Efficient FPGA Implementation of Probabilistic Gallager B LDPC Decoder

Fakhreddine Ghaffari †, Burak Unal*, Ali Akoglu*, Khoa Le†, David Declercq† and Bane Vasić*
*Dep. of Electrical and Computer Engineering, Uni. of Arizona, Tucson, AZ, 85719 USA
{burak,akoglu,vasic}@email.arizona.edu
†ETIS, UMR 8051,Univ Paris Seine, Univ Cergy-Pontoise, ENSEA, CNRS, France
{ghaffari, khoa.letrung, declercq}@ensea.fr

***Abstract*—This paper presents the performance evaluation of the Probabilistic Gallager B (PGaB), a hard decision message passing Low Density Parity Check (LDPC) Decoder, with respect to the soft decision based decoders MinSum (MS) and Offset-MinSum (OMS). PGaB algorithm relies on introducing deliberate and controlled randomness to some of the exchanged messages of the GaB decoder such that it is able to escape from local minima associated with dominant trapping sets. We show that PGaB delivers higher decoding throughput than soft decision based decoders MS and OMS while using much fewer amount of Field Programmable Gate Array (FPGA) resources. Our Monte-Carlo simulation results show that the decoding performance of the PGaB on Binary Symmetric Channel (BSC) is far better than the deterministic GaB and very close to MS and OMS performances especially in error floor region.**

Keywords: ***high-performance probabilistic hard-decision LDPC decoders, FPGA architecture, hardware complexity/decoding performance trade-off.***

I. Introduction

Low-Density Parity-Check (LDPC) codes have competently captivated research attention due to the capacity-approaching decoding performance under iterative decoding operation, which iteratively exchange information (message) between the nodes of the Tanner graph representation of the LDPC code [1]. The LDPC decoding algorithms are classified into mainly two groups: soft-decision and hard-decision decoders. Soft-decision decoders are based on the concept of belief propagation (Belief Propagation (BP), Min-Sum (MS), Offset MinSum [2] etc.) and are very powerful in terms of error correction, but require intensive amount of computation; On the other hand, hard-decision decoders, (Gallager B (GaB), Bit-Flipping (BF), Gradient Descent Bit-Flipping (GDBF) etc.), which have an error-correcting performance not as good as those of the soft-decision decoders, but they enjoy low-complexity and high-speed decoder implementations in hardware. In the Gallager B algorithm for example, a message node decides to send a message contradicting its received value if at least a predetermined number, b, of its neighbors disagree with its received value. It has been shown that GaB decoder outperforms several algorithms such as BF and Gallager A, while still maintaining the simplicity of message passing hard decision decoders. More powerful version of the GaB called Noisy Gallager B (NGaB) has been recently proposed by Vasic *et al.* [3] with far better error correction ability than the original GaB by including a simple probabilistic feature in messages exchanged between Check Node Units (CNUs) and Variable Node Units (VNUs). In NGaB, both messages exchanged between VNU and CNU nodes are flipped with probability p_0 after their respective computations. This probabilistic feature helps the NGaB to approach the soft-decoding algorithm performance. Based on simulations and cases studies, the study in [3] shows that the Noisy GaB can correct some error patterns that are uncorrectable by the GaB. However, there does not exist any systematic explanation in how the probabilistic feature helps this new algorithm to outperform its deterministic counterpart.

In this paper, we evaluate a version of the GaB based on an addition of a deliberate randomness on specific computations of the decoder. We call this algorithm as the Probabilistic GaB (PGaB) [4]. We show the benefit of the inclusion of controlled noise on decoding performance with very low impact on the hardware complexity. We conduct exhaustive performance evaluation by designing hardware architectures for both GaB and PGaB algorithms and implementing them on the Xilinx Virtex-6 Field Programmable Gate Array (FPGA). We evaluate the robustness of the PGaB with respect to MS and OMS based on both simulation and hardware implementation analysis. We conclude that PGaB is able to bridge the gap between GaB and complex soft decoding algorithms such as MS and OMS without sacrificing the throughput advantage of the deterministic GaB.

The rest of the paper is organized as follows. In section II, we first give an overview of the baseline algorithm (GaB) and then introduce Probabilistic Gallager B algorithm. In section III, we present our simulation environment and configuration parameters, followed by FPGA implementation and hardware performance analysis in section IV. Section V presents the decoding performance of PGaB with respect to GaB, MS, and OMS followed by our robustness analysis of the PGaB over two examples of LDPC codes. Finally, section VI concludes the paper .

II. Overview of Decoding Algorithms

An LDPC code is a linear and an iterative error correction code operating in binary field. LDPC codes and its decoding algorithms can be represented by a bipartite graph with two disjoint sets of variable node and check node vertices. To each of n variable nodes of a decoder, a so-called Variable Node Unit (VNU), is assigned. Similarly to each of m check nodes, a Check Node Unit (CNU) is assigned. Each edge in the Tanner

978-1-5386-1911-7/17/$31.00 ©2017 IEEE

graph connects one CNU to one VNU. The degrees of VNUs and CNUs (d_v, d_c) are defined based on their numbers of adjacent vertices. The topology of the bipartite graph is represented by a parity check matrix (H matrix). Based on a decision function applied over the received messages from each adjacent vertex, each CNU and VNU sends a message back to its adjacent vertices. This iterative message processing between nodes recover the original data, which may have been exposed to channel noise.

A. Deterministic Gallager B algorithm (GaB)

Let us denote $v2c_n^{(i)}(e)$ the messages sent on edge e from a VNU v_n to a CNU at iteration i, and $c2v_m^{(i)}(e)$ represent the messages sent on edge e from a CNU c_m to a VNU at iteration i. r_n denotes the output of the channel, received word, at a VNU v_n. The VNU and CNU calculation in Gallager B algorithm are represented using Eq. 1 and Eq. 2, respectively.

$$v2c_n^{(i)}(e) = \begin{cases} 1 & r_n + \sum_{e' \in \mathcal{N}(v_n)-e} c2v_m^{(i)}(e') > b_n^{(i)} \\ 0 & r_n + \sum_{e' \in \mathcal{N}(v_n)-e} c2v_m^{(i)}(e') < b_n^{(i)} \\ r_n & otherwise \end{cases} \quad (1)$$

where i is the iteration count and $b^{(i)}$ is the threshold calculated as $b^{(i)} = \lceil d_v/2 \rceil$

$$c2v_m^{(i)}(e) = \sum_{e' \in \mathcal{N}(c_m)-e} v2c_n^{(i)}(e') mod 2 \quad (2)$$

B. Probabilistic Gallager B Algorithm (PGaB)

The interactions between CNUs and VNUs may result in an oscillation phenomena due to the n^{th} order dependencies between CNUs and VNUs. The decoding process may take very long time or get trapped in a cyclic behavior. Decoder remains in cyclic sequences of states. It is not easy to detect all the trapping sets during the decoding process since the trapping set size is unknown and there can be many thousands of different trapping sets. In our earlier work we showed that disturbing the state of each VNU randomly helped the decoder to escape from the trapping set [4]. In the same study we presented a novel approach to modify the GaB and improved its decoding performance by up to four orders of magnitude with negligible hardware overhead in terms of FPGA resource utilization and performance loss in terms of throughput. In this section, we give an overview of our algorithm and hardware implementation approach, and highlight the critical design decisions made to make this architecture resource efficient. We introduced a heuristic which switches the execution from GaB to PGaB if the GaB decoder can not correct errors within predefined number of (k) iterations, so that we stimulate the decoder to escape from the trapping set.

$$v2c_n^{(i)}(e) = \begin{cases} 1 & p_n^{(i)} \oplus r_n + \sum_{e' \in \mathcal{N}(v_n)-e} c2v_m^{(i)}(e') > b^{(i)} \\ 0 & p_n^{(i)} \oplus r_n + \sum_{e' \in \mathcal{N}(v_n)-e} c2v_m^{(i)}(e') < b^{(i)} \\ r_n & otherwise. \end{cases} \quad (3)$$

Eq. 3 represents probabilistic execution formula derived from Eq. 1 by introducing a probability function. The random bits in PGaB are represented by a sequence of N bits, generated following a Bernoulli distribution $\mathcal{B}$ with parameter p_0. The difference in decoding architecture between GaB and PGaB is shown in Figure 1. In the GaB algorithm, a VNU $v_n^{(i)}$ message $v2c_n^{(i)}(e)$ is computed following Eq. 1, while in PGaB, a VNU $v_n^{(i)}$ message $v2c_n^{(i)}(e)$ is computed following Eq. 3, where its value is conditioned by the value of the random bit $p_n^{(i)}$.

Algorithm 1: Probabilistic Gallager B

Initialization $i = 0$, $v_n^{(0)} \leftarrow r_n, n = 1, \ldots, N$.
$s = H\mathbf{v}^{(0)^T}$
while $s \neq \mathbf{0}$ *and* $i \leq i_{max}$ **do**
  Generate $p_n^{(i)}$, $n = 1, \ldots, N$, from $\mathcal{B}(p_0)$.
  for $n = 1, \ldots, N$ **do**
    Compute $v2c_n^{(i)}(e)$, $1 \leq n \leq N$, using Eq. 3
    Compute $c2v_m^{(i)}(e)$, $1 \leq m \leq M$, using Eq. 2
    Compute $v_n^{(i+1)} = Maj(c2v_n^{(i)}(e)_{e \in \mathcal{N}(c_m)}, r_n)$
  end for
  $s = H\mathbf{v}^{(i+1)^T}$
  $i = i + 1$
end while
Output: $\mathbf{v}^{(k)}$

The PGaB algorithm is presented in Algorithm 1. For the implementation of the PGaB hardware architecture, we need to decide the position where the randomness will be applied and determine the best value of p. The decoder in the trapping set can be disturbed in various ways. For example, the random disturbance is applied both messages exchanged mutually between VNU and CNU in Noisy GaB [3]. In an other study [5], the probabilistic function is used to affect the final output decision of a VNU for deciding between flipping the channel value or not. Contrary to these studies, based on the Monte-Carlo simulations we achieved better decoding performance by introducing randomness as an input to the VNU function for computing only messages sent from VNU to CNU [4]. Based on our experimental evaluations on determining the most suitable value of p, we found that the p value of 0.8 gave the best decoding performance results for the studied codes in the range of 0 to 1, where 0 corresponds to deterministic GaB and 1 corresponds to disturbing all the VNUs. Our conclusion is disturbing only 20% of the VNUs was sufficient to escape from the trapping set.

III. Simulation Setup and Hardware Design

Our simulation environment includes the GaB, PGaB, MinSum, and Offset MinSum implementations in C programming language. We evaluate the hardware performance and resource utilization using 1296-bit LDPC codes [6], with rate of 0.5. We design and implement each decoder on the Xilinx Virtex-6 FPGA (vc6vlx240t ff1156 target speed -3) and conduct post placement and routing analysis over hardware cost in terms of logic and register usage, and hardware performance in terms of maximum clock rate, and average throughput. Similar to other

studies ([7], [8]), we calculate the system throughput using Eq. 4.

$$SystemThroughput = \frac{CodeLength \times MaxClockRate}{AvgIteration \times CyclesPerIteration} \quad (4)$$

Figure 1 is an illustration of the generic PGaB architecture with a zoomed in view of a VNU when d_v is set to 4 based on a regular QC-LDPC code for codeword length (N). The colored arrows and the *xor* gate are used by the PGaB implementation. Since the d_v is 4, each VNU receives 6 1-bit inputs, where r_i is the data received from the channel P is the 1-bit random value received from the Linear Feedback Shift Register (LFSR) based random number generator (RNG). Remaining four inputs ($c2v_n^{(i)}(1,2,3,4)$) are the 1-bit messages received from CNUs. Each majority voter unit generates a 2-bit output, where 00 represents the majority of 0s, 11 represents the majority of 1s, and 01 or 10 represent tie cases. As shown in the figure, the first majority voter unit in the VNU architecture has an additional fifth input due to non-extrinsic calculations. For the case of a tie, the *select* unit forwards the received message from the channel to its destination, otherwise the output of the majority decision is forwarded to the destination CNUs.

As shown in our earlier work [4], the FER performance of the PGaB is not sensitive to changes of p_0 values for many cases. Based on this observation, rather than utilizing a complex and high-quality random number generator studied exhaustively in the literature [9], we choose to adopt a light weight random number generator based on LFSR to feed a 1-bit random value per clock cycle to each VNU. During the iterations between the CNUs and VNUs, we generate 1-bit and use a shift register of size N to feed into each of the VNUs.

In order to emphasize the complexity and throughput efficiency of the PGaB, we make comparisons between our PGaB decoder and the MS, OMS benchmarks. The MS, OMS architecture used in this work are the quantized decoders with 4 bits for passed messages and 6 bits for AP-LLR as in [6]. We modify the architecture in [6] to apply on the BSC and the difference comes only from the channel inputs. The BSC channel provides only hard-information (1 bit) while it is a soft-information ($q = 4$ bits in [6]) in Additive White Gaussian Noise (AWGN) channel. To loose the stress in routing process, the MS, OMS decoders are implemented in the layered scheduling where each decoding iteration is divided into L layers. The detailed information on the MS, OMS architecture can be found in [6]. In this work, we choose $L = 3$ for $d_v = 3$ and $L = 4$ for $d_v = 4$ test codes. Note also that, the synthesis process of MS and OMS is configured to use only distributed memory (not the BRAMs resources).

IV. Hardware Performance Analysis

We compare the decoding algorithms based on their hardware implementations for dv3 and dv4 using Table I.

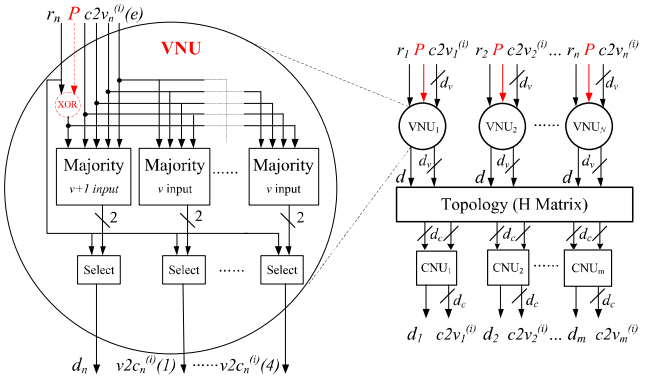

Figure 1. The VNU architecture and the overall decoder architecture for PGaB for $d_v = 4$.

The PGaB hardware implementation includes the additional random number generator (RNG) unit and the1-bit random number input to each VNU compared to the GaB implementation. We observe that these hardware modifications result with only negligible loss in the maximum clock rate and throughput performance with respect to the GaB for both codes (dv3 and dv4). However, the resource requirements in terms of register and slice LUT usage for PGaB are higher compared to the GaB for both codes. The inclusion of 1296 bits to store the 1-bit random number for each VNU and the 32-bit shift register to implement the RNG are the main sources of the demand in resource usage. We believe that the increase in resource usage is a reasonable trade-off for the gain in the decoding performance.

The architectural changes for GaB and PGaB when moving from dv3 to dv4 involve a new majority voter and increase input and output for VNU and CNU by 1 bit. Therefore 1-bit registers and LUT usage increase by 17% and 58% respectively for the GaB. We observe a similar trend for the PGaB with 17% and 42% increase in 1-bit registers and LUTs respectively. The clock rate reduces around 11% for both GaB and PGaB compared to dv3 implementations and the throughput reduces proportionally for both. We attribute this reduction primarily to 30% increase ($3 * N$ to $4 * N$) in number of connections between the CNUs and VNUs, which creates a stress on routability resulting with an increase critical path delay for the overall architecture. We further optimize the average number of iterations for PGaB by applying the randomness only after a subsequent iterations, *i.e.* decoding by GaB algorithm for $1 \leq i \leq 15$ and PGaB algorithm with $15 < i \leq i_{max}$. This modification comes from the fact that some error patterns can be easily corrected by GaB in few iterations. This way we are able to reduce the average number of iterations significantly.

The PGaB is superior in terms of decoder complexity and throughput performance when compared to the MS and OMS decoders. Indeed, the MS and OMS require more than 440% the number of registers compared to GaB while that of the PGaB is only $17\% - 20\%$. A negligible increase in LUTs compared to GaB is observed when implementing the PGaB ($23.9\% - 38.6\%$) while this increase of MS and OMS are very critical. Compared to

Table I. RESOURCE USAGE, CLOCK RATE AND THROUGHPUT COMPARISON BASED ON FPGA IMPLEMENTATIONS OF GAB, PGAB, MS, AND OMS FOR DV3 AND DV4. (* $L = 3$ FOR DV3 AND $L = 4$ FOR DV4)

	1-bit register		Slice LUTs		Fmax (MHz)		Average Throughput (Mbps) (2 clock cycles /layer for MS and OMS)	
Design	dv3	dv4	dv3	dv4	dv3	dv4	dv3	dv4
GaB	6516 (0%)	7812 (0%)	7443 (0%)	11784 (0%)	164.9	147.5	42765 (0%)	38224 (0%)
PGaB	**7845 (+20.4%)**	**9141 (+17%)**	**10320 (+38.6%)**	**14605 (+23.9%)**	**164.4**	**146**	**42610 (-0.4%)**	**37900 (-0.8%)**
MS*	37610 (+477.2%)	42794 (+447.8%)	80721 (+984.5%)	84165 (+614.2%)	55.5	55.5	600 (-98.6%)	450 (-98.8%)
OMS*	37610 (+477.2%)	42794 (+447.8%)	86445 (+1061.4%)	95721 (+712.3%)	54.9	53.6	593 (-98.6%)	434 (-98.8%)

GaB, the $d_v = 3$ MS and OMS decoders require 984.5% and 1061.4% respectively. For the $d_v = 4$ MS and OMS decoders, these increase are smaller with 614.2% of MS and 712.3% of OMS. In term of decoding throughput, PGaB provides also a significant gain over MS and OMS decoders. It offers 42610 Mbps compared to 600 Mbps of MS and 593 Mbps of OMS decoders on the $d_v = 3$ code. For $d_v = 4$ code, PGaB can operate at 37900 Mbps which is 84.2 times faster than MS (450 Mbps) and 87.3 times faster than OMS (434 Mbps).

V. LDPC DECODING PERFORMANCE ANALYSIS

Figure 2 shows the FER performance of Bit Flipping (BF), GaB, PGaB, MS and OMS based on regular LDPC code with rate of 0.5. In general, PGaB decoder is performing much better than BF and deterministic GaB decoder and approaching to soft decision decoder (MS and OMS). We show that the PGaB results with up to four orders of magnitude improvement in decoding performance compared to the GaB based on the gap between PGaB and GaB in the error floor region (at $\alpha = 0.01$). We achieve this dramatic performance increase with the utilization of hardware friendly and simple random number generator without a loss in throughput performance.

VI. CONCLUSION

In this study we extend our performance evaluation of the PGaB to comparison with respect to the OMS and MS implementations. The simulations and FPGA implementation confirm that PGaB decoder offers very low-complexity, high throughput and good performance and is a promising candidate for current and future communication standards.

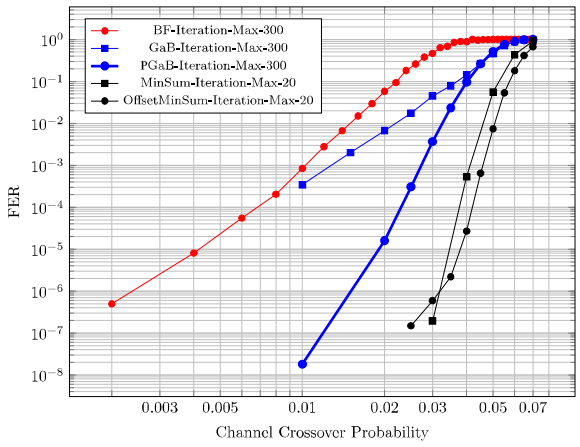

Figure 2. Performance comparison between BF, GaB, PGaB ($p = 0.8$), MS and OMS with offset factor of 1 on the $N = 1296, d_v = 4, d_c = 8$, and code rate = 0.5

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation under Grant ECCS-1500170 and CCF-1314147, the Indo-US Science and Technology Forum (IUSSTF) through the Joint Networked Center for Data Storage Research (JC-16-2014-US), and the French ANR project NAND under grant agreement ANR-15CE25-0006-01.

REFERENCES

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

[2] D. Declercq, M. Fossorier, and E. E Biglieri, *Channel Coding: Theory, Algorithms, And Applications*. Academic Press Library in Mobile and Wireless Communications, Elsevier, 2014.

[3] B. Vasic and P. Ivanis, "Error errore eicitur: A stochastic resonance paradigm for reliable storage of information on unreliable media," *IEEE Trans on Comm*, vol. 64, no. 9, pp. 3596 – 3608, 2016.

[4] B. Ünal, F. Ghaffari, A. Akoglu, D. Declercq, and B. Vasić, "Analysis and implementation of resource efficient probabilistic gallager b ldpc decoder," *2017 15th IEEE International New Circuits and Systems Conference (NEWCAS)*, pp. 333–336, June 2017.

[5] K. Le, F. Ghaffari, D. Declercq, and B. Vasic, "Efficient hardware implementation of probabilistic gradient descent bit-flipping," *IEEE Trans on Circuits and Systems I: Regular Papers*, vol. PP, no. 99, pp. 1–12, 2016.

[6] T. Nguyen-Ly, K. Le, F. Ghaffari, A. Amaricai, O. Boncalo, V. Savin, and D. Declercq, "Fpga design of high throughput ldpc decoder based on imprecise offset min-sum decoding," in *IEEE 13th Int New Circuits and Systems (NEWCAS)*, June 2015, pp. 1–4.

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

[8] K. Le, D. Declercq, F. Ghaffari, C. Spagnol, P. Ivanis, and B. Vasic, "Efficient realization of probabilistic gradient descent bit flipping decoders," in *Proc. of the 21st 2015 IEEE Int Symp on Circuits and Systems (ISCAS)*, May 2015, pp. 1–4.

[9] N. Dek, T. Gyrfi, K. Mrton, L. Vacariu, and O. Cret, "Highly efficient true random number generator in fpga devices using phase-locked loops," in *2015 20th International Conference on Control Systems and Computer Science*, May 2015, pp. 453–458.