A Probabilistic Parallel Bit-Flipping Decoder for Low-Density Parity-Check Codes

Khoa Le, Fakhreddine Ghaffari, *Member, IEEE*, Lounis Kessal, *Member, IEEE*, David Declercq, *Senior Member, IEEE*, Emmanuel Boutillon, *Senior Member, IEEE*, Chris Winstead, *Senior Member, IEEE*, and Bane Vasić, *Fellow, IEEE*

***Abstract*— This paper presents a new bit flipping (BF) decoder, called the probabilistic parallel BF (PPBF) for low-density parity-check codes on the binary symmetric channel. In the PPBF, the flipping operation is performed in a probabilistic manner which is shown to significantly improve the error correction performance. The advantage of the PPBF also comes from the fact that no global computation is required during the decoding process and that all the computations can be executed in the local computing units and in parallel. The PPBF provides a considerable improvement of the operating frequency and complexity, compared to other known BF decoders, while obtaining a significant gain in error correction. An improved version of the PPBF, called non-syndrome PPBF is also introduced, in which the global syndrome check is moved out of the critical path and a new terminating mechanism is proposed. In order to show the superiority of the new decoders in terms of hardware efficiency and decoding throughput, the corresponding hardware architectures are presented in Section II. The application-specific integrated circuit synthesis results confirm that the operating frequency of the proposed decoders is significantly improved, compared to that of the BF decoders in the literature while requiring lower complexity to be efficiently implemented.**

***Index Terms*— Low-density parity-check codes, iterative decoding, probabilistic bit flipping decoding, high decoding throughput, low-complexity implementation.**

I. Introduction

LOW-DENSITY Parity-Check (LDPC) codes were introduced by Gallager in 1963 and they have been adopted as a part of several standards, such as IEEE 802.11n, 802.16a, *etc.* due to their outstanding error correction capability [1], [2]. LDPC codes can be decoded by two classes of decoding algorithms: *soft-information* message passing algorithms, *e.g.*, Min-Sum (MS), Sum Product (SP) [3], or *hard-decision* algorithms such as Bit Flipping (BF) and Gallager-A,B [4], [5]. The soft-information decoding algorithms provide a very good decoding performance but require a large computation resources. They exhibit, therefore, very high complexity in hardware realization [3]. On the contrary, the hardware implementations of hard-decision decoders were shown to have low complexity thanks to the simple computation units and smaller connection networks, but they are weak in error correction. The increasing demand of massive data rates in several applications, such as optical communications, high-speed Ethernet [6], data storage devices [7]–[9] or the New Radio of 5G enhanced mobile broadband (eMBB) [10], will require higher decoding throughput. In such applications, hard decision decoders become the promising candidates thanks to their simple computations and hence, low complexity and high decoding throughput, provided that the decoding performance is improved. In this paper, we focus on the Bit Flipping (BF) hard decision decoder to not only enhancing the decoding throughput and reducing the decoder complexity but also improving the error correction performance. Furthermore, we focus on the decoders which require only the hard information from the channel. These decoders could be used when the channel soft-information is unable to obtain or requires a long latency to be generated, such as the storage system [11], [12].

The BF decoding concept is firstly introduced by Gallager [4]. It involves passing binary messages iteratively between two groups of nodes: the Variable Nodes (VNs) and the Check Nodes (CNs), and it uses the channel hard-information as the input values. The CN uses the exclusive-OR (XOR) operations. In each iteration, a VN is flipped if the number of unsatisfied neighboring CNs is higher than a predefined threshold. The BF has a very low complexity decoder but provides a weak error correction, compared to the soft-decision decoders. Several BF decoders have been latter proposed in literature, in which the Gradient Descent Bit Flipping (GDBF) and the Probabilistic Gradient Descent Bit Flipping (PGDBF), introduced by Rasheed *et al.* [13], could be seen as the most promising algorithms, in terms of error correction. In the GDBF, the CN operation is the XOR calculation as in the standard BF decoder, while VN computes a function called *inversion* or *energy* function derived from a gradient descent formulation. A VN is flipped when its energy

Manuscript received January 31, 2018; revised May 5, 2018 and June 11, 2018; accepted June 18, 2018. This work was supported in part by the French ANR under Grant ANR-15-CE25-0006-01 (NAND Project) and in part by the NSF under Grant ECCS-1500170. This paper was presented in part at the International Conference on Advanced Technologies for Communication, Quynhon, Vietnam, October 2017. This paper was recommended by Associate Editor G. Masera. *(Corresponding author: Khoa Le.)*

K. Le, F. Ghaffari, L. Kessal, and D. Declercq are with ETIS, UMR8051, Université Paris Sein, Université de Cergy-Pontoise, ENSEA, CNRS, 95014 Cergy-Pontoise, France (e-mail: khoa.letrung@ensea.fr; fakhreddine.ghaffari@ensea.fr; kessal@ensea.fr; declercq@ensea.fr).

E. Boutillon is with the Lab-STICC, UMR 6285, Université de Bretagne Sud, 56321 Lorient, France (e-mail: emmanuel.boutillon@univ-ubs.fr).

C. Winstead is with the Department of Electrical and Computer Engineering, Utah State University, Logan, UT 84322-4120 USA (e-mail: chris.winstead@usu.edu).

B. Vasić is with the Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ 85719 USA (e-mail: vasic@ece.arizona.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2018.2849679

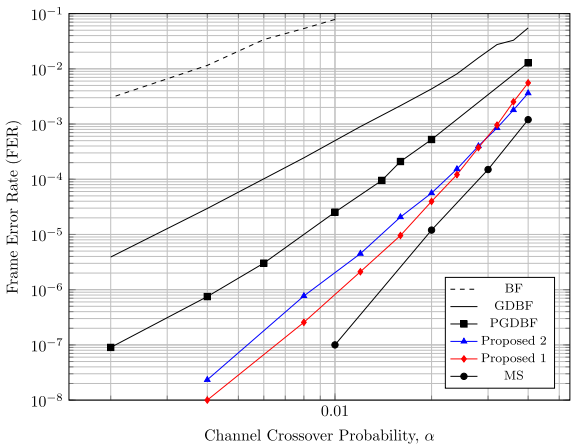


Fig. 1. The evolution in term of decoding performance of BF decoders on the Tanner code [15].

value is a maximum, compared to all other energies. The GDBF provides a very good error correction and it is better than the previously introduced deterministic BF decoders. The PGDBF is a variant of the GDBF, introduced also in [13]. The PGDBF follows precisely the decoding steps of the GDBF, and differs only in the VN flipping operations. Unlike the GDBF in which the VNs that satisfy flipping condition are automatically flipped, all the flipping candidates in PGDBF are only flipped with a probability of p ($0 < p < 1$). Interestingly, this probabilistic behavior improves the decoding performance, far better than the original GDBF and very close to MS [14]. The evolution of BF performance improvement is shown in Fig. 1 on the well-known Tanner code [15].

Another newly proposed BF decoder is introduced by Jung and Park [11], and is called Multi-Bit Flipping (MBF). The MBF scenario is in line with our target in which the decoder requires only hard information from the channel for decoding. The VNs of MBF also compute at each iteration the energy value basing on its neighbor CN values. The novelty comes from the fact that, at each iteration of MBF, only a constant number of VNs are allowed to flip to avoid the overcorrection. For example, only 4 VNs are flipped at each iteration. This truly helps the MBF avoid VN miss-flipping and offers a considerable improvement in error correction.

The GDBF and PGDBF implementations can be found in [14] and MBF is implemented in [11]. A drawback of these BF decoders is that a global operation (the Maximum Finder - MF in [14] and N-to-4 sorter in [11]) is required to identify the maximum among the VN energy values. This global function operates on the large number of energy values, and in the hardware implementations it lengthens the decoder data path and limits the maximum achievable frequency, especially when the codeword length is long. Although the reported decoding throughput of the above decoders is very promising, the global operation are the main obstacle for further improving the operating frequency and therefore, limit the increasing of the decoding throughput.

Another global computing block, which also limits the maximization of the operating frequency, is the Syndrome Check (SC) module. During the decoding process, the SC module verifies all the CN values and indicates at its output the stopping signal when all of them are satisfied, *i.e.*, when all of CN values are 0's. We show in this paper that, the SC is an important module for prior-introduced BF decoders to obtain the decoding success but it can not be fully parallelized to improve the decoding speed. The hardware implementation of the SC module has all of the CN outputs as its inputs and the data path becomes significantly long when the number of CNs is large. A BF decoder in which the SC is eliminated from the data path, would provide a significant improvement in operating frequency.

This paper presents a new probabilistic parallel BF decoder (PPBF) in which no global operation is required. The flipping decision in each VN of PPBF is made locally in the VN basing only on its channel value, neighbor CN values and a probabilistic binary signal. In the PPBF, depending on the computed energy value, the VN will be flipped with the corresponding probability. Therefore, at each iteration, all VNs in PPBF could be flipped (with different probabilities) and it is different from the PGDBF decoder where only the VNs having the maximum energy are the flip candidates. The data path is significantly shortened which results an enhancement of decoding throughput. More interestingly, by eliminating the global operation, PPBF becomes a low-complexity algorithm, comparable to the deterministic GDBF decoder, while having a better decoding performance than PGDBF and being very close to the MS performance. An improved version of PPBF, called Non-Syndrome Probabilistic Parallel Bit Flipping (NS-PPBF), is then introduced, further shortening the critical path by moving the global SC module out of the data path. In the NS-PPBF, when converging to a correct codeword, the decoder will stop flipping thanks to a specific flipping mechanism. NS-PPBF additionally improves the decoding frequency, compared to PPBF while maintaining the good error correction capability.

The rest of the paper is organized as follows. In Section II, the notations of LDPC codes and LDPC decoding algorithms are firstly introduced. The PPBF algorithm is then presented. The analysis of PPBF in improving of decoding performance and operating frequency is also provided. The improved version PPBF (the NS-PPBF) is introduced in Section III. In Section IV, the implementation architectures for PPBF and NS-PPBF are presented with detailed circuits for the probabilistic signal generator and processing units. In Section V, the hardware synthesis results and the simulated error correction performance are presented. These results confirm that the proposed decoders significantly improve the operating frequency, reduce the decoder complexity while improving the error correction capability, better than all known BF decoders in the literature. Section VI concludes the paper.

II. The Probabilistic Parallel Bit Flipping Decoders

A. Notations

An LDPC code is defined by a sparse parity-check matrix H of dimention $M \times N$, $N > M$. Each row represents a parity

check function, computed by a CN, on the VNs represented by the columns. The VN v_n $(1 \leq n \leq N)$ is checked by the CN c_m $(1 \leq m \leq M)$ if the entry $H(m,n) = 1$ and they are called neighbors. An LDPC code can also be represented by a bipartite graph called Tanner graph composing two groups of nodes, the CNs c_m, $1 \leq m \leq M$, and the VNs v_n, $1 \leq n \leq N$. The VN v_n $(1 \leq n \leq N)$ connects to the CN c_m $(1 \leq m \leq M)$ by an edge in the Tanner graph if the entry $H(m,n) = 1$. We denote the set of CNs connecting to the VN v_n (the so-called, neighbor set) as $\mathcal{N}(v_n)$ and $|\mathcal{N}(v_n)|$ is called the degree of VN v_n. Similarly, $\mathcal{N}(c_m)$ denotes all VNs connecting to CN c_m and $|\mathcal{N}(c_m)|$ is the degree of CN c_m. This paper works on the regular LDPC code, *i.e.*, $|\mathcal{N}(v_n)| = d_v$ and $|\mathcal{N}(c_m)| = d_c$, $\forall n, m$. A vector $\mathbf{x} = \{x_n\} = \{0,1\}^N$ is called a codeword if and only if $H\mathbf{x}^T = \underline{\mathbf{0}}$. $\mathbf{x}$ is sent through a transmission channel and we denote $\mathbf{y} = \{y_n\} = \{0,1\}^N$ as the channel output. The decoders presented in this paper are applied on the Binary Symmetric Channel (BSC) where each bit $x_n \in \mathbf{x}$ is flipped with a probability α, called channel crossover probability, when being transmitted, *i.e.*, $\Pr(y_n = x_n) = 1 - \alpha$ and $\Pr(y_n = 1 - x_n) = \alpha$, $1 \leq n \leq N$. We use the superscript (k) together with the node notations, *e.g.* $c_m^{(k)}$, $v_n^{(k)}$, to denote the nodes values at the k-th iteration. Also, the nodes values vector at the k-th iteration are denoted as $\mathbf{v}^{(k)} = (v_1^{(k)}, v_2^{(k)}, \ldots, v_N^{(k)})$ and $\mathbf{c}^{(k)} = (c_1^{(k)}, c_2^{(k)}, \ldots, c_M^{(k)})$.

B. The Proposed Probabilistic Parallel Bit Flipping Decoder

The proposed PPBF decoding algorithm follows the decoding procedure as in other introduced BF algorithms, in which the binary messages are iteratively passed between the VNs and CNs via the decoding iterations. The CN messages are simply the parity information and its computation is formulated as in Eq. (1), where $\oplus$ is the bit-wise Exclusive-OR (XOR) operation.

$$c_m^{(k)} = \bigoplus_{v_n \in \mathcal{N}(c_m)} v_n^{(k)} \tag{1}$$

The BF decoding process is stopped either when all CNs equations are satisfied, *i.e.*, $\mathbf{c}^{(k)} = \underline{\mathbf{0}}$, in which case, the decoding process is declared as a success and the $\mathbf{v}^{(k)}$ are the decoded codeword, or when k reaches the maximum number of iterations (denoted by It_{max}). In the latter case, the decoding process is declared as a failure.

When the above stopping conditions are not satisfied, all the CN computation results, $c_m^{(k)}$, $1 \leq m \leq M$, are sent to the neighboring VNs (called CN messages), and the VN operations are then repeated again. The computation of VN is summarized as following. Firstly, each VN evaluates its energy (reliability) value $E_n^{(k)}$, using Eq. (2), where the energy value is the sum of two terms. The first term is the similarity between its current value, $v_n^{(k)}$, and its channel output, y_n (computed by the function $\oplus$). The second term is the number of its unsatisfied neighbor CNs, *i.e.*, the number of neighboring CNs having $c_m^{(k)} = 1$. It is clear that, the energy value for a VN is an integer between 0 to $d_v + 1$, $(0 \leq E_n^{(k)} \leq d_v + 1)$. Secondly, based on the computed energy value $E_n^{(k)}$, a random bit with Bernoulli distribution, $\mathcal{B}(.)$, (denoted as $R_n^{(k)}$) is generated in each VN such that $\Pr(R_n^{(k)} = 1) = \mathbf{p}(E_n^{(k)})$ and $\Pr(R_n^{(k)} = 0) = 1 - \mathbf{p}(E_n^{(k)})$ where $\mathbf{p}$ is a pre-defined vector of real values, $\mathbf{p} = (p_0, p_1, \ldots, p_{d_v+1})$, $|\mathbf{p}| = d_v + 2$ and $\mathbf{p}(\ell)$ returns the ℓ-th element of $\mathbf{p}$, $(0 \leq \ell \leq d_v + 1)$, $p_0 = \mathbf{p}(0) = 0$, $0 < p_\ell = \mathbf{p}(\ell) \leq 1, \forall \ell = 1, \ldots, d_v + 1$. Finally, each VN will determine its new value basing on the generated random bit $R_n^{(k)}$. If $R_n^{(k)} = 1$, the VN will update its values by inverting the current value $v_n^{(k)}$, otherwise, it keeps the current value. All the VN values are sent to the neighbor CNs and a new decoding iteration will start. The PPBF decoding algorithm is described in Algorithm 1.

$$E_n^{(k)} = v_n^{(k)} \oplus y_n + \sum_{c_m \in \mathcal{N}(v_n)} c_m^{(k)} \tag{2}$$

Algorithm 1 The PPBF Decoding Algorithm

Initialization $k = 0$, $\mathbf{v}^{(0)} = \mathbf{y}$, $\mathbf{p} = (p_0, p_1, \ldots, p_{d_v+1})$
while $k \leq It_{max}$ **do**
  for $1 \leq m \leq M$ **do** ▷ CN processing
    Compute $c_m^{(k)}$, using Eq. (1).
  end for
  if $\mathbf{c}^{(k)} = \underline{\mathbf{0}}$ **then** ▷ Check syndrome
    exit while
  end if
  for $1 \leq n \leq N$ **do** ▷ VN processing
    Compute $E_n^{(k)}$ using Eq. (2).
    Generate $R_n^{(k)}$ from $\mathcal{B}(\mathbf{p}(E_n^{(k)}))$.
    if $R_n^{(k)} = 1$ **then**
      $v_n^{(k+1)} = v_n^{(k)} \oplus 1$
    end if
  end for
  $k = k + 1$
end while
Output: $\mathbf{v}^{(k)}$

It is interesting to make the comparisons between the PPBF and the state-of-the-art PGDBF decoding algorithm [13], [14] as the best BF decoder in terms of error correction. The VN and the global operations of PGDBF (the main differences between PPBF and PGDBF) are given in Algorithm 2 where $R_n^{(k)}$ also denotes the random bit generated for v_n but differs from the fact that, $\Pr(R_n^{(k)} = 1) = p$, $\Pr(R_n^{(k)} = 0) = 1 - p$ (p is a pre-defined value). The VN processing of PGDBF can be divided into 2 parts. At the first part, each VN computes its energy value $E_n^{(k)}$ using also the Eq. (2). These computed energy values are sent to a global module to find the maximum energy $E_{max}^{(k)} = \max_n(E_n^{(k)})$. At the second part, a random bit $R_n^{(k)}$ is generated for each VN. Then, the value of each VN will be flipped if and only if its energy equals to the maximum energy $(E_n^{(k)} = E_{max}^{(k)})$, and the generated random bit $R_n^{(k)} = 1$. It can be seen that the PPBF and PGDBF have the same decoding steps as well as the CN and VN energy computation formulations. They also apply the probabilistic flipping of the VN values in each iteration. The differences between them are twofold. First, the PPBF does not require the maximum energy value in making flipping selection and therefore, no global operation is required. This comes from the

fact that, in the PPBF each VN flips its value basing only on its own energy and the random signal value, regardless the energy of other VNs. Second, in the binary random generating step of the PPBF, the probability of generated bit, $R_n^{(k)}$, is dynamically changed over the VNs and over the decoding iterations. This probability is controlled by the VN energy values at each iteration while in PGDBF, it is fixed for all iterations and is equal for all VNs. Therefore, all VNs in PPBF are the flipping candidates (rather than only the energy-maximum VNs as in PGDBF) and they are actually flipped with different probabilities. This property helps PPBF be more advantageous in error correction than the PGDBF, presented in the next section.

Algorithm 2 The VN and Global Operation in PGDBF Decoder

for $1 \leq n \leq N$ **do** ▷ VN processing part 1
  Compute $E_n^{(k)}$, using Eq. (2).
end for
Compute $E_{max}^{(k)} = max_n(E_n^{(k)})$ ▷ Global operation
for $1 \leq n \leq N$ **do** ▷ VN processing part 2
  Generate $R_n^{(k)}$ from $\mathcal{B}(p)$.
  if $E_n^{(k)} = E_{max}^{(k)}$ **and** $R_n^{(k)} = 1$ **then**
    $v_n^{(k+1)} = v_n^{(k)} \oplus 1$
  end if
end for

C. The PPBF Improves the BF State-of-the-Art Error Correction Capability

The PPBF is proposed with the probabilistic strategy in flipping the VN, which provides an ability known as *oscillation breaking* to improve error correction, with respect to the deterministic BF decoders [16], [24]. Indeed, when we consider the case of the GDBF, the failure comes from the fact that, during the decoding process, GDBF decoder falls into the infinite oscillations (or loops) between the tentative states of $\mathbf{v}$, which prevent GDBF converging to the correct codeword [14], [16]. The probabilistic flip of PPBF helps break this oscillation and thus helps it converge. For illustration, we conducted a statistical analysis on the failures of GDBF decoder and the oscillation breaking of PPBF on the Tanner code. The results are plotted in Fig. 2. In Fig. 2A, we classified all the failure cases of GDBF into the t-iterations oscillation types, *i.e.*, the positions of erroneous bits repeat the same after t iterations, $\mathbf{v}^{(k+t)} = \mathbf{v}^{(k)}$, for several values of α and $It_{max} = 300$. It is confirmed by Fig. 2A that, all of the decoding failures of GDBF are due to the infinite oscillation between states of $\mathbf{v}$ and furthermore, the 2-iteration oscillations dominate other types, especially in the low error rate region. We then re-decoded the GDBF failure cases by PPBF (and also by PGDBF for comparison). The results are plotted in Fig. 2B. It is remarked that, the PPBF corrects almost the failure cases of GDBF, especially in low error rate region. This above interesting property is the source of PPBF error correction improvement.

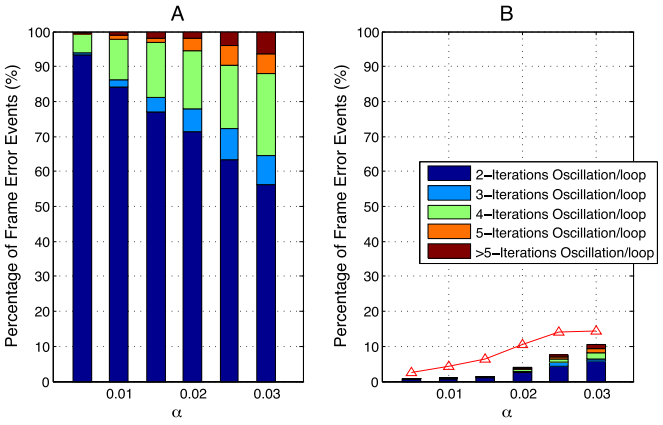


Fig. 2. The statistical analysis of decoding failure of BF decoders on the Tanner code: A. The failure types of GDBF decoder, B. The remaining after re-decoding the failure codewords in (A) by PPBF (bars) and by PGDBF (red triangle-marked curve).

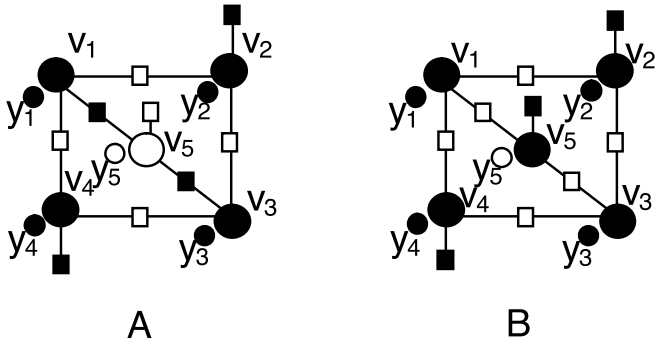


Fig. 3. An example of error pattern with which PGDBF failed to correct while PPBF can correct.

The error correction of PPBF in the above statistics is even better than that of the PGDBF decoder. This is because of the fact that, there are certain error patterns with which PGDBF failed to correct while PPBF succeeds. We show an example of these error patterns in Fig. 3 where there are 4 bits in error allocated in a Trapping Set (5, 3) [14]. Fig. 3A describes the error positions initially received from the channel. The correct VNs (satisfied CNs) are denoted as white-filled circles (squares), the erroneous VNs (unsatisfied CNs) are the black-filled circles (squares). The smaller cycles denote the channel values of each VN. When decoding this error pattern by PGDBF decoder, the VN v_5 has maximum energy ($E_5^{(1)} = 2$) (other VNs have energy of 1) and v_5 becomes the only flipping candidate. The PGDBF will either flip v_5 (if $R_5^{(1)} = 1$) or not flip v_5 (if $R_5^{(1)} = 0$). In the latter case, PGDBF keeps the same error pattern for the next iteration. In the case PGDBF flips v_5, the error locations are illustrated in Fig. 3B and in the next iteration, v_5 is again the only flipping candidate since it has energy of 2 while others have either 1 or 0. The PGDBF may again flip or not flip v_5 depending on $R_5^{(2)}$. Therefore, the PGDBF is trapped in the 2-iteration oscillation and never correct all erroneous VNs due to its maximum-energy VN flip constraint. On the contrary, in the case of PPBF, all VNs are the flipping candidates. With the above error pattern, at the

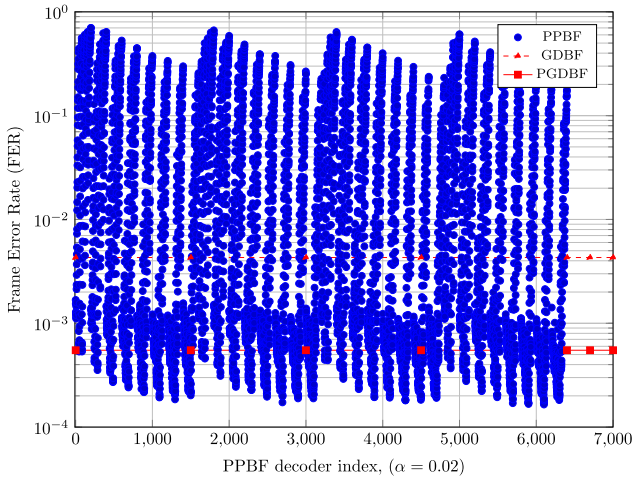


Fig. 4. The decoding performance of PPBF decoder with different values of **p** combined from Tab. I, on comparison to the GDBF and PGDBF ($p = 0.7$) on the Tanner code at $\alpha = 0.02$, $It_{max} = 300$.

first iteration, the PPBF not only flips v_5 with $\mathbf{p}(2)$ (because $E_5^{(1)} = 2$) but also other erroneous VNs (v_1, v_2, v_3 and v_4) with $\mathbf{p}(1) > 0$ (because $E_1^{(1)} = E_2^{(1)} = E_3^{(1)} = E_4^{(1)} = 1$). For example, PPBF can correct this error pattern in a single iteration with probability $(\mathbf{p}(1))^4(1 - \mathbf{p}(2)) > 0$ where it flips 4 erroneous energy-1 VNs and does not flip other VNs. Using the analysis method in [24], it is shown that PPBF will eventually correct this error pattern in a subsequent number of iterations.

The PPBF error correction capability is controlled by the values of **p**. We show that, it is feasible to choose **p** such that PPBF provides the significant decoding gain, compared to PGDBF and GDBF, by the following experiment. A statistics on the Frame Error Rate (FER) as a function of **p** was conducted on the Tanner code ($d_v = 3, d_c = 5$). The testing values of **p** were chosen with the ranges and steps as in the Table I and they are explained as following. It is obvious to choose $p_0 = \mathbf{p}(0) = 0$ since the VN having all neighbor CN satisfied and being similar to the channel received value, should not be flipped. In principle, the VN that has higher energy values should have higher probability of flip. The VN that has all neighbor CN unsatisfied and its value disagrees with the channel received value (in the Tanner code, its energy is 4), should be flipped with high probability ($p_4 = \mathbf{p}(4)$ should be close to 1). The VN that has energy of 1 should be flipped with a small probability ($p_1 = \mathbf{p}(1)$ should be close to 0) to avoid flip too many correct VNs unintentionally since there may exist many correct VNs having the energy of 1. $p_2 = \mathbf{p}(2)$ and $p_3 = \mathbf{p}(3)$ were chosen to cover a large ranges. Table I shows all of the studied configurations. A nested loops are formed to cover all combinations where p_1 is in the most inner loop, p_4 is the most outer loop. There are 6400 combinations and they are marked by an index i ($0 \le i \le 6399$) from $\mathbf{p}^{[0]}$ to $\mathbf{p}^{[6399]}$. We run the PPBF decoder with those values of $\mathbf{p}^{[i]}$ at $\alpha = 0.02$, $It_{max} = 300$ and plot the results in Fig. 4. We also show the decoding performance of

TABLE I

THE PROBABILITY RANGES FOR THE STATISTICAL EXPERIMENT

	min	max	step	number of testing cases
p_0	0	0	0	1
p_1	0	0.039	0.001	40
p_2	0.1	0.5	0.1	5
p_3	0.3	1.0	0.1	8
p_4	0.7	1.0	0.1	4
Total testing cases				6400

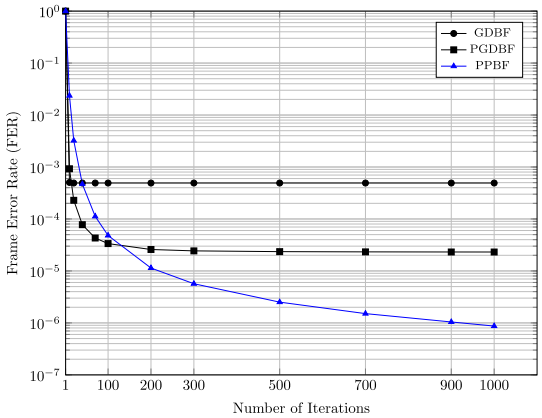


Fig. 5. FER of the proposed PPBF, the GDBF and PGDBF ($p = 0.7$) decoders as function of the number of decoding iterations on the Tanner code for $\alpha = 0.01$.

GDBF and PGDBF decoders (also at $\alpha = 0.02$, $It_{max} = 300$) to make the comparisons.

It can be seen that, there are many values of **p** with which PPBF decoder outperforms PGDBF decoder. This number is even larger when comparing to the GDBF decoder. Some of them may have 1.4 decade better than GDBF and 0.6 decade better than PGDBF. The numerous options of **p** providing a good decoding performance, are very promising since they facilitate the choice of PPBF to have the good decoding performance and simplify the hardware implementation, as shown in Section IV.

It is observed that the performance of PPBF decoder is continuously improved when increasing the number of decoding iterations. Indeed, as seen in Fig. 5 where we plot the FER as a function of the number of iterations, the PGDBF tends to saturate in correcting errors after 200 decoding iterations while the GDBF is saturated even after 10 decoding iterations. Running GDBF and PGDBF after those numbers of iterations will not provide any gain in error correction. However, this is not the case of PPBF decoder which continues to decrease the FER after the first 200 iterations. It can be seen that, when PPBF runs up to $It_{max} = 1000$ iterations, the FER si reduced around 1 decade compared to the one at 200 iterations. The PPBF is, therefore, very favorable in the applications where error correction is more important than the decoding time constraint.

As a conclusion of this section, our statistical analysis reveals that PPBF is a very promising decoder in terms of error correction. It is even more interesting since PPBF is shown

to improve significantly the decoding frequency when being implemented, which is presented in the next section. Some open research directions may be opened, *e.g.*, the theoretical optimization and the systematic choice of $\mathbf{p}$ where it may be considered as a function of α or even a function of decoding iterations. PPBF may be optimized for the additive white gaussian noise (in which PPBF may converge to the Noisy GDBF in [27]) channel and/or on irregular LDPC codes. These interesting directions are kept for our future work.

D. The PPBF Improves the Decoding Throughput and Reduces the BF Decoder Complexity

Typically, the average decoding throughput of a hardware implemented BF decoder, θ (in bits/s), is computed using the Eq. (3) where f_{max} denotes the maximum decoding frequency, N_c denotes the number of clock cycles required for each decoding iteration and It_{ave} is the number of iterations in average.

$$\theta = \frac{N * f_{max}}{It_{ave} * N_c} \tag{3}$$

In a hardware implemented LDPC decoder, the maximal operating frequency is limited by the length of the longest data path (and so-called, critical path) and f_{max} should be inferior to $1/t_D$, ($f_{max} < 1/t_D$), where t_D is the length of the critical path to avoid the timing error in run-time [25]. In order to improve the decoding throughput (assuming keeping N, It_{ave} and N_c constant), the hardware implementations of LDPC decoders tend to be parallelized to shorten the critical path [22], increasing maximally the achievable operating frequency f_{max}. However, the maximization of the operating frequency in some BF decoders reported in the literature, such as GDBF, PGDBF [14] or MBF [11] may be limited since the global operation still required in these decoders, dramatically increases the critical path [17]. In order to clarify the effect of the global operation on the critical path as well as the decoder complexity, we choose to analyze the architecture of PGDBF decoder in [14] (presented in Fig. 6). The operation of PGDBF decoder on this architecture may be found in that paper. The longest data path in PGDBF implementation is shown by the dashed green path in Fig. 6. This critical path starts from the output of D Flip-Flop storing $v_n^{(k)}$ and passes to the CN computing unit (CNU), the energy computation (summation block), the Maximum Finder (MF block), several computing units and ends at the input-D of the D Flip-Flop (the updated VN value for the next iteration, $v_n^{(k+1)}$). Many implementing methods of the MF can be found in [17]. We illustrate the implementation of MF in Fig. 6 by the binary tree of the Compare-And-Swap Units (CASUs) [11]. The CASU passes at its output the larger value of the 2 input values. The latency of the MF module can be formulated as $t_{MF} = t_c\lceil\lfloor log_2(N)\rceil$ where t_c is the delay of a single CASU module. This critical path may become longer when longer code is used. The ASIC synthesis results of the PGDBF decoder on the $d_v = 3$, $d_c = 6, N = 1296, M = 648$, code rate $R = 0.50$ regular LDPC code [22] (denoted as dv3R050N1296) show that, the MF occupies more than 40% of the total length of the critical path (the details of synthesis setup are introduced in Section V). The MF is also a large part of PGDBF in terms of complexity since it requires approximately $N-1$ CASUs. The synthesis shows also that, the MF is responsible for 20% of the total PGDBF decoder area.

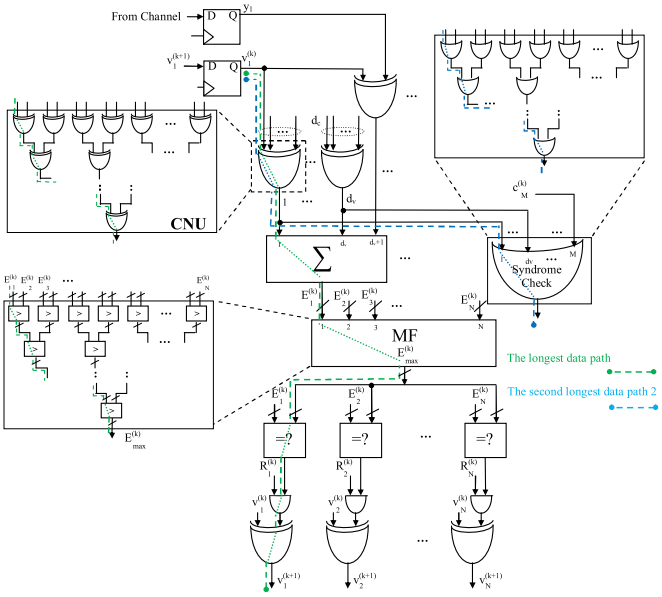


Fig. 6. The critical paths in PGDBF decoder.

The VNs of PPBF do not require the maximum energy value for their operations. This is the source of the improvement in the terms of operating frequency of PPBF. For example, it is expected that, PPBF will reduce the critical path by 40% when designed and synthesized on the same dv3R050N1296 LDPC code. Since the MF is not implemented, the decoder complexity will be reduced, compared to that of the PGDBF.

III. The Non-Syndrome Probabilistic Parallel Bit Flipping Decoder

The PPBF is expected to improve the operating frequency since the critical path is significantly reduced (as proven by the synthesis results in Section V). In the synthesized PPBF decoder, it is observed that, the critical path of PPBF starts from the output of the D Flip-Flop storing $v_n^{(k)}$ and passes to the CNU, the SC module and ends at this SC module output (which controls the enable-input of D Flip-Flop to store $v_n^{(k+1)}$ into the place of $v_n^{(k)}$). This critical path is similar to the second longest data path in PGDBF implementation illustrated in Fig. 6 by the dashed blue path. We target improving the operating frequency of the PPBF by eliminating the SC module from the data path and shortening the critical path of PPBF. This modified version of PPBF is named as NS-PPBF decoder.

In some LDPC decoding algorithms such as MS, OMS, Sum-Product or Gallager-A/B, the decoding process may continue to run when the decoder already converges to the correct codeword. This extra updating of the A-Posteriori-Probability (APP) will make the hard-decision even more solid. This property provides the possibilities that, the early stopping condition (the SC module) may not be needed.

The situation is different in the case of BF decoders such as GDBF, PGDBF, PPBF *etc.* where the SC module plays an important role in ensuring the finding of correct codewords. Indeed, when converging to the correct codeword (all CN values are zeros), the computed energy in these BF decoders is either 0 (of the correctly received VNs from channel) or 1 (of the incorrectly received VNs from channel, the value of 1 comes from the term $v_n^{(k)} \oplus y_n$ in energy computation of Eq. (2). If the SC result is not verified to stop the decoding process, the BF decoders tend to flip the energy-1 VNs in the next iteration (Note that the GDBF will flip all energy-1 VNs while PPBF and PGDBF flip a random part of them). These decoders are re-entering to the incorrect-codeword states and the correct codeword may not be found after It_{max} iterations or in a subsequent number of iterations after converging. Because of this flipping principle, the SC is inevitable when implementing these BF decoders (note further that, since the computations are simple, in the literature, these BF decoders are usually implemented such that they take only $N_c = 1$ clock cycle for 1 decoding iteration). In the PPBF, the SC module becomes a part of the critical path. Moreover, the SC is a global operation which verifies all the values of M CNs in each iteration. It occupies, therefore, a noticeable part of that critical path in PPBF. In fact, the SC introduces a delays of $t_{SC} = t_o\lceil[]log_2(M)\rceil$, where t_o is the delay of a single OR gate, when implemented as a binary tree of OR gates (as in Fig. 6). The hardware synthesis of PPBF shows that the delay of SC module is around 20% of the PPBF critical path length. Our proposed NS-PPBF decoder introduces a new stopping mechanism which does not depend on the SC result.

Algorithm 3 The VN Operations of the NS-PPBF Decoder

```
for 1 ≤ n ≤ N do                                   ▷ VN processing
    Compute E_n^(k), using Eq. (2).
    if E_n^(k) ≠ 1 or v_n^(k) = y_n then
        Generate R_n^(k) from B(p(E_n^(k))).
        if R_n^(k) = 1 then
                v_n^(k+1) = v_n^(k) ⊕ 1
        end if
    end if
end for
```

The proposed flipping mechanism targets to automatically stop flipping the VNs when the correct codeword are found, *i.e.*, all the CNs are satisfied (all CNs are 0's). The VN operation of NS-PPBF is described in Algorithm 3. It can be seen that, the flipping of NS-PPBF decoder is almost similar to that of the PPBF decoder and the difference is on the flipping when the VN energy is 1. In fact, it is sufficient to additionally manipulate on the case that $E_n^{(k)} = 1$ because when the codeword is found, the VN produces only the values 1 or 0 as its energy, and the energy-0 VNs are not flipped since $p_0 = 0$. In NS-PPBF, when the energy $E_n^{(k)} = 1$, it separates the case where the value 1 is obtained from the term $v_n^{(k)} \oplus y_n$ and the case where the value 1 is obtained from the term $\sum_{c_m \in \mathcal{N}(v_n)} c_m^{(k)}$. NS-PPBF then flips the VN having $E_n^{(k)} = 1$ with probability $p_1 = \mathbf{p}(1)$ only in the latter case.

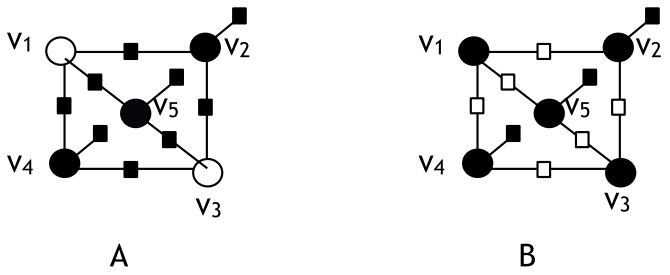


Fig. 7. An example of error pattern where the PPBF has higher probability to correct than the NS-PPBF: (A) the error configuration received from channel, (B) a trapping state during the decoding process.

It means that, NS-PPBF will not flip the VNs having energy 1 coming from the disagreement between the current value and the channel value. By using this principle, whenever a correct codeword is found, all the CNs are satisfied, the NS-PPBF will not flip any VNs and the current $\mathbf{v}^{(k)}$ is preserved. The decoding process will stop either when It_{max} iterations have been reached (if the SC is not implemented), or the parallelized implemented SC produces the stopping signal. $\mathbf{v}^{(k)}$ is then declared as the decoded codeword.

It can be seen that, the NS-PPBF decoding process will terminate without using the signal from the SC module. The SC contribution to the critical path have been eliminated, which will improve the operating frequency of the decoder. However, the constraint applied on the energy-1 VNs reduces the decoding dynamics and may lead to degradation in error correction, compared to the PPBF decoder. We observed that the performance degradation depends on the existence of the small and "harmful" sub-graphs in the Tanner graph *e.g.*, the trapping set (5,3) in the Tanner code, which prevent the decoding convergence. We show an example as in Fig. 7 where Fig. 7A shows the channel error configuration. Decoding this error configuration may lead to a special state as in Fig. 7B where all VNs in the TS(5,3) are erroneous. It can be seen that, this is a trapping state since flipping any bits out of this TS will not increase the energy of the VN in the TS. In other words, only flipping the VNs in the TS helps escape from this state and converge. With the PPBF, all 5 VNs have flipping probability p_1 while only v_2, v_4 and v_5 may be flipped in the case of the NS-PPBF. This reduces the decoding success chance of NS-PPBF. In fact, decoding this error configuration by the PPBF and NS-PPBF with $\mathbf{p} = (0, 0.0081, 0.3, 0.7, 1)$ produces FER of $4.2e^{-3}$ and $2.58e^{-2}$, respectively.

IV. The Hardware Architecture

In order to verify the efficiency of the proposed BF decoders in terms of decoding frequency and complexity, the hardware architectures implementing these decoders are introduced in this section. These hardware architectures are then synthesized on the Application-Specific Integrated Circuit (ASIC). The synthesis results and discussions are presented in the next section.

The first proposed hardware architecture of the PPBF decoder is presented in Fig. 8. Similarly to the architecture of PGDBF decoder, the CN operations is realized by the

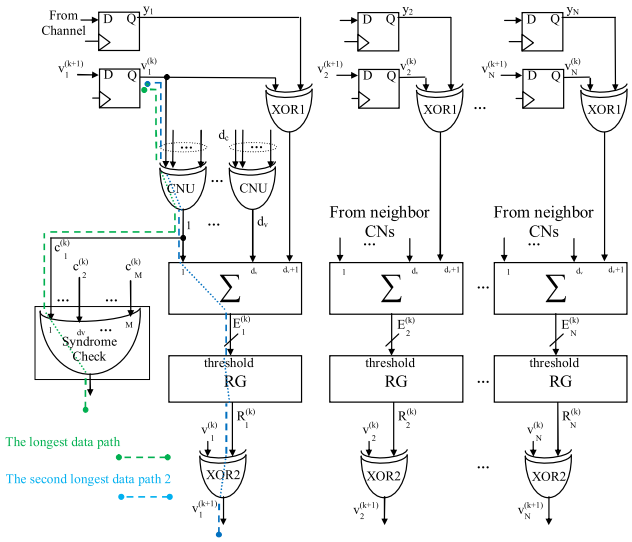


Fig. 8. The LFSR-based implementation of PPBF decoder.

d_c-input XOR gates. The VN energy value is computed by the summation block whose inputs are the CN computation results and the XOR1 (computing the term $v_n^{(k)} \oplus y_n$). The key feature of this architecture is that the computed energy values controls the threshold input of a random generator (RG). This threshold input is used to control the probability of the generated bit and by doing that, the probability of the generated bit is adapted for each VN and each iteration. One RG method could be used is the Linear Feedback Shift Register (LFSR) as in [18]. The VN value for the next iteration, $v_n^{(k+1)}$, is computed by the XOR2 gate based on the generated bit and the VN current value $v_n^{(k)}$. The $v_n^{(k+1)}$ is an inversion of $v_n^{(k)}$ if the generated bit $R_n^{(k)} = 1$ since X $\oplus$ 1 = **NOT**(X), $\forall$X. When the generated bit $R_n^{(k)} = 0$, $v_n^{(k+1)} = v_n^{(k)}$ since X $\oplus$ 0 = X, $\forall$X. The SC module is implemented to stop the decoder when the correct codeword is found. This architecture can fully execute the operations of the PPBF decoder and can also be adapted for the NS-PPBF. However, the results reported in [18] revealed that, this method may be costly since each VN requires an LFSR module and the decoder complexity may be increased due to these RGs, especially when the LDPC code with large N is used. We leave aside this architecture and focus on a low-complexity solution to generate the binary random signals, introduced in [14], called Cyclic Shift Truncated Sequence (CSTS).

A. The Probabilistic Signal Generator for PPBF and NS-PPBF Decoders

Inspired by the results of [14] where the correlation in the random generation may not much affect the error correction of the probabilistic BF decoders, we apply that CSTS approach to implement the RGs for PPBF and NS-PPBF decoders, to reduce the hardware resources. In the CSTS, a short D Flip-Flop sequence, R^t with size $|R^t| = S < N$, is allocated. R^t initially stores the generated bits where the 1's are with the ratio of p_r on the total of bits, and they are distributed in an arbitrary order. For each clock cycle, R^t is cyclically shifted.

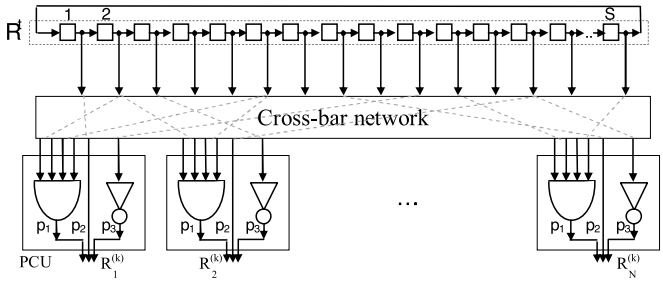


Fig. 9. The proposed probabilistic signal generator (random generator - RG) for PPBF decoder.

It can be approximated that in S clock cycles each output of R_t produces the bits 1 with probability p_r. The outputs of D Flip-Flops in R^t are used as a random bit, triggered to the Variable Node Processing Units (VNUs) and one output can be used for multiple VNUs of the PGDBF without degrading the decoding performance [14].

The RGs of the PPBF and NS-PPBF decoders are required to generate the random bits with different probabilities. In order to produce these desired probabilities, we use the logic gates as introduced in [18] and its principle is briefly recapitulated as follows. Let a and b be two independent random binary variables with $\Pr(a = 1) = p_r$ and $\Pr(b = 1) = p_r$. It can be shown that, $\Pr((a \textbf{ AND } b) = 1) = p_r^2$, $\Pr(\textbf{NOT}(a) = 1) = 1 - p_r$ and $\Pr((a \textbf{ OR } b) = 1) = 2p_r - p_r^2$. Other logic functions such as XOR *etc.* can also be formulated. We design the Probability Controlling Units (PCUs), composed by the logic gates, to produce the random bits for our PPBF and NS-PPBF decoders. The PCU has the binary values from R^t as its inputs through the crossbar network and has several binary outputs. Each output has different probability to be 1, which is transformed from the probability p_r from R^t.

In this paper, we target to implement PPBF and NS-PPBF decoders for the Tanner and the dv3R050N1296 LDPC codes with $d_v = 3$. The extension for other LDPC codes which have higher VN degree, *i.e.*, $d_v \geq 4$, is straightforward. As shown in the statistical analysis, several values of $\mathbf{p}$ may be used to provide good decoding performance gain. We use $\mathbf{p} = (0, 0.0081, 0.3, 0.7, 1)$ as an example to implement since it provides a good error correction while facilitating the hardware implementation. In fact, it can be shown that this particular $\mathbf{p}$ is equal to $(0, p_r^4, p_r, 1-p_r, 1)$ where $p_r = 0.3$. The designed RG is shown in Fig. 9 where N PCUs are designed for N VNs. Each PCU has $d_v = 3$ outputs (internally named as p_1, p_2 and p_3). The outputs have the value of 1 with the probability correspondingly to p_1, p_2 and p_3. The outputs of each PCU are triggered to the corresponding implemented VNU of the PPBF and NS-PPBF, presented in the next section.

B. The Computing Units of PPBF Decoder

PPBF decoder is realized in hardware by two connection networks connecting two groups of processing units: the VNUs and CNUs. The proposed architecture to implement PPBF decoder is presented in Fig. 10. The connection network 1 transmits the messages from VNUs to CNUs and the

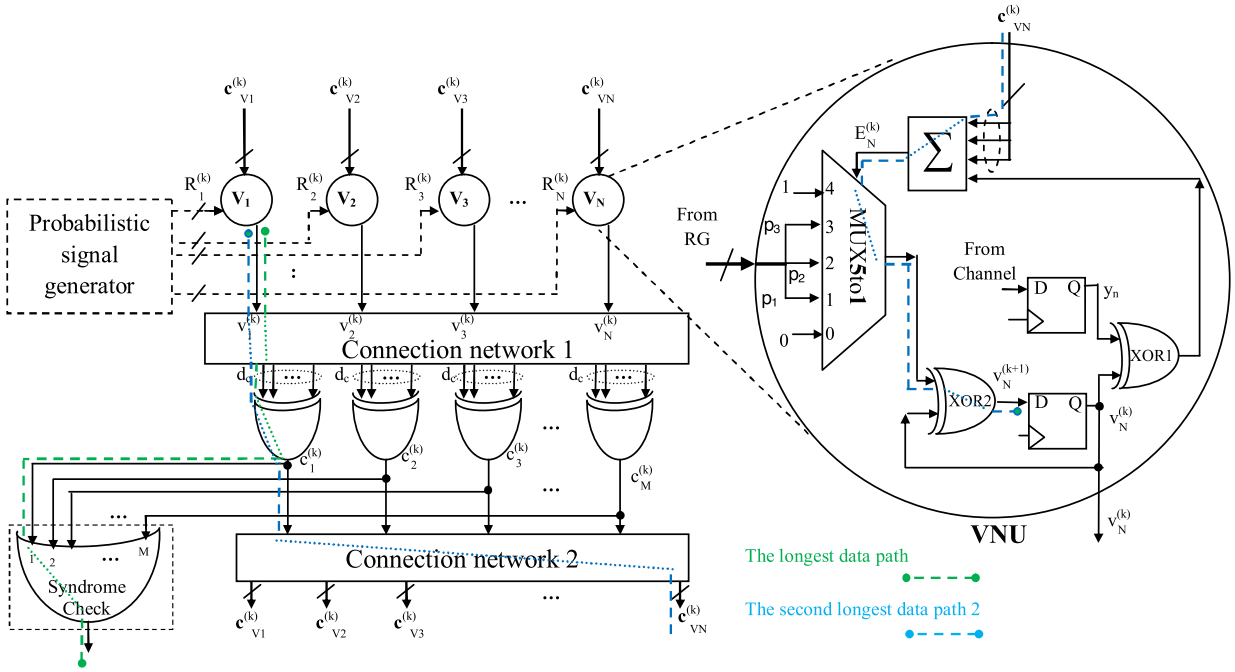


Fig. 10. The proposed architecture for the realization of PPBF decoder for $d_v = 3$ LDPC codes.

connection network 2 is to transfer the messages from CNUs to VNUs. The CNUs in PPBF decoder are also the d_c-inputs XOR gates, computing the parity check on the VN messages. The CNUs outputs are transmitted to the VNUs inputs and we denote (as in Fig. 10) $\mathbf{c}_{v_n}$ as the group of messages of $c_m \in \mathcal{N}(v_n)$, $1 \leq n \leq N$. In each VNU, the summation block computes the VN energy value and this computed energy controls the selection of the multiplexer MUX5to1 module. The MUX5to1 module is a key module which results to flip or not the VN current value, $v_n^{(k)}$, by producing 1s or 0s at its output. When the output of MUX5to1 is 1, the VN current value, $v_n^{(k)}$, is XOR-ed by 1 by the XOR2 and is flipped, otherwise, $v_n^{(k+1)} = v_n^{(k)} \oplus 0 = v_n^{(k)}$. The flipping probability of $v_n^{(k)}$ is, therefore, controlled by its energy. Indeed, the input 0 of MUX5to1 is always reset to 0 while the input 4 is set to 1 which results the fact that, the VN which has energy of 0 is not flipped and the VN having energy of 4 is flipped with probability of 1. When the energy is from 1 to 3, the corresponding flipping probability is arranged as p_1, p_2 and p_3, controlled by the RG. The SC is implemented by M-inputs OR gate whose inputs are the outputs of the CNUs. The decoder will stop when SC produces 0.

It should be noted that, this PPBF architecture requires $N_c = 1$ clock cycle for each decoding iteration. The critical path of the PPBF is denoted by the dashed green path and the second longest data path is shown by the dashed blue path in Fig. 10.

C. The Computing Units of NS-PPBF Decoder

The implementation architecture of NS-PPBF decoder is presented in Fig. 11, which follows mostly the principle of PPBF architecture. The difference comes from the added multiplexer MUX2to1. The MUX2to1 is controlled by the output of the XOR1 which indicates the similarity between the $v_n^{(k)}$ and y_n. This added MUX2to1 contributes to the flipping feature of NS-PPBF in the case when the energy-1 appears. In fact, when $E_n^{(k)} = 1$ and $v_n^{(k)}$ differs from y_n, XOR2 always receives 0's from MUX5to1 and $v_n^{(k)}$ will not be flipped. When $E_n^{(k)} = 1$ and $v_n^{(k)}$ equals to y_n, XOR2 receives signal from the RG with the probability p_1 to be equal to 1.

In the NS-PPBF hardware architecture, the SC is not implemented and the decoder will stop after It_{max} iterations. The critical path of NS-PPBF in the synthesized decoder is shown by the dashed green path in Fig. 11.

V. Synthesis Results and Decoding Performance

A. Synthesis Results

In this section, we report the ASIC synthesis results of our decoder implementations. The proposed decoders are implemented using the Digital Standard Cell Library SAED-EDK90-CORE, Process 1P9M 1.2v/2.5v. The hierarchical design flow is followed with the Synopsys Design Platform: Synopsys Design Compiler is used for VHDL synthesis, Synopsys VCS tool for simulation and DVE, a graphical user interface for debugging and viewing waveforms. Synopsys Prime Time tool is used for timing analysis, power and energy estimation. We generate the switching activity interchange format (saif) file to estimate the power consumption of the decoders. The Synopsys IC Compiler (ICC) is used for floor planning, place and route.

We make the first synthesis of our decoders on the Tanner code ($d_v = 3, d_c = 5, N = 155, M = 93$). Our strategy is to look for the minimum timing constraint to which the synthesizer can pass (which indicates the maximum achievable

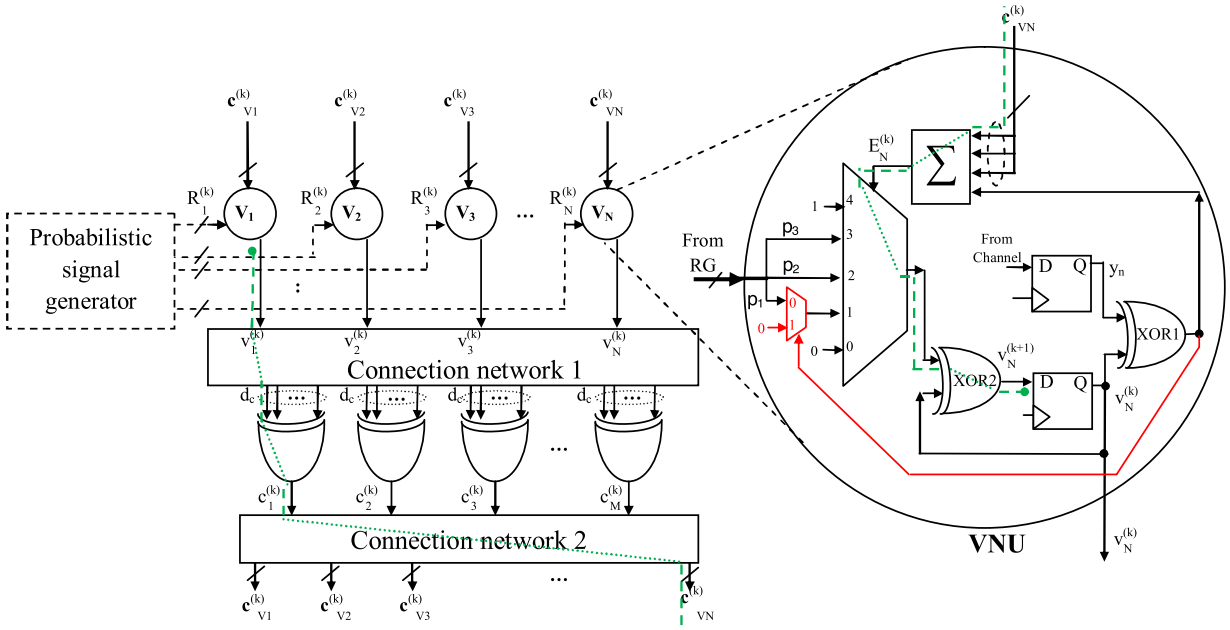


Fig. 11. The proposed architecture for the realization of NS-PPBF decoder for $d_v = 3$ LDPC codes.

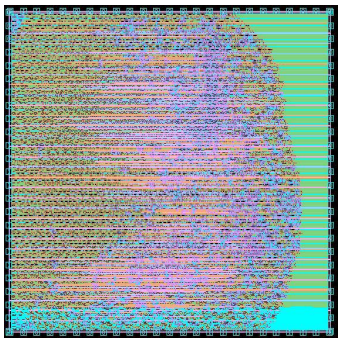


Fig. 12. Layout plot of the PPBF decoder implemented in Table III.

TABLE II

THE SYNTHESIS RESULTS OF PPBF AND NS-PPBF IN THE COMPARISONS TO GDBF AND PGDBF DECODERS ON THE TANNER CODE. ALL THE RGS ARE THE CSTS-BASED IMPLEMENTATIONS WITH $S = 155$. PGDBF IS WITH $p = 0.7$. THE AVERAGE THROUGHPUT θ AND THE AVERAGE POWER CONSUMPTION (ENERGY/BIT) ARE COMPUTED AT $\alpha = 0.012$

	Area (μm^2)	f_{max} (MHz)	θ (Gbps)	Energy/bit (pJ)
GDBF	39528 (-17.6%)	435 (+0%)	30.4	0.24
PGDBF	46472 (-0.0%)	435 (+0%)	25.2	0.33
PPBF	44881 (-3.4%)	625 (+43.7%)	28.1	0.59
NS-PPBF	45814 (-1.4%)	714 (+64.1%)	32.5 $^{(*)}$ 0.369@It_{max}	0.60$^{(*)}$ 53.18@It_{max}

$^{(*)}$ Since the SC module is not implemented in this NS-PPBF implementation, these average values of NS-PPBF are bounds on the achievable results and they are not really practical relevance.

frequency that the decoders can reach) after place-and-route synthesizing processing. The GDBF and PGDBF decoders are implemented following the architecture described in [14]. The synthesis results are reported in Table II. It can be seen, as expected, that PPBF increases more than 43% the operating frequency compared to GDBF and PGDBF while NS-PPBF is even more than 64%. The average decoding throughput at $\alpha = 0.012$ is also provided. It is shown that, the PPBF and NS-PPBF are faster in decoding speed than PGDBF (with 28.1 Gbps and 32.5 Gbps compared to 25.3 Gbps). In terms of hardware complexity, PPBF reduces 3.4% and NS-PPBF reduces 1.4% compared to the PGDBF. This complexity reduction is interesting since PPBF and NS-PPBF provide a better error correction (shown in the next sub-section) while requiring less hardware resources. The complexity reduction is modest because of the fact that, the RGs are more costly than that of the PGDBF, which diminishes the advantage of the MF removal. Our decoders consume more energy than other BF decoders. PPBF needs 0.59 pJ to decode 1 bit while NS-PPBF requires 0.6 pJ, compared to 0.33 pJ of PGDBF and 0.24 pJ of GDBF. It is because of the fact that, the proposed decoders operate at higher frequency. Also, more switching activities may be produced in PPBF and NS-PPBF due to the randomness injection in all levels of VN energy.

In the second synthesis, we implement the proposed decoders for the longer code (the dv3R050N1296 LDPC code). We present the detailed complexity of the computing units of PPBF and NS-PPBF decoders in Table III. Further comparisons of the proposed decoders with other BF-based decoders as well as the MS benchmark from the literature are shown in Table IV. It can be seen in the Table IV that, PPBF and NS-PPBF provide a significant gain in decoding frequency. PPBF can operate at $f_{max} = 476$MHz and NS-PPBF is at $f_{max} = 556$MHz, compared to 385MHz of GDBF and 357MHz of PGDBF. PPBF and NS-PPBF are much faster than MS (250MHz) and ATBF (250MHz). In terms of hardware complexity, the proposed decoders require small hardware resource to be implemented. The required area

TABLE III
THE SYNTHESIS RESULTS OF THE IMPLEMENTED PPBF AND NS-PPBF DECODERS ON DV3R050N1296: AREA PERCENTAGE OF COMPUTING UNITS. THE IMPLEMENTED LAYOUT OF THE PPBF IS SHOWN IS FIG. 12

	PPBF	NS-PPBF
Operating temperature	25°C	
Operating voltage	1.2V	
Total area (mm^2)	0.367	0.349
Frequency (Mhz)	476	556
CNU	14.2%	17.3%
VNU	67.0%	65.0%
Input buffer	13.7%	13.4%
Random Generator	4.0%	3.9%
Syndrome Check	0.7%	N/A
Controller	0.4%	0.4%

is from $0.35 - 0.37$ mm^2 which is equivalent to that of the GDBF (and PGDBF) decoder. It is only $1/5$ of SBF, $1/4$ of MS and $1/2$ of ATBF decoders. In terms of decoding throughput, we compute θ in the worst case (where It_{max} is used) and in the average case (where It_{ave} at $FER = 10^{-5}$ is used). In the former case, the proposed decoders provide the throughput from 2 to 2.5 Gbps which are higher than the GDBF (1.44 Gbps), PGDBF (1.35 Gbps) and MS (1.95 Gbps). The better decoding throughput is also observed in the average throughput to maintain the FER at 10^{-5}. In order to make the comparison in the hardware efficiency, we compute the throughput area ratio (TAR) for all the synthesis results and make comparison to other decoders. It can be seen that, the TAR of PPBF and NS-PPBF ($218 - 272$ $Gbps/mm^2$) are much higher than the other decoders such as ATBF (34.6 $Gbps/mm^2$), SBF (18.8 $Gbps/mm^2$) and MS (16.7 $Gbps/mm^2$).

B. Decoding Performance

The decoding performance of the PPBF and NS-PPBF decoders, simulated on several LDPC codes, is shown in this section. The values of $\mathbf{p}$ are chosen by empirical experiments. On the illustrating figures, we show the decoding performance comparisons with other hard decision decoders such as the conventional BF, GDBF, PGDBF ($p = 0.7$) and Gallager-B (GaB). All hard decision decoders are run with $It_{max} = 300$. The simulations are run until 1000 error frames are found. The performance of the quantized soft-decision MS decoder, implemented in layered scheduling with 4 bits for LLR (Log-LikeLihood Ratio) and 6 bits for A Posteriori Log-LikeLihood Ratio (AP-LLR) [21], are also added, with the $It_{max} = 20$. We additionally compare the BF decoding performance to the floating point sum-product (SP) decoding ($It_{max} = 20$).

Fig. 13 shows the decoding performance on the Tanner code. It can be seen that, the PPBF is the best error correction compared to all other BF decoders. NS-PPBF is better than the PGDBF especially in the low-error rate region. With a small degradation compared to PPBF, NS-PPBF is at the gains of decoding throughput as shown in the previous section. We also extend the simulation of PPBF and NS-PPBF on this Tanner code to run up to $It_{max} = 1000$. Our proposed decoders provide a significant gain in the decoding performance as shown in the statistical analysis. At $\alpha = 0.004$, PPBF

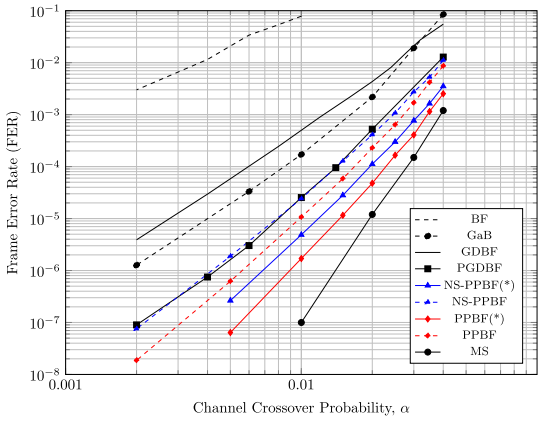


Fig. 13. The decoding performance of the PPBF and the NS-PPBF with $\mathbf{p} = (0, 0.0081, 0.3, 0.7, 1.0)$, on comparison to other BF decoders and to the MS ($It_{max} = 20$) on the Tanner code. The curves with (*) are with $It_{max} = 1000$ while the others are with $It_{max} = 300$.

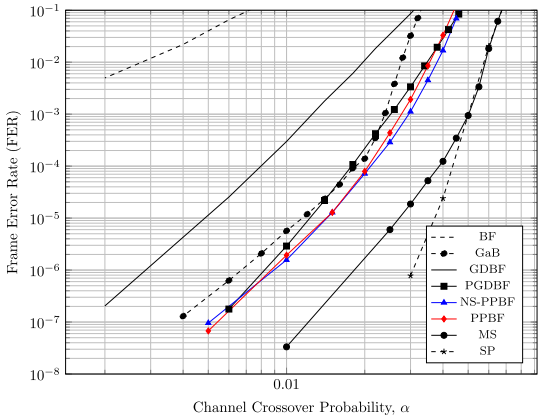


Fig. 14. The decoding performance of the PPBF and the NS-PPBF with $\mathbf{p} = (0, 0.0081, 0.3, 0.7, 1.0)$, on comparison to other BF decoders and to the MS ($It_{max} = 20$), SP ($It_{max} = 20$) on the $d_v = 3, d_c = 6, N = 1296$, code rate $R = 0.5$ LDPC code. All BF-based decoders are with $It_{max} = 300$.

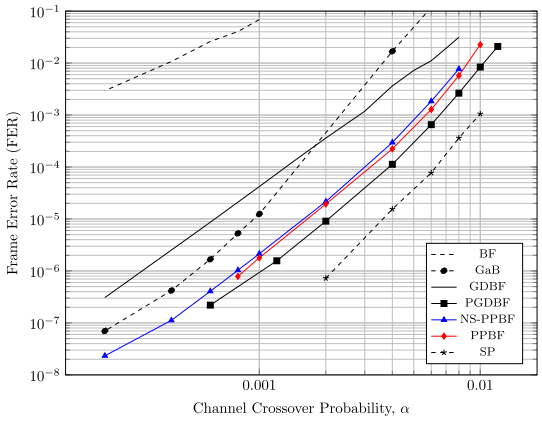


Fig. 15. The decoding performance of the PPBF and the NS-PPBF with $\mathbf{p} = (0, 0.001, 0.1, 1.0, 1.0)$, on comparison to other BF decoders and to the MS ($It_{max} = 20$), SP ($It_{max} = 20$) on the $d_v = 3, d_c = 12, N = 1296$, code rate $R = 0.75$ LDPC code. All BF-based decoders are with $It_{max} = 300$.

TABLE IV
COMPARISON BETWEEN THE PROPOSED BF DECODERS AND THE STATE-OF-THE-ART LDPC DECODERS

	This work		**K.Le'17** [26]		**M.Ismail'13** [20]	**J.Cho'10** [28]	**Nguyen-Ly'16** [21]
(N, M)	$(1296, 648)$		$(1296, 648)$		$(1008, 504)$	$(1057, 813)$	$(1296, 648)$
Decoder	PPBF	NS-PPBF	GDBF	PGDBF	ATBF	SBF	MS(4, 6)
Technology	90nm		90nm		90nm	180nm	65nm
Frequency (MHz)	476	556	385	357	250	345	250
Area (mm^2)	0.367	0.349	0.360	0.367	0.729	7.4	0.72
Area scaled to 90nm (mm^2)	0.367	0.349	0.360	0.367	0.729	1.85	1.38
It_{max}	300	300	300	300	–	40	20
Throughput @ It_{max} **(Gbps)**	2.06	2.45	1.44	1.35	–	$\theta_{min} = 0.253$	2.7
Tput scaled to 90nm (Gbps)	2.06	2.45	1.44	1.35	–	0.506	1.95
It_{ave} **@ FER=** 10^{-5}	7.71	7.58	2.24	5.48	–	–	2.34 [26]
	$@\alpha \approx 0.012$	$@\alpha \approx 0.012$	$@\alpha \approx 0.005$	$@\alpha \approx 0.011$			$@\alpha \approx 0.025$
Average throughput @ FER= 10^{-5} **(Gbps)**	80	95.1 (*)	222.8	84.4	25.2 @ $It = 10$	$\theta_{max} = 17.37$	-
Average throughput scaled to 90nm (Gbps)	80	95.1 (*)	222.8	84.4	25.2	34.74	23.1 [26]
Average TAR $(Gbps/mm^2)$ (90nm)	218	272.4 (*)	618.9	230	34.6	18.8	16.7

(*) Since the SC module is not implemented in this NS-PPBF implementation, these average values of NS-PPBF are bounds on the achievable results and they are not really practical relevance.

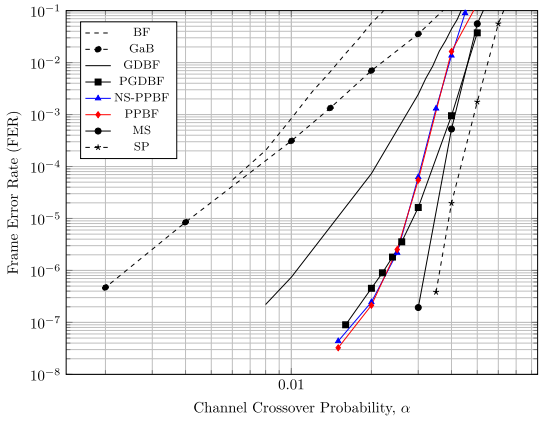


Fig. 16. The decoding performance of the PPBF and the NS-PPBF with $\mathbf{p} = (0, 0.001, 0.05, 0.3, 0.95, 0.95)$, on comparison to other BF decoders and to the MS ($It_{max} = 20$), SP ($It_{max} = 20$) on the $d_v = 4, d_c = 8, N = 1296$, code rate $R = 0.5$ LDPC code. All BF-based decoders are with $It_{max} = 300$.

provides around 2 decades gain in FER compared to PGDBF, 3.3 decades when compared to GDBF. With $It_{max} = 1000$, PPBF gains around 1 decade in FER when compared to itself with $It_{max} = 300$, which confirms the superiority of PPBF when increasing the number of decoding iterations. It is also shown that, the decoding performance of PPBF and NS-PPBF is very close to that of the MS decoder.

The decoding performance on other LDPC codes with different (VN and CN) node degrees and code rates, are shown in Fig. 14, 15 and 16. Fig. 14 shows the performance on the dv3R050N1298 ($d_v = 3, d_c = 6, N = 1296$, code rate $R = 0.5$) LDPC code. With the chosen value of $\mathbf{p}$, PPBF and NS-PPBF have the equivalent error correction and they are better than PGDBF when FER $> 1e^{-5}$. Compared to GDBF decoder, PPBF and NS-PPBF provide a significant gain in performance, half-way approaching the MS, while requiring the equivalent hardware complexity and 1.5 times faster in decoding speed, as shown in the previous section.

The good decoding performance of PPBF and NS-PPBF are also observed on $d_v = 3, d_c = 6, N = 1296$, code rate $R = 0.75$ code (Fig. 15) and $d_v = 4, d_c = 8, N = 1296$, code rate $R = 0.5$ (Fig. 16). Especially, PPBF and NS-PPBF are very close to the MS performance for the latter code.

VI. CONCLUSION

In this paper, we propose a new high-throughput, low-complexity Bit Flipping decoder for Low-Density Parity-Check (LDPC) codes on Binary Symmetric Channel (BSC), called Probabilistic Parallel Bit Flipping (PPBF). The PPBF is more advantageous than other state-of-the-art BF decoders such as the Gradient Descent Bit Flipping and Probabilistic Gradient Descent Bit Flipping decoders, in terms of operating frequency and decoder complexity, thanks to the elimination of the global operation which helps shorten the critical path and reduce the decoder hardware cost. Furthermore, in PPBF, the probabilistic flipping of the Variable Node is introduced for all levels of its energy value rather than only on the maximum values as in PGDBF. PPBF provides, therefore, a very good error correction capability and in some cases, overcomes some error patterns with which PGDBF can not correct. One improved version of PPBF, named as non-syndrome Probabilistic Parallel Bit Flipping (NS-PPBF), is then introduced in which the critical path is further shortened by eliminating the global Syndrome Check module from the data path. The operating frequency of the NS-PPBF is heightened while a negligible error correction degradation observed, with respect to those of the PPBF. The hardware architectures to implement the PPBF and NS-PPBF decoders are introduced with the detailed circuits of the random generator and computing units. The ASIC synthesis results of the proposed architectures re-confirm as expected that, PPBF and NS-PPBF are high hardware-efficiency and high decoding throughput. The proposed PPBF and NS-PPBF appear as the very low-complexity, high throughput decoders with the significant improvement in error correction, which can be seen as the competitive candidates for the next communication and storage standards.

REFERENCES

[1] *IEEE Standard for Information Technology—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput*, IEEE Standard 802.11n-2009, Mar. 2008.

[2] *IEEE Standard for Local and Metropolitan Area Networks—Part 16: Air Interface for Fixed Broadband Wireless Access Systems—Amendment 2: Medium Access Control Modifications and Additional Physical Layer Specifications for 2–11 GHz*, IEEE Standard 802.16a-2003, 2003.

[3] D. Declercq, M. Fossorier, and E. Biglieri, *Channel Coding: Theory, Algorithms, and Applications: Academic Press Library in Mobile and Wireless Communications*. New York, NY, USA: Academic, 2014.

[4] R. G. Gallager, *Low-Density Parity-Check Codes* (Research Monograph). Cambridge, MA, USA: MIT Press, 1963.

[5] B. Unal, A. Akoglu, F. Ghaffari, and B. Vasić, "Hardware implementation and performance analysis of resource efficient probabilistic hard decision LDPC decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, to be published.

[6] *IEEE Standard for Information Technology Telecommunications and Information Exchange Between Systems Specific Requirements Part 3: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Corrigendum 2: IEEE Std 802.An-2006 10GBASE-T Correction*, IEEE Standard 802.3-2005/Cor 2-2007 (Corrigendum to IEEE Std 802.3-2005), Aug. 2007.

[7] S. Jeon and B. V. K. V. Kumar, "Performance and complexity of 32 k-bit binary LDPC codes for magnetic recording channels," *IEEE Trans. Magn.*, vol. 46, no. 6, pp. 2244–2247, Jun. 2010.

[8] K.-C. Ho, C.-L. Chen, Y.-C. Liao, H.-C. Chang, and C.-Y. Lee, "A 3.46 Gb/s (9141,8224) LDPC-based ECC scheme and on-line channel estimation for solid-state drive applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2015, pp. 1450–1453.

[9] G. Dong, N. Xie, and T. Zhang, "On the use of soft-decision error-correction codes in nand flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 2, pp. 429–439, Feb. 2011.

[10] *LDPC Codes for the Enhanced Mobile Broadband (eMBB) Data Channel the 3GPP 5G New Radio*. [Online]. Available: http://www.3gpp.org/ftp/tsgran/WG1RL1/TSGR188/Report/

[11] J. Jung and I.-C. Park, "Multi-bit flipping decoding of LDPC codes for NAND storage systems," *IEEE Commun. Lett.*, vol. 21, no. 5, pp. 979–982, May 2017.

[12] Y. Du, Q. Li, L. Shi, D. Zou, H. Jin, and C. J. Xue, "Reducing LDPC soft sensing latency by lightweight data refresh for flash read performance improvement," in *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Austin, TX, USA, Jun. 2017, pp. 1–6.

[13] O. Al Rasheed, P. Ivaniš, and B. Vasić, "Fault-tolerant probabilistic gradient-descent bit flipping decoder," *IEEE Commun. Lett.*, vol. 18, no. 9, pp. 1487–1490, Sep. 2014.

[14] K. Le, F. Ghaffari, D. Declercq, and B. Vasić, "Efficient hardware implementation of probabilistic gradient descent bit-flipping," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 4, pp. 906–917, Apr. 2017.

[15] R. M. Tanner, D. Sridhara, and T. Fuja, "A class of group-structured LDPC codes," in *Proc. 5th Int. Symp. Commun. Theory Appl.*, Jul. 2001, pp. 365–370.

[16] D. Declercq, C. Winstead, B. Vasić, F. Ghaffari, P. Ivaniš, and E. Boutillon, "Noise-aided gradient descent bit-flipping decoders approaching maximum likelihood decoding," in *Proc. 9th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, Brest, France, Sep. 2016, pp. 300–304.

[17] B. Yuce, H. F. Ugurdag, S. Gören, and G. Dündar, "Fast and efficient circuit topologies forfinding the maximum of n k-bit numbers," *IEEE Trans. Comput.*, vol. 63, no. 8, pp. 1868–1881, Aug. 2014.

[18] K. Le *et al.*, "Efficient realization of probabilistic gradient descent bit flipping decoders," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Lisbon, Portugal, May 2015, pp. 1494–1497.

[19] G. Sundararajan, "Noisy gradient descent bit flip decoding of low density parity check codes: Algorithm and implementation," Ph.D. dissertation, Dept. Elect. Comput. Eng., Utah State Univ., Logan, Utah, 2016.

[20] M. Ismail, I. Ahmed, and J. Coon, "Low power decoding of LDPC codes," *ISRN Sensor Netw.*, vol. 2013, Dec. 2013, Art. no. 650740.

[21] T. T. Nguyen-Ly, T. Gupta, M. Pezzin, V. Savin, D. Declercq, and S. Cotofana, "Flexible, cost-efficient, high-throughput architecture for layered LDPC decoders with fully-parallel processing units," in *Proc. Euromicro Conf. Digit. Syst. Design (DSD)*, Limassol, Cyprus, Aug./Sep. 2016, pp. 230–237.

[22] T. T. Nguyen-Ly, V. Savin, K. Le, D. Declercq, F. Ghaffari, and O. Boncalo, "Analysis and design of cost-effective, high-throughput LDPC decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 3, pp. 508–521, Mar. 2018.

[23] D. Declercq, V. Savin, O. Boncalo, and F. Ghaffari, "An imprecise stopping criterion based on in-between layers partial syndromes," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 13–16, Jan. 2018.

[24] B. Vasić, P. Ivaniš, D. Declercq, and K. LeTrung, "Approaching maximum likelihood performance of LDPC codes by stochastic resonance in noisy iterative decoders," in *Proc. Inf. Theory Appl. Workshop (ITA)*, La Jolla, CA, USA, 2016, pp. 1–9.

[25] F. Leduc-Primeau, F. R. Kschischang, and W. J. Gross, "Modeling and energy optimization of LDPC decoder circuits with timing violations," *IEEE Trans. Commun.*, vol. 63, no. 3, pp. 932–946, Mar. 2018.

[26] K. Le, D. Declercq, F. Ghaffari, L. Kessal, O. Boncalo, and V. Savin, "Variable-node-shift based architecture for probabilistic gradient descent bit flipping on QC-LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 7, pp. 2183–2195, Jul. 2018.

[27] G. Sundararajan, C. Winstead, and E. Boutillon, "Noisy gradient descent bit-flip decoding for LDPC codes," *IEEE Trans. Commun.*, vol. 62, no. 10, pp. 3385–3400, Oct. 2014.

[28] J. Cho, J. Kim, and W. Sung, "VLSI implementation of a high-throughput soft-bit-flipping decoder for geometric LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 5, pp. 1083–1094, May 2010.

Khoa Le received the bachelor's and M.Sc. degrees in electronics engineering from the Ho Chi Minh City University of Technology, Vietnam, in 2010 and 2012, respectively, and the Ph.D. degree from the University of Cergy-Pontoise, France, in 2017. He is currently a Post-Doctoral Researcher with the ETIS Laboratory, ENSEA, France. His research interests are in error correcting code algorithms and analysis and their implementations in field-programmable gate array/application-specific integrated circuit.

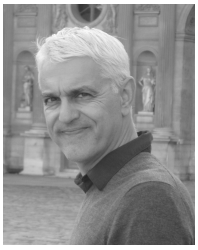
Fakhreddine Ghaffari received the Degree in electrical engineering and the master's degree from the National School of Electrical Engineering, Tunisia, in 2001 and 2002, respectively, and the Ph.D. degree in electronics and electrical engineering from the University of Sophia Antipolis, France, in 2006. He is currently an Associate Professor with the University of Cergy-Pontoise, France. His research interests include very large-scale integration design and implementation of reliable digital architectures for wireless communication applications in application-specific integrated circuit/field-programmable gate array platform and the study of mitigating transient faults from algorithmic and implementation perspectives for high-throughput applications.

Lounis Kessal received the Ph.D. degree in microelectronics from Paris-XI University in 1987. He is currently an Associate Professor with the ENSEA Engineering School, where he involved in research activities with the ETIS Laboratory. His research interests are essentially on real-time image processing. His previous research works lead him to study the interest of the dynamic reconfiguration approach in architectures dedicated to signal and image processing. He focused on reconfigurable systems on chip and application-specific integrated circuit/field-programmable gate array implementations of error correcting code algorithms.

David Declercq (SM'11) was born in 1971. He received the Ph.D. degree in statistical signal processing from the University of Cergy-Pontoise, France, in 1998. He held a junior position with the Institut Universitaire de France from 2009 to 2014. He is currently a Full Professor with ENSEA, Cergy-Pontoise. His research topics lie in digital communications and error-correction coding theory. He focused on the particular family of low-density parity-check (LDPC) codes, both from the code and decoder design aspects for several years. Since 2003, he has been developing a strong expertise on non-binary LDPC codes and decoders in high-order Galois fields GF(q). A large part of his research projects are related to non-binary LDPC codes. He mainly investigated two aspects: 1) the design of GF(q) LDPC codes for short and moderate lengths and 2) the simplification of the iterative decoders for GF(q) LDPC codes with complexity/performance tradeoff constraints. He has authored over 40 papers in major journals (IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE TRANSACTIONS ON INFORMATION THEORY, IEEE COMMUNICATION LETTERS, and EURASIP JWCN) and over 120 papers in major conferences in Information Theory and Signal Processing. He is the General Secretary of the National GRETSI Association.

Emmanuel Boutillon was born in Chatou, France, in 1966. He received the Diploma degree in engineering from the Telecom Paris Tech, Paris, France, in 1990, and the Ph.D. degree in 1995. In 1991, he joined the Ecole Multinationale Supérieure des Télécommunications, Dakar, Africa, as an Assistant Professor. In 1992, he joined Telecom Paris Tech as a Research Engineer, where he conducted research in the field of very large-scale integration for digital communications. In 1998, he spent a sabbatical year at the University of Toronto, ON, Canada. In 2000, he joined the University of Bretagne Sud as a Professor. He was the Head of the LESTER Lab from 2005 to 2007 and the CACS Department from 2008 to 2015. In 2011, he had a sabbatical year at INICTEL-UNI, Lima, Peru. He is currently the Scientific Adviser of the Lab-STICC. His research interests are on the interactions between algorithm and architecture in the field of wireless communications and high-speed signal processing. He focuses on turbo codes and LDPC decoders.

Chris Winstead received the B.S. degree in electrical and computer engineering from The University of Utah in 2000 and the Ph.D. degree from the University of Alberta in 2005. From 2013 to 2014, he was a Fulbright Visiting Professor with the Universite de Bretagne Sud, Lorient, France. He is currently an Associate Professor with the Electrical and Computer Engineering Department, Utah State University. His research interests include algorithms for probabilistic error correction and inference, particularly Bayesian networks and hidden Markov models, implementation of high-performance forward error correction techniques, especially low-density parity check codes, mixed-signal circuit design for reliable wireless communication, low-power and fault-tolerant very large-scale integration circuits, and theory of stochastic computation with applications in electronics and synthetic biology. He is a member of the Tau Beta Pi Engineering Honor Society. He received the NSF Career Award for research in low-energy wireless communication circuits in 2010.



Bane Vasić is currently a Professor of electrical and computer engineering and mathematics with The University of Arizona. He is with BIO5, the Institute for Collaborative Bioresearch. He is also the Director of the Error Correction Laboratory. He is an Inventor of the soft error-event decoding algorithm, and the Key Architect of a detector/decoder for Bell Labs data storage chips which were regarded as the best in industry. He is the Chair of the IEEE Data Storage Technical Committee, an Editor of the IEEE JSAC Special Issues on Data Storage Channels, and a member of the Editorial Board of the IEEE TRANSACTIONS ON MAGNETICS. He is a da Vinci Circle and Fulbright Scholar, and a Founder and a Chief Scientific Officer of Codelucida.