A Novel High-Throughput, Low-Complexity Bit-Flipping Decoder for LDPC Codes

Khoa Le*, Fakhreddine Ghaffari*, David Declercq*, Bane Vasic†, Chris Winstead‡
*ETIS, UMR-8051, Université Paris Sein, Université de Cergy-Pontoise, ENSEA, CNRS, France,
{khoa.letrung, fakhreddine.ghaffari, declercq}@ensea.fr
†Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, USA, vasic@ece.arizona.edu
‡Department of Electrical and Computer Engineering, Utah State University, UT, USA, chris.winstead@usu.edu

***Abstract*—This paper presents a new high-throughput, low-complexity Bit Flipping (BF) decoder for Low-Density Parity-Check (LDPC) codes on the Binary Symmetric Channel (BSC), called Probabilistic Parallel Bit Flipping (PPBF). The advantage of PPBF comes from the fact that, no global operation is required during the decoding process and from that, all of the computations could be parallelized and localized at the computing units. Also in PPBF, the probabilistic feature in flipping the Variable Node (VN) is incorporated for all satisfaction level of its CN neighbors. This type of probabilistic incorporation makes PPBF more dynamic to correct some error patterns which are unsolvable by other BF decoders. PPBF offers a faster decoding process with an equivalent error correction performance to the Probabilistic Gradient Descent Bit Flipping (PGDBF) decoder, which is better than all so-far introduced BF decoders in BSC channel. A hardware implementation architecture of PPBF is also presented in this paper with detailed circuits for the probabilistic signal generator and processing units. The implementation of PPBF on FPGA confirms that, the PPBF complexity is much lower than that of the PGDBF and even lower than the one of the deterministic Gradient Descent Bit Flipping (GDBF) decoder. The good decoding performance along with the high throughput and low complexity lead PPBF decoder to become a brilliant candidate for the next generation of communication and storage standards.**

I. Introduction

Low-Density Parity-Check (LDPC) codes have been adopted to several communication standards such as IEEE 802.11n, 802.3an, 802.16a *etc.* [1][2][3] due to their near-capacity error correction capability and their manageable complexity implementation [4]. LDPC can be effectively decoded by the either *soft-information* message passing decoders, *i.e.* Min Sum (MS), Sum Product (SP) [4], or *hard-decision* decoders, *i.e.* Bit Flipping (BF), Gallager-A,B [5]. The hard decision decoders have been recently attracted much attention due to the high decoding throughput, low-complexity implementation and the progressive improvement in error correction, approaching or even surpassing the soft-decision decoders [6]. The LDPC decoding is an iterative process where messages are iteratively passed between the computing units, Variable Nodes (VNs) and Check Nodes (CNs). The low-complexity of hard decision decoders comes from the fact that only 1-bit messages are iteratively passed between VNs and CNs which simplifies the connection networks. Furthermore, the computing units which process these 1-bit messages, are very simple and therefore, enhance the decoding throughput. In this paper, we focus on this hard decision decoder to further improve the decoding throughput and decoder complexity while maintaining (or even improving) the performance of the best BF decoder.

Several BF decoders have been proposed recently in literature where Gradient Descent Bit Flipping (GDBF) and Probabilistic Gradient Descent Bit Flipping (PGDBF) could be seen as the best algorithms, in term of error correction, on BSC channel. GDBF is proposed by Wadayama *et al.* in 2010 [7]. In GDBF, CN unit computes the parity check (Exclusive-OR operation) on all of its connected VNs, as in 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 value is a minima compared to all other energies. GDBF provides a very good error correction and it is better than prior-introduced deterministic BF decoders. PGDBF is a variant of GDBF, introduced by Rasheed *et al.* in 2014 [8], applied on Binary Symmetric Channel (BSC). PGDBF follows precisely the decoding steps of GDBF and differs only from the VN flipping operations. All the flipping candidates in GDBF are only flipped with a probability of p_0 ($p_0 < 1$) in PGDBF. This probabilistic behavior interestingly improves the decoding performance, far better than the original GDBF and very close to MS [8].

A drawback of GDBF and PGDBF is that, a global sorting (the Maximum Finder - denoted as MF, in [9]) is required to identify the minima (or maxima) among all VN energy values. This global operation lengthens the critical path of the decoders and limits the maximum achievable frequency, especially when the codeword length is extended. Although the decoding throughput of GDBF and PGDBF is very promising as shown in [9],[10] the global operation is the main obstacle in further improving the operating frequency and therefore, the decoding throughput. Furthermore, the MF is the most hardware exhaustive part in GDBF and PGDBF decoders [9].

This paper presents a new probabilistic parallel BF decoder (PPBF) in which the global operation is not required. The flipping decision in each VN of PPBF is made intrinsically in the VN basing only on its neighbor CN values and the signal from the probabilistic signal generator. The critical path is shortened and the flipping is, therefore, parallelized, which significantly enhances the decoding throughput. More inter-

978-1-5386-2896-6/17/$31.00 ©2017 IEEE

estingly, by eliminating the global operation, PPBF becomes a very low-complexity, even lower than deterministic GDBF decoder, while providing a comparable decoding performance to PGDBF.

The rest of the paper is organized as following. In Section II, the notations of LDPC codes and LDPC decoding algorithms are firstly introduced. The PPBF algorithm is then presented. It is shown that, PPBF has the similar property of PGDBF decoder in probabilistic flipping while not requiring the MF. Another difference is that, PPBF applies the VN flip possibility at all energy values rather than only the maximum one. In Section III, an implementation architecture of PPBF is presented with detailed circuit for the probabilistic signal generator and processing units. In Section IV, we show for the test case that, the PPBF requires only $1/2$ the Look-Up-Tables (LUTs) of a GDBF implementation in the literature while the obtained maximum frequency is 183% higher. The decoding performance is shown to be as good as the PGDBF decoder. The same conclusion can be drawn for another longer test code where the maximum frequency is around 161% faster while reducing 33% LUTs compared to a PGDBF implementation. The good decoding performance along with the high throughput and low complexity lead PPBF decoder to become a brilliant candidate for the next generation of communication and storage standards. Section V concludes the paper.

II. The Probabilistic Parallel Bit Flipping Decoder

A. Notations

An LDPC code is defined by a sparse parity-check matrix H (M, 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. 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 $(0 \leq m \leq M)$ by an edge in the Tanner graph if the entry $H(m, n) = 1$. We denote the set of CNs connected to the VN v_n as $\mathcal{N}(v_n)$, $(1 \leq n \leq N)$, and $|\mathcal{N}(v_n)|$ is called VN degree. Similarly, $\mathcal{N}(c_m)$, $(1 \leq m \leq M)$ denotes all VNs connected CN m and $|\mathcal{N}(c_m)|$ is the CN degree. 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 $\mathbf{x} = \{x_n\} = \{0, 1\}^N$, $(1 \leq n \leq N)$ is called a codeword if and only if $H\mathbf{x}^T = 0$. $\mathbf{x}$ is sent through a transmission channel and we denote $\mathbf{y} = \{y_n\}$, $(1 \leq n \leq N)$ as the channel output. The decoder presented in this paper is 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.*, $prob(y_n = x_n) = 1 - \alpha$ and $prob(y_n = 1 - x_n) = \alpha$, $1 \leq n \leq N$.

B. The Probabilistic Parallel Bit Flipping Decoder

The proposed Probabilistic Parallel Bit Flipping Decoder is an A Posteriori Probability (APP) message passing decoding which iteratively updates the VN values via the decoding iterations. We denote c_m (*resp.* v_n) as CN (*resp.* VN) itself while $c_m^{(k)}$ (*resp.* $v_n^{(k)}$) shows value of the CN (*resp.* VN) at iteration k.

The CN computation in PPBF is a parity check on all values of its neighbor VNs, as in the standard BF. The CN computing equation is formulated as in Equ. 1 where $\oplus$ is the bit-wise Exclusive-OR (XOR) operation. The PPBF decoding process is terminated when all CNs equations are satisfied, *i.e.* $c_m^{(k)} = 0, \forall m$, or k reaches to the maximum value K allowed ($k = K$).

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

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

At the k iteration, each VN updates its value by the following steps. First, the VN evaluates its energy function, denoted by $E_n^{(k)}$, $1 \leq n \leq N$, using Equ. 2. The energy value is a sum of the neighbors CN values and the similarity between the VN current value to its channel output (computed by $\oplus$ function). It is clear that, the energy value for any VN varies between 0 to $d_v + 1$, $(0 \leq E_n^{(k)} \leq d_v + 1)$ and that the energy computation is similar to those of the GDBF and PGDBF decoders in [8]. Second, in the flipping step, each VN v_n is flipped with a probability depending on its energy values, *i.e.* VN v_n is flipped with p_e if $E_n^{(k)} = e$ where $p_e \in \mathbf{p} = \{p_0, p_1, ..., p_{d_v+1}\}$, $p_0 = 0$, $p_{d_v+1} = 1$ and $0 < p_e < 1$ when $1 \leq e \leq d_v$. The updated values of VNs are sent to the next decoding iteration. The PPBF decoding is described in algorithm 1 where we denote $R_n^{(k)} = \{0, 1\}$ is a random value and $\mathbf{R} = \{R_n\}, 1 \leq n \leq N$.

Algorithm 1 Probabilistic Parallel Bit Flipping (PPBF)

Initialization $k = 0$, $v_n^{(0)} \leftarrow y_n, 1 \leq n \leq N$.
Initialization $\mathbf{p} = \{p_0, p_1, ..., p_{d_v+1}\}$, $p_0 = 0, p_{d_v+1} = 1$.
$\mathbf{s} = H\mathbf{v}^{(0)^T} \mod 2$
while $\mathbf{s} \neq \mathbf{0}$ *and* $k \leq K$ **do**
  Compute $c_m^{(k)}$, $1 \leq m \leq M$, using Eq. (1).
  Compute $E_n^{(k)}$, $1 \leq n \leq N$, using Eq. (2).
  Generate $R_n^{(k)}$, $1 \leq n \leq N$, from $\mathcal{B}(p), p = p_{E_n^{(k)}}$.
  for $1 \leq n \leq N$ **do**
    $v_n^{(k+1)} = v_n^{(k)} \oplus R_n^{(k)}$
  end for
  $\mathbf{s} = H\mathbf{v}^{(k+1)^T} \mod 2$
  $k = k + 1$
end while
Output: $\mathbf{v}^{(k)}$

It can be seen that, PPBF follows similarly the decoding steps of PGDBF decoder and the differences are two folds. First, PPBF does not require the maximum energy in making

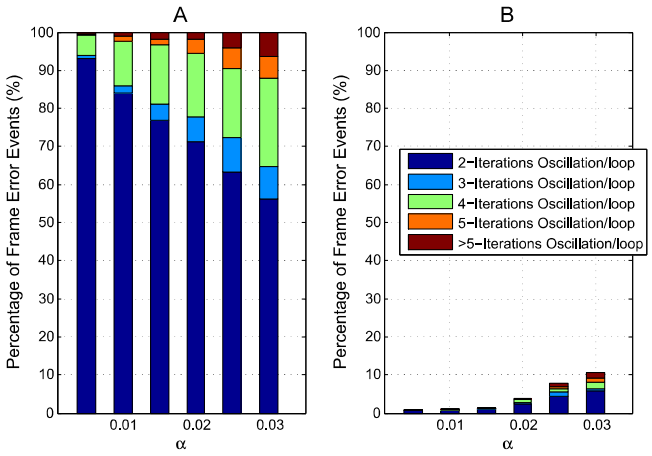

Figure 1. The statistical analysis of decoding failure of BF decoders on the Tanner code: A. The failed type in error correction of GDBF decoder, B. The remaining after re-decoding the error patterns in (A) by PPBF.

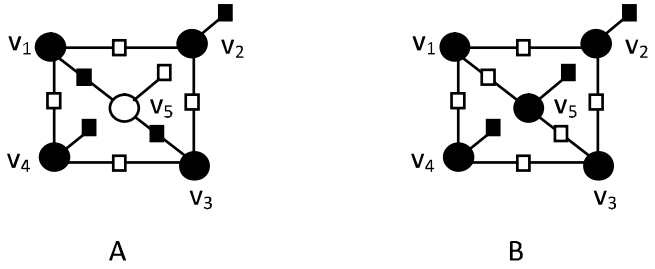

Figure 2. The 4-bits error pattern with which PGDBF failed to correct while PPBF can correct. The correct VN (satisfied CNs) are denoted as white-filled circles (squares), the uncorrect VN (unsatisfied CNs) are the black-filled circles (squares).

flip decision and therefore, no global sorting is needed. It comes from the fact that, each VN flips its value basing only on its own energy and the random signal input regardless the energies of other VNs. Second, in the binary random generating step, the probability of generated bit, $R_n^{(k)}$, is dynamically changed on the VN and the decoding iteration. This probability is controlled by the VN enegy values in each iteration while in PGDBF, it is fixed for all iterations and is equal for all VNs.

Despite the above differences, PPBF has the good property known as *oscillation breaking* to improve error correction capability [11][12]. Indeed, the failure of GDBF decoder comes from the fact that, during the decoding process, it falls into the infinite oscillations (or loops) between the states, which prevent converging to the correct codeword [9],[11]. The probabilistic flip manner of PPBF helps break these oscillations and converge. We conducted a statistical analysis on the failures of GDBF decoder on the $d_v = 3, d_c = 6, N = 155, M = 93$ [13] (denoted by Tanner code) and plot results in Figure 1. In Figure 1A, we classified all the failure cases of GDBF by the ℓ-iterations oscillation types, *i.e.* the position of bits in error repeated after ℓ iterations. It can be seen that, the 2-iterations oscillation dominates other types in the GDBF failure, especially in the low error region. We then re-decoded the GDBF failure cases by PPBF and plot the result in Figure 1B. It is remarked that, the PPBF breaks the oscillation and corrects almost the failure cases of GDBF, especially in low error region. This above interesting property is the source of PPBF error correction improvement.

The result of PPBF in the above statistic is even better than the PGDBF decoder since there are some error patterns with which PGDBF failed while PPBF could correct. These error patterns also lead PGDBF to an infinite oscillations (loops) and fail to converge to the correct codeword. We show one of these error patterns in Figure 2 where there are 4 bits in error allocated in a Trapping Set $(5, 3)$ [9]. Figure 2A describes the error positions initially received from the channel with which the VN v_5 has maximum energy ($E_5^{(0)} = 2$) (other VNs have energy of 1) and v_5 becomes the only flip candidate. In the case PGDBF flips v_5, the error locations are in Figure 2B and in this case, v_5 is again the only flip candidate since it has energy of 2 while others have either 1 or 0. PGDBF sticks between 2 states (Figure 2A and 2B) regardless the flip decision made. PPBF, on contrary, corrects this error pattern because it not only flips v_5 with p_2 but also other VN with $p_1 > 0$ and from this, it may escape from the oscillation and converge as in the statistical result. We keep the further analysis in another work.

The vector $\mathbf{p}$ affects not only to the decoding performance but also to the implementation and the average iterations. Due to the lack of space, the optimization of $\mathbf{p}$ and theoretical analysis on the decoding performance gain are not introduced and presented in another work. In this paper, we use the empirical optimized value, $\mathbf{p} = \{0, 0.01, 0.1, 0.9, 1\}$ for the $d_v = 3$ test codes, which provides a good decoding performance and facilitates the hardware implementation, as shown in the next section. One can also choose experimentally other values of $\mathbf{p}$ and it may line on the following criterions. In principle, the bit that has higher energy value should have higher probability of flip. It is obvious to choose $p_0 = 0$ since the VN having all neighbor CN satisfied and being similar to the channel received value, should not be flipped. The VN that has all neighbor CN unsatisfied and disagrees to the channel received value, should be flipped ($p_4 = 1$). The VN that has energy of 1 should be flipped with a small probability (close to 0) to avoid flip too many correct VNs unintentionally since there may exist many correct VNs having the energy of 1.

III. The Hardware Architecture for PPBF Decoder

In this section, we present the hardware architecture for the PPBF decoder. We illustrate for the $d_v = 3$ LDPC code implementation and the extension for other VN degrees is straightforward.

A. The probabilistic signal generator for PPBF decoder

A straightforward that one could apply to implement the probabilistic signal generator (random generator - RG), is

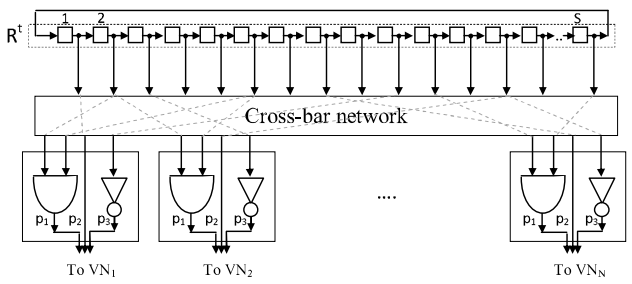

Figure 3. The proposed probabilistic signal generator for PPBF decoder.

to use the Linear Feedback Shift Register (LFSR) random generator as in [10] where the energy value controls the LFSR decision threshold. This method may be costly since each VN requires an LFSR module. In this paper, we propose to use a method called Cyclic Shift Truncated Sequence (CSTS) introduced in [9]. In this optimization, a short register sequence R^t with size $|R^t| = S < N$ is allocated and stores the randomly generated bits, being 1 with certain probability. The register outputs of this short sequence are then randomly triggered to the Probability Controlling Units (PCU) by a cross-bar network as in Figure 3. R^t is cyclically shifted from one iteration to another to make its outputs differently.

N PCU modules are designed for N VNs. Each PCU has 3 outputs internally named as p_1, p_2 and p_3. These outputs have the value of 1 with the probability correspondingly to p_1, p_2 and p_3 in $\mathbf{p}$. We use the logic gates to control the probability output as used in [10]. In this particular case for $d_v = 3$ where $p_1 = 0.01$, $p_2 = 0.1$ and $p_3 = 0.9$, R^t stores the $p = 0.1$ randomly generated sequence. Then, $p_2 = p = 0.1$, $p_3 = NOT(p) = 0.9$, $p_1 = p\text{AND}p = 0.01$ and are realized by the logic gates as in Figure 3.

An issue emerging is that, the correlation may be significant and may affect to the decoding performance when the size S of R^t is small. We have conducted a statistic on the decoding performance as a function of S as in [9] and the same conclusion was drawn. That is, S could be small (for the test code $d_v = 3, d_c = 6, N = 1296, M = 648$ LDPC code [14] - denoted by dv3R050N1296, $S = 216$ is sufficient) without performance loss.

B. The computing units of PPBF decoder

The proposed architecture to implement PPBF is presented in Figure 4. The CN processing units (CNU) are simply the d_c-inputs XOR gates which have the VN messages, propagated by the connection network 2, as the inputs. The CNU outputs are transmitted by the connection network 2 to the VNU inputs ($\mathbf{c}_{v_n}$ denotes the $\mathcal{N}(v_n)$, $1 \leq n \leq N$). In each variable node processing unit (VNU), two registers to store the channel output value (y_n) and the intermediate value of the VN at the iterations k-th, ($v_n^{(k)}$) are implemented. By taking the result of the XOR1, the summation block computes the energy value which plays as the selection input of the multiplexer module. The inputs of the multiplexer are either the output of RG or a fixed values 0 or 1. The XOR2 is in charge of flipping the bit ($v_n^{(k)}$) whenever the multiplexer output is 1, and this value is stored back to the register as the value for the next iteration. PPBF decodes 1 iteration in 1 clock cycle.

IV. Synthesis Results and Decoding Performance

A. Synthesis results

We make the comparisons by implementing PPBF decoder for a short code $N = 155$ (Tanner code) and a longer code $N = 1296$ (dv3R050N1296 code) on FPGA Xilinx Virtex 6. The strategy is to look for minimum timing constraint to which the synthesizer (the ISE 14.7) can pass (which indicates the maximum achievable frequency that the decoders can operate) after place-and-route synthesizing processing. The results are reported in Table I for Tanner code and Table II for dv3R050N1296 code. We denote θ as the target decoding throughput and compute by the equation: $\theta = \frac{N * F_{max}}{k_{ave}}$ where F_{max} is the maximum decoding frequency, k_{ave} denotes the target average iterations. By defining k_{ave}, we roughly ignore the effect of channel crossover probability α in computing θ. k_{ave} is set by 10 for all throughput computations in this paper.

In the first comparison, the hardware results of GDBF and PGDBF in Table I are extracted from [10] in literature. It can be seen that, the PPBF significantly reduces the hardware cost even compared to GDBF. It requires only 50% of registers and 46% of LUTs of GDBF implementation. It is a significant gain since PPBF offers an equivalent decoding performance to PGDBF (shown in next section) while the PGDBF implementation requires an additional 9.7% of registers and 12.1% of LUTs of GDBF. As expected, the PPBF maximum frequency is 85% higher than PGDBF (and GDBF) which leads to a higher decoding throughput, 3.78 Gbps compared to 2.04 Gbps of PGDBF.

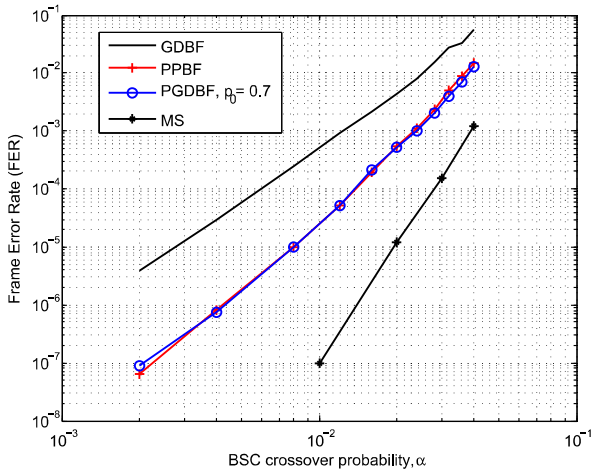

Figure 5. The decoding performance of PPBF, on comparison to other BF decoders and to MS on the Tanner code.

The synthesis results and decoding throughput when implementing on the $N = 1296$ code is presented in Table II. Since the hardware implementation results of GDBF and PGDBF in

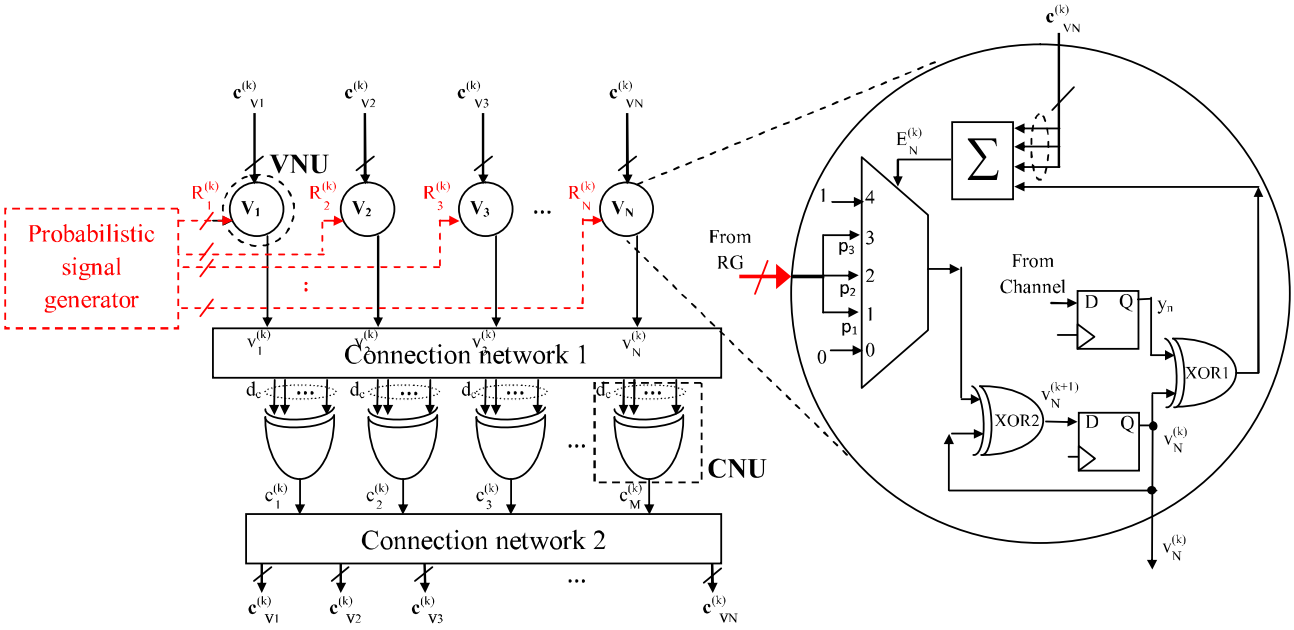

Figure 4. The proposed architecture for the realization of PPBF decoder for $d_v = 3$ LDPC codes.

Table I
SYNTHESIS RESULTS AND DECODING THROUGHPUT OF PPBF ON COMPARISON TO OTHER BF DECODERS ON THE TANNER CODE.

	1-bit Register	Slice LUTs	F_{max} (MHz)	Throughput (Gbps)
Non-Probabilistic GDBF [10]	946 (+0%)	2151 (+0%)	132 (+0%)	2.04 (+0%)
PGDBF with IVRG [10]	1038(+9.7%)	2412(+12.1%)	132 (+0%)	2.04 (+0%)
PPBF (S= 155)	476 (-50%)	991 (-54%)	244 (+85%)	3.78 (+85%)

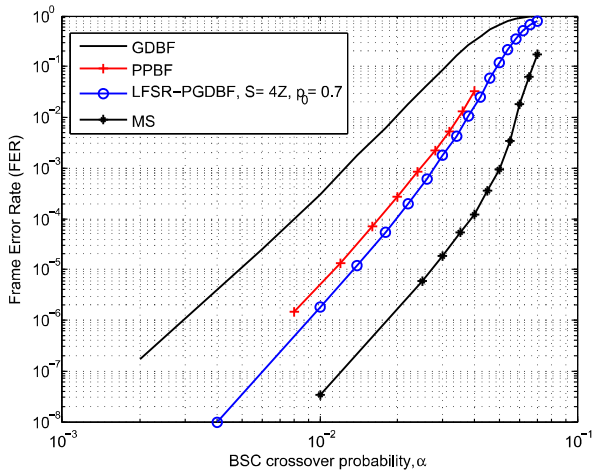

Figure 6. The decoding performance of PPBF, on comparison to other BF decoders and to MS on the dv3R050N1296 code.

[9] on FPGA were not introduced, we design these decoders by following the descriptions in that paper, synthesize on the FPGA and report the results on Table II. The same remarks can be drawn as the previous comparison. The PPBF reduces up to 23.9% of LUTs compared to GDBF while requires 8.3% of registers overhead. This extra registers required are for the RG implementation. The throughput improvement is around 60% compared to PGDBF, 15.8 Gbps compared of 9.85 Gbps of PGDBF.

B. Decoding performance

The simulated decoding performance of decoders are shown in Figure 5 for Tanner code and in Figure 6 for dv3R050N1296 code. In general, PPBF decoder is equivalent to the PGDBF decoder. For the performance in Tanner code, it is strictly close to PGDBF and far better than GDBF. Since PPBF can solve some low-weight error patterns presented above, the advantage of PPBF over PGDBF can be seen at the very low Frame Error Rate (error floor). There is a negligible performance loss of PPBF compared to PGDBF on dv3R050N129 code and both of them are halfway towards the MS decoding performance.

V. CONCLUSION

We propose in this paper 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). PPBF is more advantageous than Gradient Descent Bit Flipping and

Table II
SYNTHESIS RESULTS AND DECODING THROUGHPUT OF PPBF ON COMPARISON TO OTHER BF DECODERS ON THE DV3R050N1296 CODE.

	1-bit Register	Slice LUTs	F_{max} (MHz)	Throughput (Gbps)
Non-Probabilistic GDBF [9]	2608 (+0%)	9528 (+0%)	76 (+0%)	9.85 (+0%)
LFSR-PGDBF, $S = 216$ [9]	2858(+9.6%)	10839(+13.8%)	76 (+0%)	9.85 (+0%)
PPBF (S= 216)	2825 (+8.3%)	7250 (-23.9%)	122 (+60.5%)	15.81 (+60.5%)

Probabilistic GDBF in term of operating frequency and decoder complexity thanks to the elimination of the global operation This elimination helps shorten the critical path and reduce the decoder hardware cost. Furthermore, in PPBF, the probabilistic VN flip is introduced for all level of energy values rather than only on the maximum values as in PGDBF. PPBF provides a very good error correction capability and in some case, overcomes some error patterns with which other BF decoders can not correct. The PPBF FPGA implementations using the proposed architecture re-confirm that, PPBF is a very low-complexity, high throughput decoder. PPBF is, therefore, a competitive candidate for the next communication and storage standards.

ACKNOWLEDGMENT

This work was funded by the french ANR under grant number ANR-15-CE25-0006-01 (NAND project).

REFERENCES

[1] "Wireless lan medium access control and physical layer specications: Enhancements for higher throughput, p802.11n/d3.07," *IEEE-802.11n*, Mar. 2008.
[2] "Local and metropolitan area networks - part 16, ieee std 802.16a-2003," *IEEE-802.16*.
[3] "Ieee standard for information technology - corrigendum 2: Ieee std 802.3an-2006 10gbase-t correction," *IEEE Std 802.3-2005/Cor 2-2007 (Corrigendum to IEEE Std 802.3-2005)*, Aug. 2007.
[4] D. Declercq, M. Fossorier, and E. Biglieri, "Channel coding: Theory, algorithms, and applications," *Academic Press Library in Mobile and Wireless Communications, Elsevier, ISBN: 978-0-12-396499-1*, 2014.
[5] R. G. Gallager, "Low density parity check codes," *MIT Press, Cambridge, 1963, Research Monograph series*, 1963.
[6] G. Sundararajan, C. Winstead, and E. Boutillon, "Noisy gradient descent bit-flip decoding for ldpc codes," *IEEE Transactions on Communications*, vol. 62, no. 10, pp. 3385–3400, Oct 2014.
[7] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, "Gradient descent bit flipping algorithms for decoding ldpc codes," *IEEE Transactions on Communications*, vol. 58, no. 6, pp. 1610–1614, June 2010.
[8] O. A. Rasheed, P. Ivanis, and B. Vasić, "Fault-tolerant probabilistic gradient-descent bit flipping decoder," *IEEE Communications Letters*, vol. 18, no. 9, pp. 1487–1490, Sept 2014.
[9] K. Le, F. Ghaffari, D. Declercq, and B. Vasić, "Efficient hardware implementation of probabilistic gradient descent bit-flipping," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 4, pp. 906–917, April 2017.
[10] K. Le, D. Declercq, F. Ghaffari, C. Spagnol, E. Popovici, P. Ivanis, and B. Vasíc, "Efficient realization of probabilistic gradient descent bit flipping decoders," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 1494–1497.
[11] D. Declercq, C. Winstead, B. Vasic, F. Ghaffari, P. Ivanis, and E. Boutillon, "Noise-aided gradient descent bit-flipping decoders approaching maximum likelihood decoding," in *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Sept 2016, pp. 300–304.
[12] B. Vasić, P. Ivanis, and D. Declercq, "Approaching maximum likelihood performance of ldpc codes by stochastic resonance in noisy iterative decoders," in *Information Theory and Applications Workshop*, Feb. 2016.
[13] M. Tanner, D. Srkdhara, and T. Fuja, "A class of group-structured ldpc codes," in *Proc. 5th Int. Symp. Commun. Theory App.*, July 2001.
[14] 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 *2016 Euromicro Conference on Digital System Design (DSD)*, Aug 2016, pp. 230–237.