Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2011, Article ID 184635, 15 pages
doi:10.1155/2011/184635

Research Article

Fixed-Point MAP Decoding of Channel Codes

Massimo Rovini, Giuseppe Gentile, and Luca Fanucci

Department of Information Engineering, University of Pisa, Via G. Caruso 16, 56122 Pisa, Italy

Correspondence should be addressed to Giuseppe Gentile, giuseppe.gentile@esa.int

Received 21 June 2010; Revised 28 November 2010; Accepted 8 February 2011

Academic Editor: Olivier Sentieys

This paper describes the fixed-point model of the maximum a posteriori (MAP) decoding algorithm of turbo and low-density parity-check (LDPC) codes, the most advanced channel codes adopted by modern communication systems for forward error correction (FEC). Fixed-point models of the decoding algorithms are developed in a unified framework based on the use of the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm. This approach aims at bridging the gap toward the design of a universal, multistandard decoder of channel codes, capable of supporting the two classes of codes and having reduced requirements in terms of silicon area and power consumption and so suitable to mobile applications. The developed models allow the identification of key parameters such as dynamic range and number of bits, whose impact on the error correction performance of the algorithm is of pivotal importance for the definition of the architectural tradeoffs between complexity and performance. This is done by taking the turbo and LDPC codes of two recent communication standards such as WiMAX and 3GPP-LTE as a reference benchmark for a mobile scenario and by analyzing their performance over additive white Gaussian noise (AWGN) channel for different values of the fixed-point parameters.

1. Introduction

Modern communication systems rely upon block channel codes to improve the reliability of the communication link, as a key facet to enhance the quality of service (QoS) to the final user. To achieve this target, a block of source data is encoded into a codeword that adds some redundancy to the transmission of source (information) bits in the form of parity bits. Then, at the receiver side, the parity bits are exploited by the decoder to perform forward error correction (FEC), meaning the partial or complete correction of the errors added by the transmission over a noisy channel.

Two main categories of channel codes have gained the momentum of the scientific and industrial community, low-density parity-check codes [1], and serial or parallel concatenation of convolutional codes, SCCC, and PCCC [2]. Although LDPC codes were first designed by Gallager in the early 1960s, they were soon abandoned because of the inadequacy of the microelectronics technology, incapable of facing the complexity of the decoding algorithm. It was only in the early 1990s that channel codes became popular, when Berrou et al., sustained by an already mature very large scale of integration (VLSI) technology, revealed the turbo decoding of PCCCs [3], soon extended to SCCCs [2, 4]. This started a new age in digital communications and paved the road to many research activities and achievements in the field of information theory. Continuous advances in the VLSI technology have reinforced the success of turbo and LDPC codes and deep submicron CMOS processes (down to 65–45 nm and beyond) allow the implementation of decoders sustaining very high clock frequency, and so reaching very high processing rate or throughput. This issue is particularly felt given the iterative nature of the decoding algorithm, running for a certain number of consecutive iterations.

At the present, several communication standards specify the use of either turbo or LDPC codes or both, for FEC. These cover different applications and services, including access networks such as wireless local access networks (W-LANs) (IEEE802.11n) [5] and wireless metropolitan access networks (W-MANs) (IEEE 802.16e, also known as WiMAX)

[6], high-speed cellular networks starting from UMTS-2000 [7] and 3GPP [8] to the long-term evolution 3GPP-LTE [9], satellite broadcasting for fixed [10, 11] and hand-held terminals [12], and up to very high rate data links on optic fiber [13]. Overall, a considerable variety of code parameters is specified, such as different code rates and block lengths, along with very different requirements in terms of decoding throughput (from 2 Mb/s in UMTS to 100 Mb/s in 3GPP-LTE and even 10 Gb/s in 10GBASE-T). Hence, the design of a channel code decoder in general and in particular of a multistandard decoder is a challenging task in view of the flexibility demanded to its architecture and because of the practical restrictions on chip area and power consumption.

The definition of a fixed-point VLSI architecture of the decoding algorithm, that is, flexible, uses the smallest number of bits, and still yields very good error correction performances, is an effective means to attain an effective implementation of the related decoder, featuring both low complexity and low power consumption. On the other hand, floating- or fixed-point (16- or 32-bit) digital signal processing (DSP) units are inadequate to this aim and beside the known limitations in power consumption, they only meet the throughput requirements of the slowest standards and only with high degrees of parallelism (and so with increased power consumption).

For this reason, this paper develops an accurate fixed-point model of a decoder for turbo and LDPC codes, treated within a unified framework exploiting the inherent analogies between the two classes of codes and the related decoding algorithms.

Several works have already dealt with the same objective of fixed-point models of MAP decoding [14–16], and useful indications are provided for practical implementations of turbo [17–19] and LDPC decoders [20–22]. However, while very elegant bounds to the maximum growth of the internal signals of a turbo decoder are provided in [14, 15], the model described in this paper allows the full exploration of the complexity/performance tradeoffs. Furthermore, this model is extended to the decoding of LDPC codes, and so provides useful hints toward the design of a multistandard, multicode decoding platform.

This paper is organized like this. After this introduction, Section 2 recalls the definition of turbo and LDPC codes, and Section 3 reviews the fundamentals of the MAP decoding algorithm, going through the BCJR decoding of convolutional codes, the turbo decoding principle and the so-called horizontal layered decoding (HLD) of LDPC codes. Then, Section 4 describes the fixed-point models of the two decoding algorithms, and the dynamic range and quantization of the internal operations are discussed in detail. The performance of the fixed-point algorithms are then studied in Section 5, where frame error rate (FER) curves are shown for two turbo codes, the 3GPP-LTE binary code with block size 1504 and rate 1/3 and the WiMAX duo-binary code with size 480 and rate 1/2, and for one LDPC code, the WiMAX code with size 1056 and rate 2/3 (class B). Finally, conclusions are drawn in Section 6.

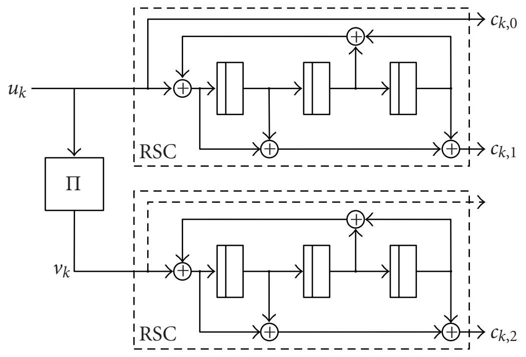


Figure 1: 3GPP-LTE turbo encoder.

2. Channel Codes

2.1. Turbo Codes. Focusing on the class of parallel concatenated convolutional code (PCCC) codes, Figure 1 shows the encoder of the 3GPP-LTE turbo code. This is composed of two stacked recursive systematic convolutional (RSC) encoders, where the upper and lower units are fed by a direct and an interleaved version of the information bits, respectively. Interleaving among the bits of the information word is performed in the block labeled Π in Figure 1. Each RSC encoder is a particular linear feedback shift register (LFSR) whose output bits c_i, $i = 0, 1$, also called parity bits, are a function of the status S of the register, of the forward/backward connections (called *taps*), and of the input bit u entering the encoder.

The performance of the turbo code closely depends on the parameters of the constituent RSCs such as the number of states, denoted as ν, and connection of the feed-back and feed-forward *taps*. The number of states ν is linked to the number of memory elements in the RSC, also referred to as the *constraint* length L ($L = 4$ in the example of Figure 1), through the relationship $\nu = 2^{L-1}$.

The encoding process of the RSC can be effectively represented by resorting to the so-called trellis graph, reported in Figure 2 for the 3GPP-LTE encoder. This is a diagram showing the evolution in time of the LFSR state and describing the transitions (also referred to as *edges*) between pairs of consecutive states: as shown in Figure 2, every edge is labeled with the pair of transmitted information symbols that caused the transition and the parity bits output by the encoder. So the RSC encoding process of a given information word can be followed as a specific path on the trellis.

Aiming at enhanced error correction capabilities, M-ary turbo codes have become widely used in recent communication standards after their introduction in the early 2000s [23]. In this case, each information symbol can assume $M > 2$ values ($M = 2$ corresponds to a binary code) that can be expressed on m bits, so that $M = 2^m$. Standards such as DVB-RCS and WiMAX define duo-binary turbo codes ($m = 2$, $M = 4$), and an example of a duo-binary encoder is shown in Figure 3. Higher values of M would further improve the error-correction performance but are not of practical use due to the excessive complexity of the related decoding algorithm.

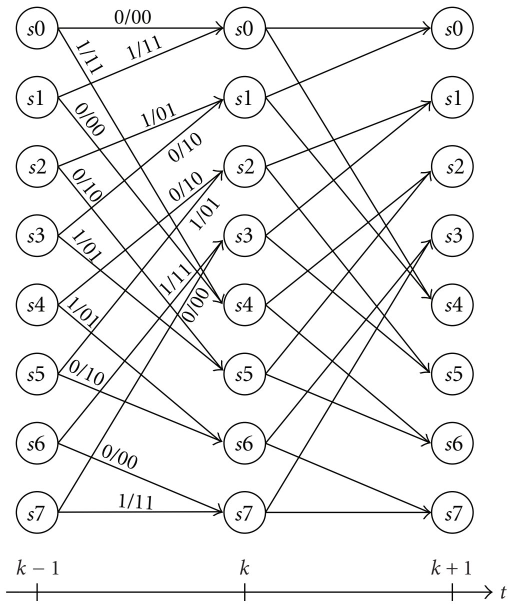
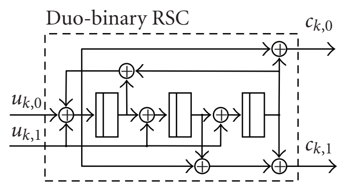
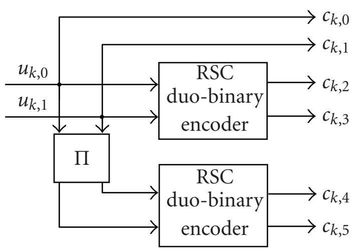


FIGURE 2: Example of an 8-state trellis diagram.



(a) Duo-binary RSC encoder



(b) Duo-binary PCCC encoder

FIGURE 3: The WiMAX turbo encoder.

2.2. LDPC Codes. LDPC codes are linear block codes defined by a sparse matrix **H** known as parity-check matrix, and **x** is a valid codeword if belongs to the null space or *kernel* of **H**, that is, $\mathbf{Hx}^T = 0$. The parity-check matrix has a number of columns N equal to the bits in the transmitted codeword and a number of rows M equal to the number of parity-check constraints, where $P = N - M$ is the number of parity bits added by the LDPC encoder. Each row of the matrix describes a parity-check constraint, with the convention that the element $h_{i,j}$ set to "1" means that the jth bit of the codeword participates into the ith parity-check constraint.

LDPC codes can be also described by means of a bipartite graph known as Tanner graph [24] which is arranged in variable nodes (VNs), represented with circles, and check nodes (CNs), represented with squares. Each VN represents

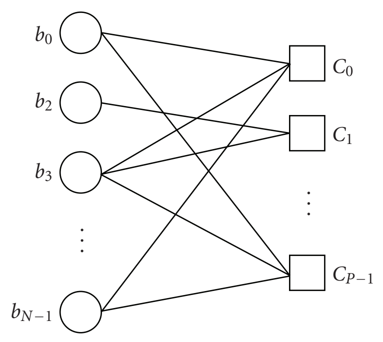


FIGURE 4: Example of a Tanner graph.

a bit of the transmitted codeword and corresponds to a column of **H**, while a CN represents a parity-check constraint, that is, a row of **H**. A connection between variable and check nodes, referred to as *edge*, corresponds to a "1" of the parity-check matrix and graphically links a parity-check constraint to a bit in the codeword. The number of edges connected to a VN (CN) is known as variable node degree, d_v (check node degree, d_c). An example of a Tanner graph is shown in Figure 4.

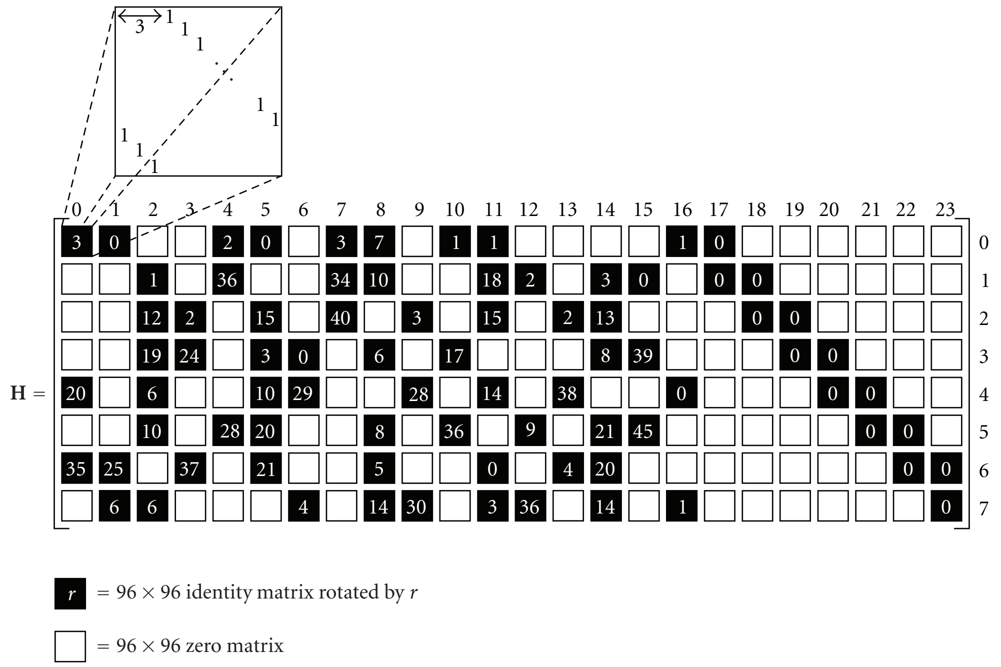
As far as the design of the parity-check matrix is concerned, it heavily affects both the error correction performance and the complexity of the LDPC decoder. Hence, joint code-decoder design techniques are usually applied [25]. Following this route, a particular class of architecture-aware- (AA-LDPC) codes [26] is currently being adopted by all modern communication standards specifying LDPC codes. The underlying idea is the arrangement of 1s in the parity-check matrix according to patterns that ease the parallelization of the decoding operations. Therefore, the parity-check matrix is partitioned in smaller squared matrices that can be either permutations or cyclic shifts of the unit matrix called *circulants* [27]. Figure 5 shows the prototype matrix of the WiMAX LDPC code 2/3a with length 2304: it is partitioned in $Z \times Z$ matrices with $Z = 96$, where a null entry corresponds to the all 0 matrix, while a nonnull entry specifies the rotation (left-shift) applied to the unit matrix.

3. Maximum A Posteriori Decoding of Channels Codes

The BCJR algorithm [28] provides the common framework to the decoding of turbo and LDPC codes as it is applied to the decoding of the two component RSC codes of a turbo code as well as to the parity-check update of an LDPC code.

3.1. The BCJR Decoding Algorithm. Figure 6 summarizes the notation used in the BCJR decoding algorithm of an M-ary convolutional code ($M = 2^m$). In particular,

(i) e is the oriented edge connecting the starting state $S_S(e)$ to the ending state $S_E(e)$, $S_S(e) \overset{e}{\to} S_E(e)$;



$$\mathbf{H} = \begin{array}{c|cccccccccccccccccccccccc|}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 \\
\hline
0 & 3 & 0 & & & 2 & 0 & & 3 & 7 & & 1 & 1 & & & & & 1 & 0 & & & & & & \\
1 & & & 1 & & 36 & & & 34 & 10 & & & 18 & 2 & & 3 & 0 & & 0 & 0 & & & & & \\
2 & & & 12 & 2 & & 15 & & 40 & & 3 & & 15 & & 2 & 13 & & & & 0 & 0 & & & & \\
3 & & & 19 & 24 & & 3 & 0 & & 6 & & 17 & & & & 8 & 39 & & & & 0 & 0 & & & \\
4 & 20 & & 6 & & & 10 & 29 & & & 28 & & 14 & & 38 & & & 0 & & & & 0 & 0 & & \\
5 & & & 10 & & 28 & 20 & & & 8 & & 36 & & 9 & & 21 & 45 & & & & & & 0 & 0 & \\
6 & 35 & 25 & & 37 & & 21 & & & 5 & & & 0 & & 4 & 20 & & & & & & & & 0 & 0 \\
7 & & 6 & 6 & & & & 4 & & 14 & 30 & & 3 & 36 & & 14 & & 1 & & & & & & & 0
\end{array}$$

Figure 5: Prototype matrix of WiMAX 2/3a LDPC code with length 2304 ($Z = 96$). Different block sizes are obtained with Z ranging from 24 to 96 in steps of 4 and rotations derived from the code with length 2304 after simple modulo or scaling operations (refer to [6] for further details).

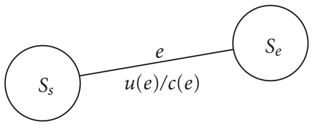


Figure 6: BCJR notation on the trellis.

(ii) $u(e)$ is the information symbol related to edge e, drawn from the alphabet $\mathcal{U} = \{0, 1, \ldots, M-1\}$, with $M = 2^m$;

(iii) $c(e)$ is the coded symbol associated to edge e, and $c_i(e)$ is the ith bit in $c(e)$, with $i = 0, 1, \ldots, n-1$.

So, against m information bits encoded in the symbol u, $n \geq m$ coded bits are generated, and the ratio $r = m/n$ is referred to as the rate of the code.

Being a particular form of MAP decoding, the BCJR algorithm aims at the maximization of the *a posteriori* probability of the transmitted bit, given the observation of the received codeword in noise. For an efficient implementation, the algorithm is formulated in terms of reliability messages having the form of log-likelihood ratios (LLRs). Given the M-ary random variable x with values in $\mathcal{X} = \{x_0, x_1, \ldots, x_{M-1}\}$, its LLR is defined as

$$\mathrm{LLR}(x = x_i) \doteq \log \frac{P(x = x_i)}{P(x = x_0)}, \tag{1}$$

where $P(\cdot)$ denotes the probability mass function and $i = 1, 2, \ldots, M-1$. In (1), x_0 is used as the reference symbol for normalization, so that only $M-1$ LLRs are associated to an M-ary random variable.

Borrowing a notation from [4], the BCJR algorithm involves the following quantities:

(i) $\lambda_{k,i}^{\mathrm{ch}}$ is the channel a priori information for the coded bit c_i at time k, with $i = 0, 1, \ldots, n-1$ and $k = 0, 1, \ldots, N-1$; being the input of the algorithm, $\lambda_{k,i}^{\mathrm{ch}}$ is also referred to as input LLR;

(ii) $\gamma_k(c(e))$ (or simply $\gamma_k(e)$) is the cumulative metric associated to the coded symbol $c(e)$ on the edge e at time k; $\gamma_k(c(e))$ is also referred to as *branch* metric;

(iii) $\lambda_k^I(u(e))$ (or simply $\lambda_k^I(e)$) is the a priori information associated to the information symbol $u(e)$ on the edge e at time k;

(iv) $\lambda_k^O(u(e))$ (or simply $\lambda_k^O(e)$) is the a posteriori *extrinsic* information associated to the to information symbol $u(e)$ on the edge e at time k;

(v) $\Lambda_k^{\mathrm{APP}}(u(e))$ (or simply $\Lambda_k^{\mathrm{APP}}(e)$) is the a posteriri probability (APP) associated to the information symbol $u(e)$ on the edge e at time k.

The BCJR algorithm first computes the *branch*-metric $\gamma_k(e)$ as

$$\gamma_k(e) = \sum_{i=0}^{n-1} c_i(e) \cdot \lambda_{k,i}^{\mathrm{ch}} \tag{2}$$

with $k = 0, 1, \ldots, N-1$ the trellis index.

Along with the a priori *extrinsic* information $\lambda_k^I(e)$, the *branch*-metric $\gamma_k(e)$ drives the forward and backward recursions α' and β', computed in the log-domain according to

$$\begin{aligned}\alpha'_{k+1}(S_i) &= \max^*_{e:S_E(e)=S_i}\left\{\alpha'_k(S_S(e))+\gamma_k(e)+\lambda_k^I(e)\right\},\\ \beta'_k(S_i) &= \max^*_{e:S_S(e)=S_i}\left\{\beta'_{k+1}(S_E(e))+\gamma_k(e)+\lambda_k^I(e)\right\},\end{aligned} \tag{3}$$

where the max*(a,b) operator is defined as

$$\begin{aligned}\max{}^*(a,b) &\doteq \log\left(e^a+e^b\right)\\ &= \max(a,b)+\log\left(1+e^{-|a-b|}\right).\end{aligned} \tag{4}$$

However, the max* can be approximated with a simpler max operation for a lower complexity implementation; in this case the decoding algorithm is referred to as max-log-MAP [4].

The forward (backward) recursion α (β) in (3) is evaluated over the set of the edges e with ending (starting) state S_i at time $k+1$ (k) and is initialized with $\alpha'_0 = \alpha_{\text{init}}$ ($\beta'_N = \beta_{\text{init}}$), at $k = 0$ ($k = N$). Indeed, the initialization value depends on the selected termination strategy, and it is $[1/\nu,\ldots,1/\nu]$ for codes not terminated and is $[1,0,\ldots,0]$ for 0-tail terminated codes, while for tail biting or circular codes it is used to propagate the value reached by either the forward or backward recursion at the previous iteration.

The state-metric recursions in (3) are in the form of logarithm of probabilities, and to increase the numerical robustness of the algorithm [14, 15], they are normalized with respect to the value taken by a reference state, typically the "zero" state S_0, as in a regular LLR. This corresponds to the following subtractions:

$$\begin{aligned}\alpha_k(S_i) &= \alpha'_k(S_i)-\alpha'_k(S_0),\\ \beta_k(S_i) &= \beta'_k(S_i)-\beta'_k(S_0)\end{aligned} \tag{5}$$

with $i = 0,1,\ldots,\nu-1$.

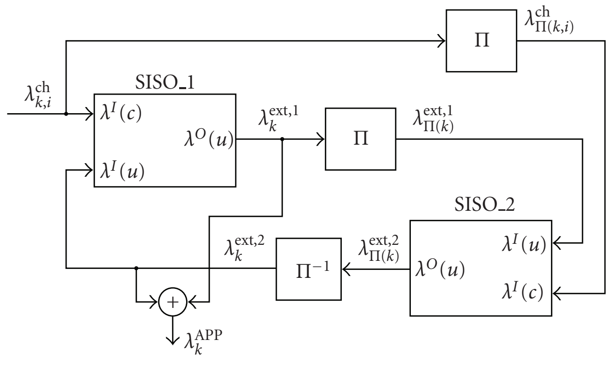
Once the state-metric recursions are available, the a posteriori estimation of the information symbol u is derived as

$$\begin{aligned}\lambda_k^O(u_i) = &\max^*_{e:u(e)=u_i}\left\{\alpha_k(S_S(e))+\gamma_k(e)+\beta_{k+1}(S_E(e))\right\}\\ &-\max^*_{e:u(e)=u_0}\left\{\alpha_k(S_S(e))+\gamma_k(e)+\beta_{k+1}(S_E(e))\right\}.\end{aligned} \tag{6}$$

Being not directly connected to the input a priori message $\lambda_k^I(e)$, the APP output $\lambda_k^O(u_i)$ is said to be *extrinsic*.

3.2. The Turbo Decoding Principle. The turbo decoding algorithm is achieved as the direct application of the BCJR algorithm to both of its constituent RSC codes, according to the block diagram of Figure 7. The two BCJR decoders are the soft-in soft-out (SISO) units labeled SISO_1 and SISO_2, and the algorithm evolves as the iterative exchange of *extrinsic* messages that are the a posteriori outputs of the SISO engines.

The algorithm is fed with the channel a priori estimations $\lambda_{k,i}^{\text{ch}}$, in the form of LLR and computed according to (1) for binary variables ($M = 2$). The output of SISO_1, called $\lambda^{\text{ext},1}$ in Figure 7, is scrambled according to the interleaving law Π before being passed to SISO_2 as a priori information. The latter also receives a scrambled version of the channel a priori estimations $\lambda_{k,i}^{\text{ch}}$ and outputs the a posteriori reliability messages $\lambda^{\text{ext},2}$. After inverse scrambling, these go back to SISO_1 as refined a priori estimations about the transmitted symbols.



$\lambda_{k,i}^{\text{ch}}$
SISO_1
$\lambda^I(c)$
$\lambda^O(u)$
$\lambda^I(u)$
$\lambda_k^{\text{ext},1}$
Π
$\lambda_{\Pi(k)}^{\text{ext},1}$
Π
$\lambda_{\Pi(k,i)}^{\text{ch}}$
SISO_2
$\lambda^I(u)$
$\lambda^O(u)$
$\lambda^I(c)$
$\lambda_k^{\text{ext},2}$
Π^{-1}
$\lambda_{\Pi(k)}^{\text{ext},2}$
+
λ_k^{APP}

Figure 7: Decoding of PCCC codes: the turbo principle.

As shown in Figure 7, the output of the turbo decoder, that is, the a posteriori estimation of the transmitted symbol, is given by the sum of the two *extrinsic* messages output by the SISO units. In formula,

$$\Lambda_k^{\text{APP}}(u_i) = \lambda_k^{\text{ext},1}(u_i)+\lambda_k^{\text{ext},2}(u_i) \tag{7}$$

with $u_i \in \mathcal{U} = \{u_0,u_1,\ldots,u_{M-1}\}$ and $k = 0,1,\ldots,K-1$.

3.3. MAP Decoding of LDPC Codes. The MAP decoding algorithm of LDPC codes is commonly referred to as belief propagation (BP) or more generally message passing (MP) algorithm [29]. BP has been proved to be optimal if the graph of the code does not contain cycles, that is, consecutive nodes connected in a closed chain, but it can still be used and considered as a reference for practical codes with cycles. In this case the sequence of the elaborations, referred to as *schedule*, considerably affects the performance both in terms of convergence speed and error correction rate.

The most straightforward schedule is the two-phase or flooding schedule (FS) [30], which proceeds through two consecutive phases, where all parity-check nodes first and all variable nodes then are updated in sequence.

A more powerful schedule is the so-called *shuffled* or *layered* schedule [26, 30–32]. Compared to FS, shuffled schedules almost double the decoding convergence speed, both for codes with cycles and cycle-free [33]; this is achieved by looking at the code as the connection of smaller *super*-codes [26] or *layers* [31], exchanging reliability messages. Specifically, *a posteriori* messages are made available to the next layers immediately after computation and not at next iteration like in FS. Layers can either be sets of consecutive CNs or VNs, and, accordingly, CN-*centric* (or horizontal) or VN-*centric* (or vertical) algorithms have been defined in [30, 32].

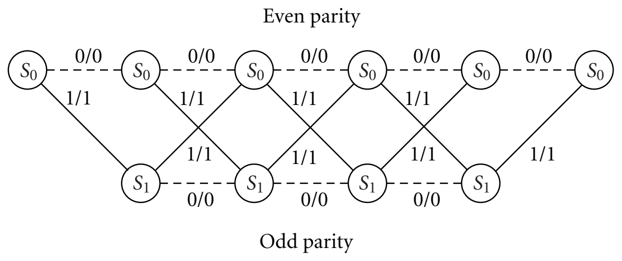


Figure 8: Two-state trellis representation of a parity-check constraint with $d_c = 5$.

3.3.1. Horizontal Layered Decoding. The HLD algorithm updates the parity-check constraints sequentially around the parity-check matrix. The key feature of HLD is the continuous update, during decoding, of a cumulative metric y_n associated to every VN in the code, $n = 0, 1, \ldots, N-1$, and called soft output (SO).

The update of CN m, with $m = 0, 1, \ldots, M-1$, is based on the availability of variable-to-check (vtoc) messages $\mu_{n,m}$ directed from VN n to CN m and computed as

$$\mu_{m,n}^{(q)} = y_n^{(q)} - \epsilon_{n,m}^{(q)}, \tag{8}$$

where $\epsilon_{n,m}^{(q)}$ is the check-to-variable (ctov) propagated by CN m toward VN n at previous iteration, $n \in \mathcal{N}_m$ denotes the set of VNs connected to CN m, and $q = 0, 1, \ldots, N_{\text{it,max}} - 1$ is the iteration index.

Refined ctov messages $\epsilon_{n,m}^{(q+1)}$ are produced as a result of the check-node update, and, based on these, the set of SOs involved in CN m, that is, y_n with $n \in \mathcal{N}_m$, is updated according to

$$y_n^{(q+1)} = \mu_{m,n}^{(q)} + \epsilon_{m,n}^{(q+1)} = y_n^{(q)} - \epsilon_{n,m}^{(q)} + \epsilon_{m,n}^{(q+1)}. \tag{9}$$

Thanks to the mechanism described in (8) and (9), check-node operations always rely on up-to-date SOs, which explains the increased convergence speed of HLD-shuffled schedule.

The HLD algorithm is initialized at iteration $q = 0$ with

$$\begin{aligned} y_n^{(0)} &= \lambda_n^{\text{ch}}, \\ \epsilon_{m,n}^{(0)} &= 0, \end{aligned} \tag{10}$$

where λ_n^{ch} is the LLR of the a priori channel estimation of the received bits in noise, $m = 0, 1, \ldots, M-1$ and $n \in \mathcal{N}_m$.

3.3.2. Check-Node Update. As far as the check-node update is concerned, it is shown in [26] that a parity-check constraint can be viewed as a 2-state convolutional code, where one state is associated to *even* parity (S_0) and the other to *odd* parity (S_1). The block size of the equivalent code is then equal to the CN degree d_c, and an example of its trellis representation is given in Figure 8.

This analogy allows the BCJR algorithm to be also employed for parity-check updates of LDPC codes, and the resulting decoding algorithm is known as turbo decoding message passing (TDMP) [26]. The algorithm is fed with vtoc messages as a priori information and produces ctov messages as a posteriori outputs, with no *branch* metric from the channel. So, in the update of CN m, the state-metric recursions are simplified into

$$\begin{aligned} \alpha_{k+1} &= \max{}^*\left\{\alpha_k, \mu_{m,n}^{(q)}\right\} - \max{}^*\left\{\alpha_k + \mu_{m,n}^{(q)}, 0\right\}, \\ \beta_k &= \max{}^*\left\{\beta_{k+1}, \mu_{m,n}^{(q)}\right\} - \max{}^*\left\{\beta_{k+1} + \mu_{m,n}^{(q)}, 0\right\}, \end{aligned} \tag{11}$$

where $k = 1, 2, \ldots, d_c(m) - 1$ is the recursion step, with $d_c(m)$ being the degree of CN m, and $n = \mathcal{N}_m(k)$ is the index of the VN involved at step k. The recursions in (11) are initialized with $\alpha_0 = 1$ and $\beta_{d_c} = 1$.

Then, the computation of a posteriori extrinsic information in (6) can be reworked in the form

$$\epsilon_{m,n}^{(q+1)} = \max{}^*\{\alpha_k, \beta_{k+1}\} - \max{}^*\{\alpha_k + \beta_{k+1}, 0\} \tag{12}$$

with $k = 0, 1, \ldots, d_c(m) - 1$ and $n = \mathcal{N}_m(k)$.

4. Fixed-Point Models

Given a positional numeric system in base δ, the fixed-point representation X of a real (i.e., floating-point) signal $x \in \Re$ is expressed as

$$X = \sum_{n=0}^{N_I - 1} a_n \delta^n + \sum_{n=1}^{N_F} b_n \delta^{-n}, \tag{13}$$

where N_I (N_F) is the number of integer (fractional) digits a_n (b_n), drawn from the set $\mathcal{D} = \{0, 1, \ldots, \delta - 1\}$. Overall, $N_x = N_I + N_F$ digits are used to represent x.

The multiplication of (13) by the factor δ^{N_F}, also referred to as *scaling* factor, is practical to get rid of the decimal point and is effective for the implementation of fixed-point DSP or VLSI systems. Focusing on binary systems with $\delta = 2$, X becomes an integer number in the form

$$X = \sum_{n=0}^{N_x - 1} x_n 2^n, \tag{14}$$

where x_n, $n = 0, 1, \ldots, N_x - 1$ are the binary digits of the integer representation of x, with $x_n = b_{N_F - n}$ for $n = 0, 1, \ldots, N_F - 1$ and $x_n = a_{n - N_F}$ for $n = N_F, N_F + 1, \ldots, N_F + N_I - 1$.

4.1. Conversion Law. Given a signal x defined in the domain of reals, that is, $x \in \Re$, its fixed-point counterpart X on N_x bits is now derived. As only a limited number of bits is available, the domain of x needs to be constrained to an interval of $\Re$, say $[-A, A]$. So a preventive saturation of the signal in the range $[-A, A]$ must be performed, and the value of A will be referred to as dynamic range in the remainder of this paper.

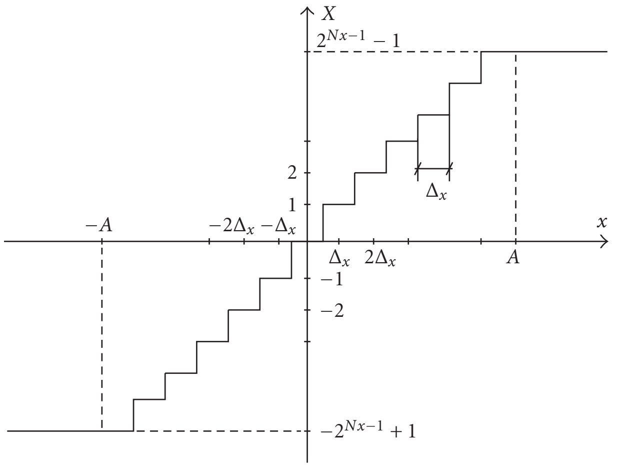


FIGURE 9: Staircase conversion function from floating- to fixed-point signals.

The operation of fixed-point conversion can be done according to the following transformation:

$$X = \min\left(2^{N_x-1}-1, \left\lfloor \frac{x}{\Delta_x}+0.5\right\rfloor\right), \quad x \geq 0,$$
$$X = \max\left(-2^{N_x-1}+1, \left\lfloor \frac{x}{\Delta_x}-0.5\right\rfloor\right), \quad x < 0, \tag{15}$$

where $\Delta_x = 2A/(2^N - 1)$ is the quantization step, that is, the maximum difference between two different floating-point values that are mapped onto the same fixed-point symbol X. The value of Δ_x is a measure of the resolution of the representation, that is, is the weight of the least significant bit (LSB) x_0 of X.

Note that (15) not only performs the quantization of the input signal, but it also limits its domain to the interval $[-A, A]$, as shown in Figure 9, as values greater (less) than A ($-A$) are saturated to the biggest positive (smallest negative) level $2^{N_x-1}-1$ ($-2^{N_x-1}-1$).

In (15), only $2^{N_x}-1$ fixed-point levels are used (the codomain of the transformation function is symmetrical with respect to the level 0); this choice prevents the algorithm from drifting toward negative levels, which otherwise would be systematically advantaged as also noted in [15].

So the pair (A, N_x) fully defines the quantization of the floating-point signal x, providing the dynamic range and the weight of the LSB Δ_x used for its representation.

This approach is similar to that described in [15] for the quantization of input LLRs and is more flexible than that generally adopted in the literature [14, 17–22], where the fixed-point format is specified by the pair $(N_I : N_F)$, disregarding the dynamic range of the underlying real signal. In other words, the dynamic range of the real signal is often put in the form $A = 2^{N_I-1}$ and is limited to a power of two. On the contrary, our approach comes through this restriction, and it is applied to every internal fixed-point elaboration.

4.2. Fixed-Point Turbo Decoding. The complete scheme of the fixed-point SISO decoder is shown in Figure 10. The algorithms described in Section 3.1 are reformulated in fixed-point domain to involve operations among integers. Following a cascade approach, all the involved operations are converted into their fixed-point counterpart one after the other.

4.2.1. Channel A Priori Information. Channel LLRs are quantized according to (15) using the threshold $A_{\lambda^{\text{ch}}}$ and $N_{\lambda^{\text{ch}}}$ bits.

4.2.2. Branch Metric. The computation of $\gamma_k(e)$ as in (2) involves the summation of n channel a priori reliabilities $\lambda_{k,i}^{\text{ch}}$, $i = 0, 1, \ldots, n-1$. So, in the worst case, where they all sum coherently, it holds $\gamma_k(e) = n \cdot A_{\lambda^{\text{ch}}}$, and the fixed-point counterpart Γ of γ needs to be represented with

$$A_\gamma = n \cdot A_{\lambda^{\text{ch}}},$$
$$N_\gamma = N_{\lambda^{\text{ch}}} + \lceil \log_2 n \rceil. \tag{16}$$

4.2.3. max Operator.* The operation $z = \max^*(x, y)$ implies the computation of the max of two signals x and y, and the addition of a correction term in the range $]0, \log 2]$; hence, the dynamic range of the z is upper bounded by

$$A_z = \max\left(A_x, A_y\right) + \log 2. \tag{17}$$

In order to let the comparison be possible, the fixed-point counterparts of x and y, X and Y, respectively, must have the same resolution, that is, $\Delta_x = \Delta_y = \Delta$; holding this, the number of bits to represent z can be derived from definition of Δ as

$$2^{N_z} = \frac{2A_z}{\Delta} + 1 = \frac{2A}{\Delta} + \frac{2\log 2}{\Delta} + 1$$
$$= \left(2^N - 1\right) + \frac{2\log 2}{\Delta} + 1 = 2^N\left(1 + \frac{\log 2}{A}\right), \tag{18}$$

where $A \doteq \max(A_x, A_y)$ and $N \doteq \max(N_x, N_y)$. Then, assuming that $A > \log 2$, as it is generally the case, expression (18) gives

$$N_z = \left\lceil \log_2\left(2^N\left(1 + \frac{\log 2}{A}\right)\right)\right\rceil = N + 1. \tag{19}$$

However, (18) and (19) strictly hold when $x = A_x = y = A_y = A$, when the contribution of the correction term is maximum; beside this very unlikely case, the additional bit required in (19) is not really exploited, and the use of $A_z = \max(A_x, A_y)$ is generally enough, so that the result can be saturated on $N_z = N$ bits. This approximation yields a very little loss of information and so has negligible impact on the algorithm performance. Therefore, the fixed-point max* operation becomes

$$Z = \max^*(X, Y) = \min\{\max(X, Y) + \text{LUT}(D), L\}, \tag{20}$$

where $L = 2^{N-1} - 1$ is the saturation threshold. In (20), the correction term is quantized using the same resolution Δ and is stored in a look-up table (LUT) addressed with $D = |X - Y|$.

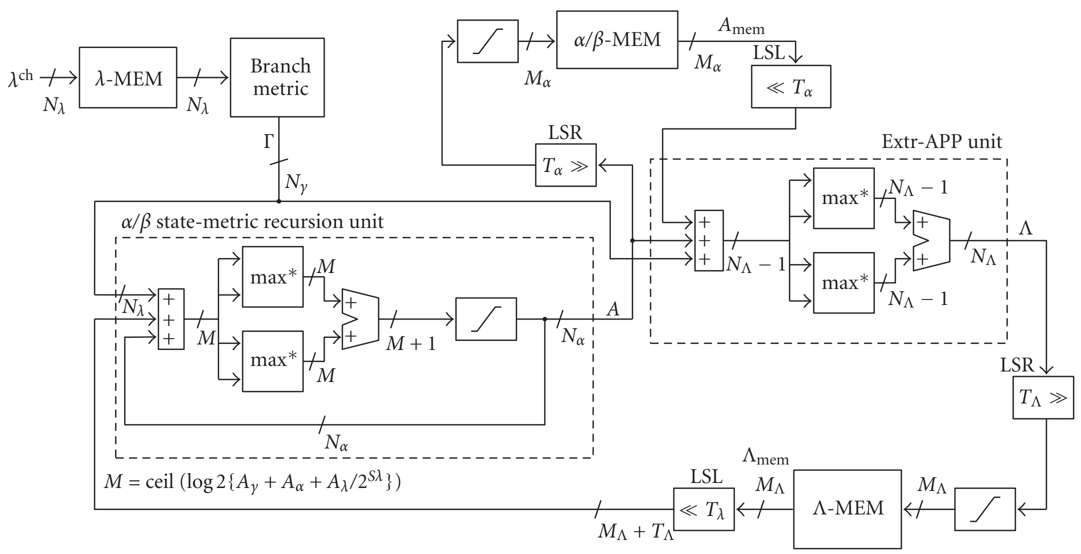


FIGURE 10: Fixed-point model of the SISO engine in a turbo decoder.

4.2.4. A Posteriori Extrinsic Information. Since a posteriori extrinsic reliabilities and forward/backward recursions are mutually dependent through the iterative turbo principle, their fixed-point representation can be studied under the assumption that the state-metric recursions are represented on $N_\alpha = N_\beta = N_\gamma + k$ bits, with k any integer. From (6), the dynamic range of λ^O is upper bounded by

$$\begin{aligned} A_{\lambda^O} &= \left(A_\alpha + A_\gamma + A_\beta\right) - \left(-A_\alpha - A_\gamma - A_\beta\right) \\ &= 2\cdot\left(2A_\alpha + A_\gamma\right) = 2A_\gamma\cdot\left(1+2^{k+1}\right), \end{aligned} \tag{21}$$

where it has been exploited that $A_\alpha = 2^k A_\gamma$ and $A_\alpha = A_\beta$.

The full precision representation of λ^O can be obtained using $N_{\lambda^O} = \lceil \log_2(2A_{\lambda^O}/\Delta_{\lambda^O}+1)\rceil$ bits, which gives

$$\begin{aligned} N_{\lambda^O} &= 1 + N_\gamma + \left\lceil \log_2\left(1+2^{k+1}\right)\right\rceil \\ &= 1 + N_\gamma + \lceil \max_2^*(0, k+1)\rceil, \end{aligned} \tag{22}$$

where the function $\max_2^*$ is the two-base max* operator defined as $\max_2^*(a,b) \doteq \max(a,b) + \log_2(1+2^{-|a-b|})$. The following cases can be distinguished:

(a) $k \geq 0$: it is easy to prove that $\lceil \max_2^*(0,k+1)\rceil = k+2$, so that $N_{\lambda^O} = N_\gamma + k + 3 = N_\alpha + 3$;

(b) $k < 0$: now it is $\lceil \max_2^*(0,k+1)\rceil = 1$ and $N_{\lambda^O} = N_\gamma + 2 = N_\alpha + 2 - k$.

In both cases N_{λ^O} is a known function of N_α and N_γ, that is, of N_α and k.

4.2.5. State-Metric Recursions. Because of its recursive computation, the magnitude of forward/backward recursions would steadily grow proceeding along the trellis unless it is controlled by means of saturation. Under the same hypothesis of Section 4.2.4, that is, $N_\alpha = N_\gamma + k$, the growth of state metrics after one update step of (3) and (5) is upper bounded by

$$2\left(A_\alpha + A_\gamma + A_{\lambda^I}\right) = 2A_\alpha\left(2 + 2^{-k} + \frac{A_{\lambda^I}}{A_\alpha}\right), \tag{23}$$

where the a priori information λ^I is indeed the a posteriori output of the companion decoder, so that $A_{\lambda^I} = A_{\lambda^O}$. Substituting (21) in (23), the latter becomes

$$2A_\alpha\left(1 + 2^{-k} + 2k^{-1}\cdot 2^{1-k}\right) = 2\left(5 + 2^{-k} + 2^{1-k}\right)A_\alpha, \tag{24}$$

meaning that the dynamic range of α would increase by the factor $2(5+3\cdot 2^{-k})$ after every recursion step. This would result in the addition of $1 + \lceil \log_2(5+3\cdot 2^{-k})\rceil$ bits. Again, two cases can be distinguished:

(a) $k \geq 0$: the term $(5+3\cdot 2^{-k})$ falls in the range from 5 to 8, resulting in the addition of 4 bits at each recursion step;

(b) $k < 0$: the term $\lceil \log_2(5+3\cdot 2^{-k})\rceil$ evaluates to $2-k$, and overall $3-k$ more bits are added at every step.

So the saturation of 4 or $3-k$ bits, respectively, prevents the uncontrolled growth of state metrics, hence represented with (A_α, N_α). In [14, 15], bounds are provided for the dynamic range of state-metric recursions, used to dimension the internal precision of the SISO engine. On the contrary, in the described approach the resolution of state-metric recursion is a free input of the model and is controlled by means of saturation. As also noted in [14], the precision of state-metric recursions is inherently linked to that of *branch* metrics and *extrinsic* messages, and if they are different, scaling of the signals participating in the update must be considered. This is achieved by means of shifting, used to re-align the precision used on different signals; in terms of quantization step Δ, the involved signals stay then in a ratio of a power-of-two.

4.3. Fixed-Point LDPC Decoding. The fixed-point model of a decoder of LDPC codes is derived following a similar cascaded approach, and its scheme is reported in Figure 11. The model allows the analysis of the independent effect on performance of the representation of three different signals, input a priori LLR, ctov messages, and state-metric recursions within check-node updates.

4.3.1. Computation of Variable to Check Messages. The computation of variable to check messages μ according to (8) involves SOs and ctov messages.

Let input LLRs be quantized with (A_λ, N_λ) and ctov messages with (A_ϵ, N_ϵ), and let Δ_λ and Δ_ϵ denote the respective resolutions. Then, let the ratio $\rho = \Delta_\lambda/\Delta_\epsilon$ be constrained to be a power of two. This assumption reduces the number of independent variables in the model (only three out of the four variables A_λ, N_λ, A_ϵ, and N_ϵ are actually independent), but it is also the only viable solution for a practical implementation of the algorithm.

If $\rho > 1$, that is, when a finer resolution is used on ctov messages, channel a priori LLRs need to be left shifted by $\sigma_\lambda = \log_2(\rho)$ bits to be brought on the same resolution of ctov messages, which in turn are not shifted ($\sigma_\epsilon = 0$); in the other case, no shift is needed on input LLRs ($\sigma_\epsilon = 0$), while ctov messages should be left shifted by $\sigma_\epsilon = -\log_2(\rho)$ bits.

As channel a priori LLRs are used to initialize SOs, the two signals have the same resolution, that is, $\Delta_y = \Delta_\lambda$. Therefore, the same relationship between the resolution of ctov messages and input LLRs holds between ctov messages and SOs. In view of this, SOs are initialized with a scaled version of input LLRs (see the input right-shift by σ_λ in Figure 11(b)) so that data stored or retrieved from the λ/SO memory use the same resolution of ctov messages. This allows the direct subtraction $Y - E$ to compute fixed-point vtoc messages.

Once available, vtoc messages are saturated in two different ways, on N_μ bits on the input of the CN update unit and on N_ν bits for the SO update in (9).

4.3.2. Update of Soft Outputs. The sum in (9) is performed between updated ctov messages E and vtoc messages M' saturated on N_ν, and its output is saturated to N_y bits. As the SO is always equal to the sum of d_v ctov messages entering a given VN, the following relationship holds:

$$N_y = N_\epsilon + \left\lceil \log_2(d_{v,\max}) \right\rceil, \tag{25}$$

where $d_{v,\max}$ is the maximum VN degree in the code. However, lower-complexity solutions can be investigated, where SOs are saturated on fewer bits than in (25).

4.3.3. State-Metric Recursions. Expression (11) combines vtoc messages with recursion metrics, and, similarly to the computation of vtoc messages, different resolutions of the two signals can be considered. Again, the ratio $\rho = \Delta_\epsilon/\Delta_\alpha$ is constrained to be a power of two. As before, ρ is used to align the fixed-point representation M and A of μ and α, respectively, so that M is shifted by $\sigma_\mu = \log_2(\rho)$ if $\rho > 1$ and by $\sigma_\mu = 0$ otherwise; dually, A is shifted by $\sigma_\alpha = -\log_2(\rho)$ if $\rho < 1$ and by $\sigma_\alpha = 0$ otherwise. So the fixed-point sum $\alpha + \mu$ in (11) becomes

$$A \cdot 2^{\sigma_\alpha} + M \cdot 2^{\sigma_\mu} \tag{26}$$

as also shown in Figure 11(a).

The remainder of the algorithm can be quantized in a very similar way to that followed for turbo decoders, with some simplifications. As also shown in Figure 11(a), if we define $B \doteq \max\{N_\alpha + \sigma_\alpha, N_\mu + \sigma_\mu\}$, the new value of A is represented on $B + 1$ bits, and, after right shift by σ_α bits, it is saturated to the desired number of bits N_α.

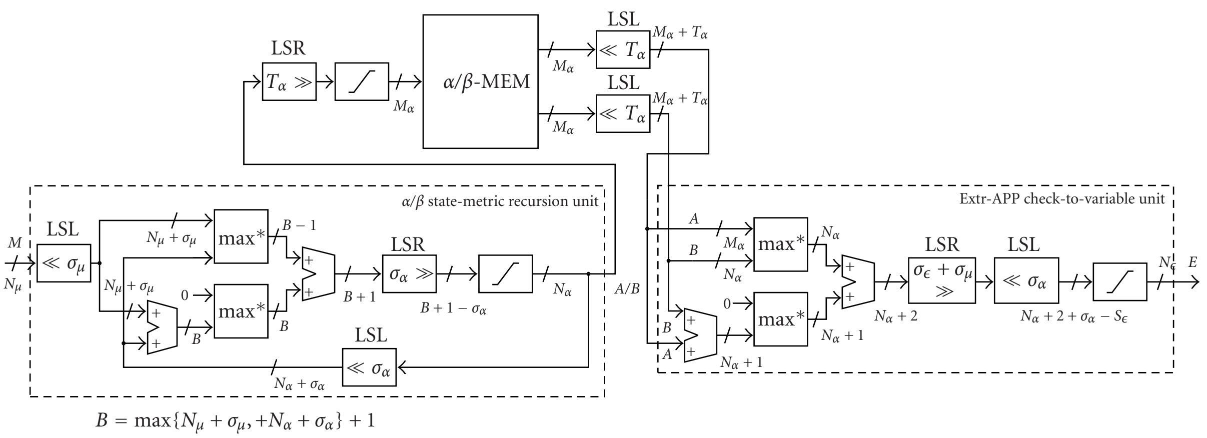
4.3.4. APP Check to Variable Messages. With reference to Figure 11(a), check to variable messages are computed with the recursion metrics taken from memory, where they are represented on M_α bits. So the full-precision representation of (12) can be represented on M_α+2 bits. Then, countershifts are performed (left shift by σ_α and right-shift by σ_ϵ) in order to go back to the resolution of ctov messages, and the final saturation restores the representation on N_ϵ bits.

4.4. Memory Size Reduction. Practical implementations of turbo and LDPC codes decoders are based on the extensive use of memory as a means to exchange *extrinsic* messages (turbo decoders), to accumulate the output estimation (LDPC decoders), and to store intermediate results (state-metric recursions in turbo and LDPC decoders, ctov messages in LDPC decoders). It follows that the overall decoder complexity is heavily dominated by memory, and techniques such as truncation of the least significant bits (LSBs) or saturation of the most significant bits (MSBs) are very effective to limit the size of data stored in memory. However, the use of saturation is preferable, as it reduces not only the size of memory but also that of the data paths accessing the memory unit. On the contrary, data truncated before storage in memory need to be left shifted after retrieval from memory to restore the original resolution (or weight of the LSB Δ) and data paths do not benefit of any reduction in size.

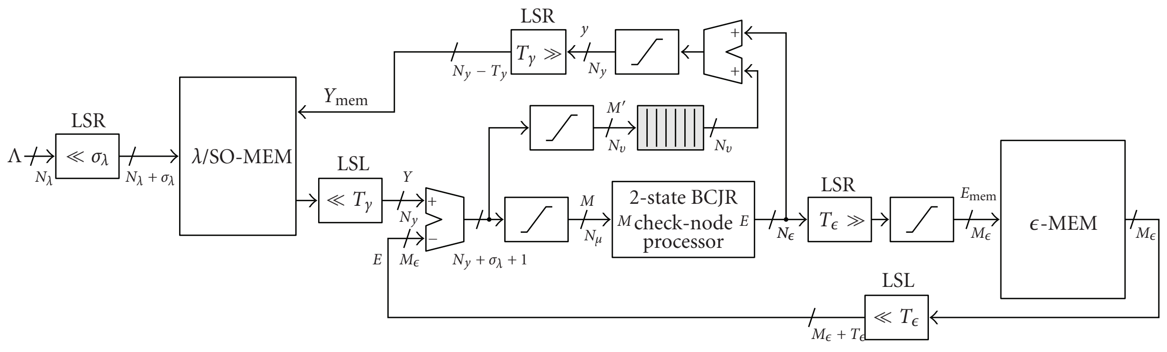
With reference to signal x, the notation T_x and S_x will denote in the remainder of this paper the number of LSBs truncated or MSBs saturated before storage in memory, respectively.

Regarding the fixed-point turbo decoder, truncation and saturation are performed on the state-metric recursions stored in the α/β-MEM memory (T_α and S_α bits, resp.) and on the a posteriori extrinsic information stored in the Λ-MEM memory (T_Λ and S_Λ bits, resp.), as shown in Figure 10.

In the LDPC decoder, truncation is operated on ctov messages (T_ϵ bits), on SOs (T_y bits), and on state-metric recursions (T_α bits); as shown in Figure 11, these signals are countershifted (left shift) just after retrieval from memory. Then, saturations are performed on ctov messages (saturated on M_ϵ bits) and α/β recursions (saturated on M_α bits), while SOs do not need any further saturation after their computation.



(a) 2-state BCJR decoder: fixed-point model



(b) Layered decoding of LDPC codes: fixed-point model

FIGURE 11: The fixed-point model of LDPC codes decoding.

TABLE 1: Reference codes for the simulation scenario.

Standard	Code	Size (m)	Length (K)	Rate (R)	Iterations N_{it}
3GPP-LTE	Turbo	1	1504	1/3	10
WiMAX	Turbo	2	480	1/2	10
WiMAX	LDPC	1	1056	2/3b	15

5. Simulation Results

The error correction performance and implementation loss (IL) of the fixed-point models developed in Section 4 have been assessed by means of simulation. A full communication system complete of encoder, modulator, transmission over AWGN channel, demodulator, and decoder has been described in C++, and the two parametric fixed-point models have been implemented as user-configurable C++ classes and methods.

Three different codes have been considered as a benchmark for the developed models, two turbo codes (a 3GPP-LTE binary code and a WiMAX duo-binary code) and one LDPC code (a WiMAX code), and their parameters are summarized in Table 1. Their fixed-point performance has been measured in the form of FER curves versus the signal to noise ratio (SNR) E_b/N_0.

5.1. Turbo Codes Performance. The first critical design issue is the identification of an optimal value for the input dynamic range $A_{\lambda^{\text{ch}}}$. Figure 12 shows the FER performance for different values of $A_{\lambda^{\text{ch}}}$. As a design constraint for a low-complexity implementation, the input LLRs λ^{ch} were coded on $N_{\lambda^{\text{ch}}} = 5$ bits while the forward/backward metrics were represented on a large number of bits ($N_\alpha = 16$) so that the IL is only due to the quantization of the inputs λ^{ch}.

Focusing on the 3GPP-LTE code (left-most bundle of curves in Figure 12), the smaller the value of A_λ^{ch}, the smaller the IL; the case $A_\lambda^{\text{ch}} = 10$ corresponds to an impairment below 0.1 dB with respect to the floating point reference model, while further increasing the dynamic range yields to very coarse resolution $\Delta_{\lambda^{\text{ch}}}$, which results in considerable losses, especially at low E_b/N_0.

Conversely, the WiMAX code (right-most curves of Figure 12) seems to be less sensitive to variations of $A_{\lambda^{\text{ch}}}$, the maximum impairment being about 0.15 dB for $A_{\lambda^{\text{ch}}} \geq 12$. This can be explained with the increased robustness to channel noise offered by duo-binary codes, paid at the cost of a bigger computational effort in the decoding algorithm.

Although Figure 12 seems to allow the use of $A_{\lambda^{\text{ch}}} = 5$, this value corresponds to a very rough quantization of the channel LLRs, where several floating point samples are saturated to the levels $\pm A_{\lambda^{\text{ch}}}$. Then, the coarser quantization of the remainder of the algorithm can yield to additional

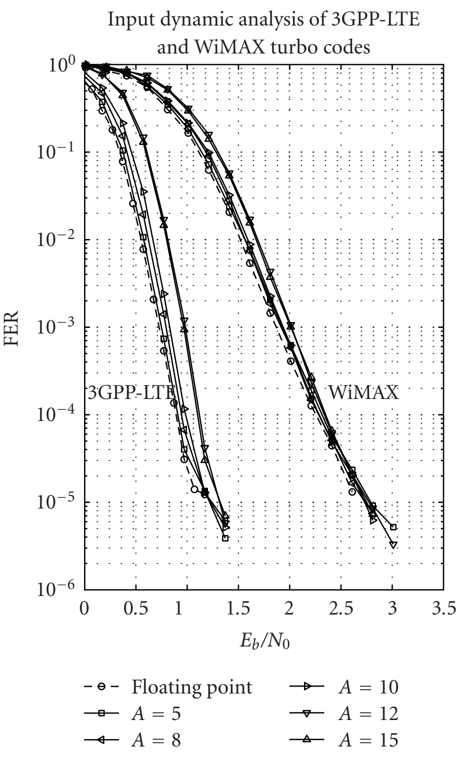


FIGURE 12: Dynamic range analysis for $N_{\lambda^{\text{ch}}} = 5$ bits.

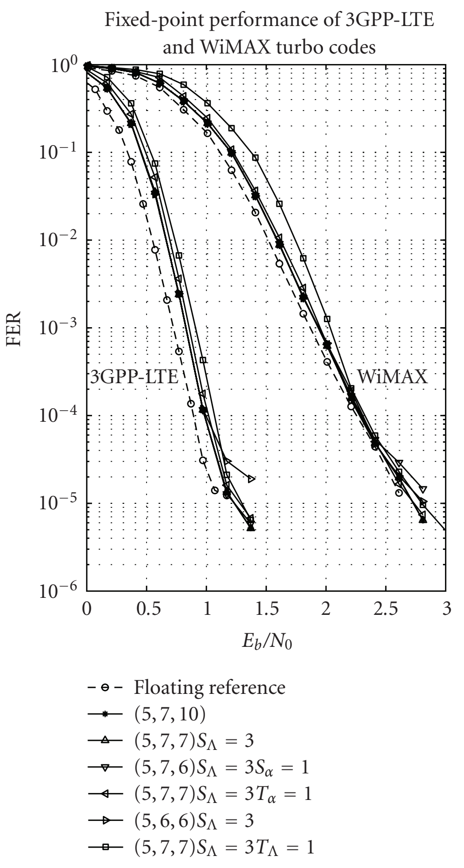


FIGURE 13: Analysis of the performance of the fixed-point turbo-decoding algorithm.

impairments due to the quantization noise, which can result in flattening the FER curve (error floor). In view of this, the value $A_{\lambda^{\text{ch}}} = 10$ is cautiously selected for the next analysis.

Note that this kind of analysis was not feasible with the quantization scheme generally adopted in the literature, or at least with a very coarser step, being $A_{\lambda^{\text{ch}}}$ constrained to powers of two.

Figure 13 shows the FER performance for different values of the fixed-point parameters defined in Section 4.2 and described with the triplet $(N_{\lambda^{\text{ch}}}, N_\alpha, N_\Lambda)$.

Focusing on the 3GPP-LTE code, it turns out that no impairment is observed by saturating three bits on the extrinsic reliabilities ($S_\Lambda = 3$) when $N_\alpha = 7$, while the saturation of only 1 bit of the recursion metrics ($S_\alpha = 1$, $N_\alpha = 6$) results in an IL of about 0.18 dB at high E_b/N_0. Not surprisingly, the truncation of either recursion metrics ($T_\alpha = 1$) or extrinsic reliabilities ($T_\Lambda = 1$) slightly spoils the FER performance at low E_b/N_0, where the LSBs bear the most of the information, while the loss becomes almost negligible at high E_b/N_0. Finally, aiming at a very low-complexity solution, recursions and extrinsic metrics can be represented on $N_\alpha = N_\Lambda = 6$ with an IL of 0.3 dB at high E_b/N_0.

As far as the WiMAX code is concerned, the robustness of duo-binary codes is once again confirmed by a negligible IL for all the simulated configurations. Indeed, only a slight deviation at high E_b/N_0 is shown when 1 bit of the recursion metrics is saturated ($S_\alpha = 1$).

Table 2 summarizes the fixed-point parameters of two optimal configurations, one (config. A) for better performances and the other (config. B) for lower complexity. The latter has been further investigated to analyze the joint effect of the input dynamic range and coarse quantization of the algorithm on the overall error correction performance. Figure 14 shows that the FER curves flatten for $A_{\lambda^{\text{ch}}} = 5$ and $A_{\lambda^{\text{ch}}} = 8$. This is a common issue to both binary and duo-binary codes and is due to the combination of the distortion introduced by the very strong saturation on channel a priori inputs with the quantization noise generated in the remainder of the algorithm. The value $A_{\lambda^{\text{ch}}} = 10$ yields the minimum convergence loss, but also shows clear signs of a floor at very high E_b/N_0. On the other hand, the use of $A_{\lambda^{\text{ch}}} = 12$ and 15 prevents any floor issue, but it is slightly penalizing in terms of convergence abscissa, the IL being about 0.25 dB for LTE and 0.1 dB for WiMAX.

TABLE 2: Optimal fixed-point parameters of the turbo decoder.

Signal	Config. A	Config. B
A priori channel LLR	$(10, 5)$	$(10, 5)$
Branch metric (RCS $r = 1/2$)	$(20, 6)$	$(20, 6)$
State-metric recursions	$(40, 7)$	$(20, 6)$
Bits saturated on α/β (S_α)	2	2
A posteriori *extrinsic* messages	$(40, 7)$	$(20, 6)$
Bits saturated on extr. msg. (S_Λ)	3	3

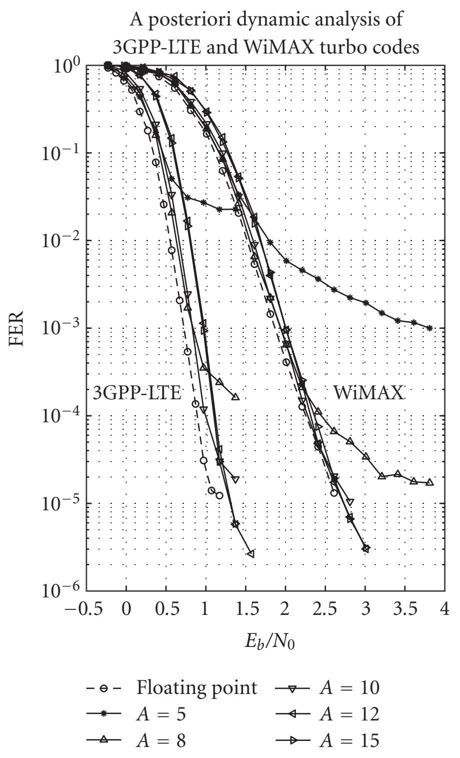


Figure 14: End-to-end fixed-point performance for different values of $A_{\lambda^{\text{ch}}}$.

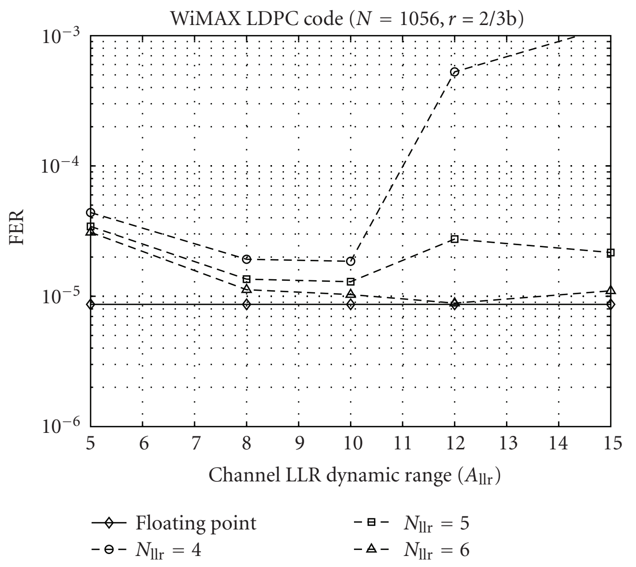


Figure 15: WiMAX LDPC code: effects of the quantization on input LLRs.

5.2. LDPC Codes Performance. The WiMAX LDPC code with rate 2/3 (class B) and $N = 1056$ features a maximum VN degree $d_{v,\max} = 6$ and a maximum CN degree $d_{c,\max} = 11$.

The effect of the quantization of the input LLRs is analyzed in Figure 15, where the FER is plotted against the dynamic range $A_{\lambda^{\text{ch}}}$ for three different values of $N_{\lambda^{\text{ch}}}$, 4, 5, and 6 bits. The three curves are simulated at $E_b/N_0 = 3.25$ dB, along with the floating point reference (solid line). The value $A_{\lambda^{\text{ch}}} = 10$ represents the best solution with 4 and 5 bits, while the use of $A_{\lambda^{\text{ch}}} = 12$ is preferable for $N_{\lambda^{\text{ch}}} = 6$ bits. However, aiming at a low-complexity implementation, the scheme $(10, 5)$ is retained for the next analysis.

An analysis similar to that of Figure 15 is repeated in Figure 16 for the quantization of the state-metric recursions within the CN processor; in this case, full FER curves are plotted as a function of E_b/N_0, and, to get rid of the losses due to the quantization of ctov and SO messages, a very fine quantization is used for both signals, based on many bits and very large dynamic ranges. In particular, the scheme $(160, 12)$ is used for ctov messages, while 3 more bits ($d_{v,\max} = 6$) were allowed for SOs and $N_y = 15$. Similarly, $N_\mu = 15$ was also used for vtoc messages, and, overall, the remainder of the algorithm was run in quasi-floating-point domain.

The curves in Figure 16 can be parted in three groups with different resolutions of the state-metric recursions: the first group contains the curves with $(A_\alpha, N_\alpha) = (10, 6)$, $(20, 7)$, and $(40, 8)$ (referred to as *lsb*-0.5 and shown in dashed lines), the second group is for the curves $(10, 5)$ and $(20, 6)$ (referred to as *lsb*-1, solid lines), and the third group contains the curves $(20, 5)$, $(40, 6)$, and $(80, 7)$ (referred to as *lsb*-2, dotted lines). It is shown that for *lsb*-0.5, at least 7 or even 8 bits are required to get rid of any flattening issue; for *lsb*-1, $N_\alpha = 6$ already provides good performances, while *lsb*-2 needs at least 7 bits to get close to the floating point reference. As a result, the scheme $(20, 6)$ can be retained for a low-complexity solution. These results agree with those reported in [17, 18] about fixed-point turbo decoding.

The quantization of ctov, SO, and vtoc messages is studied in Figure 17, where messages are on 5 bits (dashed lines) or 6 bits (dotted lines). Regarding the representation of SO, since $d_{v,\max} = 6$, 3 more bits are required in principle ($N_y = N_\epsilon + 3$), but smaller values of N_y are also simulated. Similarly, starting from the case $N_\nu = N_y$, lower-complexity solutions with $N_\nu < N_y$ are also investigated.

When ctov messages are represented on 5 bits and SOs on 8, the use of $A_\epsilon = 20$ results in a considerable IL, greater than 0.3 dB, due to the very coarse quantization; on the other hand, the IL is greatly reduced with $A_\epsilon = 10$, but the curves flatten at high SNR, due to the limit imposed to the dynamic range of ctov messages. Further reducing the number of bits of SO or vtoc only worsens this situation.

Figure 17 shows that better performance is achieved with $N_\epsilon = 6$ and $A_\epsilon = 20$, in line with the observations in [20, 21]. In this case, the curve with SO and vtoc on 8 bits stays very close to the floating point reference curve, the IL being smaller than 0.05 dB, while the saturation of only 1 bit either on vtoc (curve $(20, 6)[8, 7]$) or on SO (curve $(20, 6)[7\text{-}7]$) is enough to spoil performance.

As a result of the above analysis, the value of the fixed-point parameters yielding the best error correction performance is summarized in Table 3. Starting from this reference configuration, Figure 18 shows the effects of the reductions of α/β metrics, SO, and ctov before storage in memory. The curves are labeled according to the value of the six parameters $[T_\alpha, S_\alpha]$, $[T_\epsilon, S_\epsilon]$, and $[T_y, S_y]$. It is shown that, while 1 bit

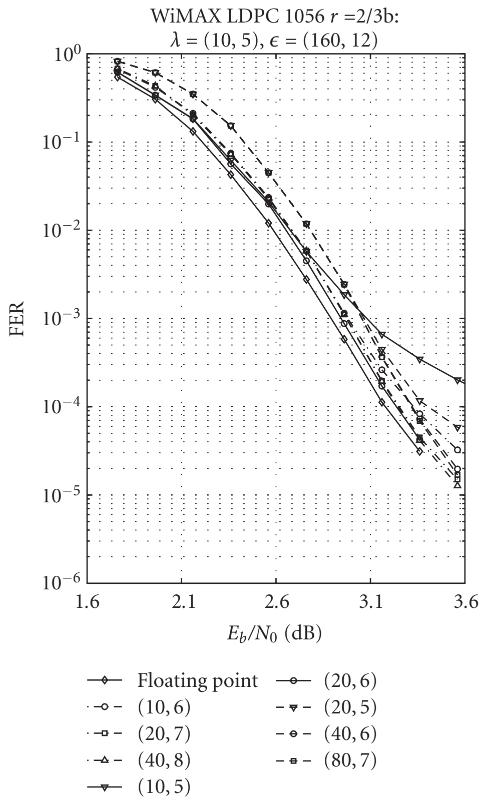


Figure 16: WiMAX LDPC code: effects of the quantization on the state-metric recursions within the CN processor.

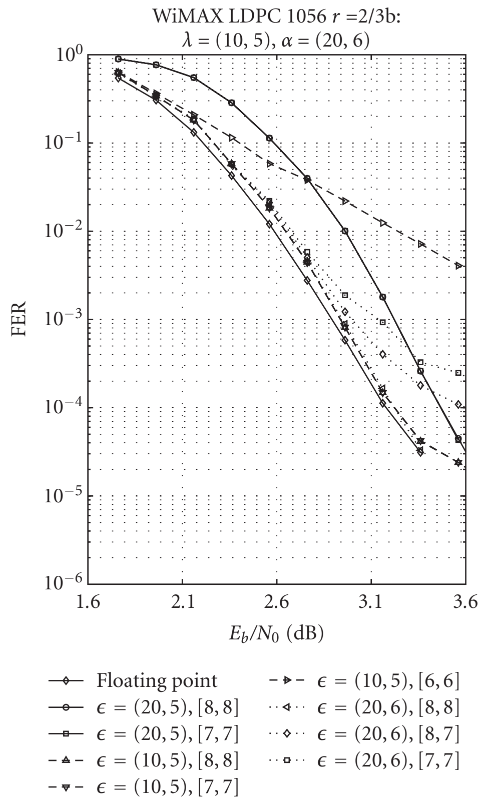


Figure 17: WiMAX LDPC code: effects of the quantization of check to variable, soft-output, and variable to check messages.

Table 3: Optimal fixed-point parameters of the LDPC decoder.

Signal	Dynamic range (A)	No. of bits (N)
A priori channel LLR	10	5
Vtoc for CN update	20	6
Vtoc for SO update	80	8
State-metric recursions	20	6
Ctov *extrinsic* messages	20	6
SO (so) metrics	80	8

can be truncated on state-metric recursions with negligible losses ($T_\alpha = 1$), the IL is more relevant when 1 bit is saturated ($S_\alpha = 1$, i.e., $M_\alpha = 5$). A similar result holds for ctov messages, where the case $T_\epsilon = 1$ is tolerated with minimum losses, while $S_\epsilon = 1$ is not. Regarding SOs, truncation of only one bit ($T_y = 1$) utterly corrupts the decoding performance.

6. Summary and Conclusion

This paper has described the fixed-point model of a decoder for turbo and LDPC codes, derived in a unified framework based on the use of the BCJR decoding algorithm.

Comparing the results of Tables 2 and 3, the following conclusions hold:

(i) a priori channel reliability can be represented with (10, 5) with negligible losses for both codes;

(ii) state-metric recursions need the representation (20, 6) or (40, 7) for turbo codes, while (20, 6) is enough for LDPC codes. This can be explained with the smaller number of edges (2) in the trellis of an LDPC code compared with a turbo code (2 in binary, 4 in duo-binary codes) and with the absence of *branch* metrics in LDPC decoding;

(iii) a posteriori *extrinsic* messages of SISO decoders are on 6 or 7 bits, while soft outputs of LPDC codes need 8 bits. Considering that the turbo decoder APP output is the sum of two extrinsic messages (see (7)), the results are in agreement;

(iv) check to variable messages of LDPC decoding need the quantization (20, 6), and no counterpart exists in turbo decoders.

These results are in line with those contained in [15, 16, 18, 19, 34] for turbo decoders or in [20–22] for decoders of LDPC codes. Compared with [14], state-metric recursions need significantly fewer bits than in the bound given, which evaluates to 9 bits for the considered codes. Also, the single-SISO architecture in [14] needs 7 bits for *extrinsic* messages, in agreement with our results.

The fixed-point quantization law described in this paper, based on the dynamic range (A) of the underlying (floating) signal and number of bits (or, equivalently, the resolution Δ), has allowed the analysis of their independent effects on performance. As opposed to the approach generally

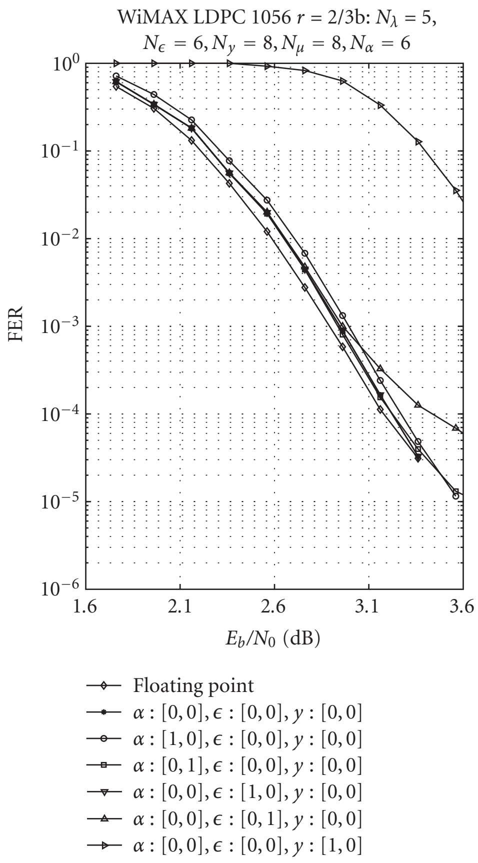


Figure 18: WiMAX LDPC code: effects of saturation and truncation before memorization.

adopted in the literature (see [14, 16, 18, 19, 34]), the dynamic range of the floating signal and the number of bits of its representation have been left independent, and the dynamic ranges not constrained to a power of two. This solution, also adopted in [15], has been extensively exploited in the proposed models, not only for the quantization of input signals but also for every internal fixed-point operation.

Also, the model described in this paper has overcome the limitation common to similar works in the field and related to the use of the same resolution Δ for every fixed-point elaboration, such computation of a priori channel messages, state-metric recursions, and ctov messages in an LDPC decoder. Furthermore, the model has allowed the exploration of memory reduction techniques based on truncation and saturation, and simulations have shown that 1 bit can be truncated on state-metric recursions before memorization in both turbo and LDPC decoders.

A final remark relates to the optimal choice of the fixed-point parameters, which does not have absolute validity, rather it depends on the operating point of the decoder and so on the desired error-correction rate; systems or applications operating above, next to, or below the error floor of the FER curve need indeed different values of the fixed-point parameters. To this extent, our results extend down to the beginning of the error-floor, above FER 10^{-6}.

References

[1] R. Gallager, *Low-density parity-check codes*, Ph.D. dissertation, Massachusetts Institutes of Technology, 1960.

[2] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design and iterative decoding," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 909–926, 1998.

[3] 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, pp. 1064–1070, May 1993.

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

[5] "Local and metropolitan area networks–specific requirements—part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 5: enhancements for higher throughput," IEEE 802.11n^{TM}-2009, October 2009.

[6] "IEEE standard for local and metropolitan area networks—part 16: Air interface for broadband wireless access systems," IEEE Computer Society, IEEE Std 802.16^{TM}-2009, May 2009.

[7] "Universal Mobile Telecommunication system," European Telecommunications Standards Institute, ETSI UMTS^{TM}-2000, June 2000.

[8] "3rd Generation Partnership Project—Technical Specification Group Radio Access Network- High Speed Downlink Packet Access (HSDPA)-Overall description," 3rd Gener. Partnership Project, September 2009.

[9] "Evolved Universal Terrestrial Radio Access (E-UTRA)," 3rd Gener. Partnership Project 2, June 2009.

[10] "Satellite digital video broadcasting of second generation (DVB-S2)," ETSI Standard EN302307, February 2005.

[11] "Digital Video Broadcasting (DVB); interaction channel for satellite distribution Systems," European Telecommunications Standards Institute, DVB-RCS^{TM}-2002, November 2002.

[12] "Framing structure, channel coding and modulation for satellite services to handheld devices (SH) below 3 GHz," Digital Video Broadcasting group, DVB-SH^{TM}-2007, July 2007.

[13] "Physical Layer and Management Parameters for 10 Gb/s Operation, Type 10GBASE-T," 802.3 Working Group, September 2006.

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

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

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

[17] T. K. Blankenship and B. Classon, "Fixed-point performance of low-complexity turbo decoding algorithms," in *Proceedings of the 53rd IEEE Vehicular Technology Conference (VTC '01)*, vol. 2, pp. 1483–1487, May 2001.

[18] M. A. Castellon, I. J. Fair, and D. G. Elliott, "Fixed-point turbo decoder implementation suitable for embedded applications," in *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, pp. 1065–1068, May 2005.

[19] A. Morales-Cortés, R. Parra-Michel, L. F. Gonzalez-Perez, and C. T. Gabriela, "Finite precision analysis of the 3GPP standard turbo decoder for fixed-point implementation in FPGA devices," in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig '08)*, pp. 43–48, December 2008.

[20] T. Zhang, Z. Wang, and K. Parhi, "On finite precision implementation of low density parity check codes decoder," in *Proceedings of the IEEE International Symposium on Circuits and systems (ISCAS '01)*, vol. 4, pp. 202–205, May 2001.

[21] Z. Zhang, L. Dolecek, M. Wainwright, V. Anantharam, and B. Nikolić, "Quantization effects in low-density parity-check decoders," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pp. 6231–6237, June 2007.

[22] R. Zarubica, R. Hinton, S. G. Wilson, and E. K. Hall, "Efficient quantization schemes for LDPC decoders," in *Proceedings of the IEEE Military Communications Conference (MILCOM '08)*, pp. 1–5, November 2008.

[23] C. Berrou, M. Jezquel, C. Douillard, and S. Keroudan, "The advantages of non-binary turbo codes," in *Proceedings of the IEEE Information Theory Workshop*, pp. 61–62, September 2001.

[24] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[25] E. Boutillon, J. Castura, and F. Kschischang, "Decoder-first code design," in *Proceedings of the International Symposium on Turbo Codes and Related Topics*, pp. 459–462, September 2000.

[26] M. M. Mansour and N. R. Shanbhag, "High-throughput LDPC decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 6, pp. 976–996, 2003.

[27] H. Zhong and T. Zhang, "Block-LDPC: a practical LDPC coding system design approach," *IEEE Transactions on Circuits and Systems I*, vol. 52, no. 4, pp. 766–775, 2005.

[28] 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. 2, pp. 284–287, 1974.

[29] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 638–656, 2001.

[30] F. Guilloud, E. Boutillon, J. Tousch, and J.-L. Danger, "Generic description and synthesis of LDPC decoders," *IEEE Transactions on Communications*, vol. 55, no. 11, pp. 2084–2091, 2006.

[31] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proceedings of the IEEE Workshop on Signal Processing Systems Design and Implementation (SISP '04)*, pp. 107–112, October 2004.

[32] E. Sharon, S. Litsyn, and J. Goldberger, "An efficient message-passing schedule for LDPC decoding," in *Proceedings of the 23rd IEEE Convention of Electrical and Electronics Engineers in Israel*, pp. 223–226, September 2004.

[33] H. Kfir and I. Kanter, "Parallel versus sequential updating for belief propagation decoding," *Physica A*, vol. 330, no. 1-2, pp. 259–270, 2003.

[34] Y. Tong, T. H. Yeap, and J. Y. Chouinard, "VHDL implementation of a turbo decoder with log-MAP-based iterative decoding," *IEEE Transactions on Instrumentation and Measurement*, vol. 53, no. 4, pp. 1268–1278, 2004.