Practical LDPC Encoders Robust to Hardware Errors

Elsa DUPRAZ[1,4], Valentin Savin[2], Satish Kumar Grandhi[3], Emanuel Popovici[3], and David DECLERCQ[4]

[1] Telecom Bretagne; UMR CNRS 6285 Lab-STICC, Brest, France
[2] CEA-LETI, Minatec Campus, Grenoble, France
[3] Department of Electrical and Electronic Engineering, University College Cork, Cork, Ireland
[4] ETIS - ENSEA / Univ. Cergy-Pontoise / CNRS UMR-8051, Cergy-Pontoise, France

***Abstract*—LDPC decoders on faulty hardware have received increasing attention over the last few years, mainly motivated by reliability issues in emerging nanotechnologies. As a main result, it was shown that LDPC decoders are naturally robust to hardware faults. LDPC encoders on faulty hardware have received less attention, and they are expected to be less robust to hardware faults. In this work, we propose an LDPC encoding solution that is robust to faulty hardware. Our encoding solution is composed of two steps. First, an Augmented Encoding method is proposed, which consists in computing an augmented codeword that contains both the codeword to be transmitted on the channel and extra parity bits. The augmented codeword is computed from a noisy encoding circuit, and then corrected by a noisy Gallager-B decoder before channel transmission. The augmented codeword is obtained from a rate-compatible construction that guarantees good decoding performance both for the augmented codeword and for the codeword to be transmitted on the channel. In order to further improve the robustness of our encoding solution, we propose a second step, consisting of a circuit-level optimization. We propose to identify the critical gates that are responsible for encoding failures, and to duplicate them in order to reduce their influence on encoding outputs. Based on Monte-Carlo simulation, we show that the proposed solution significantly improves the encoding robustness to hardware faults.**

I. INTRODUCTION

Over the past few years, reliability has become a major issue in the design of electronic devices. The expected increase in density integration coupled with significant chip size reduction will make the next generations of electronic devices much more sensitive to noise [1]. As a consequence, in the future systems of communication and storage, errors may not only come from the transmission channels, but also from the faulty hardware. In this context, there is a need to evaluate the robustness of Low Density Parity Check (LDPC) encoders and decoders running on faulty hardware.

Several recent works were devoted to the analysis and design of LDPC decoders robust to hardware faults. Hard decoders were analyzed in [2] (Gallager A) and [3] (Gallager B) by using noisy density evolution [2], while soft decoders were considered in [2] (Belief Propagation), and [4] (quantized Min-Sum). Also, the design of Finite Alphabet Iterative Decoders (FAIDs) robuts to hardware faults was proposed in [5].

For the encoding part, the level of hardware noise that can be tolerated by the LDPC encoder has been evaluated from an information theoretic perspective in [6]. However, the results of [6] do not indicate how to construct an encoder robust to hardware faults. From a practical point of view, [7] shows that most of the standard encoding solutions (systematic, lower triangular, encoding for Zig-Zag and QC-codes, etc.) completely fail under faulty hardware settings.

A robust LDPC encoding solution was proposed in [8], [9] that consists of computing extra parity bits in addition to the original codeword in order to protect the encoder. Several LDPC decoders are then embedded in the encoder, in order to correct gradually the faults introduced by the hardware. The authors of [8], [9] provide a theoretical analysis of the robustness of the proposed solution and, from this analysis, determine the hardware noise conditions that permit robust encoding. However, the proposed solution is computationally expensive, due to multiple decoding stages. Further, [8], [9] do not propose any code construction for the extra parity bits, nor practical implementation of the solution.

The objective of this paper is to propose a practical LDPC encoder robust to faulty hardware. The design of our robust encoder is composed of two parts.

We first introduce the Augmented Encoding approach that consists of computing an augmented codeword that contains both the original codeword and additional parity bits. Before channel transmission, the augmented codeword is passed through one single LDPC decoder in order to eliminate the hardware faults. Here, we propose to compute the extra parity bits from a split-extended construction [10], which belongs to the family of rate-compatible LDPC codes. The split-extended construction ensures good decoding performance both for (i) the augmented codeword, that needs to be decoded at the transmitter side in order to correct the hardware faults, and (ii) the original codeword, that needs to be decoded at the receiver side in order to correct the transmission errors.

The second part of our robust encoding solution consists of a design procedure of the circuit implementing the Augmented Encoder. We identify the *critical gates* of the circuit, in the sense that only one single error at the output of such gates may propagate to a large number of outputs of the circuit. We then propose an algorithm to duplicate the critical gates, so as to limit their error propagation impact. Based on Monte-Carlo simulations, we show that gate duplication combined with Augmented Encoding greatly improves the robustness of

the encoder to hardware faults.

The outline of the paper is as follows. Section II introduces our notation for LDPC codes and the error model we consider to represent hardware faults on the encoder. Section III presents the Augmented Encoding approach. Section IV introduces the gate duplication procedure. Section V shows the Monte Carlo simulation results.

II. LDPC codes and Error Models

In this section, we first introduce our notation for LDPC codes and present the LDPC encoding problem. We then describe the gate error model that represents the faulty hardware effect on the encoder.

A. LDPC Codes

Denote by H a binary parity check matrix of size $(m \times n)$. An LDPC code is defined as the null-space of the parity check matrix H. A binary vector $\mathbf{x}$ of length n is a codeword if and only if it verifies

$$H\mathbf{x}^{\mathrm{T}} = \mathbf{0}. \tag{1}$$

With LDPC codes, the parity check matrix H is sparse and designed such as to ensure good error correction performance under iterative message passing decoding, see [11]. Once H is fixed, the corresponding encoder has to be constructed.

B. LDPC Encoders

Denote by $\mathbf{u}$ the information sequence of length k. For the sake of simplicity, we shall assume that H is full rank and therefore $k = n - m$. The encoding operation is given by a function $\mathcal{E} : \{0,1\}^k \rightarrow \{0,1\}^n$ such that

$$\mathbf{x} = \mathcal{E}(\mathbf{u}). \tag{2}$$

The encoding function $\mathcal{E}$ has to transform the information sequence $\mathbf{u}$ into a codeword $\mathbf{x}$ that satisfies (1). In order to perform the encoding, one can construct a generator matrix G of size $(k \times n)$ that verifies $HG^{\mathrm{T}} = 0$. The generator matrix G can be obtained from H by Gaussian elimination, as described in [12]. The encoding operation is then given by

$$\mathbf{x} = \mathbf{u}G. \tag{3}$$

As was shown in [12], the matrix G is not sparse in general, and the complexity of the encoding operation (3) scales as $O(n^2)$.

We denote by $\mathcal{C}_{\mathcal{E}}$ the Boolean combinational circuit implementing the function $\mathcal{E}$ in (2). The circuit $\mathcal{C}_{\mathcal{E}}$ consists of Boolean combinational elements connected by wires. The circuit $\mathcal{C}_{\mathcal{E}}$ can be seen as a directed acyclic graph with k source nodes corresponding to primary inputs $\{u_1, \cdots, u_k\}$, n sink nodes corresponding to primary outputs $\{x_1, \cdots, x_n\}$, and a colection of L internal nodes corresponding to logic gates $\mathcal{C} = \{c_1, ..., c_L\}$. The combinational circuit is divided into a number of processing levels. Processing level-0 consists of source nodes (primary inputs), level-1 is the set of nodes with all predecessors in level-0, and recursively, level-t is the set of nodes with all predecessors in any level between 0 and $(t-1)$. We denote by P_ℓ the processing level of gate c_ℓ. We also denote by $\mathbf{I}_\ell$ the list of predecessors of gate c_ℓ, and by N_ℓ the number of elements of $\mathbf{I}_\ell$. We consider that the primary outputs $x_1, \cdots, x_n$ of the circuit are taken to be the outputs of n particular gates $\{r_1, ..., r_n\} \subset \mathcal{C}$, referred to as the output gates.

In order to obtain a Boolean circuit description of the encoding operation (2), we consider the encoding from the generator matrix (3) and we apply a logic synthesis tool to the encoding function $\mathcal{E}$ given by the generator matrix G. The logic synthesis tool outputs a Boolean combinational circuit implementing the function $\mathcal{E}$ and performs gate factorization on the circuit in order to reduce its complexity[1].

Note that several particular code constructions, such as Zig-Zag [13] or Quasy-cyclic [14] have specific encoding solutions. In this paper, we will not consider these specific code constructions and we will focus on the general solution that applies to any parity check matrix H. Before discussing the robustness to hardware faults of the previously described encoding solutions, we first present the error model we consider to represent the faulty hardware effect on the encoder.

C. Gate Error Model

Since the Boolean circuit description $\mathcal{C}_{\mathcal{E}}$ implements a linear function, it can be synthesized from XOR gates only. In order to have a fair complexity comparison between different encoding solutions, we will further assume that the Boolean circuit $\mathcal{C}_{\mathcal{E}}$ is synthesized using 2-input XOR-gates only. Consequently, we assume that hardware faults are introduced at the 2-input XOR gate level. Denote by p_{xor} the error probability of a 2-inputs XOR gate. The faulty 2-inputs XOR operator $\tilde{\oplus}$ is defined as

$$a \,\tilde{\oplus}\, b = \begin{cases} a \oplus b & \text{with prob. } 1 - p_{\text{xor}}, \\ 1 \oplus (a \oplus b) & \text{with prob. } p_{\text{xor}}, \end{cases} \tag{4}$$

where a and b are binary digits and $a \oplus b$ is the (perfect) XOR sum of a and b. The error model described in (4) is memoryless and data-independent. It is considered here as a first step of the analysis, and more accurate models will be considered in future works. At the end, the output vector of the noisy Boolean circuit $\mathcal{C}_{\mathcal{E}}$ is denoted $\tilde{\mathbf{x}}$. Note that due to hardware faults, $\tilde{\mathbf{x}}$ is not necessarily a codeword.

Under the gate error model (4), it was shown in [7] that almost all the existing encoding solutions [12]–[14] completely fail under hardware faults. For the encoding from the generator matrix (3) the high number of gates involved in the computation implies a high encoding error probability. As an example, a gate error probability $p_{\text{xor}} = 10^{-3}$ with a $(3, 6)$-code gives encoding error probabilities $P_e = 0.15$ for the encoding from the generator matrix. The encoding error probability P_e is even higher than the optimistic code ensemble threshold for the $(3, 6)$-code [11], making the channel error correction impossible.

[1]Here, the logic synthesis tool we use is *Synopsys Design Compiler* http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DCGraphical/Documents/dc_graphical_ds.pdf.

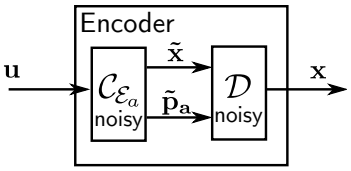

Fig. 1. Augmented Encoding

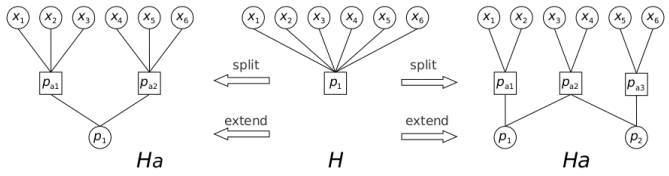

Fig. 2. Split-Extension examples

In this paper, we propose two methods in order to significantly reduce the encoding error probability prior to transmission over the channel. We first introduce the Augmented Encoding approach which consists of computing extra parity bits in order to protect the encoding operation. We then propose the gate duplication method that consists of duplicating the critical gates of the Augmented Encoding circuit in order to reduce the error propagation effect.

III. Augmented Encoding Approach

This section introduces the Augmented Encoding approach, that consists in computing an augmented codeword that contains both the original codeword to be transmitted on the channel and extra parity bits to protect the encoder from hardware faults. Before channel transmission, the augmented codeword is passed through an LDPC decoder in order to eliminate the hardware faults from $\mathbf{x}$. The original codeword $\mathbf{x}$ is then transmitted on the channel. The extra parity bits are computed from a Split-Extended construction, as we also describe.

A. The Augmented Encoding Approach

The Augmented Encoder is depicted in Figure 1. The Augmented Encoding approach consists of computing an augmented codeword $\mathbf{x}_a = [\mathbf{x}, \mathbf{p}_a]$ of length $n + n_a$ that is composed of the original codeword $\mathbf{x}$ and of n_a extra parity bits $\mathbf{p}_a$. Denote by H_a the parity check matrix of size $(m + n_a) \times (n + n_a)$ of the augmented code. The augmented codeword $\mathbf{x}_a$ then satisfies

$$H_a \mathbf{x}_a^{\mathrm{T}} = \mathbf{0}. \tag{5}$$

The augmented codeword is computed from the information sequence $\mathbf{u}$ as

$$\mathbf{x}_a = \mathcal{E}_a(\mathbf{u}) \tag{6}$$

where the function $\mathcal{E}_a : \{0,1\}^k \to \{0,1\}^{n+n_a}$ represents the encoding operation. We denote by $\tilde{x}_a$ the noisy version of the augmented codeword, which is the output of the (noisy) circuit $\mathcal{C}_{\mathcal{E}_a}$ implementing the Augmented Encoding operation.

Before transmission over the channel, the noisy augmented codeword $\tilde{\mathbf{x}}_a$ is passed through an LDPC decoder $\mathcal{D}$. The LDPC decoder has to correct the errors that were introduced by the hardware during the encoding operation. Here, the LDPC decoder will be a Gallager B decoder. As the Gallager B decoder runs on the same hardware as the encoder, it will be assumed noisy as well. The Gallager B decoder works with binary messages and is constructed from XOR and majority logic gates. The XOR gates of the Gallager B decoder can be decomposed into 2-input XOR gates which are assumed to be noisy according to the error model described in Section II-C. The majority logic gates can be decomposed as 2-input NAND and OR gates which are also assumed to follow the error model of Section II-C (defined according to NAND and OR gates, but with the same gate error probability value p_{xor}).

The LDPC decoder used at the transmitter side has to correct the hardware-induced errors on the augmented codeword $\tilde{\mathbf{x}}_a$, but only the original codeword $\mathbf{x}$ is transmitted through the channel. It is worth noting that as long as the encoding error probability on the transmitted codeword $\mathbf{x}$ is negligible with respect to the channel error probability (e.g. orders of magnitude below) the proposed encoding solution can be considered as being robust to hardware noise.

Both the LDPC code that produces $\mathbf{x}$ and the one that produces $\mathbf{x}_a$ have to lead to good decoding performance, with increased correction capabilities for the augmented code. In order to satisfy these two requirements, we consider the Split-Extended construction initially introduced in [10] in the context of coded cooperation.

B. Split-Extended Construction

We now describe the Split-Extend construction that will be used to generate the extra parity bits $\mathbf{p}_a$. The original codeword $\mathbf{x}$ satisfies all the parity-check equations defined by the rows of the parity check matrix H of the original LDPC code. The extra parity bits $\mathbf{p}_a$ can be computed by splitting the parity check equations of H.

Figure 2 provides two examples of the split of a parity check equation of H. On the middle part of Figure 2, the parity check equation

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 0 \tag{7}$$

corresponds to a row of H. On the left part of Figure 2, a new parity bit p_{a_1} is generated by splitting the original parity check into two sub-checks as follows. The set of bits connected to the check-node is partitioned into two subsets, and the parity bit p_{a_1} is generated as the sum of the bits of one of the two subsets. The splitting then gives two sub parity check equations

$$x_1 + x_2 + x_3 + p_{a_1} = 0, \quad p_{a_1} + x_4 + x_5 + x_6 = 0. \tag{8}$$

On the right part of Figure 2, applying the same principle, two new parity bits p_{a_1} and p_{a_2} are generated by splitting the original parity-check into three parity-checks. The set of bits connected to the check node is partitioned into three subsets, and p_{a_1} is generated as the sum of the bits in the first subset.

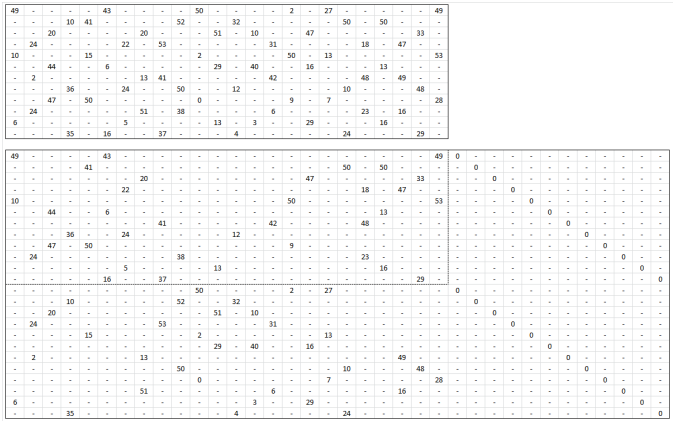

Fig. 3. Base matrix of the i-RISC QC-LDPC code with $d_v = 3$ and $r = 1/2$ (top), and corresponding split-extended base matrix (bottom)

Subsequently, p_{a_2} can be generated either as the sum of p_{a_1} and the bits in the second subset, or as the sum of the bits in the third subset. The matrix defined by these new parity check equations is denoted by H_a.

With this construction, the augmented codeword $\mathbf{x}_a$ verifies $H_a \mathbf{x}_a^{\mathrm{T}} = 0$ (5) while the original codeword $\mathbf{x}$ still satisfies $H\mathbf{x}^{\mathrm{T}} = 0$ (1). An advantage of this construction is that it can be applied to any preexisting parity check matrix H, in order to generate a number of extra parity bits that allows increasing the error correction capacity of the code. The total number n_a of extra parity bits $\mathbf{p}_a$ depends on the number of rows of H and on the number of extra bits generated for each row of H. The number of extra bits generated for each row may vary from one row to another. More details on the split-extended construction and on how to design the number of extra bits generated for each row of H, depending on noise conditions, can be found in [10].

At the end, the encoding function $\mathcal{E}_a$ can be obtained by computing the generator matrix G_a corresponding to the split-extended parity check matrix H_a, as explained in Section II-B. The logic synthesis tool is then applied to G_a to generate the corresponding circuit $\mathcal{C}_{\mathcal{E}_a}$. Some gates of $\mathcal{C}_{\mathcal{E}_a}$ may be critical, in the sense that injecting only one fault at the output of such gates may result in a very large number of errors at the output of the circuit. In the following, we propose to modify the encoding circuit $\mathcal{C}_{\mathcal{E}_a}$ by duplicating the critigal gates, so as to limit their impact on the primary outputs of the circuit.

IV. Gate duplication

A gate of the encoding circuit is critical when the output of the gate is propagated to a large number of primary outputs of the circuit. The criticality degree of a gate c_ℓ, denoted by $\mathrm{cdeg}(c_\ell)$, is defined as the number of primary outputs to which c_ℓ is connected by at least one path. Thus, injecting one error in gate c_ℓ will produce at most $\mathrm{cdeg}(c_\ell)$ errors on the output.

In order to reduce the high error propagation effect induced by critical gates, we propose an algorithm to duplicate the gates with high criticality degree. In this section, we introduce the algorithm we propose for gate duplication and provide a theoretical analysis of the algorithm.

Algorithm 1 Node duplication

Fix $\mathrm{CT} > 1$
while $\max_\ell \mathrm{cdeg}(c_\ell) > \mathrm{CT}$ **do**
  Select the gate to duplicate
  $\mathcal{L}_c \leftarrow \underset{c_\ell \in \{c_1,\ldots,c_L\}}{\arg\max}\ \mathrm{cdeg}(c_\ell)$
  $\mathcal{C}_g \leftarrow \underset{c_\ell \in \mathcal{L}_c}{\arg\max}\ P_\ell$
  $c_g \leftarrow$ one gate at random in $\mathcal{C}_g$
  Identify the successors of c_g
  $\mathcal{S}_g \leftarrow \mathrm{succ}(c_g)$
  Separate the set of successors into two sets
  $\mathcal{S}_0 \leftarrow \mathcal{S}_{g,1}$, $\mathcal{S}_1 \leftarrow \emptyset$, $\mathcal{B}_0 \leftarrow \mathcal{B}_{g,1}$, $\mathcal{B}_1 \leftarrow \emptyset$
  for $j = 2$ to $\mathrm{length}(\mathcal{S}_g)$ **do**
    if $\mathrm{length}(\mathcal{B}_0 \cap \mathcal{B}_{g,j}) > \mathrm{length}(\mathcal{B}_1 \cap \mathcal{B}_{g,j})$ **then**
      $\mathcal{S}_1 \leftarrow \mathcal{S}_1 \cup \{c_j\}, \quad \mathcal{B}_1 \leftarrow \mathcal{B}_1 \cap \mathcal{B}_{g,j}$
    else
      $\mathcal{S}_0 \leftarrow \mathcal{S}_0 \cup \{c_j\}, \quad \mathcal{B}_0 \leftarrow \mathcal{B}_0 \cap \mathcal{B}_{g,j}$
    end if
  end for
  Reconstruct the circuit with the new gate
  $\mathcal{C} \leftarrow \{c_1, \ldots, c_{L+1}\}$
  $T_{L+1} = T_g$, $N_{L+1} = N_g$, $\mathbf{I}_{L+1} = \mathbf{I}_g$
  $\mathcal{S}_g = \mathcal{S}_0$, $\mathcal{S}_{L+1} = \mathcal{S}_1$, $O_{L+1} = O_g + 1$
  for $\ell = 1$ to L **do**
    if $O_\ell > O_g$ **then**
      $O_\ell = O_\ell + 1$
    end if
  end for
  $L = L + 1$
end while

A. Duplication Algorithm

We fix a criticality threshold value $\mathrm{CT} > 0$. In the duplication algorithm, the gates with criticality degree $\mathrm{cdeg}(c_\ell) > \mathrm{CT}$ will be duplicated until the following condition is reached

$$\max_{c_\ell} \mathrm{cdeg}(c_\ell) \leq \mathrm{CT}. \tag{9}$$

The gate duplication algorithm relies on the circuit description $\mathcal{C}_{\mathcal{E}_a}$ of the encoding function $\mathcal{E}_a$. We denote $\mathcal{B}_\ell$ the list of length $\mathrm{cdeg}(c_\ell)$ of primary outputs to which gate c_ℓ is connected by at least one path.

Our gate duplication algorithm is given in Algorithm 1. At each step of Algorithm 1, one gate $c_g \in \{c_1, \ldots, c_L\}$ is selected for duplication as follows. Among the set $\mathcal{L}_c$ of gates with maximum criticality degree $\mathrm{cdeg}(c_\ell)$, we identify the set $\mathcal{C}_g \subseteq \mathcal{L}_c$ of gates with the highest processing level P_ℓ. We select the gate c_g to duplicate at random in $\mathcal{C}_g$.

We then identify the set $\mathcal{S}_g$ of direct successors of the selected gate c_g, as well as the sets $\mathcal{B}_{g,j}$ of primary outputs to which all the successors $c_j \in \mathcal{S}_g$ are connected. We separate the set $\mathcal{S}_g$ of successors into two sets $\mathcal{S}_1$ and $\mathcal{S}_2$, with as much as possible equal number of primary outputs associated

to each set. To finish, we replace g_c with two gates; each gate is allocated one of the two sets of successors $\mathcal{S}_1$ or $\mathcal{S}_2$. We reconstruct the circuit with the new gate, and update the circuit parameters.

At the end, duplicating a gate will reduce the criticality degree of the two substituted gates, at the price of a complexity increase, that is an increase in the number of 2-input XOR gates involved in the circuit. In the following, we show that the complexity increase induced by the duplication algorithm is limited to only what is needed.

B. Analysis of the Algorithm

As a justification of the construction of Algorithm 1, we show that it has the following property.

Proposition 1: If Algorithm 1 is applied to any combinational circuit for a given value CT, the sum of criticality degrees

$$\sum_{\ell=1}^{L} \text{cdeg}(c_\ell) \tag{10}$$

for the resulting circuit does not depend on CT.

Proof: First consider the duplication of one single gate c_g in the circuit. Denote c_1 and c_2 the two gates resulting from the duplication of c_g. The proof of the Proposition is done by contradiction.

First, if $\text{cdeg}(c_g) > \text{cdeg}(c_1)+\text{cdeg}(c_2)$, then one or several primary outputs to which c_g was connected is not connected to c_1, nor to c_2, which is impossible. We thus get

$$\text{cdeg}(c_g) \leq \text{cdeg}(c_1) + \text{cdeg}(c_2). \tag{11}$$

Second, let $\mathcal{B}_1$ and $\mathcal{B}_2$ be the sets of primary outputs to which c_1 and c_2 are connected, respectively. Condition $\text{cdeg}(c_g) < \text{cdeg}(c_1) + \text{cdeg}(c_2)$ is verified only if $\mathcal{B}_1$ and $\mathcal{B}_2$ are not disjoint. Since the combinational circuit is a directed acyclic graph, $\mathcal{B}_1$ and $\mathcal{B}_2$ are not disjoint in the two following cases: (i) there is a reconvergent fan-out in the circuit, which means that the signal splits into two and then reconverges, (ii) c_g has only one single successor $c_{g'}$.

Case (i) cannot appear in our case since the circuit is composed by 2-input XOR gates only. In case (ii), $c_{g'}$ is such that $\text{cdeg}(c_{g'}) \geq \text{cdeg}(c_g)$. Condition $\text{cdeg}(c_{g'}) > \text{cdeg}(c_g)$ is impossible, since from Algorithm 1 all the gates c_ℓ with $\text{cdeg}(c_\ell) > \text{cdeg}(c_g)$ have already been duplicated. Condition $\text{cdeg}(c_{g'}) = \text{cdeg}(c_g)$ is impossible neither, since $c_{g'}$ has necessarily a processing level $P_{g'}$ higher than P_g and should already have been duplicated by Algorithm 1. Thus, case (ii) cannot appear as well, and we get

$$\text{cdeg}(c_g) \geq \text{cdeg}(c_1) + \text{cdeg}(c_2) \tag{12}$$

From conditions (11) and (12), we get

$$\text{cdeg}(c_g) = \text{cdeg}(c_1) + \text{cdeg}(c_2), \tag{13}$$

which shows the Proposition. ∎

Proposition 1 shows that the algorithm we propose for gate duplication does not add any needless gate and therefore limits the complexity increase induced by gate duplication to only what is needed.

At the end, gate duplication will reduce the effect of error propagation at the price of a complexity increase. In the following, we evaluate through Monte Carlo simulations the encoding error probability of the proposed Augmented Encoding and gate duplication methods. In particular, we analyze the tradeoff between encoding error probability and circuit complexity.

V. Experimental Results

In this section we evaluate the performance of Augmented Encoding for the two following QC-LDPC codes

- dv3-r12 with $d_v = 3$, $r = 1/2$, $m = 646$, $n = 1292$,
- dv4-r12 with $d_v = 4$, $r = 1/2$, $m = 645$, $n = 1290$.

The two considered code are regular, and d_v represents the variable node degree of the code. The extra parity bits are obtained from the Split-Extended construction. The two codes dv3-r12 and dv4-r12 are such that $n+n_a = 1944$. For the two considered codes, it corresponds to splitting each parity check equation into two sub parity check equations, see Figure 3 for the code dv3-r12.

In the following, we evaluate the performance of our LDPC encoder for the two codes in terms of encoding error probability and of complexity.

A. Error Probability Analysis

In this section, we consider the two codes, and measure their Bit Error Rate (BER) with respect to the gate error probability p_{xor}. The performance is measured in terms of Bit Error Rate (BER) rather than in Frame Error Rate (FER). Indeed, at the encoding part, the objective is not to retrieve the actual codeword, but to drastically diminish the amount of errors in the codeword before channel transmission.

For the dv4-r12 code, Figure 4 represents the BER with respect to the gate error probability p_{xor} for various values of criticality threshold CT. Here, the BER is measured after the noisy Gallager B LDPC decoder used at the transmitter side, except for the case without Augmented Encoding, where it is measured at the end of the Boolean circuit.

In Figure 4, the first upper curve represents the case without Augmented Encoding. In our simulations, we observed that this curve does not change with gate duplication. Indeed, when the value of CT is high, depending on the frames, the error probability at the end of the Boolean circuit is either very small, or very high due to error propagation, which makes the LDPC decoder fail. When the value of CT is low, the error probability is the same in average, but it is less balanced from one frame to one another.

In Figure 4, we see that the curve with Augmented Encoding but without gate duplication is still very close to the uncoded curve, due to high error propagation. Then, when the value of CT decreases, the BER considerably reduces. Even for CT$= 50$, for $p_{\text{xor}} = 10^{-5}$, the BER reduces by a factor 10^3 compared to the case without Augmented Encoding. By setting CT$= 25$, we even gain a new decade in BER.

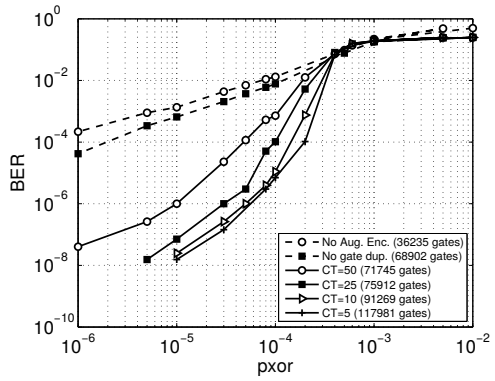

Fig. 4. Output BER with respect to p_{xor}, dv4r12 code

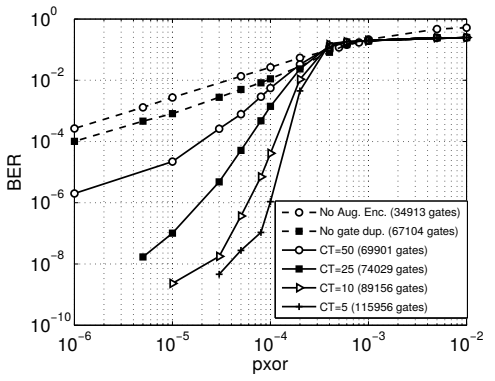

Fig. 5. Output BER with respect to p_{xor}, dv3r12 code

Figure 5 represents the BER with respect to p_{xor} for the dv3-r12 code. For CT$=50$, the BER is higher than for the previous code. On the other hand, the BER is approximately the same for CT$=25$, and it is lower for CT$=10$ and CT$=5$. At the end, Augmented encoding combined with gate duplication greatly improves the robustness of the encoding, at the price of a complexity increase, as we now discuss.

B. Complexity Analysis

In this section, we analyze the complexity of the combinatorial circuits that realize the encoding. The complexity is evaluated in terms of number of 2-input XOR gates involved in each Boolean circuit. For each considered setup, the number of gates is given in the legends of Figure 4 for the dv4-r12 code, and of Figure 5 for the dv3-r12 code.

As expected, the Augmented Encoding increases the number of gates needed by the encoder. The increase in complexity corresponds to approximately a factor 2, which is expected since every parity check equation of H is splitted into two parity check equations in H_a. Note that, for a complete complexity evaluation of the Augmented Encoder, one should also take into account the LDPC decoder used at the transmitter side.

Concerning gate duplication, for both codes, we see that the number of gates involved in the circuit does not increase much with CT, except for CT$=5$. Consequently, in terms of complexity, it is worthwhile to apply gate duplication, at least for values of CT higher than 10. For both codes, setting CT$=25$ appears to be a good tradeoff between BER performance and complexity of the circuit.

VI. Conclusion

In this paper, we proposed an LDPC encoding solution robust to hardware errors. The proposed Augmented Encoding solution consists of computing an augmented codeword that contains both the codeword to be transmitted on the channel and extra parity bits. The extra parity bits used in Augmented Encoding are obtained from a split-extended code construction. As a second part of our robust LDPC encoding solution, we proposed a gate duplication algorithm that identifies and duplicates the gates of the encoding circuit responsible for high error propagation. Through finite length simulations, we showed that Augmented Encoding combined with gate duplication greatly improves the robustness of the encoder.

Acknowlegement

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

References

[1] S. Zaynoun, M. Khairy, A. Eltawil, F. Kurdahi, and A. Khajeh, "Fast error aware model for arithmetic and logic circuits," in *IEEE 30th International Conference on Computer Design (ICCD)*. IEEE, 2012, pp. 322–328.

[2] L. Varshney, "Performance of LDPC Codes Under Faulty Iterative Decoding," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4427–4444, 2011.

[3] C.-H. Huang, Y. Li, and L. Dolecek, "Gallager B LDPC Decoder with Transient and Permanent Errors," *IEEE Transactions on Communications*, vol. 62, no. 1, pp. 15–28, 2014.

[4] C. K. Ngassa, V. Savin, E. Dupraz, and D. Declercq, "Density Evolution and Functional Threshold for the Noisy Min-Sum Decoder," *IEEE Transactions on Communications*, vol. 63, no. 5, pp. 1497–1509, 2015.

[5] E. Dupraz, D. Declercq, B. Vasic, and V. Savin, "Analysis and design of finite alphabet iterative decoders robust to faulty hardware," *IEEE Transactions on Communications*, vol. 63, no. 8, pp. 2797–2809, 2015.

[6] J. Hachem, I. Wang, C. Fragouli, S. Diggavi *et al.*, "Coding with encoding uncertainty," in *Proceedings of International Symposium on Information Theory*, 2013, pp. 276–280.

[7] E. Dupraz and D. Declercq, "Evaluation of the robustness of LDPC encoders to hardware noise," in *IEEE International Black Sea Conference on Communications and Networking*, 2015, pp. 87–91.

[8] Y. Yang, P. Grover, and S. Kar, "Can a noisy encoder be used to communicate reliably?" in *52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2014, pp. 659–666.

[9] ——, "Computing linear transformations with unreliable components," *arXiv preprint arXiv:1506.07234*, 2015.

[10] V. Savin, "Split-extended LDPC codes for coded cooperation," in *International Symposium on Information Theory and its Applications (ISITA)*, 2010, pp. 151–156.

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

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

[13] L. Ping, X. Huang, and N. Phamdo, "Zigzag codes and concatenated zigzag codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 800–807, 2001.

[14] Z. Li, L. Chen, L. Zeng, S. Lin, and W. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Transactions on Communications*, vol. 53, no. 11, pp. 1973–1973, 2005.