POLITECNICO DI TORINO
Repository ISTITUZIONALE

On Practical Implementation and Generalizations of max* Operator for Turbo and LDPC Decoders

Original
On Practical Implementation and Generalizations of max* Operator for Turbo and LDPC Decoders / Martina, M., Masera, G., Papaharalabos, S., Mathiopoulos, P.T., Gioulekas, F.. - In: IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT. - ISSN 0018-9456. - STAMPA. - 61:4(2012), pp. 888-895. [10.1109/TIM.2011.2173045]

Availability:
This version is available at: 11583/2495970 since:

Publisher:
IEEE

Published
DOI:10.1109/TIM.2011.2173045

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

17 November 2026

On Practical Implementation and Generalizations of max* Operator for Turbo and LDPC Decoders

Maurizio Martina, *Member, IEEE*, Guido Masera, *Senior Member, IEEE*, Stylianos Papaharalabos, P. Takis Mathiopoulos, *Senior Member, IEEE*, and Fotios Gioulekas.

Abstract

In this paper, we deal with practical implementation issues of the $\max^*$ operation in generalized form used for both turbo and Low-Density-Parity-Check (LDPC) codes decoding. In particular, first a unified framework for the so-called generalized $\max^*$ operation is established, which includes most of previously published algorithms already known for turbo decoding. Next, the hardware architectures used for the practical implementation of the generalized $\max^*$ operation, which is derived from this novel framework, are revealed for the first time and further analyzed, in terms of hardware complexity reduction. It is also shown how this generalized $\max^*$ operation can be adopted in LDPC decoding achieving essentially optimal BER performance with small computational complexity against other algorithms in joint turbo-LDPC architectures. This solution is useful in applications where joint decoding architectures are deployed to decode both turbo and LDPC codes. An important example of such application is in software radio receivers of 4G wireless communication systems, such as those proposed in conjunction with the WiMAX standard.

Index Terms

Maximum a posteriori (MAP) and Log-MAP algorithms, Turbo codes, LDPC codes, iterative decoding, VLSI architectures.

I. Introduction

Turbo and LDPC codes [1], [2] have been proposed as powerful error correcting codes able to approach the Shannon limit. Not only these codes have been adopted in several standards for wireless and wired communication

M. Martina and G. Masera are with the VLSI Lab, Dipartimento di Elettronica, Politecnico di Torino, Torino, Italy (e-mail: maurizio.martina@polito.it).

S. Papaharalabos is with the Institute for Space Applications and Remote Sensing (ISARS), National Observatory of Athens, 15236, P. Penteli, Greece. He is also with Athens Information Technology, P.O. Box 68, 19.5 Km Markopoulo Avenue, 19002 Peania, Attiki, Greece.

P. T. Mathiopoulos is with the Institute for Space Applications and Remote Sensing (ISARS), National Observatory of Athens, 15236, P. Penteli, Greece.

Fotios Gioulekas is with the Department of Computer Engineering & Informatics, University of Patras, Patras, Greece.

The work reported in this paper was supported, in part, by EU IST NEWCOM++ (IST-216715) project.

Manuscript received Month xx, 2011.

systems, but also their exceptional error correcting capabilities have captured the interest of scientists working in other field of research, such as magnetic disk reliability [3] and telemetry [4]. However, the hardware implementation of turbo and LDPC decoders is a challenging task, due to the high computational complexity requirements of the decoding algorithms [5]–[7]. In terms of implementation, the log-sum-exp (`lse`) function is a common operator used in Log-MAP turbo decoders and its efficient implementation has been addressed in several works [6], [8]–[11]. Essentially, all previous implementations rely on approximating the following mathematical expression

$$\texttt{lse}(x_1, x_2) = \log(e^{x_1} + e^{x_2}) = \max^* \{x_1, x_2\} \tag{1}$$

where

$$\max^* \{x_1, x_2\} \approx \max \{x_1, x_2\} + f_c(|x_1 - x_2|) \tag{2}$$

and $f_c(|x_1 - x_2|)$ is a correction term. More accurate approximations of $f_c(|\cdot|)$ lead in general to smaller bit error rate (BER) performance degradation against Log-MAP turbo decoding at the expense of small complexity increase, e.g. see [6], [9]–[11]. It has been shown in [8] that, in order for the correction term $f_c(|x_1 - x_2|)$ to achieve nearly floating point BER performance, at least three fractional bits are used to represent $|x_1 - x_2|$. Therefore, in this paper we will show experimental results obtained for data represented with three fractional bits.

Recently in [12] the approximation of the `lse` function as a robust geometric programming problem (optimization problem) has been presented. In particular, the following two conclusions reached in [12] are useful to our current research.

i) The two-terms `lse` function (1) is well approximated by an r-term piece-wise-linear (PWL) function

$$\texttt{lse}(x_1, x_2) \approx \max \{x_1, \ldots, y_i, \ldots, x_2\} \tag{3}$$

or

$$z = \max \{x_1, \ldots, y_i, \ldots, x_2\} \tag{4}$$

where $y_i = a_{r-i-1}x_1 + a_i x_2 + b_i$, $i = 1, \ldots, r-2$ and a_i, b_i are appropriate coefficients [12].

ii) The r-term PWL function obtained as in [12] is the best r-term PWL convex approximation of the bivariate `lse` function. In [13], [14] this approximation, termed as generalized $\max^*$ operator, is exploited in turbo trellis-coded modulation (TCM) and turbo codes. In particular, the generalized $\max^*$ operator shows significant, i.e. near-optimal, BER performance evaluation and hardware complexity savings, being comparable with other known `lse` approximations [6], [9]–[11]. However, to the best of our knowledge, no systematic approach has been previously proposed in the open technical literature trying to unify `lse` approximations in a general framework. This work aims to partially fill this gap and, although no new algorithm approximating the `lse` function is proposed here, to concentrate on the efficient implementation of previously known algorithms.

In particular, we extend [14] into the following novel directions: i) It is shown, for the first time, that several `lse` approximations proposed in the literature can be obtained as special cases of (3); ii) Practical hardware implementation architectures for the generalized $\max^*$ operator are proposed and analyzed. To the best of our knowledge such implementation is, for this first time, presented in the open technical literature; and iii) It is shown

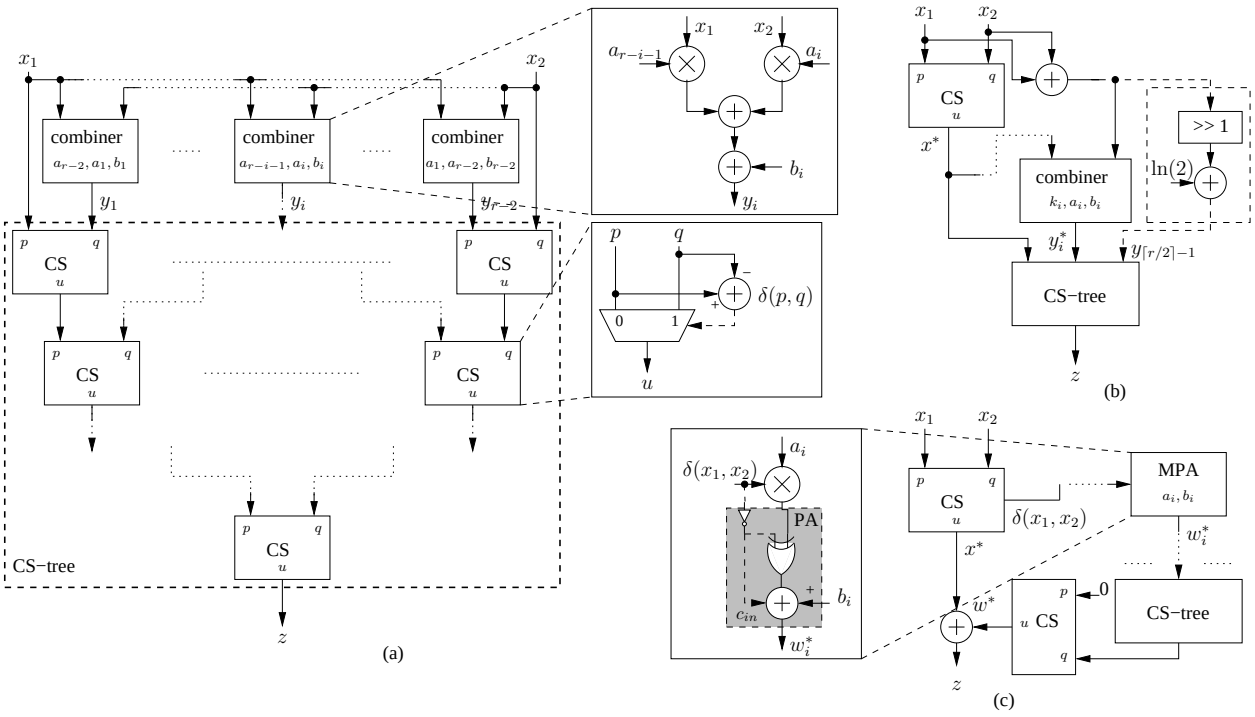


Figure 1. Block scheme of a different implementations of (4): (a) A1 architecture, (b) A2 architecture, (c) A3 architecture

that essentially optimal BER performance can be obtained by employing proper versions of (3) in LDPC decoders with adequate computational complexity. In particular, one of the architectures proposed implementing the best approximation proposed in [14] outperforms the recently published algorithm in [6], [10], in terms of both BER performance and complexity. Moreover, the proposed architectures can be used to reduce the complexity of the dual mode architectures for turbo-LDPC decoding proposed in [15], [16]. These architectures are thus identified as promising candidates for joint turbo-LDPC decoder architectures in future 4G wireless communication systems.

II. Approximated `lse` function as a generalized $\max^*$ operator

In general, direct implementation of (3) leads to high complexity. However, the implementation of (4) can be simplified by exploiting the characteristics of a_i and b_i coefficients being calculated with the algorithms detailed in [12]. As shown in Fig. 1 (a), the direct implementation of (4), referred to as A1 in the following, requires $r-2$ combiners devoted to compute y_i and an appropriate structure to find the maximum among the possible r inputs of the $\max$ function. Each combiner requires two multiplications and two additions as shown in Fig. 1 (a). A tree of two-inputs compare-select (CS) blocks is used to find the maximum among r elements; each CS selects the maximum value between its inputs p and q. Thus, each CS requires a subtracter and a multiplexer. The multiplexer selector is driven by the sign of the subtraction $\delta(p,q) = p - q$ (shown with dashed line in the right part of Fig. 1 (a)).

However, the best PWL approximation of the `lse` function, obtained as in [12], features the following properties:

$$0 < a_1 < a_2 < \cdots < a_{r-2} < 1 \tag{5}$$

$$a_i + a_{r-i-1} = 1 \tag{6}$$

$$b_i = b_{r-i-1} \tag{7}$$

with $i = 1, \ldots, r-2$. Moreover, $a_{\lceil r/2 \rceil - 1} = 0.5$ and $b_{\lceil r/2 \rceil - 1} = \ln(2)$ when r is odd. From (5) we can infer

$$a_{r-i-1} = a_i + k_i \tag{8}$$

with $k_i \geq 0$ and $i = 1, \ldots, \left\lceil \frac{r}{2} \right\rceil - 1$ where $\lceil . \rceil$ is the next highest integer value. Thus, we can conveniently group each $y_i = a_{r-i-1}x_1 + a_i x_2 + b_i$ term in (4) with the corresponding $y_{r-i-1} = a_i x_1 + a_{r-i-1}x_2 + b_{r-i-1}$ one in order to rewrite z as

$$\begin{aligned} z &= \max\left\{\max\{x_1, x_2\}, \ldots, \max\{y_i, y_{r-i-1}\}, \ldots\right\} \\ &= \max\left\{x^*, \ldots, y_i^*, \ldots\right\} \end{aligned} \tag{9}$$

where $x^* = \max\{x_1, x_2\}$, $y_i^* = \max\{y_i, y_{r-i-1}\}$ and $i = 1, \ldots, \left\lceil \frac{r}{2} \right\rceil - 1$. By means of (7) and (8) we rewrite y_i^* as

$$y_i^* = \max\{y_i, y_{r-i-1}\} = a_i(x_1 + x_2) + k_i x^* + b_i. \tag{10}$$

When r is odd $k_{\lceil r/2 \rceil - 1} = 0$ and the max arguments in (9) also include the term

$$y^*_{\lceil r/2 \rceil - 1} = 0.5(x_1 + x_2) + \ln(2). \tag{11}$$

The architecture obtained by implementing (9) and (10) is shown in Fig. 1 (b) where $>> i$ stands for a hard-wired i-position right-shift block (dashed block in the right part of Fig. 1 (b)). In the following we will refer to this solution as A2. As it can be inferred from Figs. 1 (a) and (b), A1 requires $2[r - 2 - (r \bmod 2)]$ multiplications, whereas A2 requires only $2(\lfloor r/2 \rfloor - 1)$ multiplications[1] where $\lfloor . \rfloor$ is the next lowest integer value. As a consequence, A2 additionally reduces the complexity of the CS-tree which has $\lceil r/2 \rceil + 1$ inputs instead of r.

However, the number of multiplications required to compute y_i^* can be further reduced. In fact, from (6) and (8) we obtain $k_i = 1 - 2a_i$, that substituted in (10) leads to

$$y_i^* = x^* + b_i - a_i \Delta \tag{12}$$

where $\Delta = 2x^* - x_1 - x_2$. Thus, (9) by means of (12) can be rewritten as

$$z = x^* + \max\{0, \ldots, b_i - a_i\Delta, \ldots\} \tag{13}$$

when r is even and as

$$z = \max\{x^* + \max\{0, \ldots, b_i - a_i\Delta, \ldots\}, y_{\lceil r/2 \rceil - 1}\} \tag{14}$$

[1]In both cases the trivial multiplication by 0.5 in (11) is not considered

when r is odd. As it can be observed, this implementation reduces the number of multipliers to $\lfloor r/2 \rfloor - 1$, namely by a factor of four, as compared to A1. The complexity of implementing (13) and (14) can be further reduced by applying a predication-like technique, namely since $\delta(x_1, x_2) = x_1 - x_2$ we obtain

$$\Delta = \begin{cases} \delta(x_1, x_2) & \text{if} \quad \delta(x_1, x_2) \geq 0 \\ -\delta(x_1, x_2) & \text{if} \quad \delta(x_1, x_2) < 0. \end{cases} \tag{15}$$

As a consequence, by applying this technique to (13) and (14) we obtain a simplified architecture referred to as A3. This advantage is shown in Fig. 1 (c), where the generation of Δ is no longer required. Furthermore, the sign of $\delta(x_1, x_2)$ is used to select addition or subtraction for the programmable adder (PA) in the multiply-PA (MPA) blocks (see the grey shaded block in left side of Fig. 1 (c) where c_{in} is carry-in of the full-adder corresponding to the least significant bit). When r is odd $\delta(x_1, x_2)$ is also employed to compute

$$y_{\lceil r/2 \rceil - 1} = \begin{cases} \chi^* - \delta(x_1, x_2)/2 & \text{if} \quad \delta(x_1, x_2) \geq 0 \\ \chi^* + \delta(x_1, x_2)/2 & \text{if} \quad \delta(x_1, x_2) < 0 \end{cases} \tag{16}$$

where $\chi^* = x^* + \ln(2)$. However, (15) and (16) show that (13) and (14) can be unified by rewriting (9) as

$$z = x^* + w^* \tag{17}$$

$$w^* = \max\{0, \ldots, w_i^*, \ldots\} \tag{18}$$

$$w_i^* = b_i \mp a_i \delta(x_1, x_2) = b_i - a_i |\delta(x_1, x_2)| \tag{19}$$

with $i = 1, \ldots, \left\lceil \frac{r}{2} \right\rceil - 1$.

It is interesting to note that, from what has been shown in the previous paragraphs for the generalized $\max^*$ operation some already known Log-MAP approximations for turbo decoding can be derived:

1) when $r = 2$, from (4) the $\texttt{lse}(x_1, x_2)$ function is approximated by $\max(x_1, x_2)$, leading to the well known Max-Log-MAP algorithm [8].
2) when $r = 3$, from (11) we can infer that the approximation of $\texttt{lse}(x_1, x_2)$ as A2 is the Average Log-MAP algorithm [9].
3) when $r = 3$, based on (19) A3 is the MacLaurin approximation [6], [10], if $b_1 = \ln(2)$ and $a_1 = 0.5$.
4) when $r = 4$ and the hardware friendly approximation $a_1 \approx 0.25$ is adopted, (19) becomes $w_1^* = \ln(2) - 0.25|\delta(x_1, x_2)|$; thus, A3 is the Linear Log-MAP approximation [11].

In [14], a_i and b_i coefficients were approximated as simple powers of two to ease hardware implementation of turbo decoders. In particular, we have shown in [14] for $r = 3$ and $r = 4$ that implementing

$$z \approx \max\{x^*, 0.5(x_1 + x_2 + 1)\} \tag{20}$$

$$z \approx \max\{x^*, y_1, y_2\} \tag{21}$$

with

$$y_1 = 0.25x_1 + 0.75x_2 + 0.5 \tag{22}$$

$$y_2 = 0.75x_1 + 0.25x_2 + 0.5 \tag{23}$$

causes a negligible BER performance loss with respect to the original solutions for $r = 3$ and $r = 4$ found in [12]. The implementation of (20) in [14] is very simple, as shown in Fig. 2 (a), whereas, in [14] (21) is implemented by using (10) as an A2 architecture (see Fig. 2 (b)):

$$z \approx \max\{x^*, 0.25(x_1 + x_2) + 0.5 + 0.5x^*\}. \tag{24}$$

Further complexity is saved by implementing (24) as an A3 architecture (see Fig. 2 (c)):

$$z \approx x^* + \max\{0, 0.5 \mp 0.25\delta(x_1, x_2)\} \tag{25}$$

Post synthesis results obtained with Synopsys Design Compiler on a 130 nm standard-cell technology for a target clock frequency of 200 MHz representing x_1 and x_2 on eight bits confirm that the implementation of (25) is simpler than (24). In fact, implementing (25) requires only 676 μm^2, whereas (24) [14] requires 883 μm^2, therefore the proposed architecture features a complexity reduction of about 23%. Thus, it is the lowest-complexity solution among the six near-optimal implementations as these are summarized in Table I.

Table I

OCCUPIED AREA COMPARISON (EQUIVALENT GATES) OF DIFFERENT $\max^*$ APPROXIMATIONS ON A 130 NM STANDARD CELL TECHNOLOGY

Alg.	MacLaurin approx.		Average/Linear Log-MAP		r=4 approx. [14] with proposed A3
	[6]	[10]	[9]	[11]	architecture (25)
Area (eq. gates)	190	140	178	163	113

III. APPLICATION OF GENERALIZED $\max^*$ OPERATOR IN LDPC DECODING

Apart from turbo decoding, the two original approximations $r = 3$ and $r = 4$, i.e. from (20) and (21), respectively can be applied, for the first time, to LDPC decoding. For this purpose, the check-node update of the sum-product algorithm can be expressed as [17]:

$$L(U \oplus V) = \max^* \{0, L(U) + L(V)\} - \max^* \{L(U), L(V)\} \tag{26}$$

where $\oplus$ denotes modulo-2 operation, and U and V are two statistically independent binary random variables with log-likelihood ratio (LLR) values of $L(U)$ and $L(V)$, respectively. In all performance evaluation results the coded bits are binary phase-shift keying (BPSK) modulated and transmitted with bit energy E_b over an additive white Gaussian noise (AWGN) channel with single-sided power spectral density N_o. The following reduced complexity decoding algorithms are assumed for comparison: (i) Min-Sum (MS) [18]; (ii) Normalized Min-Sum (NMS) [18]; (iii) Offset Min-Sum (OMS) [18]; (iv) MacLaurin approximation [10]; as well as (v) Sum-Product algorithm (SPA) [19] and (vi) Min-Sum plus correction [19]. The correction term of the latter algorithm is implemented with PWL approximation using six values, thus simplifying the implementation of SPA with near-optimal performance. In

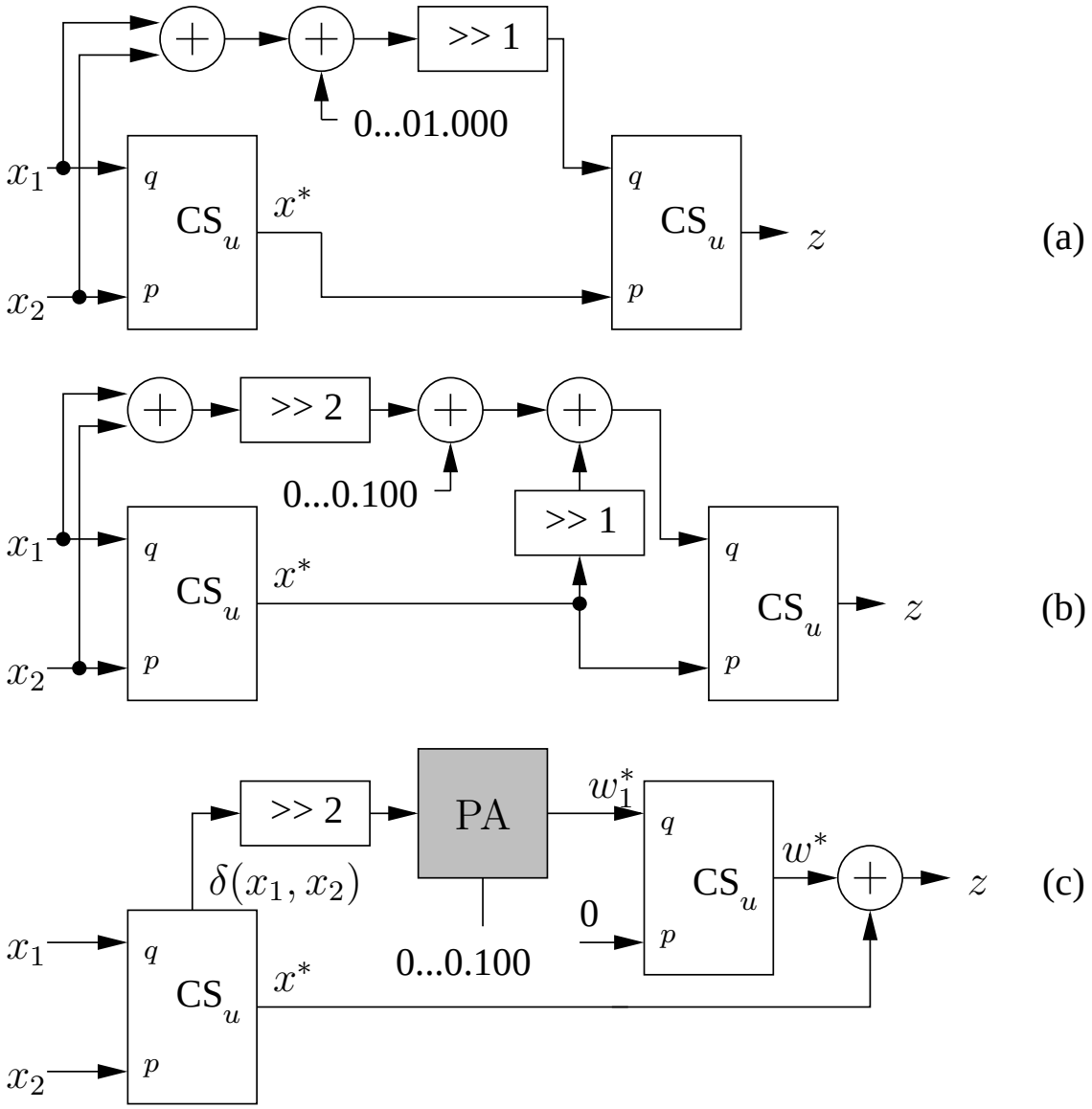


Figure 2. Block diagram of the implementations of: (a) (20) as A2 architecture [14]; (b) (24) as A2 architecture [14]; and (c) (24) as A3 architecture

[19], the multiplying factors in PWL are power of two values being easily implemented in hardware with shift operations.

A randomly constructed regular LDPC code obtained from [20] is considered first with block size $(N, K) = (8000, 4000)$, where K represents the information block size and N the coded block size, respectively. The column weight is equal to three and the coding rate is $R = 1/2$, whereas the decoder assumes maximum 100 iterations. Following [21], to further improve the BER performance scaling is applied in the extrinsic information and the best performing values are found for all investigated algorithms. In particular, the scaling factor for NMS is equal to 0.8, the offset value for OMS is equal to 0.15, and the scaling factor for the MacLaurin approximation is

equal to 0.9. The two approximations $r = 3$ and $r = 4$ have used the same scaling factor as for the MacLaurin approximation. BER performance evaluation results against E_b/N_o are illustrated in Fig 3, in the order of the least to the best performing algorithm. It can be noticed that both $r = 3$ and $r = 4$ approximations (shown always with dashed lines) are inferior to the OMS and NMS. However, when additional scaling is used, then both $r = 3$ and $r = 4$ approximations provide essentially optimal BER performance, in contrast with OMS and NMS, which are 0.1 dB inferior at BER of 10^{-5}. Finally, it is noticed that the performance of MacLaurin approximation degrades as compared with $r = 4$ approximation but when additional scaling is used both algorithms have similar BER performance.

Next, an irregular LDPC code is considered according to the Wi-MAX standard [22]. Note that both SPA and Min-Sum plus correction provide optimal BER performance without using any scaling in the extrinsic information. The block size is $(N, K) = (2304, 1152)$, the coding rate is $R = 1/2$, whereas the decoder assumes maximum 50 iterations. Different from the above, the scaling factor for NMS is equal to 0.87, and the scaling factor for $r = 3$ approximation is 0.85. BER performance evaluation results against E_b/N_o are illustrated in Fig 4, with the same order of performing algorithms and same markers. From this figure, it can be noticed that both $r = 3$, $r = 4$, and MacLaurin approximations achieve essentially optimal BER performance with the additional use of scaling, as opposed with OMS and NMS algorithms. In contrast, NMS provides the best BER performance, with respect to both OMS, $r = 3$, $r = 4$, and MacLaurin approximations without scaling. For the latter codes, Fig. 5 depicts the required average number of iterations (ANI) against E_b/N_o showing the fact that the better the algorithm performs the less number of iterations requires.

IV. Hardware implementation of joint turbo-LDPC decoder

In order to show the effectiveness of the proposed architecture for the generalized $\max^*$ operator in a joint turbo-LDPC decoder a complete, parallel 8 input check node (CN) based on (26) has been designed where the $\max^*$ operator is implemented as in Fig. 2 (c). As highlighted in Fig. 6, the proposed architecture with some multiplexers and PAs can be modified to implement two 8-input $\max^*$ blocks by means of programmable units (PU). Furthermore, these two 8-input $\max^*$ blocks can be used to compute the a-posteriori information in an 8-state turbo decoder architecture, as the Wi-MAX one, namely

$$J = \begin{cases} \max^*\{L(Ui'), L(Vi')\} & \text{if turbo} \\ - & \text{otherwise} \end{cases} \tag{27}$$

$$K = \begin{cases} \max^*\{L(Ui), L(Vi)\} & \text{if turbo} \\ L(Ui \oplus Vi) & \text{if LDPC} \end{cases} \tag{28}$$

where symbols Ui and Vi are processed in the LDPC case ($i = 0, 1, 2, 3$), and the two symbol sets (Ui,Vi) and (Ui',Vi') are processed in the turbo case. As in Section II, a maximum clock frequency of 200 MHz has been set and the synthesis has been carried out on a 130 nm standard-cell ASIC library for a data width of eight bits where the three least significant ones are devoted to represent the fractional part. In Table II the proposed dual

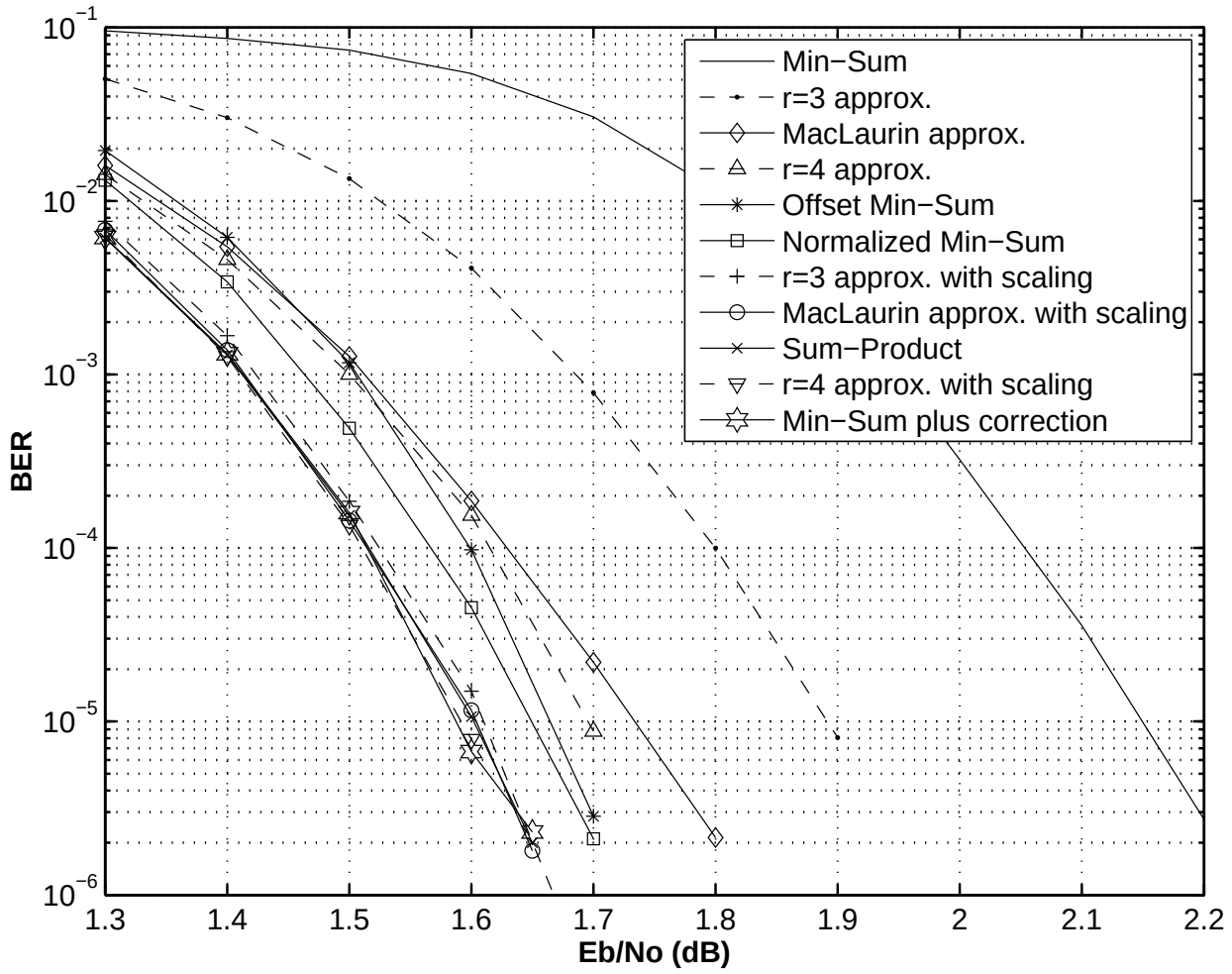


Figure 3. BER performance comparison among several reduced complexity decoding algorithms (from the least to the best performing algorithm order) assuming regular $(N, K) = (8000, 4000)$ LDPC code, coding rate $R = 1/2$, the AWGN channel and maximum 100 iterations.

mode turbo-LDPC architecture is compared with other known solutions to implement CNs. As it can be observed, Min-Sum based architectures, i.e. MS, OMS, and NMS, exhibit lower computational complexity as compared to the architectures which are based on the $\max^*$ design. However, our previously reported results (see Section III) confirm similar finding reported in [15]: that is Min-Sum based LDPC decoding algorithms have inferior BER performance as compared to the ones which are based on the $\max^*$ operation. As a consequence, $\max^*$ based architectures have been suggested as an enabling solution to implement dual mode turbo-LDPC architectures [15], [16]. Moreover, the proposed architecture can be used to reduce the complexity of the dual mode turbo-LDPC architectures proposed in [15], [16] where the correction term $f_c(|x_1 - x_2|)$ in (2) is implemented with a look-up table [8]. As a consequence, in these cases two 8-input $\max^*$ blocks ought to be added to obtain a fair comparison. Each 8-input $\max^*$ block requires 1.09 equivalent kgates (Eq. kgates), thus, in Table II the area for single mode (second and third column) and dual mode (fourth column) architectures is explicitly shown. Finally, Table III depicts an overall comparison, in

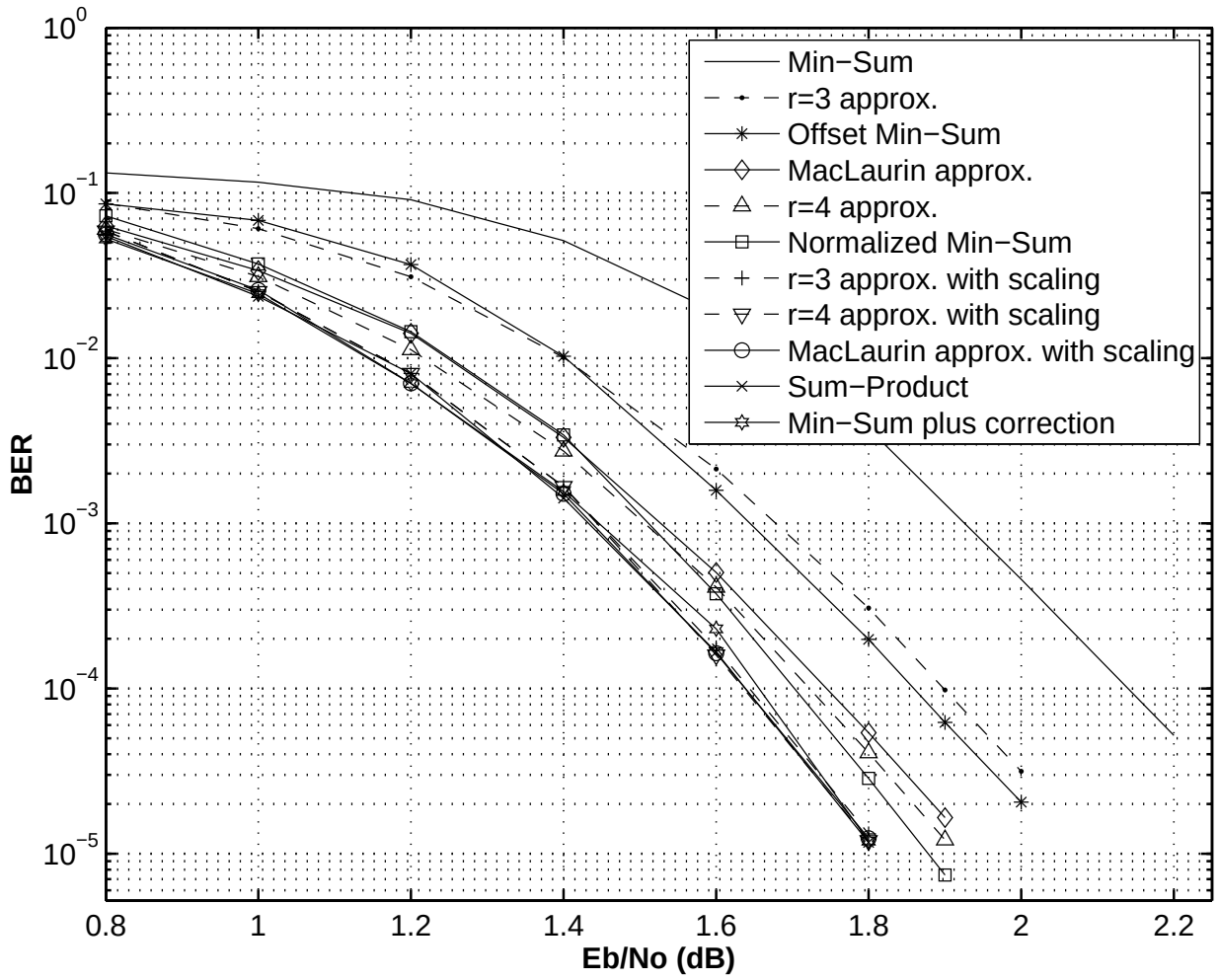


Figure 4. As in Fig. 3 but for irregular $(N, K) = (2304, 1152)$ Wi-MAX LDPC code and maximum 50 iterations.

terms of BER performance, ANI, and equivalent number of gates obtained by post synthesis results for Wi-MAX LDPC code with dual mode architecture. Note that from Table III, the Min-Sum plus correction algorithm [19] requires higher computational complexity as compared to the previously mentioned reduced complexity algorithms. From this table, it is concluded that both $r = 4$ and MacLaurin [10] approximations have the same performance in terms of BER and ANI. However, the proposed architecture outperforms MacLaurin approximation in terms of complexity. Compared with Min-Sum based implementations, the proposed architectures, in particular the one with $r = 3$, achieves better BER with a small increase in the complexity.

V. Conclusion

In this paper practical aspects of $\max^*$ implementation used in both turbo and LDPC decoding were discussed. A unified framework for the generalized $\max^*$ operation, including most of previously published algorithms, was established. Moreover, low-complexity hardware architectures to implement the generalized $\max^*$ operation were

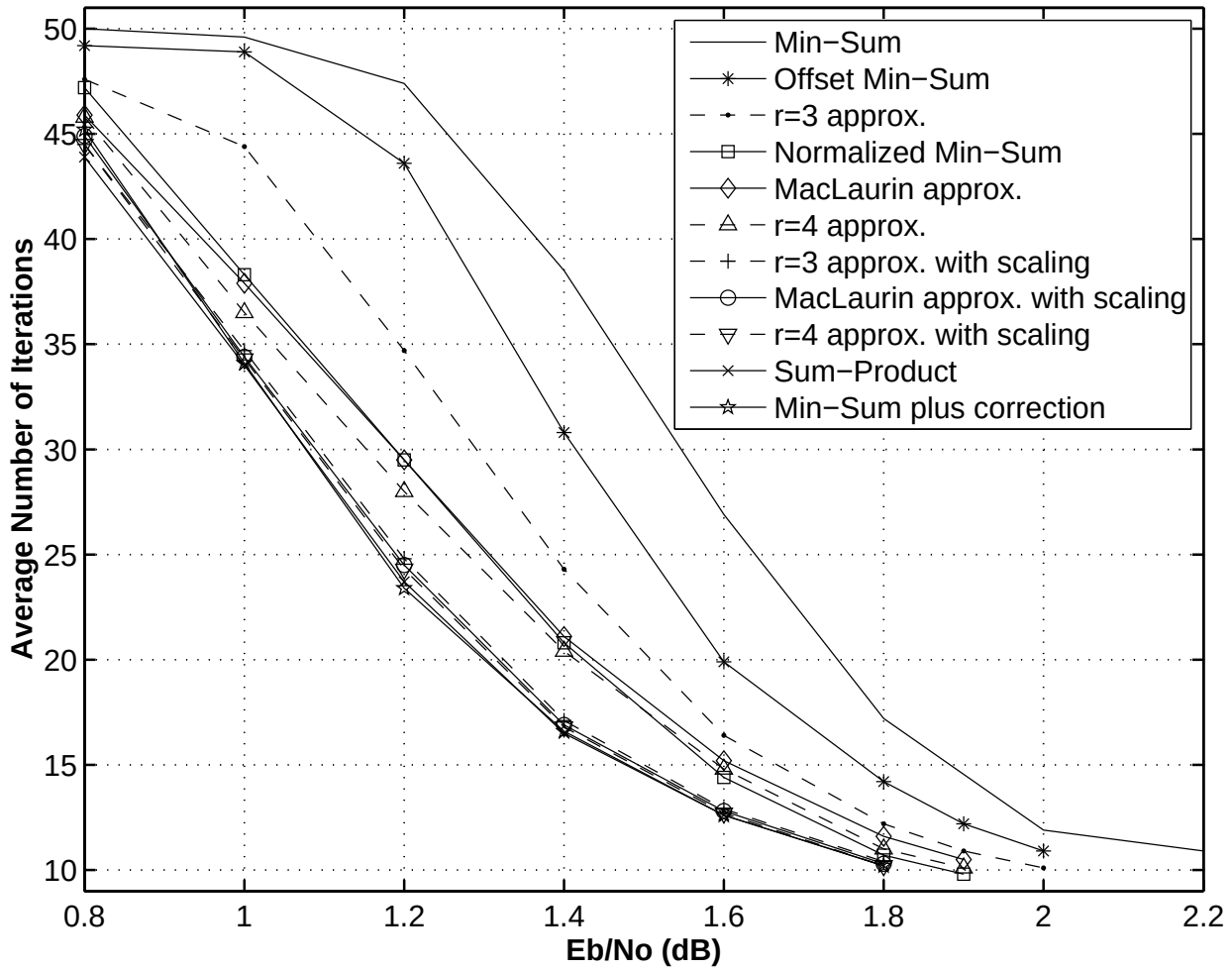


Figure 5. Average number of iterations against E_b/N_o value among several reduced complexity decoding algorithms. The Wi-MAX LDPC code parameters from Fig. 4 are assumed.

presented together with architectural and VLSI design details. Finally, the adoption of this generalized $\max^*$ operation in joint turbo-LDPC architectures was proposed, showing that generalized $\max^*$ implementations with $r = 3$ and $r = 4$ achieve essentially optimal BER performance with lower computational complexity as compared to dual mode turbo-LDPC decoding architectures.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding and decoding: Turbo codes," in *IEEE International Conference on Communications*, 1993, pp. 1064–1070.

[2] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, pp. 21–28, Jan 1962.

[3] S. Jeon and B. V. K. V. Kumar, "Binary SOVA and nonbinary LDPC codes for turbo equalization in magnetic recording channels," *IEEE Trans. on Magnetics*, vol. 46, no. 6, pp. 2248–2251, June 2010.

[4] "Consulative committee for space data systems (CCSDS): Telemetry channel coding, ser. blue book, no. 4," May 1999.

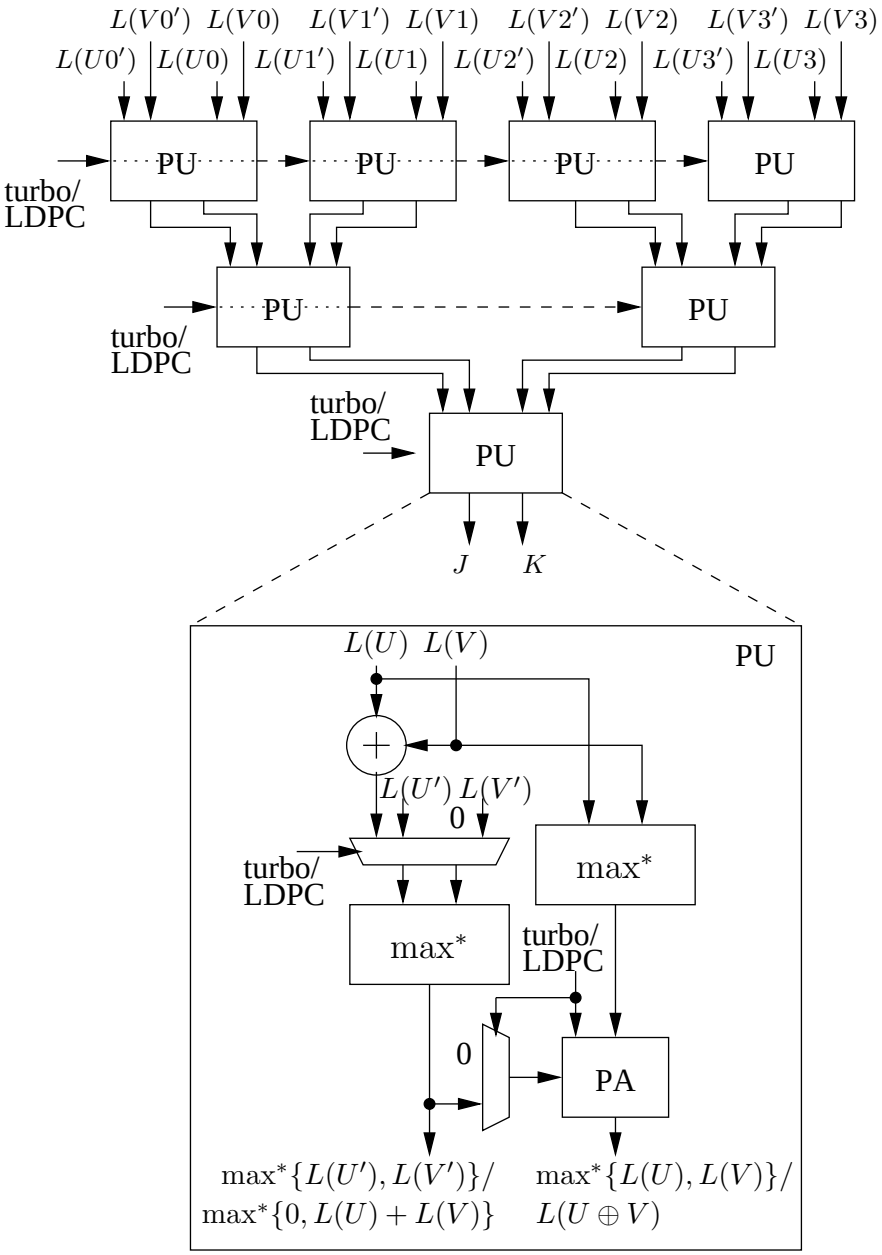


Figure 6. Joint turbo-LDPC architecture

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

[6] S. Talakoub, L. Sabeti, B. Shahrrava, and M. Ahmadi, "An improved Max-Log-MAP algorithm for turbo decoding and turbo equalization," *IEEE Trans. on Instrumentation and Measurement*, vol. 56, no. 3, pp. 1058–1063, Jun 2007.

[7] X. Yin and J. Liu, "Design and implementation of an improved 3G turbo codes interleaver for 3GPP system," in *International Conference on Electronic Measurement & Instruments*, 2009, pp. 319–321.

[8] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the Log domain," in *IEEE International Conference on Communications*, 1995, pp. 1009–1013.

Table II

COMPARISON OF DIFFERENT REDUCED COMPLEXITY DECODING ALGORITHMS TO IMPLEMENT A PARALLEL 8-INPUT CHECK NODE AND TWO 8-INPUT $\max^*$ BLOCKS ON A 130 NM STANDARD-CELL TECHNOLOGY FOR A 200 MHZ CLOCK FREQUENCY

	Single Mode		Dual Mode
Decoding Algorithm	Turbo [Eq. kgates]	LDPC [Eq. kgates]	Turbo+LDPC [Eq. kgates]
MS	2.18	1.49	3.67
OMS	2.18	1.52	3.70
NMS	2.18	1.53	3.71
$r = 3$ approx. (20) & scaling	-	5.24	5.64
$r = 4$ approx. (25) & scaling	-	5.82	6.22
MacLaurin approx. [10] & scaling	-	6.73	7.13
Min-Sum plus correction [19]	2.18	8.04	10.22
Sum-Product	2.18	16.74	18.92

Table III

COMPARISON OF DIFFERENT REDUCED COMPLEXITY DECODING ALGORITHMS $N = 2304$ BITS, $R = 1/2$, WI-MAX LDPC CODE WITH DUAL MODE ARCHITECTURE.

Decoding Algorithm	Eb/No @ BER=10^{-4}	ANI	Eq. kgates
MS	2.14	11.2	3.67
OMS	1.86	13.0	3.70
NMS	1.70	12.5	3.71
$r = 3$ approx. (20) & scaling	1.65	12.3	5.64
$r = 4$ approx. (25) & scaling	1.64	12.2	6.22
MacLaurin approx. [10] & scaling	1.64	12.3	7.13
Min-Sum plus correction [19]	1.64	12.2	10.22
Sum-Product	1.64	12.1	18.92

[9] B. Classon, K. Blankenship, and V. Desai, "Channel coding for 4G systems with adaptive modulation and coding," *IEEE Wireless Communications Magazine*, vol. 9, no. 2, pp. 8–13, Apr 2002.

[10] S. Papaharalabos and P. T. Mathiopoulos, "Simplified sum-product algorithm for decoding LDPC codes with optimal performance," *IET Electronics Letters*, vol. 45, no. 2, pp. 116–117, Jan 2009.

[11] J. Cheng and T. Ottosson, "Linearly approximated Log-MAP algorithms for turbo decoding," in *IEEE Vehicular Technology Conference*, 2000, pp. 2252–2256.

[12] K. L. Hsiung, S. J. Kim, and S. Boyd, "Tractable approximate robust geometric programming," *Springer Optimization Engineering Journal*, vol. 9, no. 2, pp. 95–118, Jun 2008.

[13] M. Sybis, P. Tyczka, S. Papaharalabos, and P. T. Mathiopoulos, "Reduced-complexity algorithms for near-optimal decoding of turbo TCM codes," *IET Electronics Letters*, vol. 45, no. 5, pp. 278–279, Feb 2009.

[14] S. Papaharalabos, P. T. Mathiopoulos, G. Masera, and M. Martina, "On optimal and near-optimal turbo decoding using generalized max*

operator," *IEEE Comm. Letters*, vol. 13, no. 7, Jul 2009.

[15] Y. Sun and J. R. Cavallaro, "Unified decoder architecture for LDPC/turbo codes," in *IEEE Workshop on Signal Processing Systems*, 2008, pp. 13–18.

[16] M. Rovini, G. Gentile, and L. Fanucci, "A flexible state-metric recursion unit for a multi-standard BCJR decoder," in *IEEE International Conference on Signals Circuits and Systems*, 2009, pp. 1–6.

[17] W. E. Ryan, *An Introduction to LDPC Codes*. in CRC Handbook for Coding and Signal Processing for Recording Systems (B. Vasic, ed.), CRC Press, 2004.

[18] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X. Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. on Communications*, vol. 53, no. 8, pp. 1288–1299, Aug 2005.

[19] X. Y. Hu, E. Eleftheriou, D. M. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes," in *IEEE Global Telecommunications Conference*, 2001, pp. 1036–1036E.

[20] D. J. C. MacKay. Online database of low-density parity-check codes. [Online]. Available: http://wol.ra.phy.cam.uk/mackay/codes/data.html

[21] J. Vogt and A. Finger, "Improving the max-log-MAP turbo decoder," *IEE Electronics Letters*, vol. 36, no. 23, pp. 1937–1939, Nov 2000.

[22] *Air interface for fixed and mobile broadband wireless access sytems: Physical and medium access control layers for combined fixed and mobile operation in licensed bands*, IEEE Std. P802.16e-2005 Amendment 2, Feb. 2006.