Design of Min-Sum-based LDPC decoders using imprecise arithmetic

Christiane L. Kameni Ngassa [*#1] Valentin Savin [*2] David Declercq [#3]

[*] *CEA-LETI, MINATEC campus, 38054 Grenoble, France*
[#] *ETIS ENSEA/UCP/CNRS UMR 8051, 95014 Cergy-Pontoise, France*
[1] christiane.kameningassa@cea.fr, [2] valentin.savin@cea.fr, [3] declercq@ensea.fr

***Abstract*—This work evaluates the robustness of Low-Density Parity-Check decoders against errors due to imprecise arithmetic. While the use of imprecise arithmetic is motivated by savings in energy, delay and area, it also causes errors during the decoding process. This is a new paradigm in coding theory, which traditionally assumes that an error correction decoder operates on exact hardware and errors can only be introduced by the transmission channel. We design imprecise arithmetic operators and investigate their use within several Min-Sum-based decoders. We show that all decoders are able to provide error protection, but most of them suffer a performance penalty compared to the exact arithmetic implementation. Remarkably, the Self-Corrected Min-Sum decoder incurs no performance penalty when imprecise arithmetic is used.**

***Index Terms*—Fault-tolerance, LDPC codes, Min-Sum-based decoders, low-power decoders, imprecise arithmetic.**

I. Introduction

In order to support the sustainable development of future communication systems, energy efficiency became one of the most important issues to be addressed. Traditionally, the power consumption has been interpreted as the transmit power. This is mainly due to long-range communications, which greatly contributed to the development of information and coding theory, for which the transmit power dominates the total power consumed by the system. Forward error correction (FEC) codes have been key components to design reliable communication systems, while limiting the transmit power to acceptable low levels. Moreover, spectacular advances in the domain of graph-based codes and iterative decoding techniques, made possible the development of new families of error correcting codes ensuring reliable communication at transmit powers closer and closer to the theoretical Shannon limit [1], [2].

Nowadays, there is an increased interest in shorter range communications, from a few meters (femto-cells, wireless sensor networks, etc.) to a few millimeters (inter-chip and on-chip communications). For such applications, it is commonly accepted that the processing power (the power consumed in processing the signals) might represent a substantial fraction of the total power [3], [4]. In this context, power consumption of the FEC decoder module is often a bottleneck, as it can require an important part of the processing power, or even more power than the receiver can supply (*e.g.* in case of low-power systems).

For applications that can trade the accuracy of the circuit for the power consumption, two main approaches are currently investigated. The first approach, consists in using aggressive voltage scaling as the basis for reliability and energy tradeoffs. This brings the signal level close to the noise level, which reduces the noise immunity of the circuit and leads to probabilistic computational models [5], [6]. Such models were used to derive low-energy computational platforms for probabilistic algorithms of for applications from the domain of image and video processing, which tolerate probabilistic behavior at the circuit level [7].

The second approach consists in using *imprecise* (also referred to as inexact) circuits, obtained by pruning the exact circuit. This amounts to removing a certain number of logic gates from the circuit (depending on the application's tolerance to errors), which may result in significant savings in energy, delay and area [8]. Imprecise arithmetic (*e.g.* adders, multipliers) proved to be particularly useful for applications from the domain of image and video processing [9]–[11].

This paper investigates the robustness of FEC decoders against errors due to imprecise arithmetic. This is a new paradigm in coding theory, which traditionally assumes that the operations of a FEC decoders are exact, and errors can only be introduced by the channel. While the use of imprecise arithmetic is motivated by savings in energy, delay and area, *the first question we have to answer is whether or not FEC decoders are able to provide reliable error protection when they operate on imprecise hardware*.

We focus on Low-Density Parity-Check (LDPC) codes [12], a class of error correcting codes that feature low complexity message-passing (MP) iterative decoding and can be optimized for a broad class of channels, with performance approaching the theoretical Shannon limit [2]. We evaluate the performance of several LDPC decoders using imprecise arithmetic operators. While the performance penalty due to imprecise arithmetic depends on the considered decoder, we show that the Self-Corrected Min-Sum decoder is inherently robust, and does not suffer any performance penalty due to imprecise arithmetic.

The remainder of the paper is organized as follows. Related works are discussed in Section II. Section III gives a brief introduction to LDPC codes and iterative decoding algorithms. Section IV is concerned with the design of imprecise Min-Sum-based decoders and their imprecise arithmetic components. Simulation results are provided in Section V. Section VI discusses future works and Section VII concludes the paper.

II. Related works

Over the last few years, there has been an increased interest in investigating the behavior of LDPC decoders operating on circuits built from probabilistic components. The motivation is two-fold. On the one hand, as mentioned in the Introduction, aggressive voltage scaling can be used to reduce the power consumption of the circuit, leading to probabilistic computational models. On the other hand, it is now widely accepted that emerging nano-electronic devices will be inherently unreliable, due to ineluctable increases in density integration and imperative requirements of low-energy consumption. Recent works studied the performance of Gallager A and Gallager B decoders on faulty hardware [13]–[15]. The focus of this paper differs from these research in several ways:

1) Our decoders are implemented on circuits built from imprecise components. Both imprecise and probabilistic components causes errors during the decoding process, but imprecise behavior is deterministic and can be "tuned" to a desired level of errors.
2) In [13]–[15], the authors investigate the asymptotic behavior of the decoder, while we are concerned with finite-length performance.
3) Previous works were concerned with Gallager A and Gallager B decoders, while we are interested in Min-Sum-based decoders, which are widely implemented in real communication systems.

III. LDPC codes and iterative decoding

Low-Density Parity-Check (LDPC) codes, have been introduced by Gallager in the early 60's [12], as a class of linear block codes defined by sparse parity-check matrices, suitable for decoding by message-passing (MP) iterative algorithms. Tanner described LDPC codes in terms of sparse bipartite graphs [16], containing two types of nodes: variable-nodes corresponding to coded bits and check-nodes corresponding to parity checks. Equivalently, variable and check nodes correspond respectively to columns and rows of the parity check matrix H, while edges connecting variable and check nodes correspond to the non-zero entries of H.

The graphical representation proposed by Tanner proved to be particularly suitable for MP decoding algorithms. Such a decoding algorithm consists of an exchange of messages along the edges of the bipartite graph. Each message provides an estimation of either the sender or the recipient variable-node (the variable-node incident to the edge), and the exchange of messages takes place in several rounds, or iterations. At each new iteration, new messages are computed in an *extrinsic manner*, meaning that a message that is sent on an edge does not depend on the message just received on the same edge. Consequently, variable-nodes collect more and more information with each new decoding iteration, which gradually improves the estimation of the sent codeword.

The use of Tanner graphs allowed reformulating the probabilistic decoding initially proposed by Gallager in terms of Belief-Propagation (BP) – also referred to as Sum-Product (SP) – a MP algorithm that performs Bayesian inference on graphical models [17], [18]. The BP decoding is known to be optimal for codes defined by cycle-free bipartite graphs, in the sens it outputs the Maximum A Posteriori (MAP) estimates of the coded bits. It is also known to achieve an excellent performance on general sparse bipartite graphs, even if it deviates from the MAP in practical cases (because bipartite graphs associated with practical codes contain cycles). However, the decoding performance is not the only relevant criterion when it comes to practical system implementation, and the BP algorithm is disadvantaged by its complexity, numerical instability, and the fact that it requires the perfect knowledge of the channel parameter (*e.g.* SNR), which may be imprecisely estimated in practical situations.

One way to deal with complexity and numerical instability issues is to simplify the computation of messages exchanged within the BP decoding. The most complex step of the BP decoding is the computation of check-to-variable node messages, which makes use of computationally intensive hyperbolic tangent functions. The Min-Sum (MS) algorithm is aimed at reducing the computational complexity of the BP, by using max-log approximations of the parity check to coded bit messages [19], [20]. The only computations required by the MS decoding are additions and comparisons, which solves the complexity and numerical instability problems. The performance of the MS decoding is also known to be independent of the knowledge of the channel parameter, for most of the usual channel models.

However, the max-log approximation used in the MS decoding leads to a performance degradation with respect to the BP decoding. Several "correction" methods were proposed in the literature in order to mitigate this performance degradation [20]–[25]. These decoding algorithms are referred to as MS-based algorithms: they are improved versions of the MS algorithm, with only a very limited increase of complexity.

In this paper we investigate the use of imprecise arithmetic circuits for the following decoders: MS decoder, Normalized-MS (NMS) decoder [21], Offset-MS decoder [21], Self-Corrected-MS (SCMS) decoder [25]. They will be described in the following paragraphs.

A. Notation

The following notation will be used throughout the paper:

- H, Tanner graph of an LDPC code,
- N, number of variable-nodes,
- M, number of check-nodes,
- d_n, degree of the variable-node n,
- d_m, degree of the check-node m,
- $n \in \{1, 2, ..., N\}$, a variable node of H,
- $m \in \{1, 2, ..., M\}$, a check node of H,
- $H(n)$, set of check-nodes connected to the variable-node n,
- $H(m)$, set of variable-nodes connected to the check-node m,
- γ_n, a priori log-likelihood ratio (LLR) of variable-node n,
- $\tilde{\gamma}_n$, a posteriori LLR of variable-node n,
- $\alpha_{m,n}$, variable-to-check message sent from n to m,
- $\beta_{m,n}$, check-to-variable message sent from m to n.

B. Min-Sum decoding

Assume that a codeword $(x_n)_{n=1,\dots,N}$ is sent over a memoryless noisy channel, and let $(y_n)_{n=1,\dots,N}$ denote the received word. The MS decoding algorithm works as follows.

Initialization

- A priori LLRs

$$\gamma_n = \log \frac{\Pr(x_n = 0 \mid y_n)}{\Pr(x_n = 1 \mid y_n)}$$

- Variable-to-check messages initialization

$$\alpha_{m,n} = \gamma_n$$

Iterations

- Check-node processing

$$\beta_{m,n} = \left(\prod_{n' \in H(m) \backslash n} \operatorname{sgn}(\alpha_{m,n'}) \right) \min_{n' \in H(m) \backslash n} (|\alpha_{m,n'}|)$$

- A posteriori LLRs

$$\tilde{\gamma}_n = \gamma_n + \sum_{m \in H(n)} \beta_{m,n}$$

- Variable-node processing

$$\alpha_{m,n} = \tilde{\gamma}_n - \beta_{m,n}$$

In the above description, γ_n and $\tilde{\gamma}_n$ are computed for each variable-node n, while messages $\alpha_{m,n}$ and $\beta_{m,n}$ are computed for each graph edge (m, n). Finally, at each iteration, coded bit estimates are computed by $\hat{x}_n = (1 - \operatorname{sgn}(\tilde{\gamma}_n))/2$, and decoding stops when whether $(\hat{x}_n)_{n=1,\dots,N}$ is a codeword or a maximum number of iterations has been reached.

C. Normalized Min-Sum decoding

As discussed in Introduction, MS decoding can be seen as a low-complex approximate version of the BP decoding. This approximation is known to result in an overestimation of check-to-variable messages. The aim of the NMS decoding is to compensate this overestimation, by introducing a normalization (scaling) factor $\lambda \in]0, 1[$ within the check-node processing step. Hence, all the other decoding steps remain unchanged, and only the check-node processing step is modified as follows:

$$\beta_{m,n} = \left(\prod_{n' \in H(m) \backslash n} \operatorname{sgn}(\alpha_{m,n'}) \right) \min_{n' \in H(m) \backslash n} (|\alpha_{m,n'}|)$$
$$\beta_{m,n} = \lambda \cdot \beta_{m,n}$$

D. Offset Min-Sum decoding

Similar to the the NMS decoding, the OMS attempts to compensate the overestimation of the check-to-variable messages. This time an offset factor $\delta > 0$ is used, and the check-node processing step is modified as follows:

$$\beta_{m,n} = \left(\prod_{n' \in H(m) \backslash n} \operatorname{sgn}(\alpha_{m,n'}) \right) \min_{n' \in H(m) \backslash n} (|\alpha_{m,n'}|)$$
$$\beta_{m,n} = \operatorname{sgn}(\beta_{m,n}) \cdot \max(|\beta_{m,n}| - \delta, 0)$$

E. Self-Corrected Min Sum decoding

The SCMS addresses the overestimation issue at the variable-node processing side of the algorithm. The rationale behind the SCMS is that the overestimation of check-to-variable messages is not critical, unless any given variable-to-check message is updated to map a different bit state. In the log likelihood ratio domain, this corresponds to a sign change. In the SCMS, any variable-to-check message that would experience a sign change is erased (that is, it is set to zero). Hence, check-node processing step is modified as shown below, while all the the decoding steps are the same as for MS decoding.

$$\alpha^{\text{tmp}}_{m,n} = \tilde{\gamma}_n - \beta_{m,n}$$
$$\alpha_{m,n} = \begin{cases} 0, & \text{if } \operatorname{sgn}(\alpha^{\text{tmp}}_{m,n}) \neq \operatorname{sgn}(\alpha_{m,n}) \text{ and } \alpha_{m,n} \neq 0 \\ \alpha^{\text{tmp}}_{m,n}, & \text{otherwise} \end{cases}$$

So, the variable-to-check message is first stored in a temporary value $\alpha^{\text{tmp}}_{m,n}$, and its sign is compared against the sign of the variable-to-check messages from the previous iteration (stored in $\alpha_{m,n}$). If a sign change is detected and $\alpha_{m,n} \neq 0$, the value of the new variable-to-check message is set to zero. Otherwise, the value the new variable-to-check message is set (as it would usually be) to $\alpha^{\text{tmp}}_{m,n}$.

It [25], the author pointed out that a variable-to-check message changes its sign between two consecutive iterations if and only if its computation tree [26] contains unreliable information. As a consequence, it has been shown that the SCMS decoding behaves as the MS decoding on a computation tree that has been pruned of its unreliable branches. The SCMS decoding ability to detect unreliable messages will prove to be particularly useful when the decoder is implemented on imprecise circuits.

IV. Imprecise Min-Sum-based decoders

In order to evaluate the impact of the imprecise arithmetic components on the performances of MS-based LDPC decoders, all the messages in the decoders must be quantized. Since the a posteriori LLR is computed as the sum of the a priori LLR and the incoming check-to-variable messages, more quantization bits have to be used to represent its value. Hence, Q bits are used for the quantization of the a priori LLRs (γ_n), as well as exchanged messages ($\alpha_{m,n}$, $\beta_{m,n}$), while $Q+1$ bits are used for the quantization of the a posteriori LLRs ($\tilde{\gamma}_n$). The imprecise arithmetic components used within the MS-based LDPC decoders are:

- Q-bit comparators, used for the implementation of the check-node processing step.
- $(Q + 2)$-bit adders used for the implementation of the a posteriori LLRs update and the variable-node processing step.

Note that an extra bit is used for the adder in order to detect the overflow. At the entry of the adder, the $(Q + 1)$-bit input operands get an extra bit by repeating their most significant (sign) bit. Throughout this paper, we shall use $Q = 6$. The design of imprecise 6-bit comparator and 8-bit adder is addressed in the following sections.

A. Kogge-Stone Adder

Consider two integers A and B, and let S denote their sum and C the corresponding carry. The simplest type of adder that can be used to compute S is the ripple carry adder which computes successively the sum and the carry for each bit starting from the least significant bit (LSB). However, this type of adder is very slow and the Carry Lookahead adders have been developed to reduce the computation time. In order to compute the sum S, they introduce two binary parameters G and P. For each bit position i, G_i is 1 if A_i and B_i are both equal to 1. The bit position i is then said to generate a carry. If only one of A_i or B_i is equal to 1, P_i is equal to 1 and the bit position i is said to propagate a carry.

All carry-lookahead adders (CLA) perform binary addition in three steps: precomputation, prefix and postcomputation. For all adder architectures, the precomputation and postcomputation steps are similar: P_i and G_i are computed for each position bit in the precomputation, while the sum S is computed in the postcomputation step. However, the prefix step is different for each adder architecture and those architectures can be classified into three categories: serial-prefix, group-prefix and parallel-prefix. Fig. 1 represents a 4-bits parallel-prefix CLA architecture.

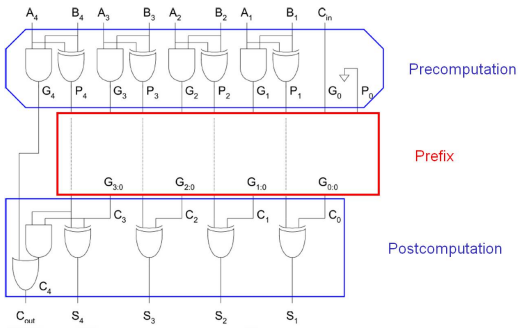


Fig. 1: 4-bit parallel-prefix CLA architecture

Kogge-Stone adder is a parallel-prefix CLA architecture [27], which generates the carry in a $O(\log n)$ time. It is one of the fastest existing adder architectures and provides the highest performances [8]. The Prefix Diagram of a 8-bit Kogge-Stone adder is represented in Fig. 2. As shown in the top of the figure, a grey cell corresponds to one OR and one AND gate, while a black cell corresponds to one OR and two AND gates.

Kogge-Stone adders have been selected as a basis for the implementation of imprecise adders. Imprecise comparators are also derived from Kogee-Stone adders, but only the path that computes the most significant (sign) bit will be used.

B. The imprecise adder

In order to design inexact arithmetic components, we suppress several logic gates in all the adders and in the comparators of the circuit. However, the objective is to control the errors introduced by pruning the circuit, hence some constraints might be defined according to the desired level of errors. For example, the magnitude of the errors can be

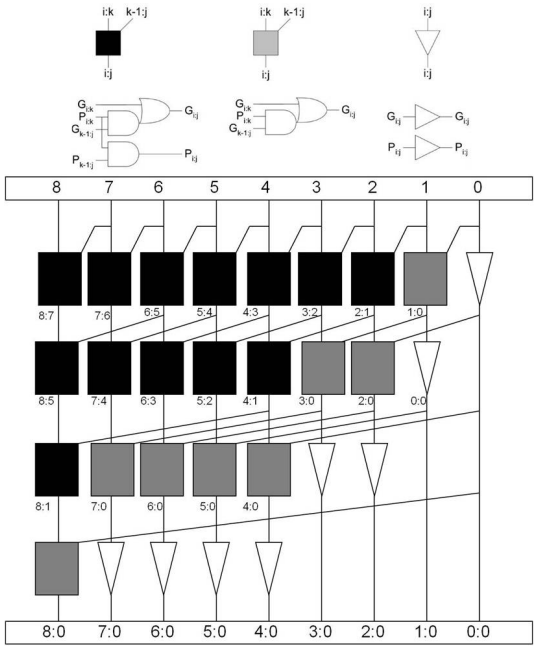


Fig. 2: 8 bits Kogge-Stone diagram

bounded or the paths of the circuit with lower probability of being used can be deleted [8]. In this work the main constraints are as follows. First, we require that the changes to the exact circuit do not impact the most significant (sign) bit. Protecting the sign of each addition from errors reduces the impact of the inexact circuit. Secondly, we require that for operands x and y, such that the value of the exact addition $x + y$ is small, the error made by the imprecise adder when computing $x + y$ must also be small (close to 0). The reason is that we do not want to give any extra confidence to the decoder, when its degree of confidence should be low. Note however that if the exact value of $x + y$ is relatively large, the error made by the imprecise adder when computing $x + y$ may also be large (but the sign is always correct).

In order to meet the above requirements, the path that computes the sign bit is kept unchanged and logic gates are suppressed starting from the least significant bit (LSB)to the most significant bit (MSB). The adder is simulated to check if all the requirements are met or not. The procedure is repeated until all the requirements are met and the suppression of any of the remainder logic gates will fail to comply with the requirements. The designed imprecise adder is shown in Fig. 3, and it has the following characteristics:

- 3 grey cells have been deleted.
- 4 black cells have been replaced by 4 grey cells.
- 288 erroneous outputs are generated out of 16129 possible additions.
- The sign bit of any output is always correct.
- For a given x, the adder always correctly computes $x - x$.

Some examples of imprecise additions are given in Table I.

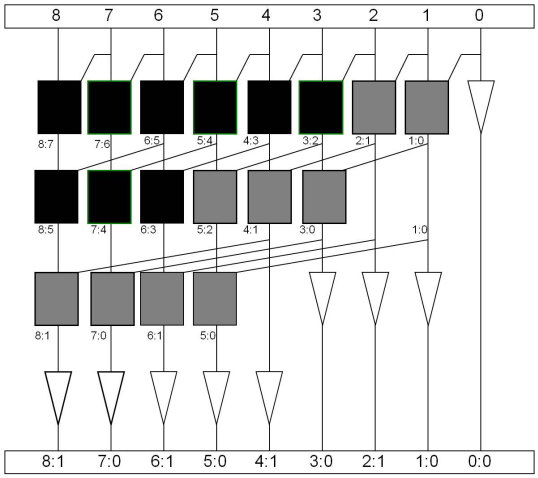


Fig. 3: Imprecise adder

TABLE I: Example of inexact additions

A	-29	-25	-21	3	7	11	-31	-27	-23	-29
B	-27	-23	-19	5	9	13	7	11	15	13
Output	-24	-16	-8	40	48	56	-56	-48	-40	-48

C. The imprecise comparator

Comparators are used to implement the check-node processing step. The sign path of a 6-bit Kogge-Stone adder is used to design the imprecise comparator. For any two operands x and y, we allowed the output of the imprecise comparator to be in error only if x and y have relatively close values. As for the imprecise adder, logic gates are suppressed starting from the LSB to the MSB until the suppression of any of the remainder logic gates will fail to comply with the requirements. The designed imprecise comparator is shown in Fig. 4b, and it has the following characteristics:

- 1 black cell and 2 grey cells have been deleted.
- 2 black cells have been replaced by 2 grey cells.
- 96 erroneous outputs are generated out of 961 possible comparisons.
- Errors happen only when $\mid x - y \mid < 2^2$.

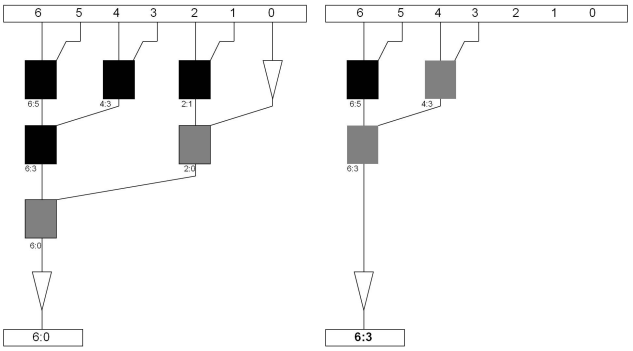
Some examples of imprecise comparisons are given in Table II. The comparator's output is 1 iff $x <_{\text{imprecise}} y$.

TABLE II: Example of inexact comparisons

A	7	-29	-21	-9	-5	-1	-5	2	3	6
B	-7	-31	-23	-11	-7	-3	-6	1	2	5
Output	1	1	1	1	1	1	1	1	1	1

D. Implementation of imprecise Min-Sum-based decoders

Let $\mathbf{q}$ denote the Q-bit quantizer used at the decoder input, and $\mathbf{s}$ denote the Q-bit saturation operation ($\mathbf{s}(x) = 1-2^{(Q-1)}$ if $x < 1-2^{(Q-1)}$, $\mathbf{s}(x) = 2^{(Q-1)}-1$ if $x > 2^{(Q-1)}-1$, and $\mathbf{s}(x) = x$ otherwise). We also denote by $\mathbf{m}_{\text{imp}}$ the imprecise minimum computation, and by $\mathbf{a}_{\text{imp}}$ the imprecise adder. The Imprecise-MS decoder is implemented as follows:



(a) Ideal 6 bits comparator (b) Imprecise Comparator

Fig. 4: 6 bits comparator architecture

Initialization

- A priori LLRs

$$\gamma_n = \mathbf{q}\left(\log \frac{\Pr(x_n = 0 \mid y_n)}{\Pr(x_n = 1 \mid y_n)}\right)$$

- Variable-to-check messages initialization

$$\alpha_{m,n} = \gamma_n$$

Iterations

- Check-node processing
  Let $H(m)\backslash\{n\} = \{n_1, \ldots, n_{d_m-1}\}$

$$\beta_{m,n} = \left(\prod_{i=1,\ldots,d_c-1} \text{sgn}(\alpha_{m,n_i})\right) \cdot \mathbf{m}_{\text{imp}}(\cdots(\mathbf{m}_{\text{imp}}(|\alpha_{m,n_1}|, |\alpha_{m,n_2}|), \cdots, |\alpha_{m,n_{d_m-1}}|)$$

- A posteriori LLRs
  Let $H(n) = \{m_1, \ldots, m_{d_n-1}\}$

$$\tilde{\gamma}_n = \mathbf{a}_{\text{imp}}(\cdots(\mathbf{a}_{\text{imp}}(\gamma_n, \beta_{m_1,n}), \cdots, \beta_{m_{d_n-1},n})$$

- Variable-node processing

$$\alpha_{m,n} = \mathbf{s}\left(\mathbf{a}_{\text{imp}}(\tilde{\gamma}_n, -\beta_{m,n})\right)$$

We note that in order to implement the check-node processing step it is actually sufficient to compute the first and the second minima of $|\alpha_{m,n_i}|$, for all $i = 1, \ldots, d_m$.

For the OMS decoder (Section III-D), the offset factor δ is subtracted from $|\beta_{m,n}|$ by using the imprecise adder $\mathbf{a}_{\text{imp}}$. We use $\delta = 1$, which is the optimal value for the exact OMS decoding.

For the NMS decoder (Section III-C), we use $\lambda = 0.75$ and the multiplication between λ and $\beta_{m,n}$ is implemented as the sum $0.75 \cdot \beta_{m,n} = \mathbf{a}_{\text{imp}}(\beta_{m,n}, \beta_{m,n}/2)$, where the exact value of $\beta_{m,n}/2$ is obtained by a right-shift of bits of $\beta_{m,n}$.

Finally, we note that the SCMS decoder does not use any extra arithmetic operation compared to the MS decoder.

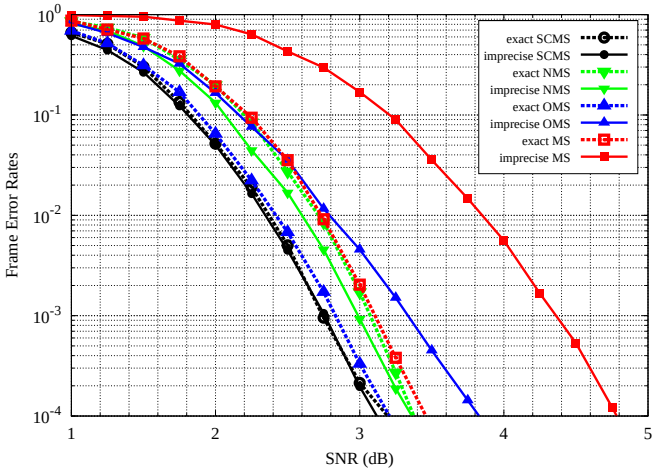


Fig. 5: Frame Error Rate for the (504, 252) regular code

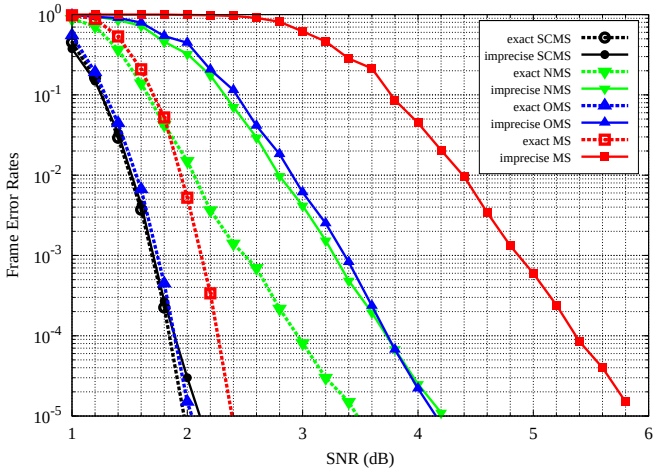


Fig. 6: Frame Error Rate for the IEEE 802.16e code

V. Simulation Results

The performance of Min-Sum-based decoders has been evaluated for two different codes.

- The first code is a short $(504, 252)$ LDPC code, constructed by Mackay and available online at [28]. It is a regular code, with all variable-nodes of degree 3 and all check-nodes of degree 6.
- The second code is a longer $(2304, 1152)$ and irregular quasi-cyclic LDPC code, specified by the IEEE 802.16e (WiMAX) standard [29].

Both codes have been simulated over the Additive White Gaussian Noise (AWGN) channel with Quadrature Phase-Shift Keying (QPSK) modulation. The above MS-based decoders have been simulated for both exact and imprecise arithmetic components, and the maximum number of decoding iterations was fixed to 100. Imprecise arithmetic components have been simulated through lookup tables.

A. Decoders' performance

The Frame Error Rate (FER) performance of MacKay and WiMAX LDPC codes are shown respectively in Fig. 5 and Fig. 6. The FER curves of the MS, NMS, OMS and SCMS decoders are plotted respectively in red, green, blue and black. In addition, for each decoder, the dashed curve (empty markers) was obtained by using exact arithmetic components, and the solid curve (full markers) was obtained by using imprecise arithmetic components.

1) Analysis of results for the Mackay code: At FER $= 10^{-4}$ the use of the imprecise arithmetic results in a loss of about 1.3 dB for the MS decoder and 0.6 dB for the OMS. The imprecise SCMS provides almost the same performance as the exact SCMS, and even outperforms it in the error floor region, while the imprecise NMS outperforms the exact NMS in the waterfall region. Both SCMS and NMS can be considered robust to imprecise arithmetic components.

2) Analysis of results for the WiMAX code: At FER $= 10^{-4}$ the use of the imprecise arithmetic results in a loss of about 3.3 dB for the MS algorithm, 1.8 dB for the OMS algorithm, and only 0.8 dB for the NMS algorithm. As previously, the performance of the imprecise SCMS is almost identical to the exact SCMS.

The excellent performance of the SCMS decoder under imprecise arithmetic settings is explained by its inherent ability to detect unreliable messages during the iterative decoding process. These results bring empirical evidence that the SCMS decoder is able to provide efficient error protection even if it operates on imprecise hardware, which represents the main contribution of this paper.

B. Complexity analysis

The exact evaluation of the savings in energy, delay and area is out the scope of this paper (but will be addressed in future works). However, we include a complexity analysis of the different decoders, where the complexity is expressed in terms of the number of logic gates of the circuit.

Let C_a be the relative complexity of the imprecise adder with respect to the exact adder, and C_c be the relative complexity of the imprecise comparator with respect to the exact comparator. They are defined as follows:

$$C_a = \frac{\text{number of logic gates of imprecise adder}}{\text{number of logic gates of exact adder}}$$

$$C_c = \frac{\text{number of logic gates of imprecise comparator}}{\text{number of logic gates of exact comparator}}$$

For the imprecise adder and comparator designed in Section IV, we have $C_a = 0.86$ and $C_c = 0.57$.

In order to evaluate the gain in complexity when imprecise arithmetic is used, we consider the ratio between the number of arithmetic operations (additions and comparisons) for imprecise and exact arithmetic. Moreover, in case of imprecise arithmetic, the number of additions is weighted by C_a and the

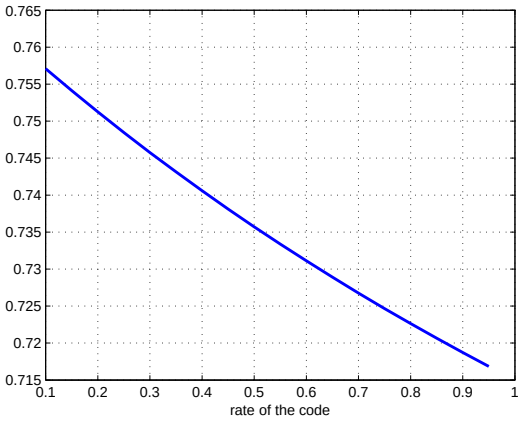


Fig. 7: Gain in complexity

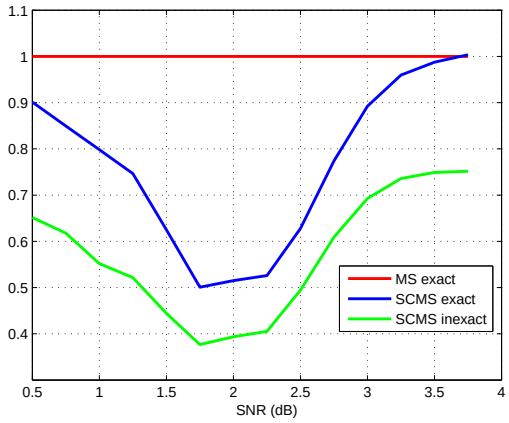


Fig. 8: Relative complexity for the (504,252) regular code

number of comparisons in weighted by C_c. Hence, we obtain:

$$\text{gain} = \frac{2(C_a + C_c)d_v - 3C_c(1-r)}{4d_v - 3(1-r)},$$

where d_v is the average variable-node degree and r is the coding rate. Fig. 7 shows the evolution of the gain when $d_v = 3$ and the rate r varies. As it can be seen, the gain decreases slowly from 0.715 to 0.71, as the rate increases from 0 to 1.

The average decoding complexity per codeword of the exact MS, the exact SCMS, and the imprecise SCMS has also been evaluated. This complexity is defined as the average number of arithmetic operations (additions and comparisons) required to decode 1 codeword. Again, in case of imprecise arithmetic, the number of additions is weighted by C_a and the number of comparisons in weighted by C_c. Fig. 8 and Fig. 9 plot the relative complexities of these decoders with respect to the exact MS (which is used as the reference). The complexity of the exact SCMS is much lower than those of the exact MS in the waterfall region, because SCMS needs in average a smaller number of decoding iterations. The imprecise SCMS exhibits the lowest complexity and the difference between its complexity and those of the exact SCMS complexity is due to the use of imprecise components.

VI. Future works

Future works will focus on two directions. The first direction is to evaluate the savings is energy, delay and area that can be obtained by using the imprecise arithmetic components. The complexity analysis from the above section gives us confidence on this direction, especially as the SCMS decoder is known to present several advantages from an energy efficiency perspective [30].

The second direction is to design Min-Sum-based decoders able to provide reliable error correction on probabilistic devices. The current work gives some hints on which parts of the circuit must be better protected. We are currently investigating the performance of the four Min-Sum-based decoders on probabilistic devices and the first results are very promising.

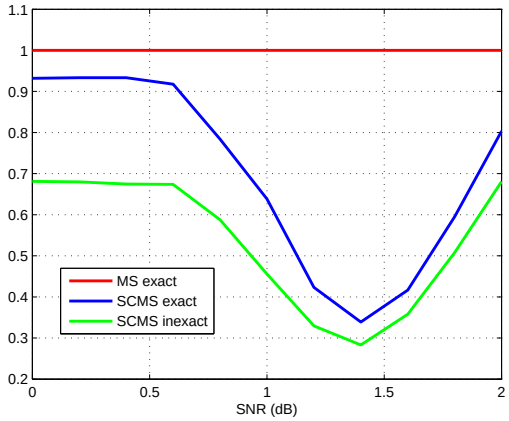


Fig. 9: Relative complexity for the IEEE 802.16e code

VII. Conclusion

In this paper we investigated the performance of several Min-Sum-based decoders on devices with imprecise arithmetic circuits. Imprecise adders and comparators have been specially design for this purpose, by pruning the exact circuits. The pruning operation has been constrained to meet specific requirements, such as to avoid a number of undesirable errors.

Simulation results have shown that MS, NMS, and OMS decoders manage to provide error protection, but the imprecise arithmetic circuits significantly degrade their performance. On the contrary, the imprecise SCMS proved to be robust to imprecise arithmetic circuits, due to its ability to detect unreliable messages during the decoding process. Hence, this work demonstrated that the SCMS decoder provides *efficient* error protection on devices with imprecise arithmetic circuits.

In addition, the complexity of the decoders has been evaluated in terms of number of logic gates of the circuits. For a code rate of $1/2$ the use of imprecise arithmetic circuits yields a complexity decrease of about 27%. Moreover, the imprecise SCMS provides a complexity reduction between 30% and 60% with respected to the exact MS decoder.

VIII. Acknowledgment

This work was supported by the Seventh Framework Programme of the European Union, under Grant Agreement number 309129 (i-RISC project).

References

[1] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. on Inf. Theory*, vol. 47, no. 2, pp. 599–618, 2001.

[2] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. on Information Theory*, vol. 47, no. 2, pp. 619–637, 2001.

[3] P. Grover and A. Sahai, "Green codes: Energy-efficient short-range communication," in *IEEE Int. Symp. on Inf. Theory (ISIT)*, 2008, pp. 1178–1182.

[4] A. Sahai and P. Grover, "The price of certainty: "waterslide curves" and the gap to capacity," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-1, Jan 2008. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-1.html

[5] K. Palem, "Energy aware computing through probabilistic switching: A study of limits," *IEEE Trans. on Computers*, vol. 54, no. 9, pp. 1123–1137, 2005.

[6] L. N. B. Chakrapani, K. K. Muntimadugu, A. Lingamneni, J. George, and K. V. Palem, "Highly energy and performance efficient embedded computing through approximately correct arithmetic: A mathematical foundation and preliminary experimental validation," in *Proc. of Int. Conf. on Compilers, Architectures and Synthesis for Embedded Systems*. ACM, 2008, pp. 187–196.

[7] B. E. S. Akgul, L. N. Chakrapani, P. Korkmaz, and K. V. Palem, "Probabilistic cmos technology: A survey and future directions," in *IFIP Int. Conf. on Very Large Scale Integration*. IEEE, 2006, pp. 1–6.

[8] A. Lingamneni, C. Enz, J. L. Nagel, K. Palem, and C. Piguet, "Energy parsimonious circuit design through probabilistic pruning," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2011, pp. 1–6.

[9] I. Chong and A. Ortega, "Hardware testing for error tolerant multimedia compression based on linear transforms," in *IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems (DFT)*, 2005, pp. 523–531.

[10] H. Chung and A. Ortega, "Analysis and testing for error tolerant motion estimation," in *IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems (DFT)*, 2005, pp. 514–522.

[11] N. Zhu, W. Goh, W. Zhang, K. Yeo, and Z. Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 8, pp. 1225–1229, 2010.

[12] R. G. Gallager, "Low density parity check codes," MIT Press, Cambridge, 1963, research Monograph series.

[13] L. Varshney, "Performance of ldpc codes under faulty iterative decoding," *IEEE Trans. on Information Theory*, vol. 57, no. 7, pp. 4427–4444, 2011.

[14] S. M. S. Tabatabaei, H. Cho, S. Mitra, and L. Dolecek, "Gallager B decoder on noisy hardware," *IEEE Trans. on Communications*, 2012, accepted for publication.

[15] S. M. S. Tabatabaei, S. Huang, and L. Dolecek, "Optimal design of a Gallager B noisy decoder for irregular LDPC codes," *IEEE Comm. Letters*, 2012, accepted for publication.

[16] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. on Inf. Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[17] J. Pearl, "Reverend Bayes on inference engines: A distributed hierarchical approach," in *Proc. of the 2nd National Conference on Artificial Intelligence (AAAI-82)*, 1982, pp. 133–136.

[18] ——, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers, 1988.

[19] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. on Communications*, vol. 47, no. 5, pp. 673–680, 1999.

[20] E. Eleftheriou, T. Mittelholzer, and A. Dholakia, "Reduced-complexity decoding algorithm for low-density parity-check codes," *IET Electronics Letters*, vol. 37, no. 2, pp. 102–104, 2001.

[21] J. Chen and M. P. Fossorier, "Near optimum universal belief propagation based decoding of low density parity check codes," *IEEE Trans. on Communications*, vol. 50, no. 3, pp. 406–414, 2002.

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

[23] J. Chen, R. Tanner, C. Jones, and Y. Li, "Improved min-sum decoding algorithms for irregular LDPC codes," in *IEEE Int. Symp. on Inf. Theory (ISIT)*, 2005, pp. 449–453.

[24] J. Zhang, M. Fossorier, and D. Gu, "Two-dimensional correction for min-sum decoding of irregular LDPC codes," *IEEE Communications Letters*, vol. 10, no. 3, pp. 180–182, 2006.

[25] V. Savin, "Self-corrected min-sum decoding of LDPC codes," in *IEEE Int. Symp. on Inf. Theory (ISIT)*, 2008, pp. 146–150.

[26] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Likoping University, Sweden, 1996.

[27] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Tran. on Computers*, vol. C-22, no. 8, pp. 786–793, 1973.

[28] D. J. MacKay. Encyclopedia of sparse graph codes. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

[29] IEEE-802.16e, "Physical and medium access control layers for combined fixe and mobile operation in licensed bands," 2005, amendment to Air Interface for Fixed Broadband Wireless Access Systems.

[30] E. Amador, V. Rezard, and R. Pacalet, "Energy efficiency of SISO algorithms for turbo-decoding message-passing LDPC decoders," in *17th IFIP Int. Conf. on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2009, pp. 95–100.