

Unequal Error Protection of memories in LDPC  
decoders

*Original*

Unequal Error Protection of memories in LDPC

decoders / Condo, Carlo; Masera, Guido; Montuschi, Paolo. - In: IEEE TRANSACTIONS ON COMPUTERS. - ISSN 0018-9340. - STAMPA. - 64:10(2015), pp. 2981-2993. [10.1109/TC.2014.2378271]

*Availability:*

This version is available at: 11583/2577757 since: 2015-10-08T07:59:40Z

*Publisher:*

IEEE - INST ELECTRICAL ELECTRONICS ENGINEERS INC

*Published*

DOI:10.1109/TC.2014.2378271

*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)

# Unequal Error Protection of memories in LDPC decoders

Carlo Condo, Guido Masera, *Senior Member IEEE*, Paolo Montuschi, *Fellow IEEE*

**Abstract**—Memories are one of the most critical components of many systems: due to exposure to energetic particles, fabrication defects and aging they are subject to various kinds of permanent and transient errors. In this scenario, Unequal Error Protection (UEP) techniques have been proposed in the past to encode stored information, allowing to detect and possibly recover from errors during load operations, while offering different levels of protection to partitions of codewords according to their importance. Low-Density Parity-Check (LDPC) codes are used in many communication standards to encode the transmitted information: at reception, LDPC decoders heavily rely on memories to store and correct the received information. To ensure efficient and reliable decoding of information, the need to protect the memories used in LDPC decoders is of primary importance. In this paper we present a study on how to efficiently design UEP techniques for LDPC decoder memories. The devised UEP method is divided in four adjustable levels, each one offering a different degree of protection. The full UEP, along with simplified versions, has been implemented within an existing decoder and its area occupation and power consumption evaluated. Comparison with the literature on the subject shows an unmatched level of protection from errors at a small complexity and energy cost.

**Index Terms**—Unequal Error Protection, LDPC, decoder

## I. INTRODUCTION

CMOS technology scaling has reached deep sub-micron level: such a degree of integration has been shown to rise reliability problems that cannot be overlooked. In fact, the achieved degree of control on the fabrication process of the latest nodes is much lower than before [1]: this leads to process variations and less reliable components. Moreover, deeply scaled integrated circuits are very sensitive to external influences even in presence of ideal components. Consequently, depending on the application, it is necessary to apply error protection techniques to memories and logic.

Memories are particularly critical devices, that are subject to various types of faults. Soft errors are faults in the stored data that are not caused by permanent hardware damage. They can be induced by different kinds of energetic particles hitting memory cells. The subsequent voltage rise can cause bit flipping and consequent faulty data [2]. The downscaling of both cell dimensions and operating voltage has increased the sensitivity of memories to such events, since a lower amount of energy is required to change logic states [1], even affecting multiple cells at the same time. Opposite to soft-errors, permanent or hard errors are caused by hardware damages, that can be the result of fabrication defects or aging: stuck-at bits, i.e. bits which value cannot be modified, are a common permanent error.

Low Density Parity Check (LDPC) codes [3] are block error correcting codes employed in a variety of communication

standards, like WiMAX, WiFi, DVB-S2, CMMB and DTMB. LDPC decoders rely on decoding algorithms that iteratively update bit error probabilities: these are stored, in between iterations, in dedicated memories, whose implementation requires the largest amount of area and at the same time accounting for most of the power consumption [4], [5]. The quantities stored in memories during LDPC decoding are soft amounts (usually represented as Logarithmic Likelihood Ratios or LLRs) that express both the value of codeword bits (sign) and their reliability (magnitude). The values of these parameters evolve as the decoding proceeds, ideally towards correct decoded bits. A property of LDPC decoders is to be moderately resistant to hardware errors: this characteristic has been exploited in past error fault tolerant designs [6]. On the other hand, since in LDPC codes each LLR is correlated with many others, an error during a read operation can be propagated, possibly compromising the whole decoding process.

This paper focuses on providing error resilience to LDPC decoders, to ensure correct functionality also in presence of permanent and transient memory error conditions under which current decoders cannot work. Our approach starts from the analysis of the impact of memory errors on the decoding performance, in order to determine the specific effect of each bit of an LLR on the decoding process. To be able to sustain high soft error probabilities and large numbers of permanent errors, different degrees of protection are proposed, leading to the development of an Unequal Error Protection (UEP) methodology [7]. The idea behind UEPs is that each bit is protected proportionally to its relevance, i.e. the higher the importance the higher the degree of protection. UEP techniques have been designed in the past targeting applications like image transmission and storage [8]–[11] and are usually based on error correcting codes. However, LDPC decoders are characterized by deep, narrow memories (usually  $< 8$  bits) which have convoluted access address patterns. It turns out that in these conditions, protection techniques based on information encoding have unsatisfying performance or cause very large overheads, since the access patterns do not allow LLRs to be grouped together. In this paper we present a special purpose UEP scheme that is, as we will see throughout the paper, particularly suitable for LDPC decoders and applications that use narrow memories with frequent accesses and complex address patterns.

The rest of the paper is organized as follows. Section II introduces LDPC decoding. Then, Section III provides an overview on previous works on UEP and error protection in LDPC decoders and Section IV analyzes the impact of errors on the performance of a generic decoder. Subsequently, Section V describes the proposed method. The UEP's im-

plementation is presented in Section VI, while performance results are reported in Section VII. Comparison with the state of the art is performed in Section VIII and conclusions are drawn in Section IX. **Finally, in the Appendix (separate file) interested readers can find the zoomed version of all the graphs shown in the paper, for an even better visual understanding of the details of our test.**

## II. LDPC DECODING

LDPC codes are characterized by a binary parity check matrix  $\mathbf{H}$  [3] with  $M$  rows and  $N$  columns. The  $\mathbf{H}$  matrix is sparse and identifies the set of valid codewords related to a particular LDPC code; all codewords  $x$  satisfying  $\mathbf{H} \cdot x' = 0$ , where  $x'$  is the transposed of  $x$ , are considered valid.

LDPC decoding is usually handled via Belief Propagation (BP) algorithm or one of its approximations, following one of two scheduling approaches: the two-phase scheduling and the layered scheduling [12]. In the following we will focus on the layered scheduling technique, which has been shown to be more performing, nearly doubling the convergence speed of the decoding process with respect to two-phase scheduling. In layered decoders the  $\mathbf{H}$  matrix is subdivided into sets of consecutive, non-communicating parity-check constraints, called layers: these can be decoded in sequence, with the extrinsic information being propagated from one layer to the following ones [12].

Let us denote with  $\lambda[c]$  the LLR of symbol  $c$ ; the bit LLR  $\lambda_k[c]$  is initialized, for column  $k$  in  $\mathbf{H}$ , to the corresponding received soft value. The sign of an LLR is associated to the value of an LDPC codeword bit, while its magnitude is a measure of how reliable the information is. A low magnitude expresses little confidence on the current bit value: its weight on the overall decoding is low, and it is more likely to be influenced by other, more reliable LLRs.

The BP algorithm involves some computationally intensive operations, including the calculation of a hyperbolic tangent [13]. Out of the several approximations present in the literature, we have considered the Self-Corrected-Min-Sum (SCMS) [14] as it combines easiness of implementation with negligible BER performance losses. For all parity-check constraints  $l$  in a given layer, and for every iteration  $i$  up to a maximum of  $I_{max}$  iterations, the following operations are executed:

$$Q_{lk}^i[c] = \lambda_k^{old}[c] - R_{lk}^{i-1} \quad (1)$$

$$\delta_{lk} = \prod_{n \in N(l), n \neq k} \text{sgn}(Q_{ln}[c]) \quad (2)$$

$$R_{lk}^i \approx -\delta_{lk} \cdot \min_{n \in N(l), n \neq k} \{|Q_{ln}[c]|\}, \quad (3)$$

$$\lambda_k^{new}[c] = Q_{lk}^i[c] + R_{lk}^i \quad (4)$$

where  $\lambda_k^{old}[c]$  is the extrinsic information coming from one of the previous layers and updated in (4), before being passed to one of the subsequent layers. The term  $R_{lk}^{i-1}$  in (1) is initialized to zero in the first iteration; the same term is then updated in (3), obtaining  $R_{lk}^i$ , and stored until the next iteration. In (2)  $N(l)$  is the set of bit indexes that contribute in the  $l^{th}$  parity check. At the end of each iteration, by observing

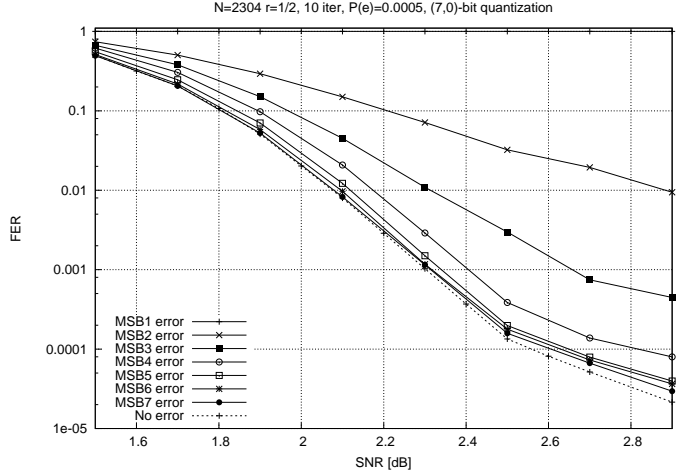


Figure 1. FER - errors on different MSBs

the sign of all  $\lambda_k[c]$ , the corrected version of  $x$  is obtained. Please observe that while  $R_{lk}$  and  $Q_{lk}[c]$  are updated only once per iteration, and are thus endowed with the iteration indicator  $i$ ,  $\lambda_k[c]$  is updated multiple times during each iteration, and the apexes ‘old’ and ‘new’ are consequently used to differentiate the values before and after each update. The SCMS approximation adds a dynamic correction factor to (1). When  $Q_{lk}^i[c]$  is computed at the  $i^{th}$  iteration, it is compared with  $Q_{lk}^{i-1}[c]$ . If their signs differ, then  $Q_{lk}^i[c]$  is substituted with zero for the current iteration, thus inducing a conservative behavior when a sign changes.

## III. PREVIOUS WORK

Several memory-protection techniques and algorithms have been presented in the specialized literature over the years. Certainly, one of the most popular frameworks is the encoding of codewords and pages before storage in order to guarantee error detection and recovery at load time [15]. Many types of codes have been experimented with, from simple Hamming and BCH [16] codes to more complex turbo codes and LDPC codes themselves [17]–[19]. The concept of unequal error protection of memories has been proposed for the first time in [7]: codewords are subdivided in slots, to which different degrees of protection are applied. From a practical point of view, it has then been studied effectively for wireless transmissions and storage of images, where a certain degree of unreliability can be tolerated [9], [20], [21]. For example, in [22] UEP is applied to image transmission on mobile channels, and a UEP SRAM coding scheme for multimedia applications has been proposed in [10].

To the best of our knowledge only a few works in literature deal with resilient LDPC decoders. Stochastic decoders [23], [24] implement an LDPC decoding approach based on an alternative representation of the bit probabilities. While inherently robust to hardware errors, in [25] statistical error compensation is used to overcome the severe performance loss brought by voltage overscaling. The work in [26] builds and updates a list of cells that are probably faulty. This allows

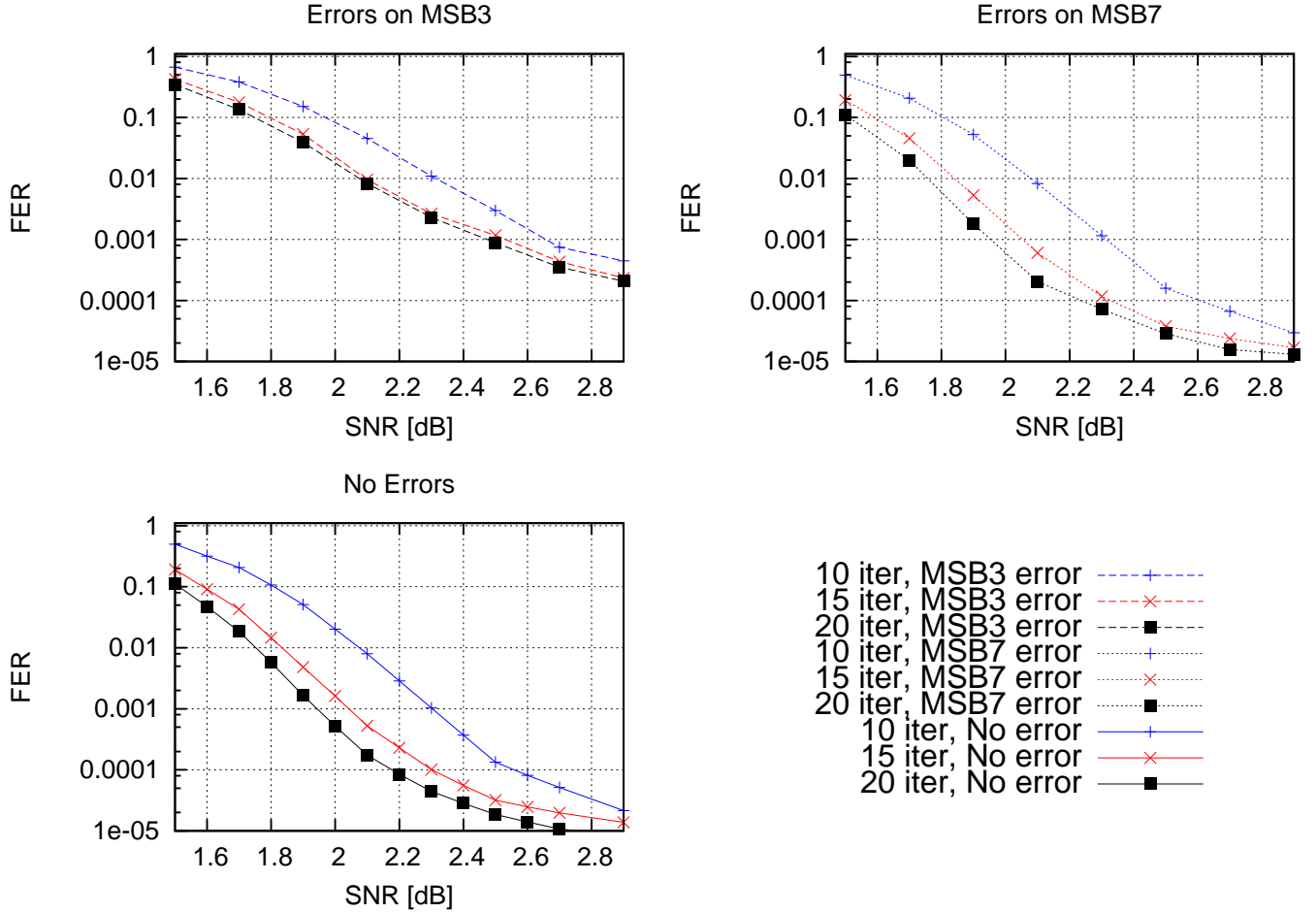


Figure 2. FER - errors on different MSBs and variation of  $I_{max}$  for  $N=2304$ ,  $r=1/2$ ,  $P(e)=0.0005$ ,  $(7,0)$ -bit quantization

parity check results to be cross-referenced and the faulty cells to be replaced: at the same time, relative computation errors are managed by ad-hoc bit flipping. The work presented in [6] applies separate protection techniques to the different functional blocks of an existing LDPC decoder architecture, according to their level of exposure to failures and importance for the correct operations of the system. Memories, as some of the most critical modules, are protected according to their role in the decoding process by doubling or tripling of the MSB and dynamic corrections.

#### IV. ERROR ANALYSIS

We now study the impact of the errors on the performance of a generic LDPC decoder.

Almost all practical implementations of LDPC decoders make use of memories to store LLRs between iterations or, in case of multi-core decoders, to exchange data information between processing elements. The number of memory read and write accesses can be extremely high both when we consider the total lifetime of the decoder and each single decoding process. For example, WiMAX LDPC codes require from 7200 to 32800 memory accesses between read and write operations for a single iteration, but this number can exceed

one million of accesses in case large codes are employed, like in DVB-S2.

The two quantities usually stored in LDPC decoder memories are  $\lambda_k[c]$  and  $R_{lk}$ , that are read in (1) and updated in (3) and (4). Depending on the involved quantities, errors in read and write operations impact differently on the decoding process. For example, a wrong  $Q_{lk}^i[c]$  (1) does not necessarily result in an incorrect  $R_{lk}^i$ . As (3) shows, only the first and second minimum among the  $Q_{lk}^i[c]$  involved in the check node computation are considered for the selection of  $R_{lk}^i$ : consequently, the error might not be propagated to other  $\lambda_k^{new}[c]$ . Moreover, since the LDPC decoding process relies on soft information, the performance degradation caused by a flipped bit in  $\lambda_k^{old}[c]$  or  $R_{lk}^{i-1}$  depends on the position of the error, as shown later in this section. In the following, for sake of clarity but with no loss of generality, we will refer to a practical case study. Assuming a quantization on  $b$  bits, let us call MSB1 the Most Significant Bit, and MSBb the Least Significant Bit. As a case of study, both  $\lambda_k^{old}[c]$  and  $R_{lk}^{i-1}$  have been quantized with  $b = 7$  (7 bits for the integer part and 0 for the fractional part) [27], [28].

In the following subsections we move to analyze the effect of errors on different bits on the metrics stored in LDPC decoder memories, and how they influence the decoding per-

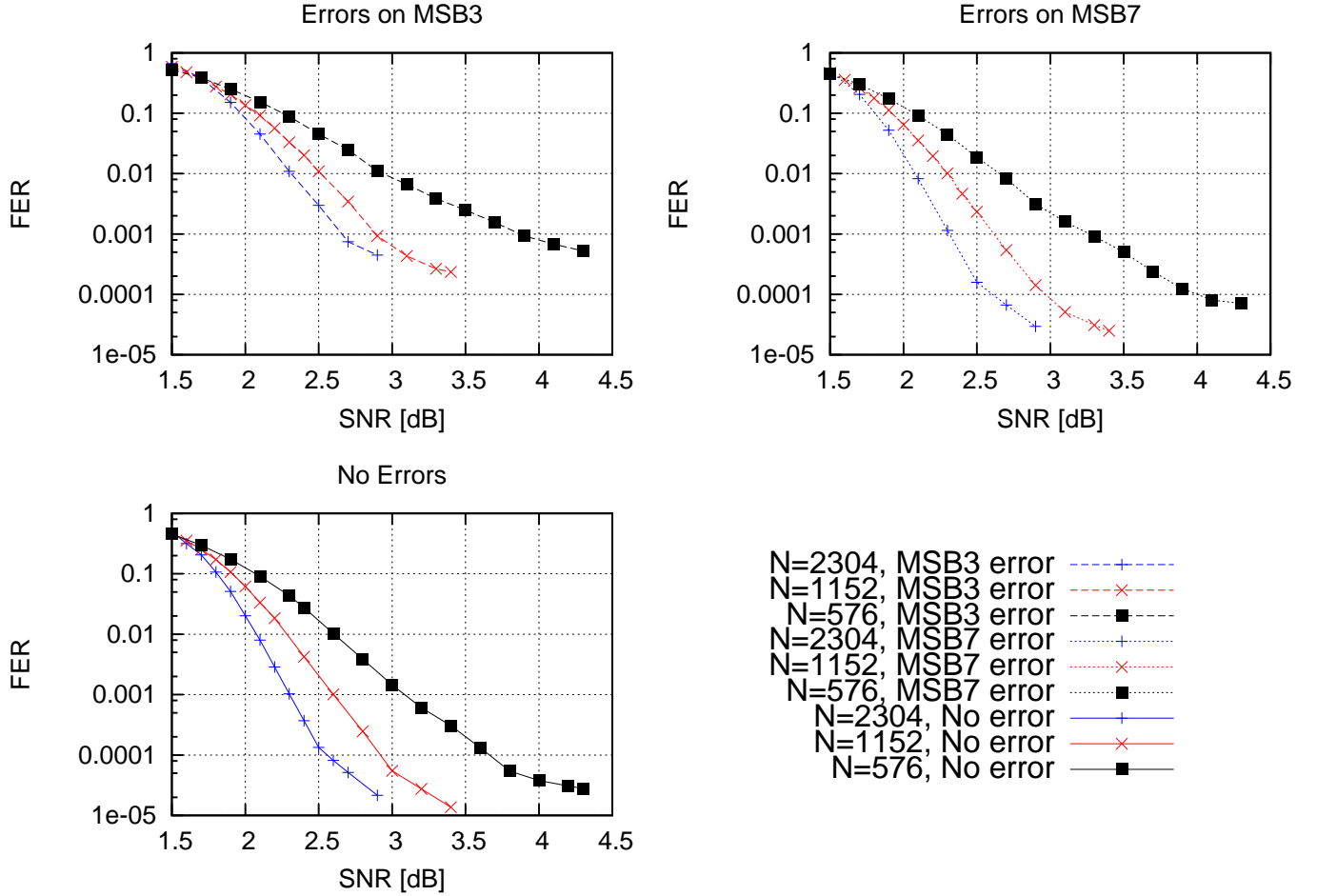


Figure 3. FER - errors on different MSBs and variation of code size for  $I_{max}=10$ ,  $r=1/2$ ,  $P(e)=0.0005$ , (7,0)-bit quantization

formance at the variation of different parameters. In particular, we plot the Frame Error Rate (FER) while varying the MSB of  $\lambda_k^{old}[c]$  and  $R_{lk}^{i-1}$  on which the error was injected, alone and in combination with the following cases, where we are varying:

- the maximum number of allowed iterations  $I_{max}$ ;
- the code rate  $r$ ;
- the code size  $N$ ;
- the quantization;
- the decoding algorithm.

Figures 1 to 5 have been obtained through simulations on a bit-accurate software model. **To better highlight the characteristics of the results of our tests, in Fig. 2-4 we have provided separate sub-figures for the different error cases.**

The curves in Fig. 1 show the Frame Error Rate (FER) for the WiMAX code of size  $N = 2304$  and rate  $r = 1/2$  in case of transmission on an Additive White Gaussian Noise (AWGN) channel. The decoding has been performed with the SCMS approximation. The “no error” curve plots the FER under ideal hardware conditions, i.e. without any error in the read and write operations. The other curves have been obtained by allowing errors on a single bit of  $\lambda_k^{old}[c]$  and  $R_{lk}^{i-1}$  with a probability equal to  $P(e)=0.0005$ . Errors injected on the sign bit (MSB1) are disruptive at every Signal-to-Noise Ratio value

(SNR), and cause unrecoverable errors. The criticalness of the sign bit in LDPC decoding is well documented and protection has been employed in the past [6]. The MSB2 and MSB3 error curves show severe degradation with respect to the ideal one. Both MSB2 and MSB3 account for a considerable percentage of the total dynamic, and can cause strong variations in both  $R_{lk}^i$  and  $\lambda_k^{new}[c]$ : particularly disruptive occurrences can easily lead to sign bit flips. Less critical is the impact of MSB4 errors, while errors injected in MSB5, MSB6 and MSB7 result in similarly small performance degradations.

#### A. Variation of $I_{max}$

The bit-per-bit error analysis has been extended to different choices of the maximum number of iterations  $I_{max}$ . The increased correction capability brought by additional iterations can in fact be dampened by the introduction of new errors. Fig. 2 plots the FER for three different error bits and three  $I_{max}$  values. While the effect of errors is almost the same with  $I_{max}=10$  and  $I_{max}=15$ , the degradation caused by erroneous bits is more consistent when  $I_{max}=20$ : in fact, we observe the existence of a larger gap between the “no error” curve and the MSB7 curve with respect to the two other cases, while the MSB3 curve is proportionally shifted.

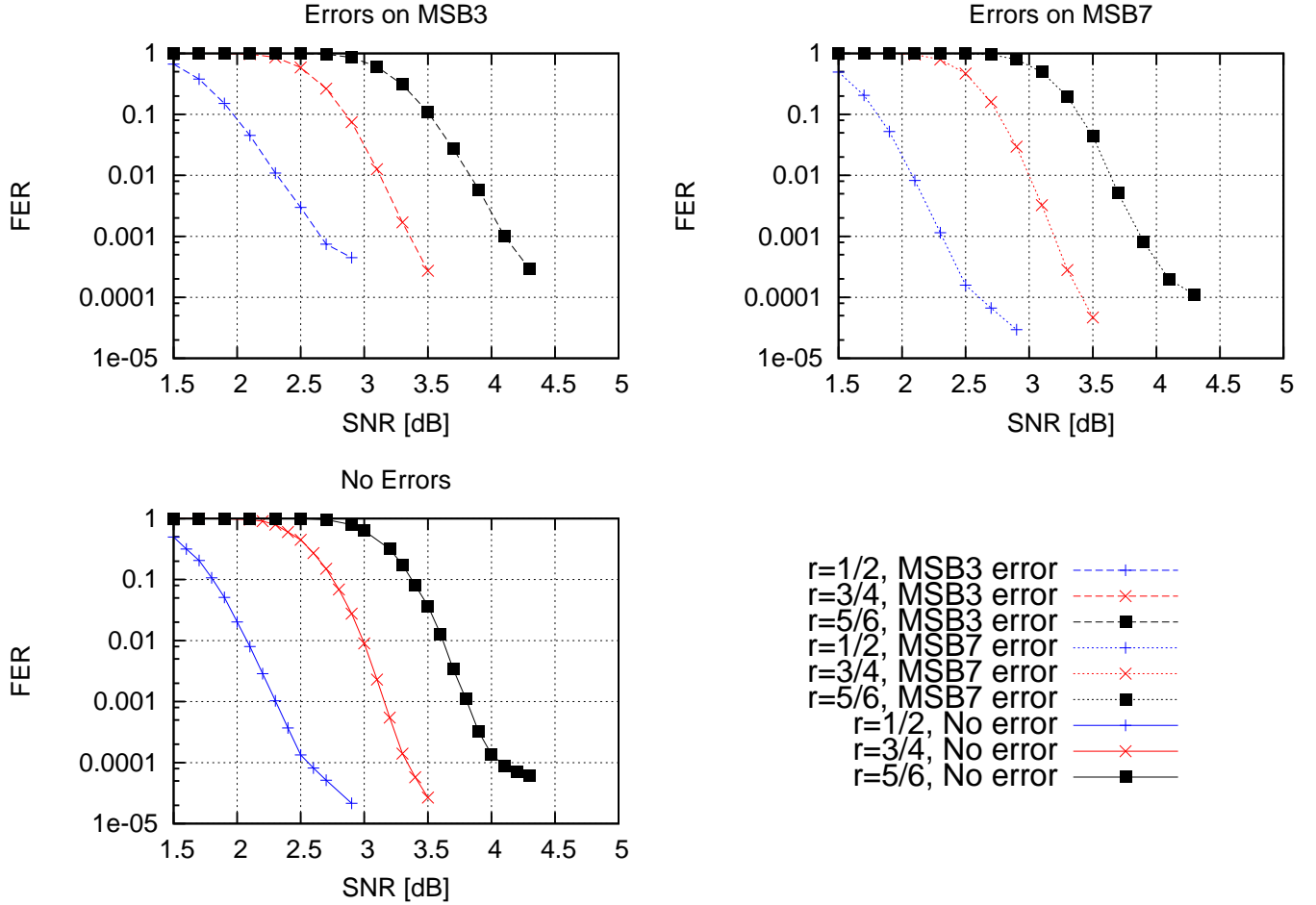


Figure 4. FER - errors on different MSBs and variation of code rate for  $I_{max}=10$ ,  $N=2304$ ,  $P(e)=0.0005$ , (7,0)-bit quantization

### B. Variation of code rate $r$ and size $N$

Code rate and code size play a major role in the overall impact of errors. Fig. 3 considers three codes with different size  $N$  and the same code rate  $r$ . The error injection probability has been kept constant at  $P(e)=0.0005$  for all the codes: this means that codes with smaller  $N$  will lead to a lower number of wrong bits per frame. However, it can be noticed how smaller codes are more sensitive to faults, regardless of the fewer injected errors. Taking as a reference point  $FER=10^{-4}$ , MSB7 errors cause a loss of 0.05 dB to the  $N=2304$  code, while 3.5 dB are lost with  $N=576$ .

Fig. 4 is complementary to Fig. 3, with fixed code size and varying rate. In the structure of WiMAX LDPC codes, increasing the code rate, the total number of  $\lambda_k^{old}[c]$  and  $R_{lk}^{i-1}$  is kept almost constant, since the rows of the  $\mathbf{H}$  matrix increase their weight. However, SCMS can often mask errors on  $R_{lk}^{i-1}$ : increments in code rate consequently lead to a lower number of potentially erroneous bits per frame. Fig. 4 shows that rate and error injection variations do not scale proportionally, as it has been already observed with changes in code size: high rate codes are more sensitive to faults. The performance loss caused by MSB7 errors at  $FER=10^{-4}$  is 0.05 dB for code rate 1/2, and 2.5 dB for code rate 5/6.

### C. Variation of quantization and decoding algorithm

Fault tolerance of LDPC decoding has been analyzed in terms of quantization too. The results meet our expectations as errors affect the decoding performance proportionally to the weight of the erroneous bit on the overall dynamic range, regardless of the quantization. Obtained FER curves are similar to what has been shown in Fig. 1.

A final practical test has been carried out by comparing different decoding algorithms. The decoding performance of the SCMS approximation is intrinsically more resistant to hardware errors than more common approximations of the BP algorithm like the Normalized-Min-Sum (NMS) [29]. Fig. 5 shows the FER degradation due to  $P(e)=0.0005$  for a code decoded with both NMS and SCMS, and the inherent resilience of SCMS can be easily noted. This is due to the puncturing of  $Q_{lk}^i[c]$  in presence of a sign change, that introduces an additional barrier to the propagation of disruptive errors.

The previous analysis has clearly shown that errors on MSBs always cause destructive effects, and therefore this issue must be tackled accordingly. On the other hand, even if errors on LSBs have a minor impact on the decoding performance, they cannot be ignored either in the more critical cases, such as in presence of high-rate codes and small-size codes.

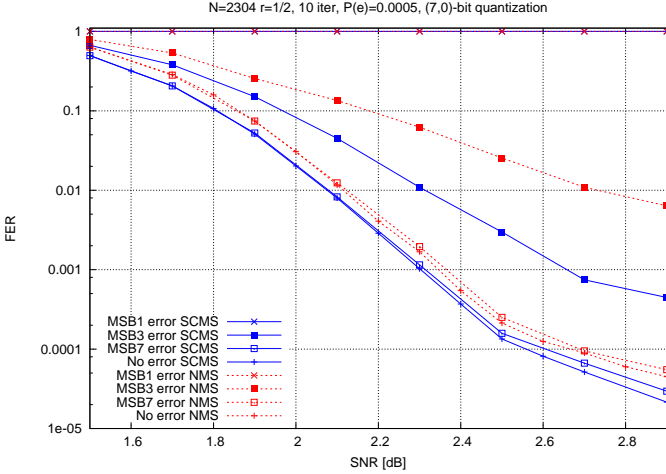


Figure 5. FER - errors on different MSBs and variation of decoding algorithm

## V. UNEQUAL ERROR PROTECTION

The analysis on the impact of the different memory errors carried out in Section IV highlighted that not all errors on the LLR bits have the same influence on the FER of LDPC decoders. This is an important result, as it identifies a characteristic of LDPC decoders that can be used to increase the reliability of the decoding process. In particular, based on the study of Section IV, we observe that it is possible to apply distinct error protection techniques to each bit or group of bits stored in memories depending on their importance and influence on the FER. The choice on the number and type of error protection techniques is subject to a suitable tradeoff. On one side, it should be guaranteed that the UEP is granted sufficient granularity to effectively act upon errors with different impacts on the decoding capability. On the other side, as this number should be kept small to save area, execution time and complexity of the decoder, bits with similar significance should be protected with the same technique. In this paper, after having analyzed different tradeoff alternatives, we have opted for a UEP subdivided into four levels of possible error protection.

### A. Level 1 - towards full recovery

The highest level of protection is applied to bits which reliability is mandatory for a correct decoding, i.e. the sign bit and possibly the magnitude MSBs. In fact, as we have seen in Section II, LDPC decoders base their hard decision on bits (i.e., the selection if a codeword bit is zero or one) on the sign of their respective LLRs. Errors on sign bits and on bits representing a large part of the total dynamic will consequently have catastrophic effects on the decoding; sign changes and sudden increments or decrements in LLRs may cause an avalanche of metrics to evolve towards misleading directions. To provide a high level of reliability and recovery, our choice has been that bits falling within the Level 1 protection level are tripled during write operations: at load time, a majority voter selects the most probable output.

### B. Level 2 - tentative recovery from critical errors

An extensive simulation campaign has been performed to observe the characteristics of  $\lambda_k^{old}[c]$  and  $\lambda_k^{new}[c]$  that are most sensitive to memory errors and that can lead to unsuccessful decoding. A very large percentage of cases in which a wrong bit in  $\lambda_k^{old}[c]$  leads to an incorrect  $\lambda_k^{new}[c]$  (4) is characterized by recognizable sequences of bits within the representation of  $\lambda_k^{new}[c]$ . It is possible to observe the occurrence of these bit patterns in case of errors and to determine their impact on the overall decoding process. We have chosen to add a parity bit to the Level 2 bits, and in case of discrepancy during load operations the pattern recognition system is activated. If  $\lambda_k^{new}[c]$  matches the critical bit pattern, recovery is possible by observing how  $\lambda_k^{new}[c]$  varies with changes of the Level 2 bits in  $\lambda_k^{old}[c]$ . The distinctive bit pattern is dependent on the total quantization of the LLRs and on the position of the wrong bit, while the Level 2 bits must be chosen with care to obtain the maximum effectiveness: thus, every case must be analyzed separately.

Level 2 is not able to give the same level of protection as Level 1, but gives a very good percentage of identification and recovery of errors that have been observed to be the main cause for LDPC decoder performance loss.

### C. Level 3 - bounding of error impact

As we have said in Section II, the magnitude of an LLR is a measure the information reliability. We have chosen to apply the same concept to the third level of protection, designed for bits of medium-to-low significance. A parity bit is added to the protected bits during write operations. When the LLR is loaded, parity is recomputed and in case of discrepancy the contribution of all the bits falling within Level 3 is nulled or reduced. With a two's complement representation bit puncturing can require bits to be flipped among Level 2 and Level 1 bits as well: to avoid complex operations, depending on Level 2 and Level 1 values, a partial puncturing can be employed.

Level 3 protection does not allow to recover from errors, but reduces their impact by decreasing the LLR magnitude, that in turn induces a conservative behavior in the decoder. This method can not be applied to bits expressing large percentages of the total dynamic, since the LLR magnitude change would be too large and cause errors.

### D. Level 4 - no protection zone

As shown in Section IV, errors on the least significant bits seldom affect the overall decoding performance. For Level 4, our choice for this set of low-importance bits has been to leave them unprotected, as possibly this will not incur in any impacting degradation.

### E. UEP - full design

The partition of memory bits among the four levels of the proposed UEP has been carried out through extensive simulations. We have considered an LLR quantization with  $b = 7$  and  $I_{max} = 10$ , a common enough choice that allows



good decoding performance without excessive overheads [30], [31]. We have then simulated a large set of codes with a wide range of possible combinations of UEP levels. The design choices reported here were selected among those simulated for their very good error protection performances (shown in detail in Section VII), granting good results regardless of strong variations in code size and rate.

In our scheme, Level 1 protection provides the highest degree of reliability, but requires two additional bits for each protected bit: therefore, its usage should be limited to the most critical sets of bits only. We are well aware, thanks to previous works like [6], that the correctness of the sign bit is the minimum and necessary requirement for a decoder to correctly operate. For this reason, Level 1 protection has been applied on both  $R_{lk}$  and  $\lambda_k[c]$  sign bits, resulting in a 28.6% increase in memory bits (i.e 7 bits + 2 bits). It has also been observed that errors on  $R_{lk}$  are less critical than those on  $\lambda_k[c]$  and less prone to propagating, thanks to the local nature of  $R_{lk}$  and the masking capabilities of the min-sum algorithm already addressed in Section IV. Consequently, Level 1 is the only UEP level applied to  $R_{lk}$  values, while the complete UEP is applied to  $\lambda_k[c]$  values.

As mentioned in Section V-B, the pattern recognition and error recovery involved in Level 2 protection must be evaluated according to the LLR quantization and to the number and position of bits assigned to Level 2. With  $b = 7$  and Level 1 protecting MSB1, Level 2 has been chosen to include MSB2 and MSB3. In fact, the set of simulations that allowed us to select the number and type of UEP levels has revealed that the precision of the identification of critical errors through output bit patterns shows a decreasing trend when we increase the number of protected bits, while the complexity of the error recovery system increases. This means that the smaller the number of bits at Level 2, the more accurate the identification of errors based on pattern analysis, and the simpler the error correction. However, as shown later in this section, a minimum of two bits must be included in Level 2 to be able to recover from errors. Let us define as  $\lambda_k^{new}[c]^C$  the correct result of (4) (i.e. the one obtained when no errors are present in Level 2 of  $\lambda_k^{old}[c]$ ), and  $\lambda_k^{new}[c]^I$  the incorrect one (obtained when an error is detected among the Level 2 bits of  $\lambda_k^{old}[c]$ ). The simulation campaign performed for Level 2 revealed that regardless of the channel conditions, the majority of cases in which errors on MSB2 or MSB3 result in uncorrectable codewords are characterized by  $\lambda_k^{new}[c]^C$  and  $\lambda_k^{new}[c]^I$  with small magnitude and opposite signs, meaning that

$$\text{MSB1} = \text{MSB2} = \text{MSB3} \quad (5)$$

for both  $\lambda_k^{new}[c]^C$  and  $\lambda_k^{new}[c]^I$ , whereas

$$\text{MSB1}(\lambda_k^{new}[c]^C) \neq \text{MSB1}(\lambda_k^{new}[c]^I) \quad (6)$$

where  $k$  is the same also for the erroneous  $\lambda_k^{old}[c]$  in (1). This pattern is observed either in case a single error is introduced in MSB2 or MSB3 of  $\lambda_k^{old}[c]$ , or in case both MSB2 and MSB3 are incorrect: this characteristic is exploited to recover from the error.

- 1) A parity bit considering MSB2 and MSB3 is added to  $\lambda_k[c]$  at storage time.

- 2) When  $\lambda_k^{old}[c]$  is loaded from the memory, MSB2-3 are XORed: if the result is different from the parity bit, an error alert is raised.
- 3) In case of error, three different versions of  $\lambda_k^{old}[c]$  are produced:  $\lambda_k^{old}[c]_1$  is the one read from the memory, while in  $\lambda_k^{old}[c]_2$  and  $\lambda_k^{old}[c]_3$  the MSB2 and MSB3 are respectively flipped. Two of them are wrong (one has a single erroneous bit, the other has two incorrect bits) and one is correct, but it is not possible to determine which is which at this stage.
- 4) Eq. (1)-(4) are evaluated with  $\lambda_k^{old}[c]_1$ ,  $\lambda_k^{old}[c]_2$  and  $\lambda_k^{old}[c]_3$  separately. Three sets of results are produced, containing  $\lambda_k^{new}[c]_1$ ,  $\lambda_k^{new}[c]_2$  and  $\lambda_k^{new}[c]_3$  respectively, with the awareness that two of them are  $\lambda_k^{new}[c]^I$  and one is  $\lambda_k^{new}[c]^C$ .
- 5) If at least one among  $\lambda_k^{new}[c]_1$ ,  $\lambda_k^{new}[c]_2$  and  $\lambda_k^{new}[c]_3$  does not follow (5), then there is a high probability that the error is not critical or that the wrong bit was the parity bit: in this case the set containing  $\lambda_k^{new}[c]_1$  is selected.
- 6) If  $\lambda_k^{new}[c]_1$ ,  $\lambda_k^{new}[c]_2$  and  $\lambda_k^{new}[c]_3$  follow (5) then two of them ( $\lambda_k^{new}[c]^I$ ) will have the same sign and the other ( $\lambda_k^{new}[c]^C$ ) will have an opposite sign (6). The set of results containing the  $\lambda_k^{new}[c]$  with the discordant MSB1 is the correct one, and is consequently selected.

This proposed technique can handle a single Level 2 memory error in every parity check computation, that usually involves between five and a few tens of LLRs depending on the LDPC code. Even though the probability of two Level 2 errors within the LLRs under consideration is very low, it can be further reduced by protecting memories against burst errors as we will describe later in this section. Together with Level 1, the additional cost in memory requirements increases to 42.9%.

To bind the impact of errors on the least significant bits of the representation of  $\lambda_k[c]$ , the remaining four bits have been divided between Level 3 and Level 4. Our choice has been to consider MSB4-5 for error protection and include them in Level 3, while not to provide any protection for the two least significant bits MSB6-7 by leaving them in Level 4. With this choice, we do not mean that they do not contribute to the decoding, but, as we have observed in our tests, occasional error events on these bits do not affect the overall performance. The additional memory cost of the complete UEP is 57.1%, i.e. the same as applying Level 1 to MSB1-2, but in this case shielding from errors five bits instead of two: the impact on the decoder architecture and the additional logic required are discussed in Section VI.

#### F. Remarks on Burst Errors

With the level of current integration the problem of burst or multi-cell errors has gathered increased interest [17], [32]. All levels of the proposed UEP are able to detect and possibly recover from single-bit errors, and do not take in account burst errors. However, it is possible to greatly limit the impact of burst errors by scrambling the bits of LLRs before storage, and rearranging them at load time. Scrambling is a technique widely used in communications and data storage, where the



MSB

LSB

Level 1	Level 2	Level 3	Level 4	Level 3	Level 1	Level 2	Level 4	Level 3	Level 2	Level 1
---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------

Figure 6. Example of LLR rearranged bits for burst error protection

probability of burst errors is larger than the probability of single errors, as it allows, under certain conditions, to avoid long error correcting codes to recover from burst errors [32]. For our design an example of possible scrambling is shown in Fig. 6. By interleaving the bits belonging to the same level with those from other levels, multiple errors are spread over the different protection techniques and can still be handled. Every bit is placed as far as possible from those belonging to the same level, with priority being Level 1 > Level 2 > Level 3 > Level 4. According to this priority sorting, Level 1 can sustain up to 9-bit burst errors depending on the involved bits, while it guarantees immunity to burst errors as large as 5-bit. Level 2 has a maximum resiliency of 8-bit burst errors, and guaranteed immunity to 3-bit bursts, while Level 3 sustains up to 6-bit bursts and guarantees protection from 2-bit bursts. Burst errors can affect various LLRs concurrently: to avoid multiple wrong LLRs within the same parity check computation (1), it is sufficient to store the various  $\lambda_k[c]$  sorted from the MSB of the LDPC frame to the LSB. In fact, the sparse structure of the parity check matrix acts as an interleaver and does not require loading consecutive LLRs.

#### G. Additional schemes

As will be shown in Section VII, the described UEP is able to guarantee a very high degree of error protection. However, such a high degree of confidence is not always necessary. As a matter of example, here we have reported two cases where, assuming that a lower protection is sufficient, a simplified version of the UEP is designed. This is achieved by including only some of the previous protection levels and has the advantage to guarantee the required lower level of protection by using a reduced overhead. Table I summarizes the previously designed UEP case of study together with the two new ones, with details being given on the protection of  $\lambda_k[c]$ . In all three cases,  $R_{lk}$  is protected with Level 1 on MSB1. UEP<sub>full</sub> refers to the case of study detailed previously in this section. The second scheme UEP<sub>sim1</sub> applies Level 2 protection to MSB1 and MSB2, while MSB3-7 are left in Level 4. Finally, in UEP<sub>sim2</sub>, MSB1 falls within Level 1, while all the other bits are in Level 4. The performance of UEP<sub>sim1</sub> and UEP<sub>sim2</sub> is evaluated along with that of UEP<sub>full</sub> in Section VII.

### VI. HARDWARE IMPLEMENTATION ARCHITECTURE AND EVALUATION

#### A. Architecture

The logic flow of the operations needed by UEP in the conditions portrayed in Section V for UEP<sub>full</sub> is shown in Fig.

Table I  
UEP ALTERNATIVE SCHEMES

	Level 1	Level 2	Level 3	Level 4
UEP <sub>full</sub>	MSB1	MSB2-3	MSB4-5	MSB6-7
UEP <sub>sim1</sub>	–	MSB1-2	–	MSB3-7
UEP <sub>sim2</sub>	MSB1	–	–	MSB2-7

7; UEP<sub>sim1</sub> and UEP<sub>sim2</sub> can be derived from it considering only the employed levels. After reading the LLR from the memory, the tripled MSB1 value is obtained through majority voting. The parity of Level 3 bits is then computed and compared to the parity bit from memory. In case of mismatch, total or partial puncturing is applied to obtain MSB4-5. Three datapaths are necessary to implement the Level 2 operations: the parity comparison is performed on the Level 2 bits, and if an error occurred, each datapath receives a different version of the LLR ( $\lambda_k[c]_1$ ,  $\lambda_k[c]_2$  and  $\lambda_k[c]_3$ ). The outputs are checked according to (5)-(6) to identify whether the error is critical or not and to select the correct  $\lambda_k^{new}[c]$ . However, if no error is detected by the parity comparison, all datapaths work with the same set of data. The three outputs can be used to recover from possible errors in the datapath logic, and the value of every LLR is decided with a majority voter.

Based on the logic flow of Fig. 7, the hardware structure depicted in Fig. 8 has been designed. The light gray blocks identify functions pertaining to the different UEP levels, whereas LLR bits subdivided in the different levels are shown at the top of the picture. The L1 bit and its tripled versions TR enter the Level 1 block, that implements a simple majority voting to decide on MSB1. The output of Level 1 block is used, along with the rest of the LLR as read from memory, within the Level 3 block to decide on what kind of puncturing to apply, if any. The OR operation is applied to bits belonging to the same level to identify the relative positions of 1s within the LLR. MSB4-5 are either cleared or set, depending on MSB1, the ORed bits, and the parity bit comparison. The dark gray Datapath blocks implement the serial datapath detailed in [30]. The computed MSB1 and MSB4-5 are given as inputs to the standard Datapath and to the Level 2 block, that comprises the two additional Datapath blocks. If no discrepancy is detected in the parity comparison among L2 bits, the three datapaths run the same calculations: at the output, majority voters add another layer of reliability to the system against errors within the logic. Two inverters flip MSB2-3, that are fed to the additional datapaths in case the parity comparison presents a mismatch: the output majority voters are cut off, and the pattern recognition system

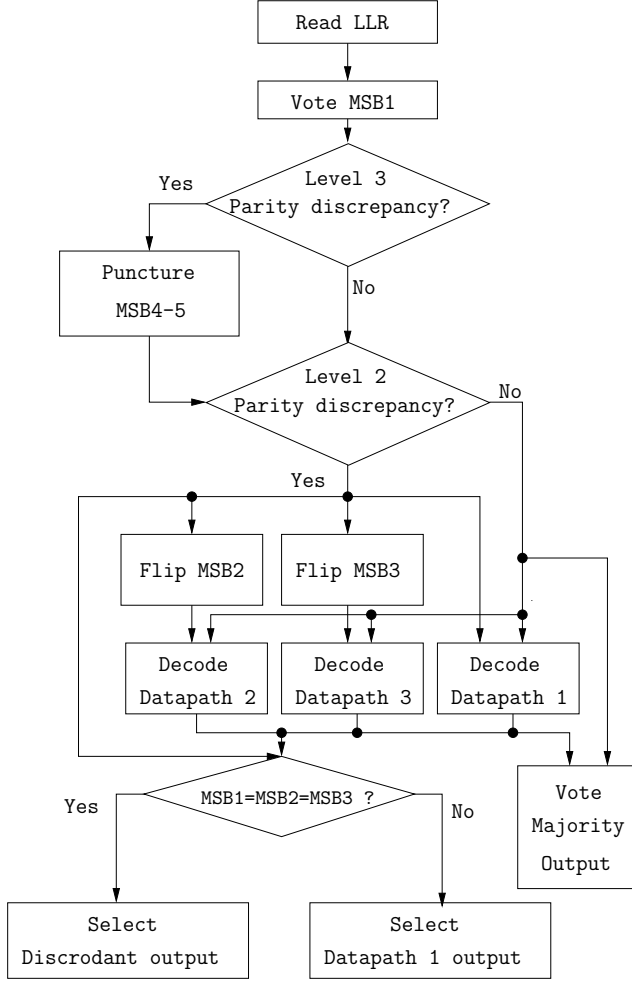


Figure 7.  $UEP_{full}$  logic flow

chooses the correct output.

### B. Implementation evaluation

The described architecture for UEP has been implemented in 90 nm CMOS technology with a target frequency of 200 MHz. Table II reports the area occupation of the logic necessary for the different levels and for a single datapath. It can be seen that while the complexity of Level 1 and Level 3 is negligible, Level 2 introduces a consistent overhead due to the two additional datapaths, that dominate the UEP logic area overhead. The total power consumption and area occupation introduced by UEP over a complete decoder architecture for  $UEP_{full}$ ,  $UEP_{sim1}$  and  $UEP_{sim2}$  has been reported in the last part of Table II. The considered decoder is the one presented in [30] as **A**, after adaptation to support only LDPC codes and to use the SCMS decoding algorithm; quantization of LLRs is  $b = 7$ . This version of the **A** implementation has been named **A<sub>REF</sub>** and has been used as a reference in the following comparisons. It relies on 22 Processing Elements (PEs) connected by means of a Kautz [33] network-on-chip,

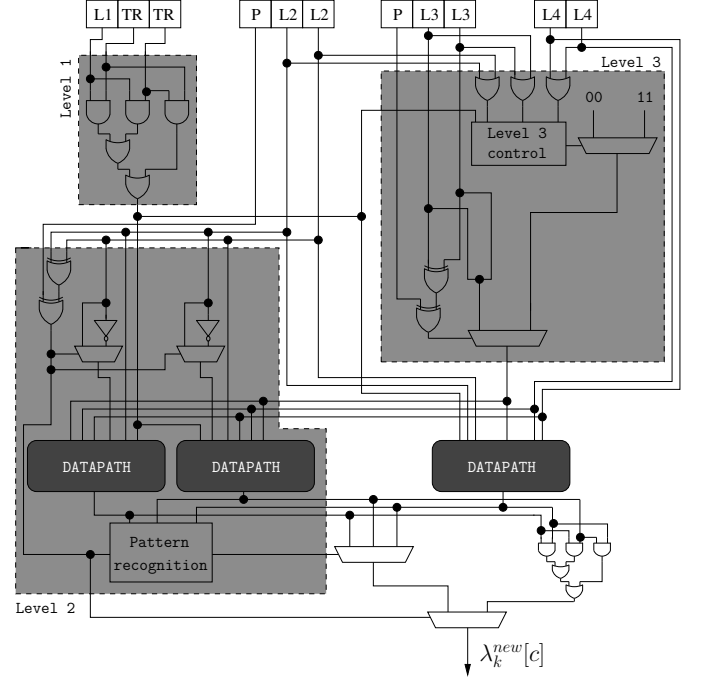


Figure 8.  $UEP_{full}$  hardware structure

with routing elements of degree three. The decoder supports all LDPC codes in WiMAX and WiFi standards, with  $N$  ranging between 576 and 2304, and  $r$  between 1/2 and 5/6. To help a fair evaluation of the overhead introduced by UEP with respect to alternative schemes, a straightforward “brute force” error protection technique has been applied in **A<sub>BF</sub>**, where MSB1-4 of both  $\lambda_k[c]$  and  $R_{lk}$  are tripled and voted. The additional memory bits result in +78.7% area occupation and +92.8% power consumption.

Between additional logic and extra memory bits, the implementation of  $UEP_{full}$  in **A<sub>full</sub>** leads to a 36.3% area overhead with respect to **A<sub>REF</sub>**, while a 43.9% increment is noticed in power consumption. Both area and power overheads are less than half than those shown by **A<sub>BF</sub>**, even though their error correction capabilities are comparable. Compared to **A<sub>REF</sub>**, both **A<sub>sim1</sub>** and **A<sub>sim2</sub>** show similar area increments (22.8% and 19.6% respectively), mainly due to the additional Level 2 datapaths in **A<sub>sim1</sub>** and to Level 1 extra memory bits in **A<sub>sim2</sub>**. The difference in power consumption increments (24.9% and 23.3%) is even smaller, since the Level 1 memory bits in **A<sub>sim2</sub>** contribute to a higher percentage of the total power consumption.

To prove that the devised UEP does not influence the performance of the decoder in terms of throughput and maximum frequency, Table III reports the delay introduced by each UEP level and by the whole architecture for different target frequencies in 90 nm CMOS technology. The synthesis process has been carried out with exact mapping and low mapping effort. Regardless, timing requirements have been met for frequencies as high as 1 GHz, much higher than those commonly used in current LDPC decoders, thus showing that the critical path of the complete system is not affected by the

Table II  
UEP - AREA OCCUPATION, POWER CONSUMPTION (90 NM CMOS, 200 MHz)

	Area	Power
Level 1 unit	$8 \mu\text{m}^2$	102.0 nW
Level 2 unit	$13655 \mu\text{m}^2$	503.5 $\mu\text{W}$
Level 3 unit	$102 \mu\text{m}^2$	12.7 $\mu\text{W}$
Single Datapath	$6667 \mu\text{m}^2$	278.1 $\mu\text{W}$
$A_{\text{REF}}$	$2.59 \text{ mm}^2$	97.1 mW
$A_{\text{BF}}$	$4.63 \text{ mm}^2$	187.5 mW
$A_{\text{full}}$	$3.53 \text{ mm}^2$	139.8 mW
$A_{\text{sim1}}$	$3.18 \text{ mm}^2$	121.3 mW
$A_{\text{sim2}}$	$3.10 \text{ mm}^2$	119.7 mW

Table III  
UEP - ACHIEVABLE FREQUENCY AND DELAY (90 NM CMOS)

Target Frequency	Delay [ns]			
	L1	L2	L3	UEP <sub>full</sub>
200 MHz	0.06	3.25	0.34	3.59
500 MHz	0.06	1.75	0.33	1.93
1 GHz	0.06	0.93	0.29	0.93

proposed UEP.

## VII. UEP PERFORMANCE

This section presents the performance evaluation of the proposed UEP under the same conditions of Section IV and Section V, showing the impact of each level of UEP as described in UEP<sub>full</sub>. Both  $R_{lk}$  and  $\lambda_k[c]$  are represented in two's complement and quantized with  $b = 7$ , where all bits are used to represent the integer part. In our evaluations we have considered SCMS algorithm, because of its enhanced error resiliency w.r.t. other min-sum approximations. Fig. 9-13 are referred to a WiMAX  $N = 2304$ ,  $r = 1/2$  LDPC code, with a maximum of 10 iterations per frame.

The decoders in [31] and [30] present similarities with many other decoders in the state of the art (serial core, min-sum-based layered decoding, partial parallelism, shared or dedicated memories, either high-throughput or flexible design). We have decided to consider them for our performance evaluation, as they are representative examples of the current literature on the subject. Before entering the details of our discussion, we wish to highlight that the effect of errors in an LDPC decoder is strongly dependent on the speed of the decoder. Usually, in literature errors in memories and logic are mostly identified by the Mean Time Between Failures (MTBF, [s]) or Failure In Time (FIT, number of errors in  $10^9$  hours). This turns out that, for example, a small MTBF will have little effect on the performance of a fast decoder like [31]: on the other hand, the same MTBF could be disruptive for [30], as it works at less than 1/5 of the frequency and has a lower degree of parallelism, thus requiring a higher number of clock cycles to complete an iteration. A fair error measure can consequently be the average number of errors encountered by the decoder during each iteration, here defined as Average Failures Per Iteration (AFPI). The frequencies and memory structure of [31] and [30] have been used to provide three error scenarios

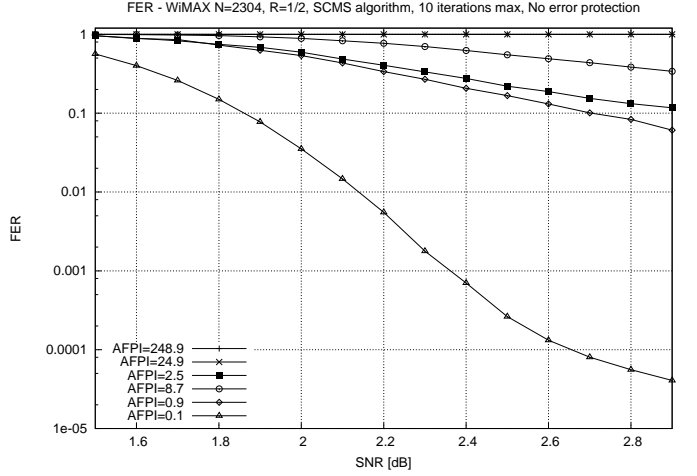


Figure 9. FER with different Average Failures Per Iteration (AFPI) - No error protection

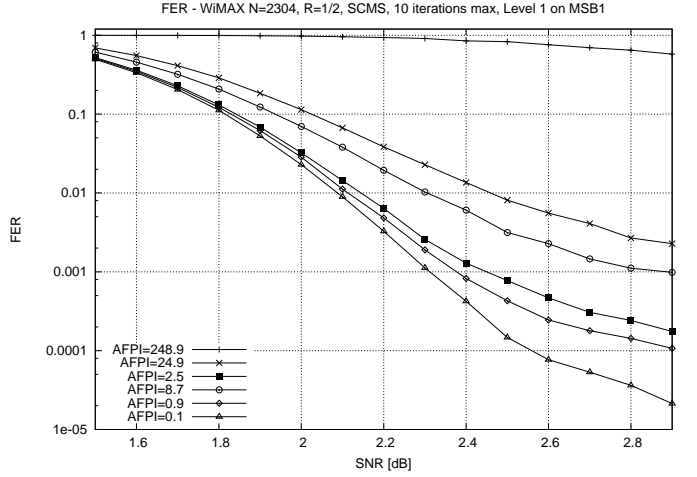


Figure 10. FER with different AFPI - Level 1 on MSB1

each (low, medium and high error probability), resulting in six AFPI values, i.e. 248.9, 24.9 and 2.5 for [30], and 8.7, 0.9 and 0.1 for [31].

Fig. 9 shows the FER of the considered code under the influence of the aforementioned six AFPI values: each bit of the stored  $R_{lk}$  and  $\lambda_k[c]$  has an equal probability of being flipped in presence of a soft error. From Fig. 9 we observe that, as it is reasonable, high values of AFPI highly degrade the FER well beyond any possibility of recovering. We also notice that even low AFPI values greatly affect the decoder.

Fig. 10 has been obtained by applying only Level 1 protection of UEP<sub>full</sub>. A relevant improvement can be noticed for all the AFPI values except the largest (i.e. AFPI=248.9), while no degradation is observed for the smallest AFPI=0.09 case.

The curves plotted in Fig. 11 have been obtained by considering both Level 1 and Level 2 of UEP<sub>full</sub>. Error protection is complete for AFPI=(0.1, 0.9, 2.5), and consistent improvements are observed for AFPI=(8.7, 24.9) with respect to Fig. 10. At the reference point of FER= $10^{-4}$ , for AFPI=8.7 there is a 0.1 dB performance loss, whereas values slightly less

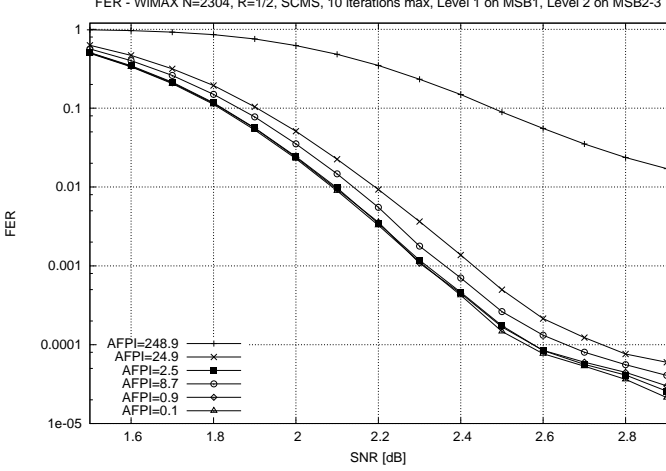


Figure 11. FER with different AFPI - Level 1 on MSB1, Level 2 on MSB2-3

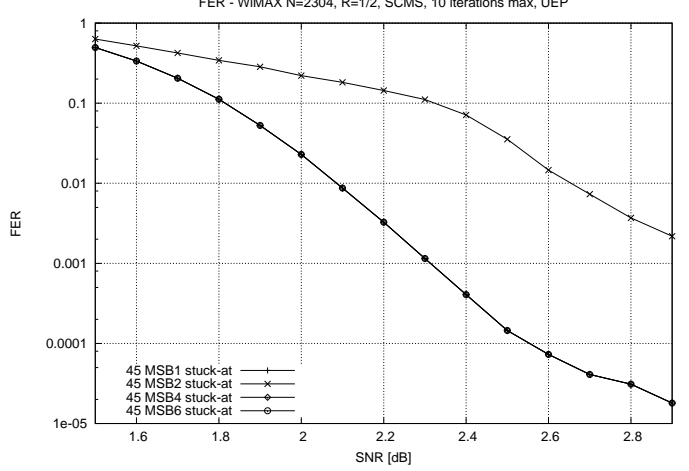


Figure 13. UEP performance in presence of stuck-at bits

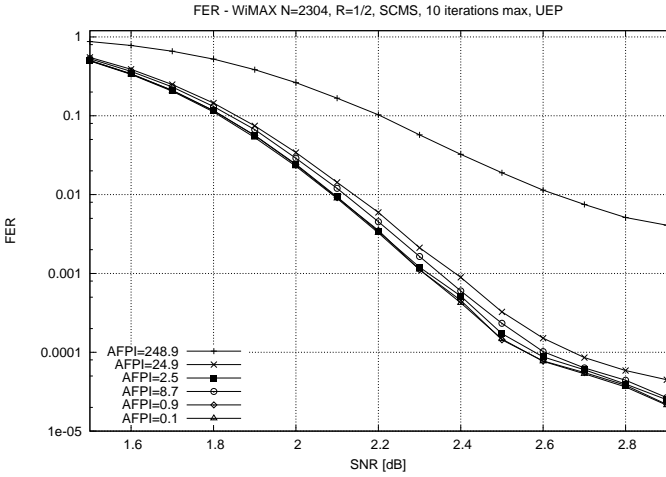


Figure 12. FER with different AFPI - Complete UEP

than 0.2 dB are observed for AFPI=24.9. Degradation is still strong for AFPI=248.9.

The complete UEP<sub>full</sub> has been employed to obtain the curves in Fig. 12. In this case, UEP<sub>full</sub> is able to bind the performance degradation caused by low AFPIs to very small values (e.g. 0.1 dB loss for AFPI=24.9 at FER=10<sup>-4</sup>), with positive effects also on AFPI=248.9, which impact is highly reduced.

Let us move to the evaluation of another characteristic of the proposed protection technique: stuck at bits errors. In Fig. 13 and Table IV we have reported the results. In particular, the resilience of UEP<sub>full</sub> against permanent memory errors is shown in Fig. 13, where each curve is affected by 45 stuck-at bits affecting different groups of bits. Level 1 gives total protection from single stuck-at bits, and Level 3 is sufficient to protect relatively critical MSB4 and MSB5. As with soft errors, MSB6 and MSB7 can be left without protection without any noticeable performance degradation. 45 stuck-at bits, however, are equivalent to a minimum of AFPI=45, a rate of failures that can not be afforded by the single Level 2 when

Table IV  
UEP - SINGLE STUCK-AT BITS THAT CAN BE RECOVERED

Code $N, r$	MSB Level 1	MSB2-3 Level 2	MSB4-5 Level 3	MSB6-7 Level 4
2304, 1/2	$\infty$	24	190	256
1152, 1/2	$\infty$	21	175	223
576, 1/2	$\infty$	19	152	206
2304, 5/6	$\infty$	17	138	191
1152, 5/6	$\infty$	15	119	152
576, 5/6	$\infty$	12	107	136

all errors are on MSB2. This is reflected also in the results presented in Table IV, where the number of stuck-at bits that can be withstood without performance degradation by different codes on every bit is shown. It can be noticed how the number of stuck-at bits that can be recovered is influenced not only by UEP, but also by the inherent characteristics of the code.

The error protection capability of UEP<sub>sim1</sub> and UEP<sub>sim2</sub> with respect to UEP<sub>full</sub> can be measured by observing the AFPI value for which a 0.1 dB loss is observed at FER=10<sup>-4</sup>. For UEP<sub>full</sub> the 0.1 dB loss is encountered with AFPI=24.9, while UEP<sub>sim1</sub> is able to sustain up to AFPI=1.8. Finally, UEP<sub>sim2</sub> can stand an even lower AFPI=0.3.

## VIII. COMPARISON

The resilient LDPC decoder designed in [6] protects both memories and logic from errors. The decoder works at a frequency of 400 MHz, obtaining a throughput of 333 Mb/s with 30 iterations and the WiMAX code with N=2304 and  $r=1/2$ , and implements the  $\lambda$ -min algorithm, that gives better performance than traditional min-sum approximations, since more than two minimums are considered. A dedicated 9-bit RAM is used to store  $\lambda_k[c]$  values, and it is protected with MSB1 tripling (+22% memory increment), while the initial LLRs received from the channel are stored in a 6-bit RAM and protected with MSB1 duplication and puncturing in case of discrepancy (+23% memory increment). The 6-bit  $R_{lk}$  values are not protected. The maximum memory error resiliency obtained with this method is MTBF=2 ms, corresponding to

AFPI=0.0017. Our UEP<sub>full</sub> proposal is more expensive in terms of memory (+57% for 7-bit  $\lambda_k[c]$  and +28% for 7-bit  $R_{lk}$ ), and the tripling of the whole datapath foreseen by Level 2 adds more complexity than the MSB1 duplication with algorithmic puncturing employed for the datapath in [6]. On the other hand, the proposed UEP targets much more degraded environments, since total error protection is achieved in presence of AFPI four orders of magnitude greater than those in [6]: moreover, this work tackles permanent errors as well, together with burst errors, both neglected in [6]. If we take in account the less expensive UEP<sub>sim1</sub> and UEP<sub>sim2</sub>, we have memory increments that are comparable to those observed in [6]: for  $\lambda_k[c]$ , +14.2% in UEP<sub>sim1</sub> and +28.4% in UEP<sub>sim2</sub>, while +28% for  $R_{lk}$  in both UEP<sub>sim1</sub> and UEP<sub>sim2</sub>. Still, the sustained AFPI is much larger than that of [6]. We can also observe how the achieved frequency is lower than that that can be reached by the proposed UEP implementation, as shown in Table III.

The two-level UEP devised in [32] splits each codeword in two partitions: in the most important one burst errors can be corrected, while in the less important partition only single errors can be corrected. Assuming to fit our case of study into [32], MSB1-3 are assigned to the important part, while MSB4-7 are left in the other. The number of redundancy bits required by the encoding is linked to the size of the sustained bursts: five bits (a +71% memory increment in our case) are necessary to correct 2-bit bursts, while much larger bursts are considered in our case. The work in [32] can potentially reach performance similar to this work's, but at a much higher complexity cost.

The statistical error correction scheme devised in [25] is built around a concept different from the proposed UEP: voltage overscaling is introduced in the decoder to save energy, and the performance loss brought by the timing errors caused by this technique must be compensated. Regardless of the different target, a fair comparison with this work is difficult, since the system performance in [25] has been evaluated with random regular codes, which characteristics and performance are far from those used in practical communication systems and in this paper. The obtained results in terms of error resiliency are promising, being comparable to those presented in this paper. The error estimator/detector circuit introduces a small power consumption overhead, and requires a considerable level of precision to guarantee acceptable BER levels. To these overheads must be added those linked with the calculation of the statistic itself, that requires non negligible computational power.

## IX. CONCLUSION

This paper proposes a novel Unequal Error Protection technique for memories used in LDPC decoders. It is divided in four levels, that can be adjusted and applied according to the decoder parameters and desired degree of protection. A complete design is presented, together with results for other two alternative schemes, showing a high level of resilience to transient and permanent errors, both single-bit and multi-bit. The design of an hardware architecture implementing the

UEP is proposed, and applied to an existing LDPC decoder to evaluate area and power consumption overheads. Comparison with the state of the art shows superior error resiliency even at comparable complexity overheads.

## ACKNOWLEDGMENT

The authors would like to thank Eyuel Zewdu Teferi for his valuable contribution in the VHDL description and validation of UEP.

## REFERENCES

- [1] "Process integration, devices and structures," *International Technology Roadmap for Semiconductors (ITRS)*, 2011. [Online]. Available: <http://www.itrs.net/Links/2011ITRS/2011Chapters/2011PIDS.pdf>
- [2] S. Srinivasan, A. Gayasen, N. Vijaykrishnan, M. Kandemir, Y. Xie, and M. Irwin, "Improving soft-error tolerance of FPGA configuration bits," in *Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on*, 2004, pp. 107–110.
- [3] R. G. Gallager, "Low density parity check codes," *IRE Trans. on Information Theory*, vol. IT-8, no. 1, pp. 21–28, Jan 1962.
- [4] C.-H. Liu, S.-W. Yen, C.-L. Chen, H.-C. Chang, C.-Y. Lee, Y.-S. Hsu, and S.-J. Jou, "An LDPC decoder chip based on self-routing network for IEEE 802.16e applications," *IEEE Jour. of Solid-State Circuits*, vol. 43, no. 3, pp. 684–694, 2008.
- [5] T.-C. Kuo and A. Willson, "A flexible decoder IC for WiMAX QC-LDPC codes," in *Custom Integrated Circuits Conference, 2008. CICC 2008. IEEE*, 2008, pp. 527–530.
- [6] M. May, M. Alles, and N. Wehn, "A case study in reliability-aware design: A resilient LDPC code decoder," in *Design, Automation and Test in Europe, 2008. DATE '08*, 2008, pp. 456–461.
- [7] B. Masnick and J. Wolf, "On linear unequal error protection codes," *Information Theory, IEEE Transactions on*, vol. 13, no. 4, pp. 600–607, 1967.
- [8] K. Namba and F. Lombardi, "Parallel decodable multi-level unequal burst error correcting codes for approximate computing," Northeastern University, MA, Tech. Rep., 2014.
- [9] C. W. Yung, H. F. Fu, C. Y. Tsui, R. Cheng, and D. George, "Unequal error protection for wireless transmission of MPEG audio," in *Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on*, vol. 6, 1999, pp. 342–345 vol.6.
- [10] X. Yang and K. Mohanram, "Unequal-error-protection codes in SRAMs for mobile multimedia applications," in *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, 2011, pp. 21–27.
- [11] M. Chen and M. Murthi, "Optimized unequal error protection for voice over IP," in *Proc. of Acoustics, Speech, and Signal Processing (ICASSP) Conference*, vol. 5, 2004, pp. V-865–8 vol.5.
- [12] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *IEEE Workshop on Signal Processing Systems*, 2004, pp. 107–112.
- [13] M. Martina, G. Masera, S. Papaharalabos, P. T. Mathiopoulos, and F. Gioulekas, "On practical implementation and generalizations of max\* operator for turbo and LDPC decoders," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 4, pp. 888–895, Apr 2012.
- [14] V. Savin, "Self-corrected Min-Sum decoding of LDPC codes," in *Proc. of IEEE International Symposium on Information Theory*, jul. 2008, pp. 146–150.
- [15] W. Liu, J. Rho, and W. Sung, "Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND flash memories," in *Signal Processing Systems Design and Implementation. IEEE Workshop on*, 2006, pp. 303–308.
- [16] R. C. Bose and D. K. Ray-Chaudhuri, *Information and Control*, 1960.
- [17] C. Argyrides, H.-R. Zarandi, and D. Pradhan, "Multiple upsets tolerance in SRAM memory," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, 2007, pp. 365–368.
- [18] T. Miller, R. Thomas, J. Dinan, B. Adcock, and R. Teodorescu, "Parichute: Generalized turbocode-based error correction for near-threshold caches," in *Microarchitecture (MICRO), 2010 43rd Annual IEEE/ACM International Symposium on*, 2010, pp. 351–362.
- [19] R. Motwani and C. Ong, "Design of LDPC coding schemes for exploitation of bit error rate diversity across dies in NAND flash memory," in *Computing, Networking and Communications (ICNC), 2013 International Conference on*, 2013, pp. 950–954.

- [20] A. Naghdinezhad, M. Hashemi, and O. Fatemi, "A novel adaptive unequal error protection method for scalable video over wireless networks," in *Consumer Electronics, 2007. ISCE 2007. IEEE International Symposium on*, 2007, pp. 1–6.
- [21] Y. C. Chang, S.-W. Lee, and R. Komiya, "An efficient FEC allocation algorithm for unequal error protection of wireless video transmission," in *Advanced Information Networking and Applications, 2009. AINA '09. International Conference on*, 2009, pp. 175–181.
- [22] D.-F. Yuan, Z.-W. Li, A.-F. Sui, and J.-M. Ning, "Research on unequal error protection with punctured convolutional codes in image transmission system over mobile channels," in *Wireless Communications and Networking Conference, 2000. WCNC. 2000 IEEE*, vol. 3, 2000, pp. 1253–1257 vol.3.
- [23] S. Tehrani, S. Mannor, and W. Gross, "Fully parallel stochastic LDPC decoders," *Signal Processing, IEEE Transactions on*, vol. 56, no. 11, pp. 5692–5703, 2008.
- [24] A. Naderi, S. Mannor, M. Sawan, and W. Gross, "Delayed stochastic decoding of LDPC codes," *Signal Processing, IEEE Transactions on*, vol. 59, no. 11, pp. 5617–5626, 2011.
- [25] E. Kim and N. Shanbhag, "Energy-efficient LDPC decoders based on error-resiliency," in *Signal Processing Systems (SiPS), 2012 IEEE Workshop on*, 2012, pp. 149–154.
- [26] C. Huang, Y. Li, and L. Dolecek, "Gallager B LDPC decoder with transient and permanent errors," *Communications, IEEE Transactions on*, vol. PP, no. 99, pp. 1–14, 2013.
- [27] G. Gentile, M. Rovini, and L. Fanucci, "A multi-standard flexible turbo/LDPC decoder via ASIC design," in *International Symposium on Turbo Codes & Iterative Information Processing*, 2010, pp. 294–298.
- [28] M. Alles, T. Vogt, and N. Wehn, "FlexiChAP: A reconfigurable ASIP for convolutional, turbo, and LDPC code decoding," in *International Symposium on Turbo Codes and Related Topics*, 2008, pp. 84–89.
- [29] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. on Comm.*, vol. 47, no. 5, pp. 673–680, May 1999.
- [30] C. Condo, M. Martina, and G. Masera, "VLSI implementation of a multi-mode turbo/LDPC decoder architecture," *IEEE Trans. on Circuits and Systems I*, vol. 60, no. 6, pp. 1441–1454, 2013.
- [31] K. Zhang, X. Huang, and Z. Wang, "A high-throughput LDPC decoder architecture with rate compatibility," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 58, no. 4, pp. 839–847, 2011.
- [32] K. Namba and E. Fujiwara, "Unequal error protection codes with two-level burst and capabilities," in *Defect and Fault Tolerance in VLSI Systems, 2001. Proceedings. 2001 IEEE International Symposium on*, 2001, pp. 299–307.
- [33] M. Imase, M.; Itoh, "A design for directed graphs with minimum diameter," *IEEE Trans. on Computers*, vol. C-32, no. 8, pp. 782–784, Aug. 1983.



Carlo Condo obtained the M.Sc. in Electrical and Computer Engineering from Politecnico di Torino, Italy, and the University of Illinois at Chicago, USA, in 2010. He obtained the Ph.D. in Electronic and Telecommunication Engineering from Politecnico di Torino and Telecom Bretagne, France, in 2014. His research interests include design and implementation of architectures for channel coding and digital signal processing.



Guido Masera (SM07) received the Dr. Ing. Degree (summa cum laude) in 1986 and the Ph.D. degree in electronic engineering from the Politecnico di Torino, Torino, Italy, in 1992. From 1986 to 1988, he was a Researcher with the Centro Studi e Laboratori in Telecomunicazioni (CSELT), Torino, Italy, involved in the standardization activities for the GSM system. Since 1992, he has been an Assistant Professor and then Associate Professor with the Electronic Department, where he is member of the VLSI-Lab group. His research interests include several aspects in the design of digital integrated circuits and systems, with special emphasis on high-performance architecture development (especially for wireless communications and multimedia applications) and on-chip interconnect modeling and optimization. He has co-authored 230 journal and conference papers in the areas of ASIC-SoC development, architectural synthesis, VLSI circuit modeling and optimization. In the frame of competitive National and European research projects, he has been co-designer of several ASIC and FPGA implementations in the fields of Artificial Intelligence, Computer Networks, Digital Signal Processing, Transmission and Coding. He is an associate editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II.



Paolo Montuschi (M90-SM07-F14) is a Professor of Computer Engineering at Politecnico di Torino, Italy, where he

served as Chair of Department from 2003 to 2011, and as Chair or Member of several Boards, including the Board of Governors. He is currently serving as Associate Editor-in-Chief of the IEEE Transactions on Computers, as a member of the steering committee and Associate Editor of the IEEE Transactions on Emerging Topics in Computing and as a Member of the Advisory Board of Computing Now. He is also serving as Member of the IEEE Publication Services and Products Board. Previously, he served as Chair of the Magazine Operations, of the Electronic Products and Services and of the Digital Library Operations Committees, Member-at-Large of the Computer Society Publications Board, and Member of the Board of Governors of the IEEE Computer Society. He served as Guest and Associate editor of the IEEE Transactions on Computers from 2000 to 2004 and from 2009 to 2012, and co-chair, program and steering committee member of several conferences. His current main research interests and scientific achievements are in computer arithmetic, computer architectures, computer graphics, electronic publications, semantics & education, and new frameworks for the dissemination of scientific knowledge. Montuschi is a Fellow of the IEEE and a Computer Society Golden Core Member. In April 2014 he has been inducted into the International Academy of Sciences of Turin. In June 2014 he has been appointed to serve as Editor-in-Chief of the IEEE Transactions on Computers for the term 2015-16. Montuschi obtained a PhD in computer engineering in 1989, and since 2000 he has been full Professor.