Energy-efficient multi-standard early stopping criterion for low-density-parity-check iterative decoding

Energy-efficient multi-standard early stopping criterion for LDPC iterative decoding

[†]Carlo Condo, [‡]Amer Baghdadi, [†]Guido Masera

[†] Politecnico di Torino, Torino, Italy – name.surname@polito.it

[‡] Telecom Bretagne, Brest, France – Name.Surname@telecom-bretagne.eu

Abstract

Low-density-parity-check (LDPC) codes decoding relies on powerful iterative algorithms, whose implementation is often expensive in terms of complexity and power consumption. Several Early Stopping Criteria (ESCs) have been proposed to reduce the number of iterations performed by a decoder with no (or limited) degradation of error correction performance. However, most of the existing ESCs have considered a reduced set of system parameters for validation and often have ignored the impacts related to a real hardware implementation. This paper proposes a novel Multi-Standard Early Stopping Criterion (MSESC) able to adapt dynamically to changes of code parameters, quantization and channel conditions. A dedicated hardware architecture is devised and integrated in a multi-standard decoder, and compared to existing techniques. Post-layout results of the proposed MSESC show a small area increment (+1.3%) and a large decrement of the average energy consumption (up to 87.2%) with respect to the same decoder implemented with no ESC. Moreover, it is shown that MSESC offers an energy consumption reduction with respect to State-of-the-Art ESCs ranging from 4% (at high SNR) to 16% (at low SNR).

Index Terms

LDPC, early stopping, VLSI

I. Introduction

Low-Density-Parity-Check (LDPC) [1] codes are used in a wide range of communication standards, such as WiMAX [2], WiFi [3], DVB-S2 [4], DTMB [5] and the standards suggested by CCSDS [6]. Energy efficient LDPC decoders have been designed adopting both generic low-power methodologies [7], [8] and application specific optimizations [9], [10]. These approaches often exhibit large potential for energy saving, but not always they are suitable for multi-standard implementations [11]–[14], which impose the support of several heterogeneous codes and different channel conditions.

Incorporation of Early Stopping Criteria (ESC) in an iterative LDPC decoder enables energy saving by limiting the average number of decoding iterations [15]–[19]. In this approach, the evolution of a certain metric is analyzed over the iterations and a proper stopping rule is set. Quite a large number of ESCs have been described in the open literature: however, in most cases, the real potential of these methods in a multi-standard decoder has not been shown. The performance of ESCs is often provided only in terms of saved number of iterations, for a limited set of codes, and over the Additive White Gaussian Noise channel (AWGN).

The hardware implementation of ESCs in a real decoder introduces overheads in terms occupied area and energy dissipated by the additional logic and memory. Therefore, the actual benefit offered by the incorporated ESC needs to be evaluated by taking into account these overheads and accurate post-layout information is required to this purpose. Moreover, ESC performance often depend on code features and channel conditions. Therefore, in the context of multi-standard implementations, it is important to show the obtained performance for a wide set of codes and assuming realistic channel models.

In this paper, we propose a novel ESC that has been integrated in a complete multi-standard decoder and tested on a wide range of LDPC codes, with both AWGN and Rayleigh fading channel models. First, the Multi-Standard Early Stopping Criterion (MSESC) for LDPC decoding is introduced: it outperforms existing ESCs in terms of number of iterations and energy consumption, and it is adaptive to various LDPC codes and channel conditions, thanks to on-the-fly threshold computation. Then, the MSESC is incorporated into an already available decoder and post-layout accurate evaluations of occupied area and consumed energy are derived.

The rest of the paper is organized as follows. Section II introduces the concepts behind LDPC decoding and overviews the state of the art on ESCs, while Section III and IV detail the proposed MSESC and its performance. In Section V the proposed hardware structure is described and power reduction results are provided under real world conditions and compared to those obtained with other ESCs. Finally, Section VI draws the conclusions.

II. LDPC DECODING

LDPC codes are identified by a sparse parity check matrix **H** [1] with M rows and N columns. The **H** matrix defines all valid LDPC codewords, since a codeword x is valid if and only if it satisfies $\mathbf{H} \cdot x' = 0$, with $(\cdot)'$ the transposition operator.

Belief Propagation (BP) algorithm is most often used for LDPC decoding, mainly with the two-phase scheduling or the layered scheduling [20] scheme, that nearly doubles the convergence speed of the decoding process with respect to two-phase scheduling. In a layered decoder, sets of consecutive, non-communicating parity-check constraints are grouped in layers that are decoded in sequence by propagating extrinsic information from one layer to the following ones [20].

Let us define as $\lambda[c]$ the Logarithmic Likelihood Ratio (LLR) of symbol c: bit LLR $\lambda_k[c]$ is initialized, for column k in **H**, to the corresponding received soft value. The following operations are then executed for all parity constraints l in a given layer:

$$Q_{lk}[c] = \lambda_k^{old}[c] - R_{lk}^{old} \tag{1}$$

$$A_{lk} = \sum_{n \in N(l), n \neq k} \Psi(Q_{ln}[c]) \tag{2}$$

$$\delta_{lk} = \prod_{n \in N(l), n \neq k} \mathrm{sgn}(Q_{ln}[c]) \tag{3}$$

$$R_{lk}^{new} = -\delta_{lk} \cdot \Psi^{-1}(A_{lk}) \tag{4}$$

$$\lambda_k^{new}[c] = Q_{lk}[c] + R_{lk}^{new} \tag{5}$$

where $\lambda_k^{old}[c]$ is the extrinsic information received from one of the previous layers and updated in (5) to be propagated to

one of the subsequent layers. The term R_{lk}^{old}, pertaining to element (l,k) of $\mathbf{H}$ and initialized to 0 in the first iteration, is used to compute (1); the same quantity is then updated in (4), R_{lk}^{new}, and stored to be used again in the following iteration. In (2) $N(l)$ is the set of bit indexes that contribute in the l^{th} parity check. The decoded vector y is obtained by observing the sign of all $\lambda_k[c]$ at the end of each iteration, and the syndrome SYN is consequently obtained as

$$\mathrm{SYN} = \sum_{l=1}^{M} \mathbf{H}_l \cdot y' \tag{6}$$

The BP algorithm involves some complex computations, in particular $\Psi(\cdot)$ that requires the calculation of a hyperbolic tangent. According to the Self-Corrected-Min-Sum (SCMS) approximation [21], it can be simplified with a limited degradation of error correction performance as:

$$R_{lk}^{new} \approx -\delta_{lk} \cdot \min_{n \in N(l), n \neq k} \{|Q_{nk}|\}, \tag{7}$$

together with a dynamic correction. When $Q_{lk}^{i}[c]$ is computed at the i^{th} iteration, it is compared with $Q_{lk}^{i-1}[c]$. If their signs are different, $Q_{lk}^{i}[c]$ is substituted with zero for the current iteration, inducing a conservative behavior in presence of the uncertainty represented by a sign change.

ESCs allow to evaluate, after each iteration, if it is worth to proceed with the decoding process or not. In case the conditions of the ESC are met, the decoding is stopped, either interrupting the already initiated iteration or not even starting it. The Genie ESC is an unrealizable technique commonly utilized as reference for comparisons: using foreknowledge of the information bits, the decoding is initiated only if the outcome is going to be successful. Practical ESCs can be classified into two subgroups: ESCs identifying *successful decoding* of a frame, and ESCs identifying *impossible decoding*.

An efficient *successful decoding* ESC is the *parity check* method [22]. Since a received vector y is a valid LDPC codeword if and only if $\mathbf{H} \cdot y' = 0$, the syndrome calculation unfailingly identifies correct codewords. The *parity check* is performed after each iteration: if the codeword is correct the decoding is stopped, if not it proceeds until success or until the maximum number of iterations has been reached.

Much more varied is the pool of *impossible decoding* ESCs. With "impossible" we refer to the decoding processes that are not going to be successful within the maximum number of allowed iterations. In [16] the Convergence of Mean Magnitude (CMM) method is presented. It measures the slope or changing ratio of the a posteriori LLR mean magnitude. In case of *impossible decoding*, this measure converges to and fluctuates around zero. If this situation is detected for long enough, the decoding is interrupted. In [17] the proposed ESC monitors the evolution of the syndrome SYN of the decoder. The decoding is stopped if SYN is larger than a certain threshold for three consecutive iterations, showing no sign of convergence. Two metrics are observed in the ESC described in [18]: the Check Node Mean Magnitude (CNMM) monitors the average value of $Q_{lk}[c]$ messages, while the Check Node Mean Checksum (CNMC) calculates the syndrome substituting $\lambda_k[c]$ with $Q_{lk}[c]$. If CNMM keeps decreasing and CNMC increasing for a set of It_{ESC} consecutive iterations, the codeword is deemed undecodable, and the process is interrupted. The ESC proposed in [19] is tied to the SCMS algorithm, since it keeps track of the number of erased messages (Erased Messages ESC, EMESC). In case of *impossible decoding*, this number does not converge to zero and the decoding process shows a high degree of unreliability.

III. Multi-Standard Early Stopping Criterion

The proposed Multi-Standard Early Stopping Criterion (MSESC) relies on the computation of two metrics whose behavior over different iterations can be observed to detect an *impossible decoding*. The two considered metrics are defined as follows:

$$\mathrm{CNMM}^i \equiv \sum_{l=1}^{M} \min_{k \in N(l)} \left| R_{lk}^i \right| \tag{8}$$

$$\mathrm{SYN}^i \equiv \sum_{l=1}^{M} \bigoplus_{k \in N(l)} \left(\mathrm{HD}\left(\lambda_k^i[c] \right) \right) \tag{9}$$

where i is the iteration number, R_{lk}^i is the value obtained in (4) at the current iteration, $\mathrm{HD}(\lambda_k^i[c])$ stands for the Hard Decision on bit k based on the $\lambda_k^i[c]$ metric, and $\oplus$ is the XOR logic operation. The CNMM metric [18] is a measure of the reliability of the codeword as estimated by the parity checks. The CNMM is compared to the previous iteration's value to evaluate its slope. The syndrome computation SYN at i^{th} iteration as expressed in (9) is equivalent to (6), but highlights the simple operations involved. MSESC exploits CNMM and SYN to detect both *impossible* and *successful decoding*. **The evolution of both CNMM and SYN has been analyzed for a large number of codes. In particular, codes from the WiMAX, WiFi, DTMB, CMMB and DVB-S2 standards have been considered. Both metrics have shown similar trends and behaviors in all codes, on which the proposed MSESC has been developed. While their respective evolution is connected, they allow for two separate points of view on the state of the decoding process that help accurate decisions on the early stopping of iterations. The value of SYN and its trend throughout iterations are able to give a general snapshot of the decoding process. Large SYN and slow convergence can signal uncorrectable codewords: this information, however, must be read taking in account CNMM, that gives a measure of the general reliability of the decoding. A large and steadily growing CNMM value signifies high confidence, that might allow correction of even numerous errors.**

The complete MSESC flow within the decoding process is described in Algorithm 1. The Impossible Decoding Detection (IDD) mode is initially activated. At the end of each iteration SYN^i is evaluated. The *parity check* ESC is then run (lines 5 - 6 in Algorithm 1) to see if a *successful decoding* is reached. The second ESC check (lines 8 to 21), aimed at detecting *impossible decoding*, is only run if IDD is active (line 7). The counter CNT is incremented at the end of the current iteration if CNMM is decreased and SYN is increased w.r.t. previous iteration (lines 11 - 15). When CNT reaches threshold It_{ESC}, *impossible decoding* is detected and the decoder is stopped (lines 16 - 17). Moreover, a stopping criterion has been introduced to identify slow convergence behavior, i.e. a codeword that is decodable per se, but not within the currently allowed maximum iterations of the decoder It_{max} (lines 19 - 20).

In case the decoding is going to be successful within It_{max} iterations, it is desirable to deactivate IDD to avoid unnecessary power consumption. For this reason, an adaptive deactivation function has been devised. If after few iterations (for example two) CNMM is larger than a threshold $T2$ or the syndrome SYN is smaller than $T3$, unsuccessful decoding is a very low probability event, and IDD is deactivated for the current codeword, avoiding further computations (lines 9 - 10). **The different checks within Alg. 1 are a result of the CNMM and SYN observation on a wide number of codes. While the decoding proceeds effectively, CNMM rises and SYN decreases: $T2$ and $T3$ thresholds must be tuned so that when the check of line 9 is verified the decoder has an extremely high probability of completing the decoding successfully. At the same**

Algorithm 1 MSESC

1: $\mathrm{CNT} \leftarrow 0$, Calculate T1, T2, T3
2: Activate IDD (Impossible Decoding Detection)
3: **for all** iterations $1< i < \mathrm{It}_{max}$ **do**
4: Decode, Calculate SYN^i
5: **if** $\mathrm{SYN}^i = 0$ **then**
6: Stop Decoding (*successful decoding*)
7: **else if** IDD is active **then**
8: Calculate CNMM^i
9: **if** $i \geq 2$ and ($\mathrm{CNMM}^i > T2$ or $\mathrm{SYN}^i < T3$) **then**
10: Deactivate IDD
11: **else if** $\mathrm{CNMM}^i < \mathrm{CNMM}^{i-1}$ and $\mathrm{SYN}^i > \mathrm{SYN}^{i-1}$ **then**
12: $\mathrm{CNT} \leftarrow \mathrm{CNT} + 1$
13: **else**
14: $\mathrm{CNT} \leftarrow 0$
15: **end if**
16: **if** $\mathrm{CNT} = \mathrm{It}_{ESC}$ **then**
17: Stop Decoding (*impossible decoding*)
18: **end if**
19: **if** $i \geq 0.6 \cdot \mathrm{It}_{max}$ and $\mathrm{SYN}^i > T1$ **then**
20: Stop Decoding (*impossible decoding*)
21: **end if**
22: **end if**
23: **end for**

time, it has been observed that if at a certain point SYN is not low enough ($T1$, line 19), it is very unlikely that the remaining iterations are sufficient to correct the errors. Contrary to MSESC, a few theoretically-based ESCs have been investigated in the past, especially regarding turbo decoding [23], [24]. However, these methods, that usually rely on cross entropy, provide excellent performance at a very high computational complexity, which introduces large area and power overheads.

To allow the decoder to dynamically adapt to the used code, on-the-fly computation of $T1$, $T2$ and $T3$ has been devised:

$$T1 = \mathrm{M} \cdot 2^{-6} \tag{10}$$

$$T2 = \mathrm{M} \cdot 2^{bits_f} \tag{11}$$

$$T3 = \mathrm{M} \cdot 2^{-5} = T1 \cdot 2 \tag{12}$$

where $bits_f$ is the number of bits assigned to the representation of the fractional part of $\lambda_k[c]$.

Threshold $T3$ sets an upper limit for SYN, and expresses a percentage of wrong parity checks that is low enough to be likely corrected in case of *successful decoding*, i.e. 1/32 of M, where M is the number of rows of the LDPC parity check matrix. Fig. 1 and 2 plot the average SYN for a short WiMAX code and a long DTMB code. With *successful decoding*, SYN is monotonically decreasing and very steep in the first iterations: the number of wrong parity checks quickly descends below $T3$, while it is confined at much higher values in presence of *impossible decoding*. The choice of $T3$ scales well with the frame size N: long codes have high error correction capabilities and are able to sustain large $T3$ values, while short codes are less performing and require consequent $T3$ reduction. The same concept as $T3$ applies to $T1$, that identifies a percentage

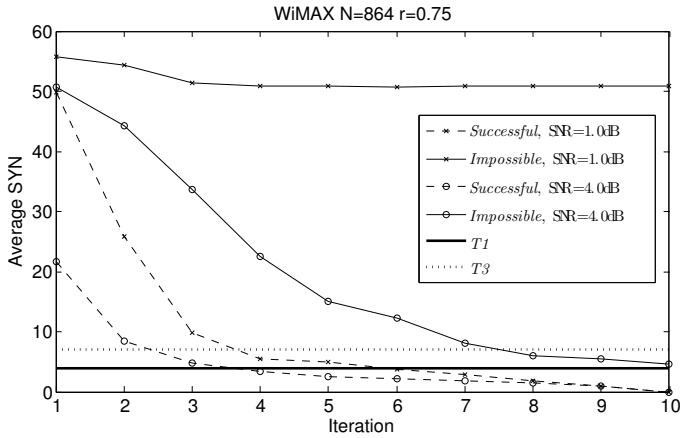

Figure 1: Average SYN for WiMAX N=864 r=3/4 in case of *successful* and *impossible decoding*

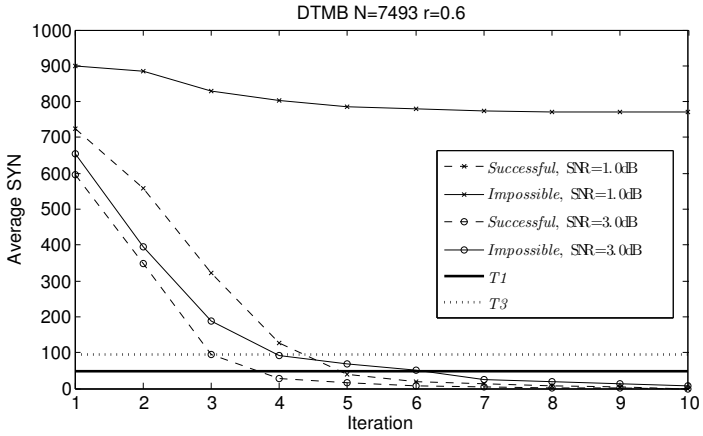

Figure 2: Average SYN for DTMB N=7493 r=3/5 in case of *successful* and *impossible decoding*

of M (1/64) not likely to be corrected within It_{max} due to slow convergence. As shown in Fig. 1 and 2, at high SNR also *impossible decoding* cases can manage to satisfy the condition in line 9, but this occurrence is taken care by $T1$ (line 19). The trajectories plotted in Fig. 1 and 2 show how both $T1$ and $T3$ depend on N and M. On the contrary, when min-sum decoding is adopted, the CNMM metric does not depend on the degree of check nodes, since only the minimum of $|R_{lk}^i|$ is considered for every parity check. Finally, $T1$ and $T3$, as well as $T2$, are independent of the considered channel mode, since different models only result in SNR shifts at reception [25].

$T2$ is compared in line 9 of Algorithm 1 to the value of CNMM at each iteration. The $|R_{lk}^i|$ elements that are summed in (8) are quantized with $bits_{tot} = bits_{int} + bits_f$: to take in account the initial left-shift of the integer part, in (11) also the value of M is left-shifted by the same number of positions $bits_f$. Consequently, for CNMM> $T2$ to be verified, the average value of $\min|R_{lk}^i|$ must be > 1. This is a condition verified early in case of *successful decoding*, while $|R_{lk}^i|$ values in *impossible decoding* tend to float much closer to zero. Fig. 3 and 4 show the average CNMM for a WiFi and a DVB-S2 code respectively, under different channel conditions. At low SNR, it can be seen how the average CNMM stays below $T2$ in case of *impossible decoding*, while it quickly rises above it in case of *successful decoding*. At high SNR, where also *impossible decoding* cases have CNMM> $T2$, the decoding is stopped by the condition in line 19.

The curves in Fig. 1 to 4 have been obtained with $bits_{tot}$ = 10 and $bits_f$ = 3. Typical internal quantization of bit LLRs, that

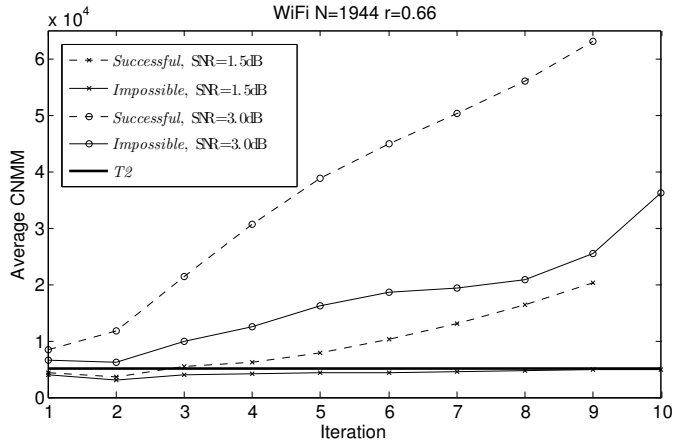

Figure 3: Average CNMM for Wifi N=1944 r=2/3 in case of *successful* and *impossible decoding*

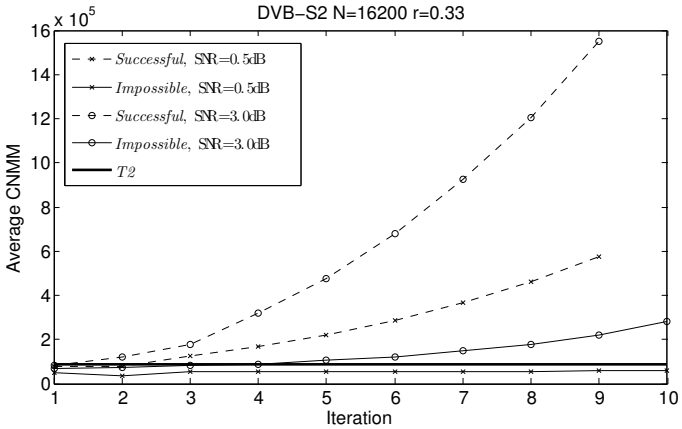

Figure 4: Average CNMM for DVB-S2 N=16200 r=1/3 in case of *successful* and *impossible decoding*

does not cause signicant BER degradation, can be reduced to $bits_{tot}$ = 7 and $bits_f$ = 1, or even to $bits_{tot}$ = 6 and $bits_f$ = 0. The effectiveness of the proposed MSESC has been validated for various quantization parameters. The chosen thresholds maintain their validity with different quantizations: changes in $bits_f$ are taken in account by $T2$, while $bits_{int}$ will only impact the upper bound of the CNMM metric.

Equation (10)-(12) have been deduced by empirical observations on the behavior of CNMM and SYN. Since both metric behavior and evolution is dependent on a number of parameters that changes with the code, the computed thresholds can be considered a best-fit: however, as shown later in Section IV and Fig. 8, the on-the-fly values guarantee quasi-optimal performance.

IV. MSESC PERFORMANCE

In Fig. 5 statistics are plotted for the activation and deactivation of MSESC with SCMS. Results have been averaged over several millions frames. At each simulated SNR, the black bars indicate the percentage of frames for which an *impossible decoding* has been detected at some point within It_{max}, i.e. an effective early stopping. The white bars, on the contrary, show the percentage of frames for which IDD has been deactivated (line 10 in Algorithm 1). This happens when the algorithm decides that the *impossible decoding* event will not occur for the current frame: in this case, a *possible decoding* is detected

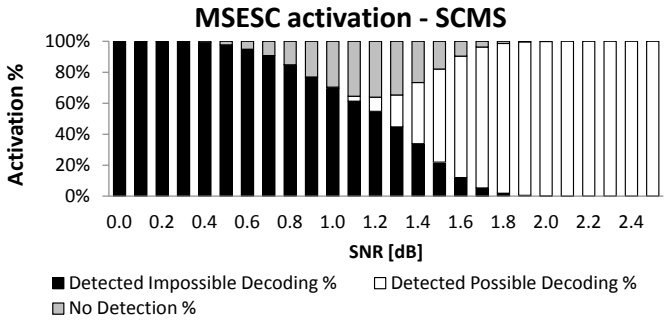

Figure 5: Activation percentages of MSESC for WiMAX 2304, 1/2

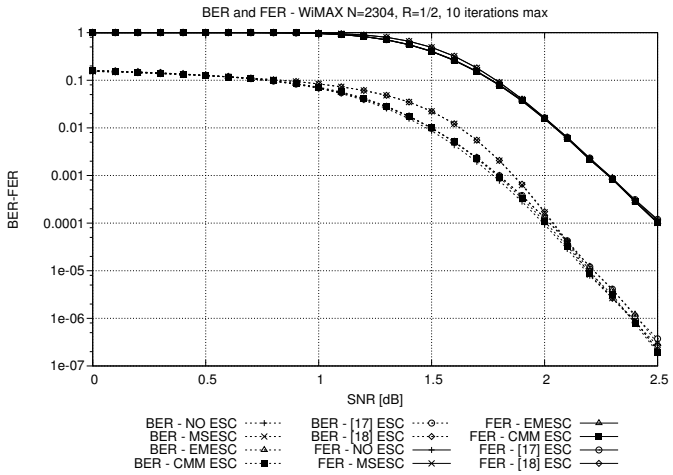

Figure 6: BER and FER curves with Early Stopping Criteria

and a *successful decoding* will be reached in a later iteration. It is possible to see how, when the SNR increases, the *impossible decoding* detection percentages decrease while the IDD deactivation percentages increase. However, there is a third possible situation, in which IDD is not deactivated but no early stopping is enforced. This case is represented by the gray *no detection* bars, which only cover a small range of SNRs.

The proposed stopping criterion has been compared to existing ESCs in terms of Bit and Frame Error Rate (BER and FER) degradation and iteration reduction capabilities. Fig. 6 and 7 present simulation results for WiMAX code with N= 2304, R= $1/2$ on an AWGN channel, with It_{max} = 10. For the threshold $T2$, we considered in these simulations $bits_f$ = 3 as the number of quantization bits to represent the fractional part of the bit LLR $\lambda_k[c]$.

The internal quantization of the decoding algorithm is 10 bits, including $bits_f$ = 3. However, MSESC is affected only by the number of $bits_f$ (taken in account by $T2$) and not by the total number of bits: its effectiveness is consequently guaranteed for all quantizations.

Fig. 6 plots the BER (dotted lines) and FER (continuous lines) curves for the *impossible decoding* ESCs analyzed in Section II: minor performance degradations are present early in the waterfall region.

Fig. 7 shows the average number of iterations (It_{AVG}) performed by various ESCs: the Genie ESC curve is used as a reference plot. All ESC are supposed to work in conjunction with *parity check* ESC for the sake of a fair comparison. Most of the curves join the "parity check" curve at around SNR=1.6 dB, where the number of detected undecodable frames is close to

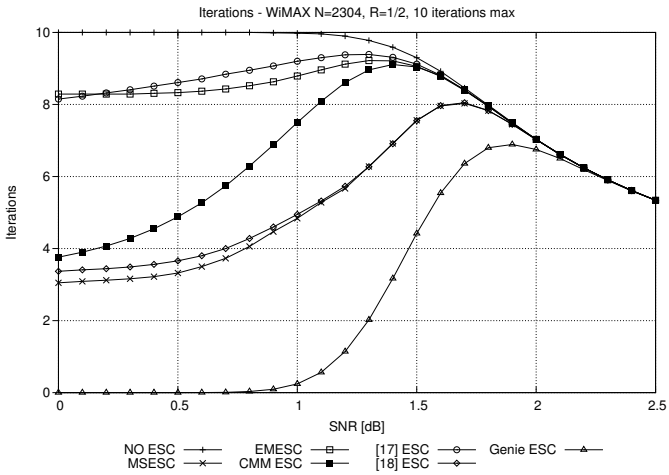

Figure 7: Average number of iterations with different ESCs

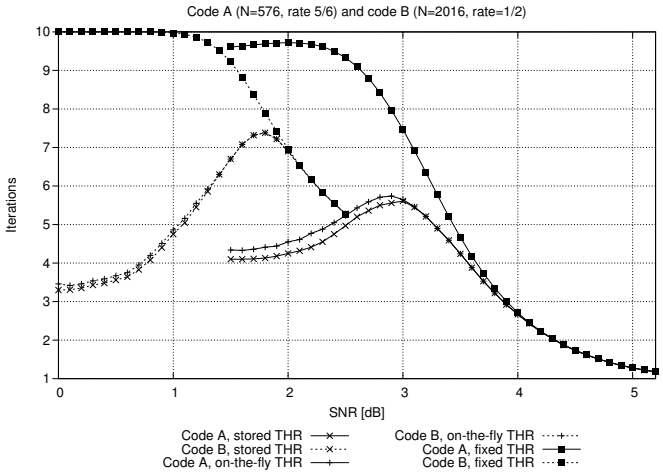

Figure 8: Average number of iterations with MSESC and different thresholds

zero. The [17] ESC and EMESC show similar performance, with two saved iterations at zero SNR, while CMM performs much better at low SNR. A large number of saved iterations can be observed for MSESC and [18] ESC. The dynamic correction in SCMS can in fact result in sudden changes in the CNMM measure, that can occasionally lead to additional errors (Fig. 6). However, these rapid metric variations allow for early detection of *impossible decoding*, with consistent saving of iterations. MSESC is the most effective method, with three average iterations performed at low SNR.

While the results shown so far have been obtained with the SCMS algorithm, MSESC is equally effective also under the other min-sum approximations [26], i.e. Normalized Min-Sum (NMS), Offset-Min-Sum (OMS) and traditional Min-Sum (MS). In Fig. 9 it is shown how the curve of performed iteration is almost the same in case of SCMS, NMS and OMS, with a maximum of 0.75 iterations of difference at SNR=0. The MS curve shows a generally higher number of performed iterations: this is to be expected since it is not caused by MSESC, but by the algorithm performance itself. In fact, the traditional MS algorithm is strongly suboptimal w.r.t. the other approximations. MSESC maintains its effectiveness even when switching from the layered scheduling to the two-phase scheduling. In the two scheduling schemes the decoder operations (1)-(5) remain the same, what changes is the number and order of LLR updates per iteration. While influencing the speed at which CNMM and SYN evolve, they do not change their overall behavior and

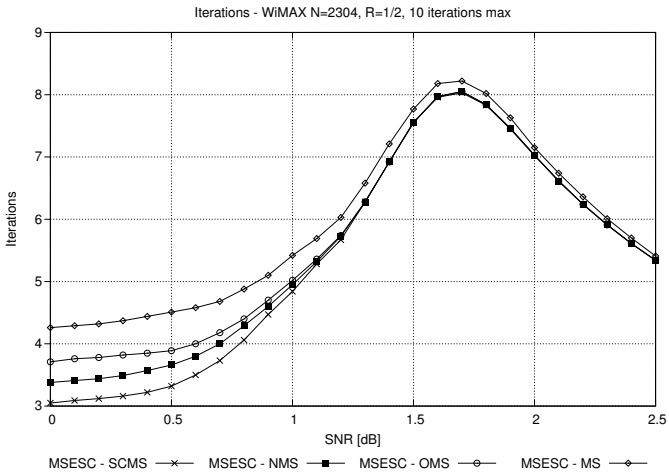

Figure 9: **Average number of iterations with different decoding algorithms**

trend. This is taken in account by the It$_{max}$ of two-phase decoders, that is usually set higher than in layered decoders to overcome the slower algorithm convergence.

In order to illustrate the potential of the on-the-fly threshold computation incorporated in MSESC, in Fig. 8 the average iteration number for two codes A (N=576, R=5/6) and B (N=2016, R=1/2) are evaluated with different $T1$, $T2$ and $T3$ management policies.

- *Stored THR*: in this case, thresholds are individually optimized for each code, via extensive simulations. These thresholds are pre-computed, stored, and read according to the code currently in use, allowing for the best MSESC average iterations results.
- *On-the-fly THR*: in this case, thresholds are computed on-the-fly according to (10), (11) and (12). The differences observed with respect to the stored threshold methods are negligible, and no memory has to be allocated to store the thresholds required for each supported code.
- *Fixed THR*: these curves show the effect on MSESC performance in case thresholds are fixed for all considered codes. In particular, the effects of strongly suboptimal thresholds are reported: to code A are applied thresholds fine-tuned for code B, and vice versa. Since size and rate are substantially different for the two codes, the optimal thresholds can vary consistently, causing severe performance degradation.

Similar behavior has been observed for a wide analyzed range of LDPC codes (WiFi, WiMAX, DTMB, CMMB, DVB-S2). The results illustrate how the proposed on-the-fly computation of thresholds combines both high flexibility and high accuracy.

V. MSESC Hardware Structure and Implementation

In order to evaluate the benefits of an ESC in terms of energy consumption, it is necessary to integrate it in a real hardware channel decoder and to compare the overheads related to its integration against the benefits related to the reduced number of decoding iterations. To that purpose, we applied the proposed MSESC to a fully characterized multi-standard channel decoder [14] based on a message passing multiprocessor NoC-based architecture [27]. The implementation referred in [14] as **A** has been considered after adaptation to support only LDPC codes and to use the SCMS decoding algorithm. This version of [14].**A**

Table I: Effect of MSESC on complexity (A), average power consumption (P) and energy per decoded frame (E_F) (It_{max} = 10, CMOS 90 nm technology, 1.0 V supply, post-layout)

	CODE	SNR	$\mathbf{A}_{\mathrm{REF}}$	$\mathbf{A}_{\mathrm{PC+CMM}}$		$\mathbf{A}_{MSESC}$	
A [mm^2]	-	-	2.40	2.49	+3.8%	2.43	+1.3%
P [mW]	-	-	90.2	95.5	+5.9%	92.9	+3.0%
E_F [μJ]	2304 1/2	0.0 dB	1.49	0.69	-53.7%	0.46	-69.2%
		1.0 dB		0.79	-47.0%	0.75	-50.4%
		2.0 dB		1.10	-26.2%	1.05	-29.5%
	960 2/3	0.0 dB	0.66	0.31	-53.1%	0.20	-69.7%
		1.0 dB		0.35	-47.0%	0.33	-50.0%
		2.0 dB		0.49	-25.8%	0.46	-30.3%
	1944 3/4	0.0 dB	1.32	0.61	-53.8%	0.41	-68.9%
		1.0 dB		0.70	-47.0%	0.66	-50.0%
		2.0 dB		0.97	-26.5%	0.93	-29.6%
	1440 5/6	0.0 dB	0.98	0.46	-53.1%	0.30	-69.4%
		1.0 dB		0.52	-47.0%	0.50	-49.0%
		2.0 dB		0.73	-25.5%	0.69	-29.6%

has been named $\mathbf{A}_{\mathrm{REF}}$ and has been used as a reference in the following comparisons. It relies on 22 Processing Elements (PEs) connected by means of a Kautz [28] network-on-chip, with routing elements of degree three. The decoder supports all LDPC codes in WiMAX and WiFi standards.

All of the operations required by the proposed MSESC can be implemented with a very low hardware cost. On-the-fly thresholds computation, which is required only when switching from a code to a new one, simply implies shift operations. Therefore, its contribution to power consumption is negligible. On the contrary, power consumption overhead related to the implementation of parity check ESC is not negligible, although it is rarely reported in previous papers on ESC methods. The great majority of current decoders relies on multiple PEs, and the data need to be gathered from all of them to compute SYN, thus usually implying non negligible overheads. Partially parallel decoders can exploit the idea behind the on-the-fly syndrome calculation proposed in [19]: after the execution of (5), the sign bits of the $\lambda_k^{new}[c]$ involved in the current parity check are XORed. These values are concurrently available to the PE without requiring additional memory accesses. During each iteration, the resulting bit in every PE is summed to the bits of the preceding parity checks through an adder modulo $\lceil \mathrm{M/P} \rceil$, where P is the number of PEs. At the end of an iteration, each PE holds a partial syndrome value: these are gathered through dedicated connections and summed one at a time to obtain the global syndrome value. Since the number of PEs is typically much smaller than M, such architecture achieves good compromises in terms of complexity and latency. CNMM can be implemented with the same on-the-fly approach by accumulating R_{lk}^{new} of every parity check as soon as they are computed. As with SYN, partial CNMM values are retrieved and summed together.

Table I reports area, power and energy consumption of different implementations: $\mathbf{A}_{\mathrm{REF}}$ is the reference architecture mentioned above, $\mathbf{A}_{\mathrm{PC+CMM}}$ implements both CMM [16] and *parity check*, $\mathbf{A}_{\mathrm{MSESC}}$ includes the proposed multi-standard ESC. **CMM is the ESC, among those considered, whose performance remains more stable when changing code: it has consequently been selected for implementation and comparison**. These three architectures have been implemented targeting 90 nm CMOS technology. The simple additional logic required in $\mathbf{A}_{\mathrm{MSESC}}$ results in +1.3% area occupation and +3.0% average power consumption after layout, while the area and power consumption overheads are higher with $\mathbf{A}_{\mathrm{PC+CMM}}$. In fact, in [16] it is described how a single CMM threshold value can be used for all codes, leading to small differences in performance. While this is true for the SNR intervals considered in [16], a single threshold causes large differences in the number of performed iterations at low SNR, as it has been shown in Fig. 8. Moreover, BER degradation can be observed at all

SNR when very different codes use the same CMM threshold. It is consequently necessary a memory to store a threshold value for every code supported, and in a multi-standard decoder like the one considered the cost of such memory is not negligible. Between WiFi and WiMAX, a total of 131 different LDPC codes are specified: thus, thresholds are calculated off-line and stored in a dedicated memory at least 131 words deep and with width greater or equal to the number of bits assigned to the representation of LLRs. This memory and the more complex CMM computations are the main contributions behind the +3.9% area and +5.9% power consumption in the $\mathbf{A}_{\mathrm{PC+CMM}}$ case. This overhead is avoided in $\mathbf{A}_{\mathrm{MSESC}}$ thanks to on-the-fly threshold computation.

The benefits of ESCs can be appreciated only over the whole decoding process: E_F expresses the average energy spent in the decoding of a frame for different codes and SNR points. It is defined as:

$$\mathrm{E}_F = \frac{\mathrm{P} \cdot cycles_{it} \cdot It}{f} \tag{13}$$

where P is the average power consumption, $cycles_{it}$ the clock cycles needed to complete an iteration, f the clock frequency and It the number of performed iterations. In $\mathbf{A}_{\mathrm{REF}}$ no ESC is present, and $It = It_{max} = 10$ for all the considered codes. Consequently, the consumed energy does not vary with the SNR, but only with the code change. The situation is different in case of $\mathbf{A}_{\mathrm{MSESC}}$, in which the performed iterations follows a pattern close to that of Fig. 7. Moreover, thanks to the activation and deactivation function (line 9 of Algorithm 1) the contribution of MSESC to E_F must be weighted according to the percentages displayed in Fig. 5. E_F is calculated as

$$\begin{aligned} \mathrm{E}_F = & \frac{(\mathrm{P}_{\mathrm{REF}} + \mathrm{P}_{\mathrm{PC}}) \cdot It \cdot cycles_{it}}{f} \\ & + \frac{cycles_{it} \cdot \mathrm{P}_{\mathrm{IDD}} \cdot (It_D \cdot DP_{\%} + It \cdot (1 - DP_{\%}))}{f} \end{aligned} \tag{14}$$

where $\mathrm{P}_{\mathrm{REF}} + \mathrm{P}_{\mathrm{PC}}$ is the power consumption of $\mathbf{A}_{\mathrm{REF}}$ and the *parity check* part of MSESC, $\mathrm{P}_{\mathrm{IDD}}$ is the power consumption due to the IDD part of MSESC, $DP_{\%}$ is the deactivation percentage at the considered SNR point, and It_D the average deactivation iteration. Note that at high SNR $DP_{\%}$ is close to 1, leading to a very small $\mathrm{P}_{\mathrm{IDD}}$. Table I displays E_F for four codes taken from the supported standards, considering three SNR points for each code. At SNR=0.0 dB MSESC manages to reduce by around 70% the energy consumption with respect to $\mathbf{A}_{\mathrm{REF}}$, while at SNR=1.0 dB the gain is reduced to 50%. At higher SNR the IDD is mostly deactivated, while the *parity check* part of MSESC allows for an average 30% gain. The gains are less significant for $\mathbf{A}_{\mathrm{PC+CMM}}$. At low SNR, CMM implies higher average number of iterations than MSESC, with $\mathbf{A}_{\mathrm{MSESC}}$ saving around 16% more energy than $\mathbf{A}_{\mathrm{PC+CMM}}$, while the increased power consumption due to the threshold memory and the absence of a deactivation function leads to larger E_F (4-5%) also at high SNR.

Another interesting assessment can be done by considering application-specific operational conditions in terms of SNR probability distribution. In fact, real world applications place their SNR working point taking in account large variation margins, in order to have very low probabilities of SNR that prevent the correct functionality. For example, for a typical indoor WiFi connection the industry suggests a minimum of 20 dB SNR to guarantee a very good level of connectivity [29]. Assuming a Rayleigh fading channel model that remains stable for the duration of a frame, it is possible to correlate the corresponding BER curves with those obtained with an AWGN channel [25]. This property can be joined with the instantaneous SNR probability

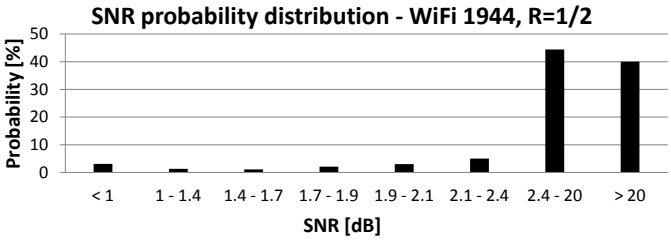

Figure 10: SNR probability distribution for WiFi 1944, 1/2

Table II: Implementations comparison on average iterations (It_{AVG}) and energy (E_F) for Wifi N=1944, R=1/2 code SNR probability distribution (CMOS 90 nm, 1.0 V, post-layout)

	It_{MAX}	It_{AVG}	E_F [μJ]	$\mathbf{A}_{\mathrm{MSESC}}$ gain
$\mathbf{A}_{\mathrm{REF}}$	10	10	1.285	-77.7%
	20	20	2.571	-87.2%
$\mathbf{A}_{\mathrm{PC}}$	10	2.47	0.321	-10.6%
	20	3.13	0.407	-19.4%
$\mathbf{A}_{\mathrm{MSESC}}$	10	2.20	0.287	-
	20	2.52	0.328	-

density function (p.d.f.) of a Rayleigh channel to map an equivalent SNR p.d.f. over the AWGN channel. For example, using the WiFi code with block N=1944 and R=1/2, at 10 dB on a Rayleigh channel the BER is 10^{-5}, and the same level of BER is obtained at 2.2 dB on AWGN. Since on the Rayleigh channel there is a 10% probability of SNR< 10 dB when the average SNR is 20 dB [25], there is an equal probability of SNR< 2.2 dB on the equivalent AWGN. Fig. 10 shows the probability distribution of SNR for a WiFi LDPC code [25], [29]. It is possible, at this point, to weigh the average number of iterations at each SNR point with the probability for the decoder to work under said conditions, in order to get an It_{AVG} valid for all considered SNR points. Table II shows the energy consumption and It_{AVG} obtained for the aforementioned WiFi code with It_{max}=10 and It_{max}=20. The implementation labeled as $\mathbf{A}_{\mathrm{PC}}$ was obtained from $\mathbf{A}_{\mathrm{REF}}$ by including the *parity check* ESC. $\mathbf{A}_{\mathrm{MSESC}}$ yields very good E_F figures, with a gain ranging from 77.7% to 87.2% with respect to $\mathbf{A}_{\mathrm{REF}}$, and decreasing the energy consumption of 10.6% and 19.4% compared to $\mathbf{A}_{\mathrm{PC}}$.

VI. Conclusion

In this work a novel early stopping criterion for LDPC decoding, namely MSESC, is presented. It yields very good performance in terms of average number of performed iterations with negligible BER degradation. Based on the combination of three low-complexity metrics, associated threshold expressions, and appropriate activation/deactivation control, MSESC enables efficient adaptation to various LDPC code parameters and SNR operational conditions. Besides error correction performance and reduction of the number of iterations, post-layout hardware complexity and energy consumption were evaluated and compared with state-of-the-art techniques for various system parameters. The obtained results show that the low increase in the area (1.3%) is counterbalanced by a high energy reduction (up to 70% for AWGN) w. r. t. the same decoder with no ESC, and reduction between 4% and 16% w.r.t. other ESCs. Finally, an assessment of the impact of more realistic channel conditions was proposed, illustrating energy reductions ranging from 10.6% to 87.2%.

References

[1] R. G. Gallager, "Low density parity check codes," *IRE Transactions on Information Theory*, vol. IT-8, no. 1, pp. 21–28, Jan 1962.

[2] *IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless*, IEEE Std 802.16e-2005 Std., 2006.

[3] *IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks*, IEEE Std 802.11n-2009 Std., 2009.

[4] A. Morello and V. Mignone, "DVB-S2: The second generation standard for satellite broad-band services," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 210 –227, 2006.

[5] *Quality Supervision and Quarantine, GB 20600-2006, digital terrestrial television broadcasting transmission system frame structure, channel coding and modulation*, Beijing: China Standard Press Std., 2006.

[6] *TM Synchronization and Channel Coding*, Consulative Committee for Space Data Systems (CCSDS) Std. 131.0-B-2, Aug. 2011.

[7] A. Cohen and K. Parhi, "A low-complexity hybrid LDPC code encoder for IEEE 802.3an (10GBase-T) ethernet," *IEEE Transactions on Signal Processing*, vol. 57, no. 10, pp. 4085–4094, 2009.

[8] Y. Cui, X. Peng, Z. Chen, X. Zhao, Y. Lu, D. Zhou, and S. Goto, "Ultra low power QC-LDPC decoder with high parallelism," in *Proc. of IEEE International SOC Conference*, sept. 2011, pp. 142 –145.

[9] Y.-L. Ueng, Y.-L. Wang, L.-S. Kan, C.-J. Yang, and Y.-H. Su, "Jointly designed architecture-aware LDPC convolutional codes and memory-based shuffled decoder architecture," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4387–4402, 2012.

[10] C.-S. Park, S. woon Kim, and S.-Y. Hwang, "Design of a low-area, high-throughput LDPC decoder using shared memory banks for DVB-S2," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 2, pp. 850–854, 2009.

[11] F. Naessens, B. Bougard, S. Bressinck, L. Hollevoet, P. Raghavan, L. Van der Perre, and F. Catthoor, "A unified instruction set programmable architecture for multi-standard advanced forward error correction," in *Proc. of IEEE Workshop on Signal Processing Systems*, oct. 2008, pp. 31 –36.

[12] C. Zhou, Y. Ge, X. Chen, Y. Chen, and X. Zeng, "An area-efficient LDPC decoder for multi-standard with conflict resolution," in *Proc. of IEEE International Conference on Application-Specific Systems, Architectures and Processors*, sept. 2011, pp. 105 –112.

[13] C. Condo, M. Martina, and G. Masera, "A network-on-chip-based turbo/LDPC decoder architecture," in *Proc. of IEEE Design, Automation Test in Europe Conference Exhibition*, march 2012, pp. 1525 –1530.

[14] ——, "VLSI implementation of a multi-mode turbo/LDPC decoder architecture," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 6, pp. 1441–1454, 2013.

[15] Z. Chen, X. Zhao, X. Peng, D. Zhou, and S. Goto, "An early stopping criterion for decoding LDPC codes in WiMAX and WiFi standards," in *Proc. of IEEE International Symposium on Circuits and Systems*, jun. 2010, pp. 473 –476.

[16] J. Li, X.-H. You, and J. Li, "Early stopping for LDPC decoding: convergence of mean magnitude (CMM)," *IEEE Communications Letters*, vol. 10, no. 9, pp. 667 –669, sept. 2006.

[17] T. Mohsenin, H. Shirani-Mehr, and B. Baas, "Low power LDPC decoder with efficient stopping scheme for undecodable blocks," in *Proc. of IEEE International Symposium on Circuits and Systems*, May. 2011.

[18] Z. Cai, J. Hao, and U. Sethakaset, "Efficient early stopping method for LDPC decoding based on check-node messages," in *Proc. of Asilomar Conference on Signals, Systems and Computers*, oct. 2008, pp. 466 –469.

[19] E. Amador, R. Knopp, R. Pacalet, and V. Rezard, "High throughput and low power enhancements for LDPC decoders," *International Journal on Advances in Telecommunications*, vol. 4, no. 1, aug. 2011.

[20] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. of IEEE Workshop on Signal Processing Systems*, 2004, pp. 107 – 112.

[21] V. Savin, "Self-corrected Min-Sum decoding of LDPC codes," in *Proc. of IEEE International Symposium on Information Theory*, jul. 2008, pp. 146 –150.

[22] F. Kienle and N. Wehn, "Low complexity stopping criterion for LDPC code decoders," in *Proc. of IEEE Vehicular Technology Conference*, vol. 1, 2005, pp. 606–609.

[23] M. Moher, "Decoding via cross-entropy minimization," in *Global Telecommunications Conference, 1993, including a Communications Theory Mini-Conference. Technical Program Conference Record, IEEE in Houston. GLOBECOM '93., IEEE*, 1993, pp. 809–813 vol.2.

[24] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *Information Theory, IEEE Transactions on*, vol. 42, no. 2, pp. 429–445, 1996.

[25] S. R. Saunders and A. A. Zavala, *Antennas and Propagation for wireless communication systems*. Wiley, 2007.

[26] M. Martina, G. Masera, S. Papaharalabos, P. Mathiopoulos, and F. Gioulekas, "On practical implementation and generalizations of $\max^*$ operator for turbo and LDPC decoders," *Instrumentation and Measurement, IEEE Transactions on*, vol. 61, no. 4, pp. 888–895, 2012.

[27] S. Tota, M. Casu, M. Ruo Roch, L. Rostagno, and M. Zamboni, "MEDEA: a hybrid shared-memory/message-passing multiprocessor NoC-based architecture," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, 2010, pp. 45–50.

[28] M. Imase, M.; Itoh, "A design for directed graphs with minimum diameter," *IEEE Transactions on Computers*, vol. C-32, no. 8, pp. 782–784, Aug. 1983.

[29] J. Geier. (2005) SNR cutoff recommendations. [Online]. Available: http://www.wi-fiplanet.com/tutorials/article.php/3468771