrons, resulting in thermal noise at the receiver, which has the power spectral density of $N_0[\mathrm{dBJ}] = 10 \times \log(k \cdot T)$, where $k = 1.3806503 \times 10^{-23} JK^{-1}$ is the Boltzmann constant. For the case of the room temperature $T = 300K$, we obtain $N_0 = -203.8$ dBJ. Note that depending on the operating conditions, co-user interference is often more significant than the thermal noise. To model this, N_0 can instead be replaced with the noise power spectral density that is expected in the operating conditions of the wireless link [31].

Considering the above channel effects, we can therefore relate the energy per bit at the receiver $E_{\mathrm{b}}^{\mathrm{rx}}$ in terms of the energy dissipated at the transmitter $E_{\mathrm{b}}^{\mathrm{tx}}$ and the channel conditions, according to

$$E_{\mathrm{b}}^{\mathrm{rx}}[\mathrm{dBJ}] = 10\log_{10}(E_{\mathrm{b}}^{\mathrm{tx}}) - A - P_{\mathrm{l}}(d) + G_{\mathrm{tx}} + G_{\mathrm{rx}}, \quad (3)$$

where all quantities are expressed in dB, except $E_{\mathrm{b}}^{\mathrm{tx}}$ which is expressed in Joules. Note that if shadowing or fading is prevalent in the particular wireless environment considered, then (3) can be modified to model this by additionally subtracting corresponding fading margins [32].

D. Turbo coded receiver

In the receiver of Figure 4, the BPSK-modulated signal provided by the receive antenna is passed to a Low Noise Amplifier (LNA). This is employed to boost the weak received signals, while introducing only a minimal amount of additional noise, which is quantified by its Receiver Noise Figure (RNF). The amplified signal is mixed down from the RF range to the baseband, where it is filtered to remove the out-of-band noise, down-sampled and provided to the BPSK demodulator.

The role of the BPSK demodulator is to extract information pertaining to the turbo-encoded bits from the received signal. However, the BPSK demodulator can never be certain of the correct value for each bit, owing to the unpredictable nature of the degradation imposed by the channel. Rather than making a binary *hard decision* of '1' or '0' for each bit, superior error correction performance can be obtained if the demodulator makes a *soft decision*. Here, a soft decision expresses not only *what* the most likely value of the bit is, but also *how likely* this value is. More specifically, the demodulator, which is also often referred to as a demapper, can express the soft information pertaining to a particular bit using a Logarithmic Likelihood Ratio (LLR), which represents the probabilities associated with the value of the bit b according to $\tilde{b} = \ln[\Pr(b = 1)/\Pr(b = 0)]$. Here, the sign of an LLR expresses whether a value of '1' or '0' is more likely for the corresponding bit, while the magnitude of the LLR is commensurate with how likely this value is. When employing BPSK modulation, it can be shown that each LLR is directly proportional to the corresponding sample provided by the down-sampler [33]. As shown in Figure 4, the BPSK demodulator generates the LLR sequences $\tilde{\mathbf{b}}_1^{\mathrm{u,s}}$, $\tilde{\mathbf{b}}_2^{\mathrm{u,a}}$ and $\tilde{\mathbf{b}}_2^{\mathrm{l,a}}$, which pertain to the bit sequences $\mathbf{b}_1^{\mathrm{u}}$, $\mathbf{b}_2^{\mathrm{u}}$ and $\mathbf{b}_2^{\mathrm{l}}$ respectively. Furthermore, an interleaver π is employed for converting $\tilde{\mathbf{b}}_1^{\mathrm{u,s}}$ into the LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{l,s}}$, which pertains to the bit sequence $\mathbf{b}_1^{\mathrm{l}}$. These LLR sequences are then provided to the turbo decoder, which is invoked for mitigating the corresponding uncertainty and for eliminating transmission errors. As shown in Figure 4, the turbo decoder comprises two Log-BCJR decoders, which correspond to the two convolutional encoders of the turbo encoder.

The turbo decoder is operated in an iterative manner, with the switch labeled 'S1' in Figure 4 being left open during the first decoding iteration. This enters the LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{u,s}}$ provided by the BPSK demodulator directly into the upper Log-BCJR decoder $\tilde{\mathbf{b}}_1^{\mathrm{u,a}}$. As shown in Figure 4, the upper Log-BCJR decoder's other input $\tilde{\mathbf{b}}_2^{\mathrm{u,a}}$ is supplied by the BPSK demodulator. The upper Log-BCJR decoder combines the old (or "*a priori*") information provided by its two input LLR sequences, in order to extract new (or "extrinsic") information for the output LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{u,e}}$. Since this LLR sequence pertains to the uncoded bit sequence $\mathbf{b}_1^{\mathrm{u}}$, the interleaver π may be used for converting it into information pertaining to the bit sequence $\mathbf{b}_1^{\mathrm{l}}$. Following this, the resultant interleaved LLR sequence may be added on a bit-by-bit basis to the values in the LLR sequence $\tilde{\mathbf{b}}_{1,s}^{\mathrm{l}}$ provided by the BPSK demodulator, which also pertains to $\mathbf{b}_1^{\mathrm{l}}$. The resultant LLR sequence is then forwarded to the lower Log-BCJR decoder's input $\tilde{\mathbf{b}}_1^{\mathrm{l,a}}$, as shown in Figure 4. Meanwhile, the lower Log-BCJR decoder's other input $\tilde{\mathbf{b}}_2^{\mathrm{l,a}}$ is supplied by the BPSK demodulator. In turn, the lower Log-BCJR decoder combines these *a priori* LLR sequences, in order to obtain the extrinsic LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{l,e}}$, completing the first decoding iteration.

In the second and in all subsequent decoding iterations, the switch labelled 'S1' in Figure 4 is closed. This allows the extrinsic LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{l,e}}$ to be deinterleaved π^{-1} and added on a bit-by-bit basis to the values in the LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{u,s}}$, in order to generate an improved *a priori* LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{u,a}}$ for the upper Log-BCJR decoder. This motivates the repeated operation of the upper Log-BCJR decoder, in order to produce an improved extrinsic LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{u,e}}$. In turn, this may be interleaved and added on a bit-by-bit basis to the values in the LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{l,s}}$, in order to generate an improved *a priori* LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{l,a}}$ for the lower Log-BCJR decoder. Likewise, the operation of the lower Log-

BCJR decoder may be repeated for obtaining an improved extrinsic LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{l,e}}$. This process may be repeated during the third iteration and during all further iterations, in order to gradually improve the quality of the iteratively exchanged LLR sequences. However, as we will show in Section II-G, each additional iteration yields a diminishing return, until convergence is eventually achieved, whereupon additional iterations provide no further improvement. Once a sufficient number I of iterations has been performed, we may obtain a final output by adding the LLR sequences $\tilde{\mathbf{b}}_1^{\mathrm{u,a}}$ and $\tilde{\mathbf{b}}_1^{\mathrm{u,e}}$ on a bit-by-bit basis. The resultant LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{u,p}}$ contains all (or "*a posteriori*") information pertaining to the turbo encoder's input bit sequence $\mathbf{b}_1^{\mathrm{u}}$. Finally, these soft-valued LLRs may be converted into hard-valued bits by considering the sign of each LLR, where a positive value corresponds to a '1' and a negative value corresponds to a '0'.

E. Log-BCJR decoder

In this section, we provide an overview of the Log-BCJR algorithm [34], which is employed both by the upper and lower Log-BCJR decoders of Figure 4. Note that the Log-BCJR algorithm is a reduced-complexity version of the BCJR algorithm, as will be discussed in greater detail in Section II-F, together with a discussion of other variants of the BCJR algorithm. Here, we use an example, where the trellis of Figure 5 is imposed for combining the example *a priori* LLR sequences $\tilde{\mathbf{b}}_1^{\mathrm{a}} = [-5, 4, 1, 6, -2]$ and $\tilde{\mathbf{b}}_2^{\mathrm{a}} = [3, 5, -4, -2, -1]$, in order to obtain the extrinsic LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{e}}$. Note that these example LLRs have been rounded to the nearest integer, for the sake of simplicity. The Log-BCJR algorithm comprises four intermediate steps, in which four sets of metrics are calculated, namely the $\gamma(T)$, $\alpha(S)$, $\beta(S)$ and $\delta(T)$ values, where T refers to a particular transition in the trellis and S refers to a particular state, as detailed in the following discussion. We will show that the calculations of each step can be decomposed into simple Add-Compare-Select (ACS) operations. Further detailed discussions are available in [18], [35].

In the first step of the Log-BCJR algorithm, a $\gamma(T)$ value is calculated for each transition in the trellis of Figure 5. This $\gamma(T)$ value represents the *a priori* probability that the transition T was selected during the convolutional encoding process. The $\gamma(T)$ value for a particular transition T in the trellis of Figure 5 is calculated according to

$$\gamma(T) = a_1(T) \cdot \tilde{b}^{\mathrm{a}}_{1,i(T)} + a_2(T) \cdot \tilde{b}^{\mathrm{a}}_{2,i(T)}, \tag{4}$$

where $a_1(T)$ and $a_2(T)$ are described in Figure 5. Here, $i(T)$ is the index of the bits that are represented by the transition T, while $\tilde{b}^{\mathrm{a}}_{1,i(T)}$ is the LLR having that specific index $i(T)$ in the sequence $\tilde{\mathbf{b}}_1^{\mathrm{a}}$. Likewise, $\tilde{b}^{\mathrm{a}}_{2,i(T)}$ is the corresponding LLR in the sequence $\tilde{\mathbf{b}}_2^{\mathrm{a}}$. In Figure 6 each transition is labeled with the particular $\gamma(T)$ value that results for our example. Note that relatively high $\gamma(T)$ values result for transitions where the *a priori* LLRs match with that transition's $a_1(T)$ and $a_2(T)$ combination. Since $a_1(T)$ and $a_2(T)$ have binary values, each $\gamma(T)$ value is given by 0, $\tilde{b}^{\mathrm{a}}_{1,i(T)}$, $\tilde{b}^{\mathrm{a}}_{2,i(T)}$ or $\tilde{b}^{\mathrm{a}}_{1,i(T)} + \tilde{b}^{\mathrm{a}}_{2,i(T)}$. Therefore, the entire set of $\gamma(T)$ values can be calculated using only addition and selection operations.

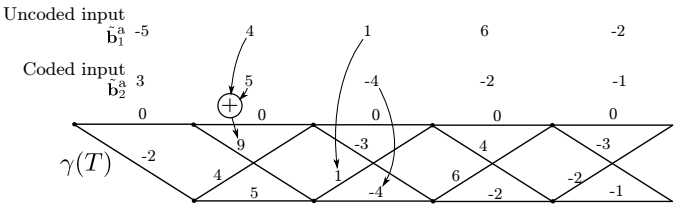


Fig. 6. Calculating the $\gamma(T)$ values for some example *a priori* LLRs.

The second step of the Log-BCJR algorithm is to calculate an $\alpha(S)$ value for each state S in the trellis. These $\alpha(S)$ values represent the probability that a particular state was entered into during the encoding process. This is obtained by considering the probabilities of the previous states having been entered into during encoding, as well as the probabilities that the transitions between these pairs of states have been taken. Owing to these dependencies between the probabilities associated with consecutive states, a forward recursion is required in order to calculate the $\alpha(S)$ values for the states of the trellis in a specific order, evolving from left to right. The calculation for an $\alpha(S)$ for a particular state S is given by

$$\alpha(S) = \max_{T \in to[S]}^{*} \left[\gamma(T) + \alpha(fr[T])\right], \tag{5}$$

where $to[S]$ returns the set of all transitions merging into the state S, while $fr[T]$ returns the particular state that the transition T emerges from. The operation $\max^*$ for two inputs A and B is defined as $\max^*(A, B) = \max(A, B) + \ln(1 + e^{-|A-B|})$. Since this operation is associative, it can be readily extended to more inputs. In the example of Figure 7, each state in the trellis is labeled with its $\alpha(S)$ value, where the $\max^*$ operator has been approximated using the max operation for simplicity. As shown in Figure 7, the forward recursion is initialized by setting the $\alpha(S)$ value of the state at the far left of the trellis to zero. Note that the $\alpha(S)$ values are calculated using only addition and $\max^*$ operations, which can be further decomposed into only ACS operations, as we shall show in Section II-F.

In the third step of the Log-BCJR algorithm, a $\beta(S)$ value is calculated for each state in the trellis, using a similar process to that of the $\alpha(S)$ values. While the $\alpha(S)$ values depend on the previous $\alpha(S)$ values in the trellis, the $\beta(S)$ value of a particular state depends on those of the next states in the trellis. Therefore the $\beta(S)$ values must be calculated in order, using a backward recursion order, evolving from the right end of the trellis to the left end. This is achieved according to

$$\beta(S) = \max_{T \in fr[S]}^{*} \left[\gamma(T) + \beta(to[T])\right], \tag{6}$$

where $fr[S]$ returns the set of all transitions that emerge from the state S, while $to[T]$ returns the particular state that the transition T merges into. Once again, the $\beta(S)$ values for our example are shown on the states of Figure 7, where the $\max^*$ operator has been approximated using the max operation for simplicity. As shown in Figure 7, the backward recursion is

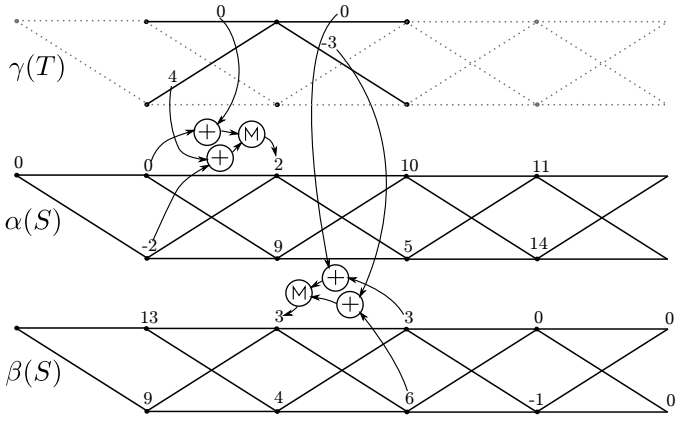


Fig. 7. Calculating the $\alpha(S)$ and $\beta(S)$ values. The previous $\gamma(T)$ trellis is shown for reference. The circled 'M' represents the $\max^*$ operation, which we have approximated using the max operation in the presented calculations, to maintain integer values for simplicity.

initialized by setting the $\beta(S)$ values of the states at the far right of the trellis to zero. Like the $\alpha(S)$ values, the $\beta(S)$ values can be calculated using only ACS calculations.

The fourth set of metrics required for the Log-BCJR algorithm are the $\delta(T)$ values, which combine the results from previous metrics in order to represent the *a posteriori* probabilities that the transitions were followed in the encoder. The $\delta(T)$ value of a particular transition T is calculated by adding its $\gamma(T)$ value to the $\alpha(S)$ value of the state it emerges from and the $\beta(S)$ value of the state it merges into, according to

$$\delta(T) = \alpha(fr[T]) + \gamma(T) + \beta(to[T]). \tag{7}$$

The $\delta(T)$ calculations detailed for our example can be seen in Figure 8. Since the $\delta(T)$ values are calculated using only additions, they can be decomposed into ACS operations.

Finally, the Log-BCJR algorithm can combine the $\delta(T)$ values in order to calculate the output extrinsic LLRs. This is achieved according to

$$\tilde{b}^{\mathrm{e}}_{1,i} = \max^*_{T\left|\begin{smallmatrix}a_1(T)=1\\ i(T)=i\end{smallmatrix}\right.} [\delta(T)] - \max^*_{T\left|\begin{smallmatrix}a_1(T)=0\\ i(T)=i\end{smallmatrix}\right.} [\delta(T)] - \tilde{b}^{\mathrm{a}}_{1,i}, \tag{8}$$

where $T\left|\begin{smallmatrix}a_1(T)=0\\ i(T)=i\end{smallmatrix}\right.$ is the set of all transitions for which the represented uncoded bit value $a_1(T)$ is zero and the index $i(T)$ of that uncoded bit is i. As shown in the example of Figure 8, this corresponds to the grouping of the $\delta(T)$ values into two sets, which are then combined using $\max^*$ operations. Following this, the *a priori* LLR $\tilde{b}^{\mathrm{a}}_{1,i}$ is subtracted from the difference between these two $\max^*$ calculations. Note that the extrinsic LLRs are calculated using only subtraction and $\max^*$ operations, which can be further decomposed into only ACS operations, as we shall show in Section II-F. This completes the Log-BCJR decoding process.

F. Algorithmic modifications to the Log-BCJR decoder

The Log-BCJR algorithm is universally preferred for implementation over the BCJR algorithm owing to its reduced computational complexity. More specially, the BCJR algorithm

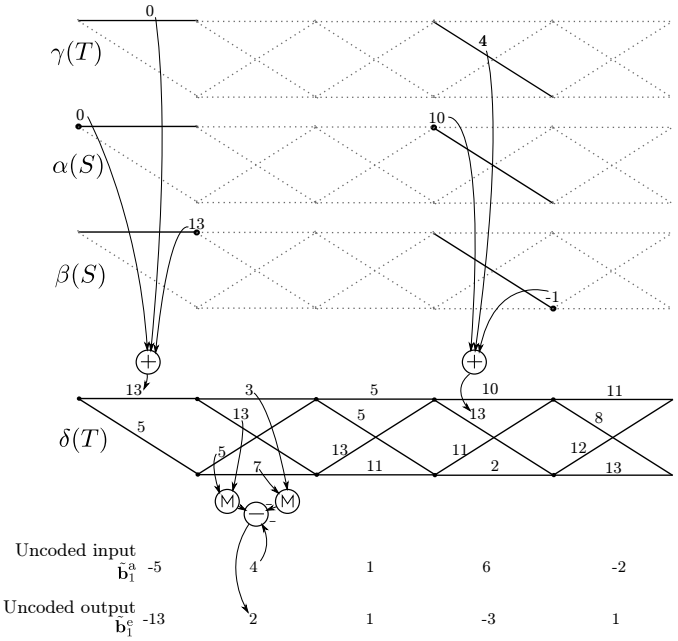


Fig. 8. Calculating the $\delta(T)$ values and the extrinsic LLRs. The circled 'M' represents the $\max^*$ operation, which we have approximated using the max operation in the presented calculations, to maintain integer values for simplicity.

operates in the normal domain, requiring addition and multiplication operations for calculating the bit probabilities. Since these probabilities have a high dynamic range, a large number of bits are required for their digitial representation. By converting the equations of the BCJR algorithm into the logarithmic domain, the Log-BCJR algorithm replaces multiplications with additions, and replaces additions with the $\max^*$ operation. These operations have a lower computational complexity, and representing the probabilities in the logarithmic domain requires fewer bits.

As shown in Section II-E, the $\max^*$ operation of the Log-BCJR algorithm is defined by $\max^*(A, B) = \max(A, B) + f(|A - B|)$, where the correction term is given by $f(|A - B|) = \ln(1 + e^{-|A-B|})$. Since the logarithmic and exponential functions of $f(|A - B|)$ are costly to implement in hardware, they are often approximated in practical applications of TCs. In the Maximum Log-BCJR (Max-Log-BCJR) approximation [36] of the Log-BCJR algorithm, $\max^*(A, B)$ is approximated using $\max(A, B)$. As shown in Figure 9, the value of $f(|A - B|)$ is always in the range $[0, 0.69]$, which is typically small compared to $\max(A, B)$, justifying this approximation. The Max-Log-BCJR approximation imposes a low computational complexity, but its error correction capability is lower than that of the original Log-BCJR algorithm [34]. This motivates the conception of a Look-Up-Table based Log-BCJR (LUT-Log-BCJR) algorithm [37], which uses a Look-Up Table (LUT) for approximating $f(|A - B|)$. As shown in Figure 9, the range of $|A - B|$ values for which $f(|A-B|)$ has a significant value is limited, meaning the LUT size can be small. Figure 9 shows how as few as four values given by $\{0, 0.25, 0.5, 0.75\}$ can be used for approximating $f(|A - B|)$, hence offering an error correction capability for

the LUT-Log-BCJR which approaches that of the Log-BCJR algorithm [37], as shown in Figure 15.

Both of the Max-Log-BCJR and the LUT-Log-BCJR algorithms can be implemented using only ACS operations. Firstly, the $\max(A, B)$ operation is performed by comparing A and B, and selecting the largest value. Based on the knowledge of $\max(A, B)$, the subtraction $|A - B|$ of the LUT-Log-BCJR can be carried out so that a positive number is returned. By comparing this result to the boundary points of the LUT, the approximate value for $f(|A - B|)$ can be selected, and then added to the value of $\max(A, B)$.

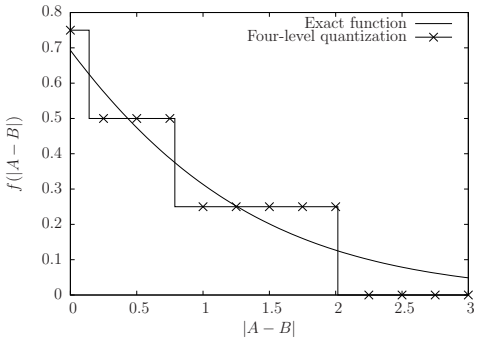


Fig. 9. Plot of $f(|A - B|) = \ln(1 + e^{-|A-B|})$, showing the quantization steps and points when a four-level quantization is employed.

As shown in Section II-E, the $\alpha(S)$ and $\beta(S)$ calculations require forward and backward recursions respectively due to their data dependencies between consecutive states. Owing to this, the Log-BCJR and its variants are not naturally suited to parallel processing. Furthermore, a large amount of memory is required, since the $\alpha(S)$ and $\beta(S)$ values are calculated in different directions along the trellis. More specifically, in order to generate the first output extrinsic LLR $\tilde{b}^{\mathrm{e}}_{1,1}$, it is necessary to have first calculated the $\beta(S)$ values for every state in the trellis and then to store them for the calculation of the subsequent output extrinsic LLRs.

An appealing technique for overcoming the data dependency issue is to decompose the trellis into N/w_s number of smaller windows [38], each having the above-mentioned length w_s. The Log-BCJR algorithm (or one of its approximations) can be applied to each window independently, significantly reducing the memory required for storing metrics. However, with this approach, it is necessary to initialize the $\alpha(S)$ values of the states at the left end of each window, as well as the $\beta(S)$ values of the states at the right end. If the windows are processed sequentially in a left to right ordering, the boundary $\alpha(S)$ values can be passed from the right end of each window to the left end of the subsequent window. However, this approach cannot supply boundary $\beta(S)$ values for the right end of each window, requiring a pre-backward recursion to generate these boundary conditions [39]. This technique generates boundary conditions by starting to calculate the $\beta(S)$ values ahead of the window, then carrying out a backwards recursion towards the edge of the window. The first $\beta(S)$ values used by the pre-backward stage are initialized to zero, then the pre-backwards length w_p is chosen for ensuring that the beta values generated at the boundary of the window converge to those values in the non-windowed Log-BCJR algorithm. Further detailed reading on the pre-backward technique is available in [40]. Other windowing techniques include the Previous Iteration Value Initialization (PIVI) technique of [39], [41], which is also known as State-Metric Propagation (SMP) [42]. This avoids the extra computation associated with the pre-backwards step by initializing the windows during the current turbo decoding iteration using the boundary conditions 'inherited' from the previous iteration.

G. Turbo Code Performance

When analyzing the performance of error correcting codes, typically the BER of the code is plotted against the *SNR per bit* $E^{\mathrm{rx}}_{\mathrm{b}}/N_0$, where $E^{\mathrm{rx}}_{\mathrm{b}}$ is the energy received per message bit. A TC's BER plot can be used for determining the minimum $E^{\mathrm{rx}}_{\mathrm{b}}/N_0$ required for reliable communication.

Figure 10 provides a BER plot for a $R = 1/3$ LTE turbo code, which uses the schematic shown in Figure 4. Figure 10 shows that the error correction performance improves with successive iterations of the decoder, until about 8 iterations have been completed. Beyond this convergence point however, there are diminishing returns, resulting in very little further improvement.

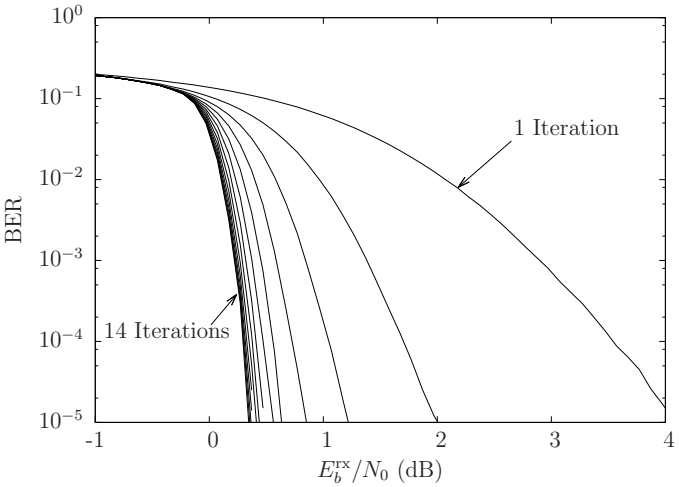


Fig. 10. BER performance of a 6144-bit $R = 1/3$ LTE turbo code employing varying number of iterations $I = 1, 2, 3, ..., 14$, for communication over an AWGN channel.

A specific feature of turbo codes is that they perform better with the aid of longer interleavers. Figure 11 shows the attainable BER performance for the message lengths of $N = 40$, 440 and 6144 bits, as well as for the uncoded BPSK case. While all of the turbo coded schemes offer an improved BER for $E^{\mathrm{rx}}_{\mathrm{b}}/N_0$ values above 0 dB, the longer frame lengths have a much steeper cliff than shorter ones. Owing to this, shorter frame lengths N correspond to higher $E^{\mathrm{rx}}_{\mathrm{b}}/N_0$ requirements for achieving reliable communication. Figure 11 shows that the LTE TC provides a coding gain G_c of around 8 dB over the uncoded scheme, which equates to a corresponding transmission energy saving at the transmitter.

Using (3) we can express the transmission energy per message bit $E^{\mathrm{tx}}_{\mathrm{b}}$ required to achieve a particular target BER

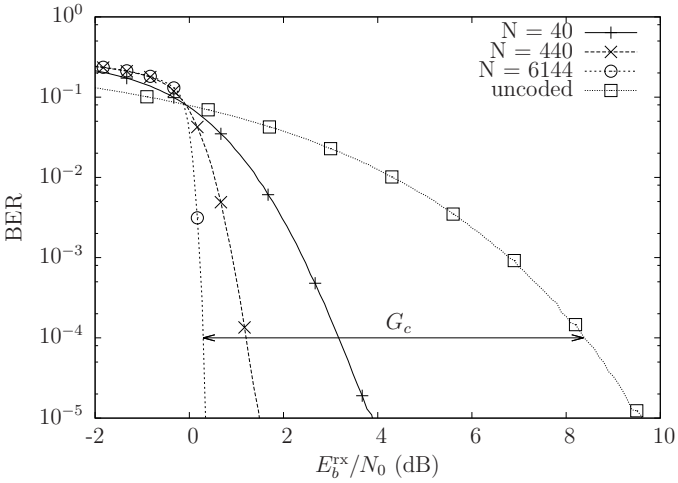


Fig. 11. BER performance of $R = 1/3$ LTE turbo codes employing 10 iterations and having different frame lengths when communicating over an AWGN channel. The coding gain $G_{\rm c}$ is achieved by the turbo code relative to the uncoded case, when achieving a target BER of 10^{-4}.

as

$$E_{\rm b}^{\rm tx}[{\rm dBJ}] = S_{\rm t} + N_0 + {\rm RNF} + P_{\rm l} + A - G_{\rm tx} - G_{\rm rx}, \quad (9)$$

where all quantities are expressed in dB, and $S_{\rm t}$ is the minimum SNR per bit $E_{\rm b}^{\rm rx}/N_0$ that is required to achieve the target BER.

III. Turbo Decoder Architectures

In this section, we will commence by reviewing the existing approaches to low power design for turbo decoders. We shall then narrow our focus to three major areas for formulating design considerations. Firstly, Section III-A considers the most significant challenges in energy-efficient datapath design, as well as in architectural solutions to these. Secondly, Section III-B considers the issues of algorithm control, where the scheduling of the decoder by the controller will be investigated, under the consideration of beneficial modifications to the algorithm that achieve a lower energy consumption at a minimal loss to error correction performance. This performance loss can then be considered during the holistic design stage, when minimizing the overall energy consumption of the system, as shown in Figure 3. Finally the various aspects of energy-efficient memory usage is discussed in Section III-C.

Table I shows a range of ASIC turbo decoder architectures disseminated in the literature, which have been designed for meeting a variety of design goals. In particular, the authors of [50] designed their low-dissipation architecture for low-throughput applications, where the energy consumption of the receiver is of primary concern. This architecture employs the LUT-Log-BCJR of [37], which provides a superior error correction performance and a reduced transmission energy compared to the faster, less complex Max-Log-BCJR [36] approximation. Low throughput turbo decoders also tend to have a reduced chip area, which results in a reduced static energy consumption and a reduced cost, which is often a concern in these applications. This is in contrast to conventional turbo decoder architectures [48], [49], [54], which are typically designed for bandwidth-constrained applications, such as cellular telephony, WLAN and broadcast systems. More specifically, these architectures are designed to have a high processing throughput, in order to match the high transmission throughputs that are sought in these applications. As a trade-off, these applications use the Max-Log-BCJR, which allows for a simpler approximation of the $\max^*$ calculation to support higher throughputs, but comes at the expense of both a degraded BER performance and an increased transmission energy requirement. Section III-B below discusses this trade-off, as well as methods aimed at mitigating their performance loss.

This section will concentrate on conventional decoders, however alternate approaches have also been proposed for implementing the BCJR algorithm, which will be briefly discussed here. Firstly, stochastic decoders [55] represent each LLR as a series of bits, where the value of the sequence is represented by how many '1's or '0's there are in the sequence. In contrast to the conventional fixed-point binary representation, each bit in a stochastic sequence has the same significance. During decoding, each bit in these LLR sequences is processed sequentially by the stochastic decoder. The decoder only processes one bit of each LLR in each clock cycle, which results in a significant reduction of the number of gates required in the decoder. However, since long LLR sequences are required for a high error correction performance, stochastic decoders typically require many more clock cycles compared to a conventional decoder, hence resulting in lower throughputs.

Another alternative architecture is constituted by the family of analog turbo decoders [56]. In these architectures, soft information is represented with the aid of analog currents, while the various operations of the decoder are performed using analog arithmetic circuits. Analog decoders have shown a promising decoding energy consumption $E_{\rm b}^{\rm dec}$, outperforming the comparable digital turbo decoders. However, they also impose additional challenges which have limited their potential. For example, the difficulties in matching analog circuits on a large scale leads to a potential performance degradation [57]. Furthermore, accurately simulating the BER performance of the circuit before its fabrication is not feasible or accurate. In [58] an analog architecture, which supports long frames is described, although this is associated with other challenges. In particular, a sampling circuit is required at each input of the decoder, which holds the analog value constant during decoding. However, these analog values cannot be readily maintained for extended periods of time, hence affecting the achievable error correction performance.

In the following subsections, we consider three salient aspects of conventional digital decoders, namely the design issues of the data path, of the controller and of the memory.

A. Data Path Considerations

Some of the designs listed in Table I rely on architectures that were designed for meeting the requirements of the latest telephony standards, resulting in optimizations for very high throughputs. These conventional architectures typically employ dedicated modules for each of the different steps in the

TABLE I
SUMMARY OF TURBO DECODING ARCHITECTURES FOR A VARIETY OF APPLICATIONS

Work	Algorithm	E_b/N_0 [dB] at BER = 10^{-3}	Technology [nm]	Voltage [V]	Area [mm^2]	Area at 90nm [mm^2]	Iterations	Throughput [Mb/s]	Energy consumption [nJ/bit/iter.]	Energy consumption @ 90nm, 1 V [nJ/bit/iter.]
Bickerstaff M. et al. 2002 [43]	LUT-Log	0.32	180	1.8	9	2.25	10	2	14.6	2.3
Bickerstaff M. et al. 2003 [44]	LUT-Log	0.37	180	1.8	14.5	3.63	8	24	11.1	2.0
Li F. et al. 2008 [45]	LUT-Log	-	180	1.8	8.2	2.05	6.5	4.17	12.7	1.59
Benkeser C. et al. 2009 [46]	Max-Log	0.46	130	1.2	1.2	0.58	5.5	20.2	0.54	0.26
May M. et al. 2010 [47]	Max-Log	-	65	1.1	2.1	4	6	150	0.31	0.35
Sun Y. et al. 2010 [48]	-	-	65	0.9	8.3	15.9	6	1280	0.11	0.19
Studer C. et al. 2011 [49]	Max-Log	0.63	120	1.2	3.57	2	5.5	390.6	0.37	0.19
Li L. et al. 2011 [50]	LUT-Log	0.32	90	1.0	0.35	0.35	5	1.03	0.4	0.4
Studer C. et al. 2012 [51]	Max-Log	-	180	1.8	0.45	0.11		542	0.84	0.13
Ilnseher T. et al. 2012 [52]	Max-Log	-	65	1.1	7.7	14.8	6	2150	-	-
Belfanti S. et al. 2013 [53]; Roth C. et al. 2014 [42]	Max-Log	0.66	65	1.2	2.49	4.8	5.5	1013	0.17	0.16

LUT-Log-BCJR decoding algorithm. More specifically, they use separate hardware for calculating each of the α, β, δ values and the extrinsic LLRs. However, this can result in a long critical path in the hardware implementation, which precludes having a high processing energy efficiency for the following three reasons:

1) Firstly, a lengthening of the critical path implies a greater variety of data path lengths. The differences amongst the data path lengths in the circuit may improse significant energy wastage owing to spurious transitions (glitches) [59]. Indeed, spurious transitions may account for a significant part of the dynamic energy consumption of ASIC implementations [60]. Reducing spurious transitions requires the lengths of the paths that converge at each register in the circuit to be roughly equal.
2) Secondly, a long critical path prevents the decoder from employing a high clock frequency. In order to implement the conventional LUT-Log-BCJR architecture at a high clock frequency, it is necessary to employ additional hardware during the synthesis for the sake of shortening the critical path. This is achieved by employing more complex circuits, such as the 'look-ahead adder' for minimizing their long datapaths. Unfortunately, this increases the chip area of the datapaths, hence resulting in a higher EC. On the other hand, operating at a lower clock frequency in order to avoid introducing this additional hardware would result in some of the hardware resources associated with shorter datapaths remaining idle for longer, hence increasing the static EC. The energy wasted by the static EC becomes more and more significant, when the process technology is scaled down [61].
3) Thirdly, the high complexity of the conventional architecture imposed by its circuits dedicated to the different tasks increases the requirements imposed on the clock tree and on the buffers for multiple input signal loads [62]. Hence, this may impose a significant additional energy dissipation on the decoder.

On this basis, we shall now discuss a pair of techniques, which can be employed for mitigating the energy inefficiencies inherent in designs having a long critical path.

The first method we will discuss is pipelining, which is employed extensively within the architectures of [48], [51], [52]. Pipelining reduces the critical path between two registers by adding additional registers to the middle of this path. This has the result of shortening the paths so that a higher clock frequency can be employed, but also adds latency to the circuit, since the number of clock cycles required before a result is available is increased for every pipeline stage that is added. This can therefore result in a slow down of a circuit's operation, if one part has to wait for a pipelined calculation to become available.

Figure 12 shows an example of pipelining in the turbo decoder of [51], which uses a similar decoder core to that proposed by the authors of [46], [49]. High-throughput turbo decoders, such as those proposed by [49], [52], [53], typically employ a multitude of these cores in parallel. The architecture of Figure 12 employs separate hardware units for calculating the α (forward state-metrics) and β (reverse state-metrics), each having dedicated hardware for generating the γ values. Since this architecture utilizes windowing, a separate dummy state-metric-recursion unit is used for generating the boundary conditions of the windows, as described in Section II-F. This parallelization within each decoder core facilitates higher throughputs than the alternative approaches. To perform the pipelining, registers are placed between the branch metric computation units that are used for calculating the γ values, as well as between the ACS Units that are used for calculating the α or β values. Note that due to their recursive nature, no pipelining can take place within the ACS Units. This is because the values for one bit depend on that of an adjacent bit, which is calculated in the preceding clock cycle. Adding pipelining to the ACS Unit then increases the number of cycles it takes for a new value to be calculated, hence slowing down the operation of the decoder, rather than speeding it up.

With careful pipelining, the critical paths in a design can be kept low and the path length can be kept more similar, therefore mitigating the previously mentioned impediments. However, as mentioned above, pipelining cannot be used in the recursive parts of the BCJR algorithm and the additional chip area as well as the EC associated with the pipeline registers must also be considered.

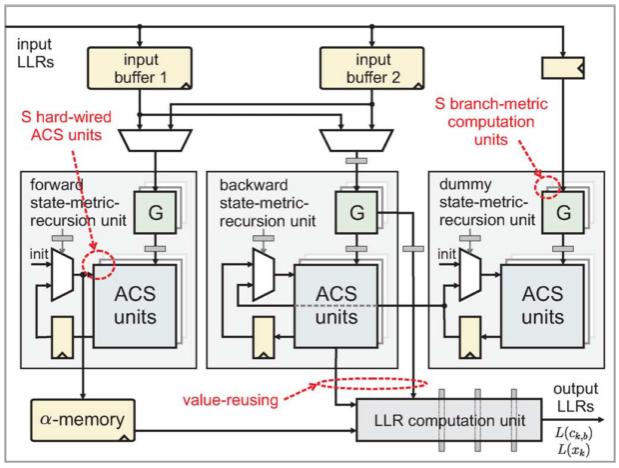


Fig. 12. Pipelining in a high throughput Max-Log-BCJR decoder. From ©IEEE [51]. The pipelining registers are shown by the gray rectangles.

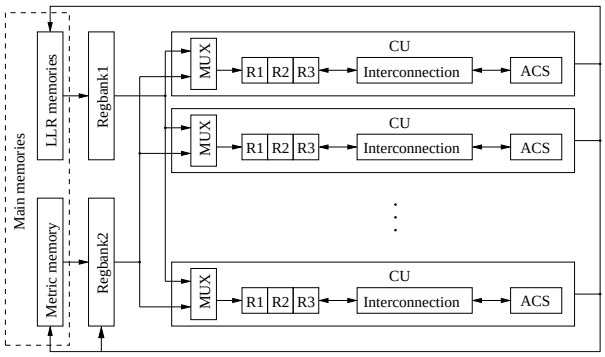


Fig. 13. The LUT-Log-BCJR architecture of [50].

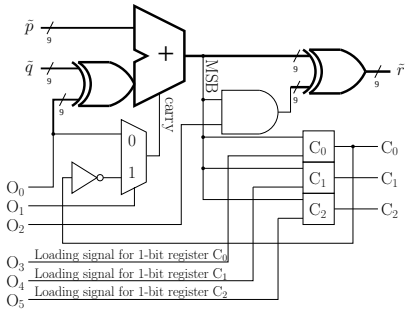


Fig. 14. The ACS unit of [50].

Building upon the pipelining philosophy, we shall now focus our attention on the turbo decoder architecture of [50], which is shown in Figure 13. This architecture has been specifically designed for a low processing EC for energy-constrained wireless communication applications, such as WSNs and the IoT. The philosophy of this architecture is to redesign the timing of the conventional architecture into a series of small steps, each with the same length, in a similar manner to that which is achieved when adding pipeline stages. In contrast to the high-throughput architectures discussed previously, where each of the pipelined stages are performed at the same time, the architecture of [50] sequentially carries out the operations using small functional units. This produces an architecture comprising only a low number of inherently low-complexity functional units, which are collectively capable of implementing the entire LUT-Log-BCJR algorithm at a high hardware efficiency. Further wastage is avoided, since the critical paths of the functional units are naturally short and have a similar length, hence eliminating the requirement for additional hardware to manage them.

Figure 14 shows an ACS Unit, which perform the necessary operations of the LUT-Log-BCJR algorithm, as previously discussed in Section II-F. The ACS Unit is based around an adder and an XOR gate, with the control signals O[5:0][1], and the status flags C[2:0]. We can see that changing the control signals changes the operation of the ACS unit. For example, the input code O[5:0] = 000000_2 performs addition, and O[5:0] = 100000_2 performs subtraction. As detailed in [50], this ACS Unit can perform a LUT-$\max^*$ operation over four clock cycles, when external control logic is used for correctly sequencing the ACS Unit. When calculating the LUT-$\max^*$, the status flags C[2:0] hold the result of the LUT operation, which is then used for selecting which of the four quantized values gleaned from Figure 9 are added on to the result in the final cycle of the LUT-max calculation.

Due to the short critical path and owing to the serial nature of this approach, it naturally results in a low chip-area and a high clock frequency, which implies having a low static EC. The architecture is based on the fact that the LUT-Log-BCJR comprises only addition, subtraction and $\max^*$ operations, which can be further decomposed into three fundamental operations, namely the ACS operations, as shown in Section II-E.

[1]This indicates the signal O is comprised of 6 bits

B. Algorithm Control

In this section, we consider the control of the architecture, where the controller instructs both the datapath and the memory to carry out a particular sequence of operations, in order to implement the algorithm.

In the LUT-Log-BCJR algorithm, the basic operation that imposes the highest computational overhead is the LUT-$\max^*$ operation [37]. This is of particular concern in high-throughput decoders, where the $\max^*$ calculation is used within the forward- and backward-recursive loops, preventing its pipelining for speeding up the decoder, as described in Section III-A. By contrast, the low-power, low-throughput architecture of Figure 13 does not suffer from this problem, since it performs all algorithmic steps using the same set of functional units, which are all capable of performing the same tasks, rather than having dedicated hardware for each part of the Log-BCJR algorithm. Owing to this, there are no parts of the decoder that are required to wait, while another part completes the operation of a slower task.

It is therefore desirable to favour the Max-Log BCJR over the LUT-Log-BCJR in applications, requiring a higher throughput. However, the naive employment of the Max-Log-BCJR results in a performance loss, when compared

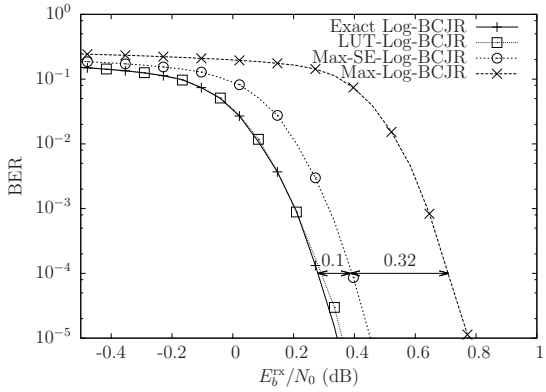


Fig. 15. Error correction performance of 6144-bit turbo decoders employing extrinsic scaling (Max-SE-Log-BCJR), the Max-Log-BCJR and the LUT-Log-BCJR, in relation to that offered by the exact Log-BCJR. A LUT comprising 8 entries was used for the LUT-Log-BCJR, and a scaling factor of 0.7 was employed for the Max-SE-Log-BCJR.

to the LUT-Log-BCJR. This motivates the employment of a technique known as extrinsic LLR scaling, which is capable of mitigating some of this performance loss [46], [63]. Figure 15 compares the error correction performance of the Log-BCJR, LUT-Log-BCJR, Max-Log-BCJR and Maximum with Scaled Extrinsic Log-BCJR (Max-SE-Log-BCJR) decoders. It can be seen that the extrinsic scaling technique improves the performance, which will be within a small margin of 0.1 dB of that offered by the Log-BCJR algorithm. This is a typical margin that may be observed for other turbo code parameterizations designed for communicating over AWGN and Rayleigh fading [64] channels.

The Max-SE-Log-BCJR decoder relies on multiplying the extrinsic LLR output of the decoder blocks in the receiver by a constant value of less than 1. This represents a reduction of confidence in the extrinsic LLRs, which is due to the non-optimal implementation of the $\max^*$ calculation. The author of [65] discuss the optimal selection of this constant, which is found to be between 0.6 and 0.8, depending on the SNR at the receiver. However, practical implementations tend to use a fixed scaling value [64]. A typical choice for the extrinsic scaling factor is one that leads to a simple hardware implementation using just adders. For example, a scaling factor of 0.75 can be achieved using fixed point arithmetic by simply adding the extrinsic output right-shifted once, to the extrinsic output right-shifted twice.

Extrinsic LLR scaling is also used in the Max-Log-BCJR architecture of [46], resulting in a 45% reduction in area and a 50% improvement in throughput, when compared to a similar architecture, which uses the LUT-Log-BCJR algorithm instead. The reduction in the number of logic gates required for the $\max^*$ calculation also results in a reduced EC.

As described above, the use of extrinsic LLR scaling in conjunction with the Max-Log-BCJR results in an error correction performance loss relative to the LUT-Log-BCJR decoder. This equates to more *transmit* energy being required, but offers the advantage of requiring lower *decoding* energy. Note that the holistic design method discussed in Section V will address these conflicting design choices. This conflict demonstrates the importance of considering both the architecture and the algorithm jointly, since a holistic design approach facilitates striking the right balance between the algorithm and the architecture, resulting in the lowest overall EC and the best overall performance for the system.

Another beneficial technique for the implementation of turbo decoders is the Radix-4 transformation of [44], [52], which combines two trellis stages into a single one. Owing to this, the decoder considers twice the number of *a priori* LLRs at once and the number of transitions emerging from each state of the Radix-4 trellis is squared. However, this technique halves the number of state metrics that have to be calculated and stored, since it halves the number of stages in the trellis. In the most common case, where only two transitions emerge from each state, the total number of transitions per frame will remain constant. This leads to a moderate area increase for radix-4 decoders over radix-2 decoders [49], partly because more ACS operations per transition are required, when considering several transitions at once. The main advantage of radix-4 decoders is that by transversing two states at once, the degree of parallelism can be doubled, hence facilitating higher throughputs.

There are a number of other techniques that may be employed in turbo decoder implementations, as follows.

- Early Stopping [46], [53], which terminates the turbo decoding process early, if the correct bit-stream is unlikely to be found, thus saving energy. This technique considers the values of the LLRs, and detects if their quality no longer improves in successive decoding iterations, indicating that the remaining errors in the message will not be corrected. Furthermore, early stopping can also stop the iterative decoding process once the correct message is found, as verified using a Cyclic Redundancy Check (CRC).
- Modulo normalization [46], [51], which allows the state metrics to overflow, relying on the nature of the two's complement arithmetic to correct this overflow, instead of requiring a larger number of bits to represent these metrics. An additional logic gate is required for the max logic, in order to allow it to correctly process numbers, which have experienced an overflow.
- Voltage scaling [46], [49], which reduces the supply voltage when the throughput requirements are lower, or when less iterations are required, because the SNR is higher, resulting in a reduced energy consumption.

C. Memory Considerations

Turbo decoder architectures require a large amount of memory for their operation. This memory is required for storing the *a priori* LLRs, the extrinsic LLRs generated by each of the Log-BCJR decoders and the intermediate α or β values of the Log-BCJR decoder, as discussed in Section II-E. While Section II-F discussed beneficial techniques, such as windowing for reducing the required memory, frequent access will still be required of this memory. Since accessing this memory dissipates energy [66], having an EC comparable to that of the datapath [50], it is desirable to minimize the number

of memory accesses, in order to reduce the overall EC of an architecture.

To address this issue, the architecture shown in Figure 13 additionally employs two register banks, namely Regbank1 and Regbank2, which act as a cache memory between the main memory and the processing units. The combined usage of both the dedicated registers and of the register banks allows an entire Log-BCJR stage of the trellis to be processed without requiring access to the main memory. A similar approach is pursued in [67], where a cache memory is employed between the LLR memories and the decoder. This reduces the required number of memory accesses, since each of the hardware blocks for the α, β and output LLR units access the cache rather than directly accessing the main memory.

As described in Section II-F, the Log-BCJR algorithm's data dependencies require an entire forward-recursion or backward-recursion to be carried out, before any extrinsic LLRs can be generated. This gives rise to the memory requirement for storing the α or β values calculated during this recursion. The authors of [52] have proposed an additional method for reducing the storage requirement of state metrics during this initial forwards- or backwards-recursion. This 're-computation' method reduces the number of values stored in the memory during on the initial recursion, which is achieved by storing only every nth set of state metrics. However, this requires the missing state metrics to be recalculated as and when needed, during the subsequent pass through the trellis. The implementation advocated in [52] opted for storing every 6th set of state metrics, since it was found that the extra hardware required for the re-computation circuit occupied a smaller area than the memory, which would otherwise have been required.

For any design, the required amount of memory storage and the number of memory accesses can be traded-off against the requirement of repeating the computation of unstored values in the decoder. However, as a minimum, the *a priori* LLRs have to be fetched from memory into the Log-BCJR decoder, while the extrinsic LLRs have to be stored from the Log-BCJR decoders into memory. The values, which require minimal computation may be readily recomputed as and when required, such as the γ values, which typically necessitate no more than a single addition per transition. Conversely, due to the data dependencies, memory will be required for at least some of the forwards- or backwards- recursion values, so that they can be stored until they are needed for the duration of a window.

In high-throughput decoders, that employ parallelization by concurrently operating multiple decoder cores, accessing the shared LLR memories may cause contention. As described in Section II-B, the interleavers within the turbo decoder dictate the memory accesses of the decoder cores. In particular, the interleavers enforce the requirement for the *a priori* and extrinsic LLR memories to be shared between each of the decoder cores, rather than having independent LLR memories for each of the decoding cores. In the case where there are M decoder cores, it is desirable for the LLR memories to be split into M separate memory blocks, with the interleaver designed for ensuring that only one decoder core requires access to each memory block at a time. An interleaver that meets this criterion for some values of M is said to be contention-free [68]. However, an interleaver which is not contention-free will cause inefficiencies in the decoder, since some of the decoding blocks will have to stall their operation, while they wait to individually access the memory.

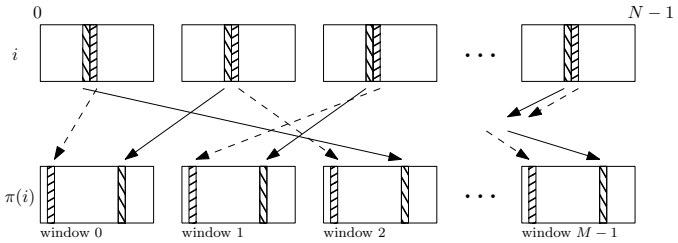


Fig. 16. Contention-free interleaver using the same indices within each window, where address i is interleaved to yield the address $\pi(i)$. The interleaving pattern is shown for two sets of addresses.

While contention-free interleavers allow the LLR memories to be broken into separate memory blocks, the address decoding logic has to be duplicated for each of these memory blocks, hence increasing both the chip area and the associated EC. It is therefore also desirable for each decoder core to fetch or store the LLRs using the same addresses for their corresponding one from the set of these M blocks of memory. This design of the interleaver will allow contention-free memory accesses to be implemented using a single address decoding circuit, since each decoder core uses the same address. As shown in Figure 16, each decoder core carries out its fetching or storing action using a different memory window, but the index used within each window is the same. A pair of specific interleaver designs which meet both of these criteria are constituted by the so-called ARP and QPP interleavers [68]. The QPP design was chosen for the LTE standard [1]. The specific interleaver design has a significant affect on the BER performance of a turbo code, hence requiring a careful design of the interleaver for meeting the contention-free implementation requirement, as well as the BER performance requirement [69]. The authors of [49], [70] demonstrated how to facilitate contention-free memory accesses, where a permutation network is employed for routing the LLRs between the memory and the decoder cores.

IV. Processing Energy Consumption Estimation

In this section, we will discuss various techniques invoked for characterizing the expected EC of a turbo decoder architecture. Referring to Figure 3, accurately processing EC estimation is important for the holistic design process, since it typically makes a similar contribution to the overall EC as the transmission energy in energy-constrained scenarios. Indeed, in some applications, the processing energy can actually exceed the transmission energy. We commence by briefly considering the most common design characterization methods, before focusing our attention on the method of [27]. This work parametrizes the estimated EC per bit per iteration, therefore aiding the joint design of the architecture and the algorithm, as it will be further discussed in Section V. The energy per bit per iteration is employed as the metric for comparing the

EC of different architectures, since it is independent from the algorithm which is being operated. Furthermore, the EC per bit metric is preferred over the power consumption per bit, since the EC is independent of the decoder's throughput.

For the majority of the ASIC architectures proposed by the authors listed in Table I, the EC of the architecture is obtained from post-layout simulations. The EC per bit per iteration can then be readily derived by taking into consideration both the throughput and the number of iterations employed. However, when characterizing the EC as a function of the TC parameters, the above-mentioned approach has the disadvantage of having to modify the design and to rerun the post-layout simulations for each of the different parameters that are considered during the holistic optimization. Table I lists a range of architectures designed for a variety of applications, resulting in a diverse range of throughputs and EC figures. These EC results are obtained from simulations using only a single particular parameterization of the design.

By contrast, a different framework was proposed in [27] for estimating the EC of a Log-BCJR decoder as a function of its parameters, which can be generalized to any turbo decoder. The objective of this framework is to quantify the EC during the TC design stage, in order to assist the designer in selecting appropriate parameters for the code.

In order to provide accurate EC predictions, the authors of [27] stipulate some assumptions, which are based on the later implementation stages. In particular, the Integrated Circuit (IC) fabrication process technology [49], the supply voltage and the clock frequency of the implemented circuit can all have a significant impact on the EC. When the designer wishes to consider a range of technology nodes or supply voltages (V_{dd}), the chip area, throughput and energy consumption can be scaled according to the scaling rules as follows [53].

$$s = l_{\mathrm{old}}/l_{\mathrm{new}}, \tag{10}$$

$$\mathrm{Area} \sim 1/s^2, \tag{11}$$

$$t_{\mathrm{pd}} \sim 1/s, \tag{12}$$

$$P_{\mathrm{dyn}} \sim 1/s(V'_{\mathrm{dd}}/V_{\mathrm{dd}})^2, \tag{13}$$

where s is the scaling factor between the two technology nodes, t_{pd} is the propagation delay, and P_{dyn} is the dynamic power consumption. Reducing t_{pd} increases the clock frequency the IC can operate at, which results in an increased throughput. The power consumption reduces with the technology node, which results in a corresponding reduction of $E_{\mathrm{b}}^{\mathrm{dec}}$. This allows the energy analysis to be performed only once, and then scaled to allow holistic design decisions to be taken. The specific parameters which affect the overall EC are summarized in Table II. When using the technique of [27] for estimating the Log-BCJR decoder's EC, the designer has the ability to change these parameters, in order to investigate their impact on the EC.

In order to derive an overall EC estimate for a turbo decoder, the EC is divided into three main components which will be discussed here. Each of these steps focuses on the three areas discussed in Section III, namely on the datapath, on the scheduling of the decoder by the controller and on the memories.

TABLE II
SUMMARY OF THE VARIABLES IN THE ENERGY ESTIMATION FRAMEWORK.

k	the number of inputs of each component encoder
m	the number of memory elements of each component encoder
n	the number of non-systematic outputs of each component encoder
N	the block length
I	the number of decoding iterations performed
v	the supply voltage
f	the clock frequency
l	the technology node
z	the number of bits employed in the fixed-point number representation

1) Datapath functional unit characterization: The first step of the technique conceived in [27] is to analyze the EC of each of the sub-blocks that comprise the datapath of the architecture. More specifically, the energy used by the different sub-blocks as they perform the tasks of addition, subtraction and max* is characterized as functions of the related parameters. It was found in [27] that the complexity of some sub-blocks varies according to some of the parameters of Table II. In particular, the number of states in the decoder and the number of bits used for number representations have a significant effect upon the EC. It is therefore suggested that the Register-Transfer Level (RTL) design [71] of the functional units should be written in a way that allows the parameters to be readily changed, in order to characterize a whole range of EC results.

2) Timing analysis: Next, the base operations undertaken by the decoder as instructed by the controller are analyzed for each time-step. More specifically, for a given set of turbo code parameters and a given set of implementation parameters of Table II, the total number of addition, subtraction, max* and idle operations undertaken by each of the functional units of the decoder can be characterized. This therefore characterizes how often each of the operational modes of the datapath are used during decoding. This allows the designer to promptly characterize the effect of the different parameters of Table II, which can be used in the ensuing steps to examine, how the EC is affected by changing the parameters.

The results from the previous two steps can be combined to estimate for the EC of the datapath, when considering a set of given parameters and a particular scheduling of operations. By multiplying the energy used per operation of step 1) and the number of operations per bit from step 2) an accurate estimate if the overall EC can be made. It was shown in [27] that this method of estimating the EC has at most 7% error, when compared against the EC simulation of the entire decoder.

3) Memory power usage: The databook provided for the memories by the standard library developer [72] provides specifications, which allow the EC to be calculated. For a technology scale of $l = 90\mathrm{nm}$, the Taiwan Semiconductor Manufacturing Company (TSMC) 90 nm databook [72] states that the power consumption of a particular memory module size can be estimated by considering both the accessing rate a in units of accesses per clock cycle, as well as the clock frequency f and the supply voltage v. In the standard cell

library, the power consumption of the SRAM used in the architecture can be estimated using the reference table of [72]. Here, the typical memory access power consumption p_{a} and leakage current I_{l} are given for memory blocks having various sizes and operand-widths. The power consumption P_{a} can be used for calculating the dynamic EC, when the memory is being accessed. Similarly, the leakage current I_{l} can be used for calculating the static EC of the memory, when it is idle.

Similarly to the EC of the datapath and of the memories discussed above, the EC of the interleaver and of the controller may also have to be considered. The authors of [27] provided the analysis of the EC of these components. However, it was found that their contribution is minor compared to that of the datapath and memories. Furthermore, their EC per bit per Log-BCJR decoder activation is unlikely to change between different parameterizations. Owing to this, when making comparisons between two candidate scheme parameterizations, any error in the interleaver or controller EC estimation will be common to both schemes, hence having little effect on the comparison.

V. Holistic Design Characterization

In this section, we explore a range of methods capable of characterizing and holistically parameterizing an overall wireless communications system, while investigating the energy efficiency of different TCs and the effect their parameters. We shall explore the techniques outlined by the authors of [4] and [27], showing how these techniques can be applied to a specific scenario and architecture, in order to demonstrate the holistic design approach and to show the effect of the various system parameters on the overall EC. By considering the energy consumption in both the transmitter and the receiver, the candidate TCs may be evaluated holistically for employment in energy-constrained applications, such as WSNs and the IoT. More specifically, the transmitter's energy consumption is comprised of the turbo encoder's processing energy consumption $E_{\mathrm{b}}^{\mathrm{enc}}$, the modulator's energy consumption $E_{\mathrm{b}}^{\mathrm{mod}}$ and the PA energy consumption $E_{\mathrm{b}}^{\mathrm{tx}}$. Likewise, the receiver energy consumption is comprised of the demodulator's energy consumption $E_{\mathrm{b}}^{\mathrm{dem}}$ and the turbo decoder's processing energy consumption $E_{\mathrm{b}}^{\mathrm{dec}}$. The techniques discussed in this section are similar to various other examples of holistic characterization that are available in the literature [73]–[75]. For example, the authors of [75] considered the holistic optimization of cellular networks, while the authors of [73], [74] investigated whether Multiple-Input Multiple-Output (MIMO)-based sensor networks can provide energy savings over conventional networks.

The conventional design method optimizes the algorithm and architecture separately, without considering the processing EC at the receiver alongside the transmission EC. By contrast, the methods explored in this section allow the TC to be used for reducing the overall EC of a wireless communication system. As highlighted in Figure 3, the holistic design characterization bridges the algorithm design and the implementation design, allowing the parameters of each to be combined, when considering the performance of the eligible schemes for a particular design scenario.

The objective of the design methods described is to determine the particular parametrization of the TC design that optimizes the overall EC of the system over the range of operating conditions expected in a particular scenario. The component encoder of the design is specified by the parameters k, m and n, as well as by the generator polynomial. Furthermore, different turbo coding schemes may be employed, which may use different arrangements of the component encoders. For example, Multiple-Component Turbo Codes (MCTCs) [76] employ multiple parallel component encoders, where the number of encoders employed also becomes a parameter of the scheme. Further parameters to be considered are those, which relate to the hardware implementation, such as which $\max^*$ approximation to utilize, as well as the number of bits used for representing the LLRs and other internal variables. Additionally, the number of decoding iterations performed also affects both the decoding EC $E_{\mathrm{b}}^{\mathrm{dec}}$ and the minimum required transmission EC $E_{\mathrm{b}}^{\mathrm{tx}}$ quite significantly.

The holistic design approaches of [4] and [27] go beyond the approaches proposed by the authors of [76], [77]. In these contributions, the decoder complexity is quantified by the number of operations undertaken in the decoder, which is related both to the number of Log-BCJR decoder activations and to the number of states in the trellis. This measure of complexity is used for representing the relative energy consumption of different codes. However, as shown in Section III, the absolute energy consumption heavily depends on the architecture, as well as on factors such as the amount of memory in the design. As an example, [27] shows that two different schemes having the same operations-based complexity have a 45% difference in their processing EC $E_{\mathrm{b}}^{\mathrm{dec}}$. Furthermore, schemes having a lower coding rate also result in the modulation of more bits on to the channel, resulting in a higher $E_{\mathrm{b}}^{\mathrm{mod}}$ and $E_{\mathrm{b}}^{\mathrm{dem}}$. This illustrates that while the complexity-based comparison of [76] is useful for comparing the relative processing EC of schemes where the Log-BCJR decoders are similar, it does not allow the overall EC to be optimized, since it does not facilitate a fair comparison between different architectural parameterizations. This is because it has no knowledge of how the architecture performs the decoding, wherein different parameterizations of the architecture will cause different activation of blocks in the decoder. During the holistic optimization, the designer may also wish to compare the performance of different architectural parameterizations, which is not provided by the approaches disseminated in [76], [77].

In order to demonstrate the holistic design techniques, this tutorial considers a scenario, which is representative of a low-power, relatively low-throughput receiver, as is typical in WSNs and in the IoT. By using the energy estimation techniques discussed in Section IV, we may obtain a reliable estimation of the processing EC for different turbo code parameters. We shall consider a Twin-Component Turbo Code (TCTC) as discussed in Section II-D, as well as two MCTCs [76] having three and four constituent codes, which we refer to as 3MCTC and 4MCTC, respectively. The TCTC and 3MCTC schemes are both $R = 1/3$-rate codes, while the 4MCTC is a $R = 1/4$-rate code. These TCs employ the generator polynomials $(17, 15)_o$, $(4, 7)_o$ and $(2, 3)_o$, respectively. A fourth

scheme considered is provided by the uncoded case, which is associated with no TC processing energy $E_{\mathrm{b}}^{\mathrm{dec}}$, allowing us to explore the specific situations, where using TCs is the most energy efficient. A number of different parameters will also be considered for each of these codes. Furthermore, we will investigate the effect of employing various approximations of the Log-BCJR algorithm, namely the LUT-Log-BCJR, the Max-Log-BCJR and the Max-SE-Log-BCJR [46]. In particular, we will explore which approximations are most appropriate, when attempting to reduce the overall EC. The number of iterations in the receiver will also be considered in the holistic characterization.

TABLE III
ENVIRONMENT ASSUMPTIONS AND SYSTEM SPECIFICATION FOR THE TWO CONSIDERED WSNS.

Parameter	Scenario 1	Scenario 2
Transmission frequency (f_c)	5.8 GHz	433 MHz
Power amplifier efficiency loss (A)	4.8 dB	5 dB
Transmit antenna gain (G_{tx})	2 dBi	-2 dBi
Receive antenna gain (G_{rx})	7 dBi	3 dBi
Receiver noise figure (RNF)	6 dB	15 dB
Path loss exponent (p)	4	2
BER target	10^{-5}	10^{-5}
Temperature	300 K	300 K
Thermal noise (N_0)	-203.8 dBJ	-203.8 dBJ
Transmitter modulator power consumption (P^{mod})	4.6 mW	1.7 mW
Receiver demodulator power consumption (P^{dem})	6.5 mW	3.3 mW
Symbol period (t_{s})	$1\mu s$	$4\mu s$

Table III shows two different operating scenarios, which will be considered in this tutorial, representing a range of environmental factors faced by energy-constrained systems. The power consumption figures P^{mod} and P^{dem} are representative of those achieved by a particular low-power state-of-the-art transceiver [78]. While naturally, only a limited number of parameterizations are considered in this tutorial, the designer of a real communications system may wish to consider a wider range of candidate schemes. For example, error correction codes such as LDPC [17], Repeat Accumulate (RA) [79], or Reed-Solomon (RS) [80] codes may provide a lower overall energy consumption, depending on the scenario. For example, TCs out perform LDPC codes at lower coding rates [81], while LDPC and RA codes lend themselves to be conveniently implemented in parallel, albeit at the expense of a large chip area. Likewise, the designer may wish to consider Hybrid Automatic Repeat Request (HARQ) [82] as an alternative method of reducing the EC. While these rate-compatible schemes have been shown to reduce the transmission EC of a scheme, usually the EC of the decoder is not considered. The techniques detailed in this section can be extended to both HARQ and to other similar techniques, however for reasons of space-economy, they are not discussed here.

A. Methodology

To evaluate the transmission energy required for each candidate scheme, first the BER requirement must be specified based on the target application. Here, a BER of 10^{-5} is assumed as the maximum tolerable BER. Table IV shows the $E_{\mathrm{b}}^{\mathrm{rx}}/N_0$ required for achieving a BER of 10^{-5} for each of the candidates considered in this tutorial. The corresponding BER simulation results of these candidate schemes are provided in [26], while the relative performance of different approximations of the Log-BCJR algorithm are taken from [64]. Next, the path loss model given in Section II-C is used for calculating the transmission EC per information bit $E_{\mathrm{b}}^{\mathrm{tx}}$, by invoking Equation (9). This path loss model may be substituted by alternative channel models, such as a Rayleigh fading [83] channel, if this is more appropriate for the design scenario. The assumptions and specifications for the target scenario of Table III are applied to the specific path loss model, having the parameters defined in Section II. Furthermore, the decoding EC $E_{\mathrm{b}}^{\mathrm{dec}}$ of the candidate schemes can be estimated using the techniques discussed in Section IV. The decoding EC $E_{\mathrm{b}}^{\mathrm{dec}}$ for the example architecture of Figure 13 is shown alongside the required $E_{\mathrm{b}}^{\mathrm{rx}}/N_0$ in Table IV. Using the coding rates of the candidate schemes, the modulation and demodulation energy consumption can be calculated according to

$$E_{\mathrm{b}}^{\mathrm{mod}} = P^{\mathrm{mod}} \times t_{\mathrm{s}}/\eta, \tag{14}$$

$$E_{\mathrm{b}}^{\mathrm{dem}} = P^{\mathrm{dem}} \times t_{\mathrm{s}}/\eta. \tag{15}$$

The encoding energy $E_{\mathrm{b}}^{\mathrm{enc}}$ is typically considerably lower than the decoding EC [84], and therefore in this design example it is assumed to be negligible. Finally, the overall EC of the candidates can be calculated by summing these figures according to $(E_{\mathrm{b}}^{\mathrm{tx}} + E_{\mathrm{b}}^{\mathrm{mod}} + E_{\mathrm{b}}^{\mathrm{enc}} + E_{\mathrm{b}}^{\mathrm{dec}} + E_{\mathrm{b}}^{\mathrm{dem}})$, in order to obtain the combined energy consumption per bit.

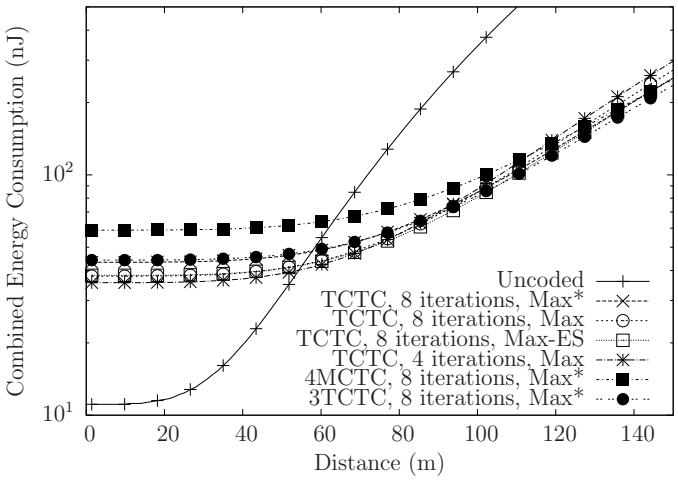


Fig. 17. Combined EC for the four candidate schemes, when operating in scenario 1 of Table III

B. Results

Figures 17 and 18 show the combined transmission and processing EC for the four candidate schemes, when operating in Scenarios 1 and 2, respectively. These graphs show how the combined EC increases with the transmission distance, allowing the designer to make decisions based on the range of required distances. It can be seen that for very short transmission distances, the uncoded candidate scheme has

TABLE IV
THREE TURBO CODE SCHEMES WITH THEIR DECODING ENERGY CONSUMPTION PER BIT ($E_{\mathrm{b}}^{\mathrm{dec}}$), AND THE E_{b}/N_0 FOR WHICH A BER OF 10^{-5} IS ACHIEVED.

	Max				Max-SE (Scaled Extrinsic)				LUT-Max*			
	$E_{\mathrm{b}}^{\mathrm{dec}}$ (nJ)		E_{b}/N_0 (dB)		$E_{\mathrm{b}}^{\mathrm{dec}}$ (nJ)		E_{b}/N_0 (dB)		$E_{\mathrm{b}}^{\mathrm{dec}}$ (nJ)		E_{b}/N_0 (dB)	
Iterations	4	8	4	8	4	8	4	8	4	8	4	8
Uncoded	0	0	9.5	9.5	0	0	9.5	9.5	0	0	9.5	9.5
TCTC	2.31	4.63	1.4	1.0	5.00	10.01	1.0	0.6	5.00	10.01	0.9	0.5
3MCTC	2.52	5.04	1.6	0.6	5.45	10.90	1.2	0.2	5.45	10.90	1.1	0.1
4MCTC	3.36	6.72	1.6	0.6	7.27	14.53	1.1	0.2	7.27	14.53	1.0	0.1

TABLE V
THE COMBINED TRANSMIT, RECEIVE, AND DECODING ENERGY CONSUMPTION ($E_{\mathrm{b}}^{\mathrm{tx}}$ + $E_{\mathrm{b}}^{\mathrm{dec}}$+ $E_{\mathrm{b}}^{\mathrm{mod}}$+ $E_{\mathrm{b}}^{\mathrm{dem}}$) [NJ] OF THE THREE SCHEMES UNDER THE DIFFERENT CONDITIONS AT A RANGE OF DISTANCES d. BRACKETS SHOW NUMBER OF ITERATIONS PERFORMED AND APPROXIMATION USED. BOLD FONT INDICATES THE LOWEST ENERGY CONSUMPTION OBTAINED FOR EACH TRANSMISSION DISTANCE.

				Scenario 1						Scenario 2				
Scheme	$E_{\mathrm{b}}^{\mathrm{dec}}$	E_{b}/N_0	η	d [m]	10	20	50	100	250	2000	5000	10000	20000	50000
uncoded	0.0	9.5	1.0		**11.1**	**11.6**	**31.9**	343.6	13001.3	**23.9**	**44.3**	117.3	409.1	2451.7
TCTC (4,Max)	2.3	1.4	0.3		35.6	35.7	38.8	87.4	2056.8	62.9	66.1	77.4	122.9	440.7
TCTC (4,Max-SE)	2.4	1.0	0.3		35.7	35.8	38.7	82.7	1870.6	63.0	65.9	**76.2**	117.4	405.9
TCTC (8,Max)	4.6	1.0	0.3		37.9	38.0	40.8	84.4	1851.8	65.2	68.0	78.2	119.0	404.2
TCTC (8,Max-SE)	4.9	0.6	0.3		38.2	38.2	40.8	**80.7**	1700.1	65.4	68.0	77.3	**114.6**	376.0
TCTC (8,Max-LUT)	10.0	0.5	0.3		43.3	43.4	45.9	84.7	1660.0	70.5	73.0	82.1	118.4	372.6
3MCTC (4,Max-SE)	2.6	1.2	0.3		36.0	36.0	39.0	85.4	1966.7	63.2	66.3	77.1	120.5	424.1
3MCTC (8,Max-LUT)	10.9	0.1	0.3		44.2	44.3	46.6	82.1	**1525.8**	71.3	73.7	82.0	115.3	**348.3**
4MCTC (4,Max-SE)	3.5	1.1	0.3		47.9	48.0	51.0	96.4	1942.8	84.1	87.1	97.7	140.3	438.2
4MCTC (8,Max-LUT)	14.5	0.1	0.3		58.9	59.0	61.3	96.8	1536.7	95.0	97.3	105.6	138.8	371.2

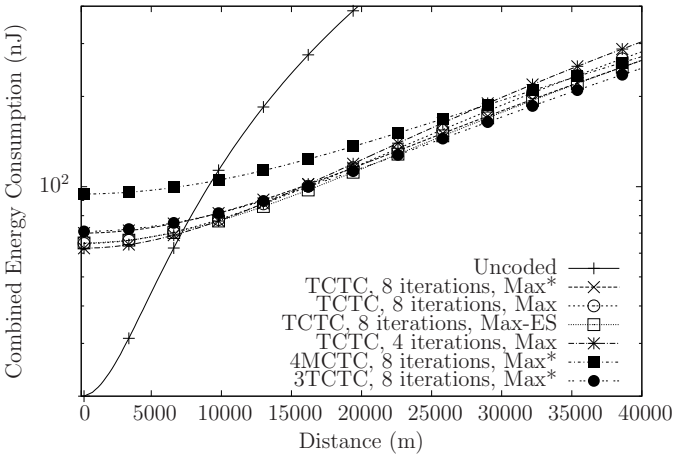


Fig. 18. Combined EC for the four candidate schemes, when operating in scenario 2 of Table III

the lowest EC due to the processing overhead of the turbo coded schemes, as well as the additional modulator $E_{\mathrm{b}}^{\mathrm{mod}}$ and demodulator $E_{\mathrm{b}}^{\mathrm{dem}}$ energy required for transmitting the additional parity bits. As the distance increases, the turbo coded schemes overtake the uncoded one, since they facilitate a lower transmit energy. The low-complexity TC schemes have an advantage for shorter transmission distances, while the transmit power dominates the EC over longer distances, where the best performing scheme becomes the one having the best BER performance.

Table V summarizes the combined EC for a selection of the candidate schemes over a range of distances. It can be seen that the Max-SE-BCJR decoder offers an attractive trade-off. At short distances, it offers an energy saving due to its lower processing energy $E_{\mathrm{b}}^{\mathrm{dec}}$ compared to the LUT-Max-BCJR decoder, while at higher distances its slight BER performance degradation results in only a small increase of the overall EC. Compared to the Max-BCJR decoder, the Max-SE-BCJR decoder offers an improvement for the majority of distances considered. Indeed, the only distances for which the Max-BCJR decoder offers a lower combined EC than the Max-SE-BCJR decoder is at distances, where the uncoded scheme provides the lowest EC.

The schemes offering the best BER performance are 3MCTC and 4MCTC of Table IV, however they also have the highest decoding EC $E_{\mathrm{b}}^{\mathrm{dec}}$. As a result, these schemes provide the lowest overall EC at longer distances, especially, when compared to the TCTC schemes, which have a slightly worse BER performance.

The authors of [4] refer to the point at which using an error correcting code becomes beneficial over uncoded transmission as the critical distance d_{cr}. This can be expressed as follows

$$\begin{aligned} &E_{\mathrm{b}}^{\mathrm{tx,e}}(d_{\mathrm{cr}}) + E_{\mathrm{b}}^{\mathrm{mod,e}} + E_{\mathrm{b}}^{\mathrm{dem,e}} + E_{\mathrm{b}}^{\mathrm{dec}} + E_{\mathrm{b}}^{\mathrm{enc}} \\ =&E_{\mathrm{b}}^{\mathrm{tx,u}}(d_{\mathrm{cr}}) + E_{\mathrm{b}}^{\mathrm{mod,u}} + E_{\mathrm{b}}^{\mathrm{dem,u}} \end{aligned} \tag{16}$$

where $E_{\mathrm{b}}^{\mathrm{tx,u}}(d_{\mathrm{cr}})$ is the transmission EC of the uncoded scheme at the critical distance, $E_{\mathrm{b}}^{\mathrm{tx,e}}(d_{\mathrm{cr}})$ is the transmission EC of the coded scheme at the critical distance, and $E_{\mathrm{b}}^{\mathrm{mod,e}}$, $E_{\mathrm{b}}^{\mathrm{dem,e}}$ and $E_{\mathrm{b}}^{\mathrm{mod,u}}$, $E_{\mathrm{b}}^{\mathrm{dem,u}}$ are the energy consumptions for the respective encoded and uncoded modulator and demodulator components. The critical distance depends on the particular error correction code used, as well as on all of the other factors shown in Table III. Figure 19 shows how the critical distance varies both with the carrier frequency and with the path loss coefficient p for a variety of schemes.

The case study of [76] offers a simple example for demonstrating the philosophy of the proposed holistic design method. Naturally, numerous idealized simplifying assumptions of the

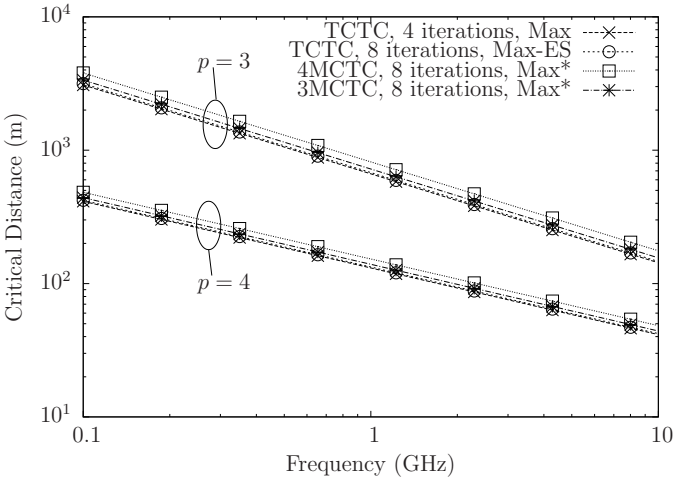


Fig. 19. Critical distances for path loss exponents of p=3 and p=4

environment and of the WSN specifications had to be stipulated here for avoiding distraction from the holistic design methodology. As a benefit, the design methodologies discussed here are capable of assisting the designer in holistically optimizing a TC design by considering numerous different design aspects. For example, apart from the basic parameters of TC schemes that were considered in our example, the longest block length N of a TC determines both the memory requirement of the hardware implementation. The number of decoding iterations performed has a significant effect on both the BER performance and on the decoding EC. Additionally, the number of hops employed in a multi-hop network determines the average transmission range and the sensor densities. All of these aspects directly affect both the transmission EC and the decoding EC. As a result, these design methods can be used for optimizing a wide variety of related specifications for improving the system's energy efficiency.

VI. Conclusions and Design Guidelines

In conclusion, energy-constrained scenarios such as WSNs and the IoT constitute emerging applications for TCs, where they can be employed for reducing the overall EC of communication systems. To achieve this goal, new design methodologies are required for TCs, which consider the EC throughout the entire design process. The key issue is to ensure that the potentially high-complexity turbo decoder has only a moderate EC $E_{\mathrm{b}}^{\mathrm{dec}}$, while also reducing the transmission energy $E_{\mathrm{b}}^{\mathrm{tx}}$. By achieving this goal, a holistically-considered overall EC $(E_{\mathrm{b}}^{\mathrm{tx}} + E_{\mathrm{b}}^{\mathrm{dec}} + E_{\mathrm{b}}^{\mathrm{enc}} + E_{\mathrm{b}}^{\mathrm{mod}} + E_{\mathrm{b}}^{\mathrm{dem}})$ can be realized. In this tutorial, the parameters of TCs were detailed and turbo decoder architecture design techniques were presented, particularly for the case of TCs specifically designed for reducing the EC of the decoder, without impacting the error correcting performance. Furthermore, energy estimation methods were conceived for estimating $E_{\mathrm{b}}^{\mathrm{dec}}$ during an early design stage. Based on these three topics, holistic TC design methods were proposed for reducing the overall EC. The selected design guidelines may be summarized as follows:

- Determine the environmental parameters of the target scenario, as exemplified in Table III.
- Establish the path loss model, as exemplified by the path loss model of Section II-C.
- Select the code design candidates, for example the candidates shown in Table IV including the specific design parameters of Table II and the architectural approximations discussed in Section III.
- Invoke the energy estimation framework of Section IV for estimating the processing EC $E_{\mathrm{b}}^{\mathrm{dec}}$ of the candidates.
- Using BER simulations and the path loss model, estimate the transmission EC $E_{\mathrm{b}}^{\mathrm{tx}}$ of the candidates, as demonstrated in Section V.
- Compare the overall ECs $(E_{\mathrm{b}}^{\mathrm{tx}} + E_{\mathrm{b}}^{\mathrm{dec}} + E_{\mathrm{b}}^{\mathrm{enc}} + E_{\mathrm{b}}^{\mathrm{mod}} + E_{\mathrm{b}}^{\mathrm{dem}})$ to find the most energy-efficient design, as demonstrated in Section V.
- Implement this most energy efficient design.

Using the discussed holistic design method, a specific design example was presented. The results demonstrated that the design methods presented are capable of finding the most desirable TC design and architectural choices from an energy efficiency point of view. The conventional approach, which used the BER and computational complexity derived from the number of states and decoder activations, could not achieve this optimal design decision due to its separate consideration of the architecture and algorithm.

In a communications system there are a wide variety of conflicting design trade-offs. The holistic design techniques of this paper allow all of the relevant trade-offs to be considered together, for the sake of minimizing the overall energy consumption. In particular, a joint optimization of these trade-offs can be used for holistically improving the entire system.

References

[1] *ETSI TS 136 212 LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); multiplexing and channel coding*, V10.2.0 ed., 2011.

[2] *IEEE 802.11n-2009 Standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - Part 11: wireless LAN Medium Access Control (MAC) and Physical Layer (PHY)*, 2009.

[3] "Digital video broadcasting (DVB); framing structure, channel coding and modulation for satellite services to handheld devices (SH) below 3 GHz," 2010.

[4] S. L. Howard, C. Schlegel, and K. Iniewski, "Error control coding in low-power wireless sensor networks: When is ECC energy-efficient?" *EURASIP Journal of Wireless Communications and Networking, Special Issue: CMOS RF Circuits for Wireless Applications*, vol. 2006, Apr., pp. 1–14, 2006.

[5] N. Sadeghi, S. Howard, S. Kasnavi, K. Iniewski, V. Gaudet, and C. Schlegel, "Analysis of error control code use in ultra-low-power wireless sensor networks," in *Proceedings of International Symposium on Circuits and Systems*, Island of Kos, 2006, pp. 3558–3561.

[6] M. E. Pellenz, R. D. Souza, and M. Fonseca, "Error control coding in wireless sensor networks," *Telecommunication Systems*, vol. 44, no. 1-2, pp. 61–68, 2009.

[7] A. Brokalakis and I. Papaefstathiou, "Using hardware-based forward error correction to reduce the overall energy consumption of WSNs," in *2012 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, April 2012, pp. 2191–2196.

[8] J. Misic, "Enforcing patient privacy in healthcare WSNs using ECC implemented on 802.15.4 beacon enabled clusters," in *Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, 2008.

[9] M. Dermibas, "Wireless sensor networks for monitoring of large public buildings," *University at Buffalo, Tech. Rep*, 2005.

[10] V. Potdar, A. Sharif, and E. Chang, "Wireless sensor networks: a survey," *2009 International Conference on Advanced Information Networking and Applications Workshops*, pp. 636–641, May 2009.

[11] M. R. Yuce, "Implementation of body area networks based on MICS/WMTS medical bands for healthcare systems," in *IEEE Engineering in Medicine and Biology Society Conference (IEEE EMBC08)*, August 2008, pp. 3417–3421.

[12] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *Communications Magazine, IEEE*, vol. 40, no. 8, pp. 102–114, 2002.

[13] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error correcting coding and decoding: Turbo codes," in *Proceedings of the IEEE International Conference on Communications*, vol. 2, Geneva, Switzerland, 1993, pp. 1064–1070.

[14] 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.

[15] *IEEE Standard for Local and Metropolitan Area Networks. Part 16: Air Interface for Fixed Broadband Wireless Access Systems, IEEE 802.16-2004*, IEEE Std., 2004.

[16] C. E. Shannon, "A mathematical theory of communications," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423,623–656, 1948.

[17] R. Gallager, "Low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, January 1962.

[18] L. Hanzo, T. H. Liew, B. L. Yeap, R. Tee, and S. X. Ng, *Turbo Coding, Turbo Equalisation and Space-Time Coding*. John Wiley & Sons Inc, 2011.

[19] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 493–497, 1967.

[20] C. Desset, B. Debaillie, V. Giannini, A. Fehske, G. Auer, H. Holtkamp, W. Wajda, D. Sabella, F. Richter, M. J. Gonzalez, H. Klessig, I. Godor, M. Olsson, M. A. Imran, A. Ambrosy, and O. Blume, "Flexible power modeling of LTE base stations," in *2012 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, April 2012, pp. 2858–2862.

[21] T. Limberg, M. Winter, M. Bimberg, R. Klemm, E. Matus, M. B. Tavares, G. Fettweis, H. Ahlendorf, and P. Robelly, "A fully programmable 40 GOPS SDR single chip baseband for LTE/WiMAX terminals," in *ESSCIRC 2008 - 34th European Solid-State Circuits Conference*. IEEE, September 2008, pp. 466–469.

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

[23] G. Auer, V. Giannini, and C. Desset, "How much energy is needed to run a wireless network?" *Wireless Communications, IEEE*, vol. 18, pp. 40–49, 2011.

[24] S. Sugiura and L. Hanzo, "MIMO-aided near-capacity turbo transceivers: taxonomy and performance versus complexity," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 2, pp. 421–442, 2012.

[25] M. El-Hajjar and L. Hanzo, "EXIT charts for system design and analysis," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 127–153, 2014.

[26] H. Chen, R. G. Maunder, and L. Hanzo, "A survey and tutorial on low-complexity turbo coding techniques and a holistic hybrid ARQ design example," Jan. 2013. [Online]. Available: http://eprints.soton.ac.uk/347846/1/Hanzo79.pdf

[27] L. Li, R. G. Maunder, B. M. Al-Hashimi, M. Zwolinski, and L. Hanzo, "Energy-conscious turbo decoder design: A joint signal processing and transmit energy reduction approach," *IEEE Transactions on Vehicular Technology*, vol. 62, pp. 3627–3638, 2013. [Online]. Available: http://eprints.soton.ac.uk/344400/

[28] F. Boccardi, R. Heath, A. Lozano, T. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74–80, Feb. 2014.

[29] P. Elias, "Coding for noisy channels," *IRE Int. Convention Record*, vol. 3, no. 4, pp. 37–46, 1955.

[30] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge: Cambridge University Press, 2005.

[31] V. S. Annapureddy and V. V. Veeravalli, "Gaussian interference networks: sum capacity in the low-interference regime and new outer bounds on the capacity region," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3032–3050, July 2009.

[32] L. Hanzo, S. X. Ng, T. Keller, and W. T. Webb, *Quadrature Amplitude Modulation: From Basics to Adaptive Trellis-Coded, Turbo-Equalised and Space-Time Coded OFDM, CDMA and MC-CDMA Systems*, 3rd ed. Wiley-IEEE Press, 2004.

[33] J. Hagenauer, "The turbo principle: Tutorial introduction and state of the art," in *Proc. International Symposium on Turbo Codes*, 1997, pp. 1–11.

[34] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proceedings of IEEE International Conference of Communication*, vol. 2, Seattle, WA, USA, 1995, pp. 1009–1013.

[35] L. Hanzo, J. P. Woodard, and P. Robertson, "Turbo decoding and detection for wireless applications," in *Proceedings of the IEEE*, vol. 95, no. 6, 2007, pp. 1178–1200.

[36] J. Vogt, K. Koors, A. Finger, and G. Fettweis, "Comparison of different turbo decoder realizations for IMT-2000," in *Seamless Interconnection for Universal Services. Global Telecommunications Conference. GLOBECOM'99. (Cat. No.99CH37042)*, vol. 5. IEEE, 1999, pp. 2704–2708.

[37] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding," *European Transactions on Telecommunications*, vol. 8, no. 2, pp. 119–125, 1997.

[38] C. Schurgers, F. Catthoor, and M. Engels, "Memory optimization of MAP turbo decoder algorithms," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 2, pp. 305–312, 2001.

[39] C.-H. Lin, C.-Y. Chen, E.-J. Chang, and A.-Y. Wu, "A 0.16nJ/bit/iteration 3.38mm2 turbo decoder chip for WiMAX/LTE standards," in *13th International Symposium on Integrated Circuits (ISIC)*, 2011, pp. 168–171.

[40] Y. Sun and J. R. Cavallaro, "Efficient hardware implementation of a highly-parallel 3GPP LTE/LTE-advance turbo decoder," *Integration, the VLSI Journal*, vol. 44, no. 4, pp. 305–315, September 2011.

[41] D. Yoge and N. Chandrachoodan, "GPU implementation of a programmable turbo decoder for software defined radio applications," *25th International Conference on VLSI Design (VLSID)*, pp. 149–154, 2012.

[42] C. Roth, S. Belfanti, C. Benkeser, and Q. Huang, "Efficient parallel turbo-decoding for high-throughput wireless systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 6, pp. 1824–1835, June 2014.

[43] M. A. Bickerstaff, D. Garrett, T. Prokop, C. Thomas, B. Widdup, G. Zhou, L. M. Davis, G. Woodward, C. Nicol, and R.-H. Yan, "A unified turbo/viterbi channel decoder for 3GPP mobile wireless in 0.18-μm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1555–1564, 2002.

[44] M. Bickerstaff, L. Davis, C. Thomas, D. Garrett, and C. Nicol, "A 24Mb/s radix-4 logMAP turbo decoder for 3GPP-HSDPA mobile wireless," in *2003 IEEE International Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC.*, vol. 1. Ieee, 2003, pp. 150–484.

[45] F.-M. Li, C.-H. Lin, and A.-Y. Wu, "Unified convolutional/turbo decoder design using tile-based timing analysis of VA/MAP kernel," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 10, pp. 1063–8210, 2008.

[46] C. Benkeser, A. Burg, T. Cupaiuolo, and Q. Huang, "Design and optimization of an HSDPA turbo decoder ASIC," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 98–106, 2009.

[47] M. May, T. Ilnseher, N. Wehn, and W. Raab, "A 150Mbit/s 3GPP LTE turbo code decoder," pp. 1420–1425, Mar. 2010.

[48] Y. Sun and J. R. Cavallaro, "Efficient hardware implementation of a highly-parallel 3GPP LTE, LTE-advance turbo decoder," *Integration, the VLSI Journal*, vol. 44, no. 1, pp. 1–11, 2010.

[49] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, "Design and implementation of a parallel turbo-decoder ASIC for 3GPP-LTE," *IEEE Jouranal of Solid-State Circuits*, vol. 46, pp. 8–17, 2011.

[50] 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 Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1–9, 2011. [Online]. Available: http://eprints.soton.ac.uk/271820/

[51] C. Studer, S. Fateh, C. Benkeser, and Q. Huang, "Implementation trade-offs of soft-input soft-output MAP decoders for convolutional codes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 11, pp. 2774–2783, 2012.

[52] T. Ilnseher and F. Kienle, "A 2.15 GBit/s turbo code decoder for LTE advanced base station applications," Aug 2012, pp. 21–25.

[53] S. Belfanti, C. Roth, M. Gautschi, C. Benkeser, and Qiuting Huang, "A 1Gbps LTE-advanced turbo-decoder ASIC in 65nm CMOS," in *VLSI Circuits (VLSIC), 2013 Symposium on*, 2013, pp. C284–C285.

[54] J. Chen and J. Hu, "High throughput stochastic turbo decoder based on low bits computation," *Signal Processing Letters, IEEE*, vol. 20, no. 11, pp. 1098–1101, 2013.

[55] Q. T. Dong, M. Arzel, C. Jego, and W. J. Gross, "Stochastic decoding of turbo codes," *IEEE Transactions on Signal Processing*, vol. 58, no. 12, pp. 6421–6425, Dec. 2010.

[56] D. Vogrig, A. Gerosa, A. Neviani, A. Amat, G. Montorsi, and S. Benedetto, "A 0.35-m CMOS analog turbo decoder for the 40-bit rate 1/3 UMTS channel code," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 3, pp. 753–762, Mar. 2005.

[57] V. Gaudet and P. Gulak, "A 13.3-Mb/s 0.35-μm CMOS analog turbo decoder IC with a configurable interleaver," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 11, pp. 2010–2015, Nov. 2003.

[58] M. Arzel, C. Lahuec, F. Seguin, D. Gnaedig, and M. Jezequel, "Semi-iterative analog turbo decoding," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 6, pp. 1305–1316, June 2007.

[59] S. Devadas and S. Malik, "A survey of optimization techniques targeting low power VLSI circuits," in *Proceedings of the 32nd ACM/IEEE conference on Design automation conference - DAC '95*. New York, New York, USA: ACM Press, January 1995, pp. 242–247.

[60] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," in *Proceedings 29th ACM/IEEE Design Automation Conference*. IEEE Comput. Soc. Press, 1992, pp. 253–259.

[61] T. Karnik, S. Borkar, and V. De, "Sub-90nm technologies: challenges and opportunities for CAD," in *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design - ICCAD '02*. New York, New York, USA: ACM Press, November 2002, pp. 203–206.

[62] J.-S. Yang, J. Pak, X. Zhao, S. K. Lim, and D. Z. Pan, "Robust clock tree synthesis with timing yield optimization for 3D-ICs," in *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, IEEE Press. IEEE Press, Jan. 2011, pp. 621–626.

[63] D. Kim and T. Kwon, "A modified two-step SOVA-based turbo decoder with a fixed scaling factor," *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, 2000.

[64] J. Vogt and A. Finger, "Improving the max-log-MAP turbo decoder," *Electronics letters*, vol. 36, pp. 1937–1939, 2000.

[65] M. van Dijk, "Correcting systematic mismatches in computed log-likelihood ratios," *European Transactions on Telecommunications*, vol. 14, pp. 227–244, 2003.

[66] Y. Cao, H. Tomiyama, T. Okuma, and H. Yasuura, "Data memory design considering effective bitwidth for low-energy embedded systems," *Proceedings of the 15th international symposium on System Synthesis*, 2002.

[67] C.-C. Lin, Y.-H. Shih, H.-C. Chang, and C.-Y. Lee, "A low power turbo/viterbi decoder for 3gpp2 applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 4, pp. 426–430, 2006.

[68] A. Nimbalker, Y. Blankenship, B. Classon, and T. K. Blankenship, "ARP and QPP interleavers for LTE turbo coding," *2008 IEEE Wireless Communications and Networking Conference*, pp. 1032–1037, March 2008.

[69] S. Crozier, P. Guinand, and A. Hunt, "Estimating the minimum distance of turbo-codes using double and triple impulse methods," *IEEE Communications Letters*, vol. 9, no. 7, pp. 631–633, July 2005.

[70] A. Ardakani, M. Mahdavi, and M. Shabany, "An efficient VLSI architecture of QPP interleaver/deinterleaver for LTE turbo coding," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*. IEEE, May 2013, pp. 797–800.

[71] D. E. Thomas, *Algorithmic and Register-Transfer Level Synthesis: The System Architects Workbench: The System Architect's Workbench*. Springer, 1990.

[72] Taiwan Semiconductor Manufacturing Company, "TSMC 90nm low power high density synchronous single port with redundancy SRAM compiler databook," 2007.

[73] S. Jayaweera, "Virtual MIMO-based cooperative communication for energy-constrained wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 5, pp. 984–989, May 2006.

[74] S. Cui, A. Goldsmith, and A. Bahai, "Energy-efficiency of MIMO and cooperative MIMO techniques in sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 6, pp. 1089–1098, Aug. 2004.

[75] E. Calvanese Strinati and L. Hérault, "Holistic approach for future energy efficient cellular networks," *e & i Elektrotechnik und Informationstechnik*, vol. 127, no. 11, pp. 314–320, Nov. 2010.

[76] H. Chen, R. G. Maunder, and L. Hanzo, "Low-complexity multiple-component turbo-decoding-aided hybrid ARQ," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 4, pp. 1571–1577, May 2011.

[77] R. Demo Souza, M. Pellenz, and T. Rodrigues, "Hybrid ARQ scheme based on recursive convolutional codes and turbo decoding," *IEEE Transactions on Communications*, vol. 57, no. 2, pp. 315–318, February 2009.

[78] A. Wong, M. Dawkins, G. Devita, N. Kasparidis, A. Katsiamis, O. King, F. Lauria, J. Schiff, and A. Burdett, "A 1V 5mA multimode IEEE 802.15.6/bluetooth low-energy WBAN transceiver for biotelemetry applications," in *2012 IEEE International Solid-State Circuits Conference*. IEEE, Feb. 2012, pp. 300–302.

[79] D. Divsalar, H. Jin, and R. J. McEliece, "Coding theorems for turbo-like codes," in *Proceeding of 36th Allerton Conf. on Communication, Control and Computing*, Allerton, Illinois, 1998, pp. 201–210.

[80] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *SIAM Journal of Applied Math*, vol. 8, pp. 300–304, 1960.

[81] K. S. Andrews, D. Divsalar, S. Dolinar, J. Hamkins, C. R. Jones, and F. Pollara, "The development of turbo and LDPC codes for deep-space applications," *Proceedings of the IEEE*, vol. 95, no. 11, pp. 2142–2156, Nov. 2007.

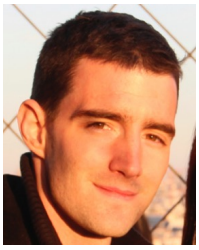
[82] K. Narayanan and G. Stuber, "A novel ARQ technique using the turbo coding principle," *IEEE Communications Letters*, vol. 1, pp. 49–51, 1997.

[83] B. Sklar, "Rayleigh fading channels in mobile digital communication systems .I. Characterization," *IEEE Communications Magazine*, vol. 35, no. 7, pp. 90–100, July 1997.

[84] L. Li, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "An energy-efficient error correction scheme for IEEE 802.15. 4 wireless sensor networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 3, pp. 233–237, 2010. [Online]. Available: http://eprints.soton.ac.uk/267462/

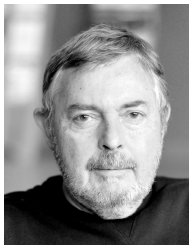
Matthew F. Brejza recieved a first class honors BEng in Electronic Engeering from the University of Southampton, UK, in July 2012, where he is currently working toward the Ph.D. degree with the Communications Research Group, School of Electronics and Computer Science. His research interests include flexible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

Liang Li received his B.S. degree in Mircoelectronics from Peking University, Beijing, China, in 2006 and his M.Sc. degree from University of Southampton, Uk, in 2008. He has joined the Communication Research Group, University of Southampton since 2008 to do study energy-efficient hardware architectures for turbo or LDPC codes and received his Ph.D, degree in 2012. He is currently working in Cirrus Logic Inc. focusing on low-power audio chip design for mobile devices.



Robert G. Maunder has studied with Electronics and Computer Science, University of Southampton, UK, since October 2000. He was awarded a first class honors BEng in Electronic Engineering in July 2003, as well as a PhD in Wireless Communications in December 2007. He became a lecturer in 2007 and an Associated Professor in 2013. Rob's research interests include joint source/channel coding, iterative decoding, irregular coding and modulation techniques. For further information on this research, please refer to http://users.ecs.soton.ac.uk/rm.

Bashir M. Al-Hashimi 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.



Claude Berrou (F'09) is a Professor at Telecom Bretagne (Institut Mines-Telecom), Brest, France. In the early 90's, he introduced the concept of probabilistic feedback into error correcting decoders (in collaboration with Prof. Glavieux) and developed a new family of quasi-optimal error correction codes, that he nicknamed turbo codes. He also pioneered the extension of the turbo principle to joint detection and decoding processing, known today as turbo detection and turbo equalization. His current research interest is computational intelligence in the light of information theory. Prof. Berrou has received several distinctions, amongst which the 1998 IEEE (Information Theory) Golden Jubilee Award for Technological Innovation, the 2003 IEEE Richard W. Hamming medal, the 2003 Grand Prix France Telecom de l'Acadmie des sciences and the 2005 Marconi Prize. He has been elected a member of the French Academy of sciences in 2007.

Lajos Hanzo (http://www-mobile.ecs.soton.ac.uk) FREng, FIEEE, FIET, Fellow of EURASIP, DSc received his degree in electronics in 1976 and his doctorate in 1983. In 2009 he was awarded the honorary doctorate "Doctor Honoris Causa" by the Technical University of Budapest. During his 38-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 about 100 PhD students, co-authored 20 John Wiley/IEEE Press books on mobile radio communications totalling in excess of 10 000 pages, published 1400+ research entries 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 100-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. His research is funded by the European Research Council's Senior Research Fellow Grant. For further information on research in progress and associated publications please refer to http://www-mobile.ecs.soton.ac.uk Lajos has 22 000+ citations.