A New Probabilistic Gradient Descent Bit Flipping Decoder for LDPC Codes

Hangxuan Cui, Jun Lin, Suwen Song, and Zhongfeng Wang
School of Electronic Science and Engineering, Nanjing University, P. R. China
Email: haxcui@163.com, jlin@nju.edu.cn, suwsong@sina.com, zfwang@nju.edu.cn

***Abstract*—Probabilistic gradient descent bit-flipping (PGDBF) is the state-of-the-art hard-decision algorithm for decoding low-density parity-check (LDPC) codes on binary symmetric channel (BSC). However, there still exists a considerable performance gap between the PGDBF algorithm and soft-decision algorithms, especially in the error-floor region. To bridge this performance gap, a tabu-list aided PGDBF (T-PGDBF) algorithm is proposed in this paper. In the T-PGDBF algorithm, a tabu-list is employed to help the decoding escape from trapping sets, which is the main cause of the error-floor phenomenon. The bits which are flipped in the current iteration will be added to the tabu-list to prevent them being flipped in the next iteration. Simulation results show that the T-PGDBF algorithm offers a significant performance gain when compared to the PGDBF algorithm, which can reach that of soft-decision algorithms. We also present the hardware architecture to implement the T-PGDBF algorithm. Synthesis results show that the improved performance offered by the T-PGDBF algorithm can be obtained with a small hardware overhead.**

***Index Terms*—Probabilistic gradient descent bit-flipping, low-density parity-check codes, tabu-list, trapping sets, high-throughput decoder.**

I. Introduction

Due to the remarkable coding performance and efficiency, low-density parity-check (LDPC) codes [1] have received increasing attentions in the past several decades. Among the various decoding algorithms, the soft-decision algorithms, such as the Belief-Propagation (BP) [1] and Min-Sum (MS) algorithms [2], offer the optimal performance. However, the high computation complexity limits the throughput of the soft-decision decoders. On the contrary, the hard-decision algorithms, such as the Bit-Flipping (BF) algorithm [3] and the Gallager-B algorithm [4], have very low computation complexity but suffer from the poor decoding performance. In order to obtain a high-performance and high-throughput decoder, many researchers devote to improving the performance of hard-decision algorithms while keeping the low-complexity property of them.

The BF algorithm has been investigated extensively due to its simple computation units. In the BF algorithm, the decoding is governed by the inversion function, which indicates the reliability value of each bit. In each iteration, the bit will be flipped if its reliability value falls below a specific threshold. Gradient Descent BF (GDBF) algorithm [5] shows a performance superior to most known BF variants. In the GDBF algorithm, the decoding process is considered as the maximization process of a non-linear objective function. The gradient descent method is applied to solve this maximization problem and the inversion function is obtained according to the steepest descent algorithm based on ℓ_1-norm [6]. Recently, the authors of [7] introduced the probabilistic GDBF (PGDBF) algorithm which is used for Binary Symmetric Channel (BSC). Compared to the GDBF algorithm, only a randomly selected subset of bits which have the minimum reliability value is flipped in the PGDBF algorithm. This modification makes the PGDBF algorithm offer a significant performance improvement compared to other hard-decision algorithms. However, there still exists a considerable performance gap between it and soft-decision algorithms.

In this work, we propose an algorithm named tabu-list aided PGDBF (T-PGDBF) algorithm. A tabu-list is designed by observing the structure of trapping sets. The bits which are flipped in the current iteration will be added to the tabu-list in order to avoid the repeating flip problem. Simulation results show that the T-PGDBF algorithm offers a performance which in some cases is even better than that of the MS algorithm. Besides, we provide an architecture to implement the T-PGDBF algorithm. Compared to the PGDBF decoder [8], only about 10% additional hardware resource is needed. In the case of offering similar performances, the throughput of the T-PGDBF decoder is 3.1 times higher than that of the corresponding MS decoder [9].

The rest of this paper is organized as follows. Section II introduces the notations and the PGDBF algorithm. In Section III, the T-PGDBF algorithm is described in detail. Section IV presents the simulation results including performance comparisons. The hardware architecture is presented in Section V. Finally, Section VI concludes the paper.

II. Preliminaries

A. Notations

Consider a binary linear LDPC code $\mathcal{C}$ defined by an $M \times N$ parity check matrix $\boldsymbol{H}$. Each row of $\boldsymbol{H}$ corresponds to a parity check equation which is represented by a Check Node (CN). Each column represents a codeword symbol which is denoted by a Variable Node (VN). CNs and VNs are indexed by $\mathcal{J} = \{0, 1, \cdots, M-1\}$ and $\mathcal{I} = \{0, 1, \cdots, N-1\}$, respectively. The code rate $R = 1 - M/N$. A length-N binary vector $\boldsymbol{u}$ is a codeword if all parity check equations are satisfied, i.e., $\boldsymbol{u}\boldsymbol{H}^T = \boldsymbol{0}$. $\mathcal{N}_c(j)$ is the set of VNs connected to the j-th CN, i.e., $\mathcal{N}_c(j) = \{\ i \mid h_{j,i} = 1\}$. Similarly, $\mathcal{N}_v(i)$ denotes the neighborhoods of the i-th VN. The degree

978-1-7281-0397-6/19/$31.00 ©2019 IEEE

of the i-th VN $d_v^i = |\mathcal{N}_v(i)|$ and the degree of the j-th CN $d_c^j = |\mathcal{N}_c(j)|$. For the regular LDPC code, $d_c^j = d_c$ and $d_v^i = d_v$ for arbitrary $i \in \mathcal{I}$, $j \in \mathcal{J}$. This work focuses on a regular LDPC code transmitted over BSC, in which the bits are flipped with crossover probability α.

B. The PGDBF Decoding Algorithm

The PGDBF algorithm is derived from the GDBF algorithm, which is proposed for Additive White Gaussian Noise (AWGN) channel. In the GDBF algorithm, the inversion function is defined as:

$$\triangle_k(\boldsymbol{c}) = (2c_k - 1)y_k + \sum_{j \in \mathcal{N}_v(k)} (2s_j - 1). \tag{1}$$

$\boldsymbol{c}$, $\boldsymbol{y}$, and $\boldsymbol{s}$ are the estimated codeword, the received vector from AWGN channel, and the syndrome vector, respectively. $s_j = \oplus_{i \in \mathcal{N}_c(j)} c_i$ and $\oplus$ represents the bit-wise exclusive-OR (XOR) operation. The value of the inversion function is called energy value. In each iteration, the bit with the maximum energy value is flipped.

The modified inversion function for BSC is shown in Eq. (2), where $\boldsymbol{r}$ is the received codeword from BSC channel.

$$\Lambda_k(\boldsymbol{x}) = c_k \oplus r_k + \sum_{j \in \mathcal{N}_v(k)} s_j. \tag{2}$$

The range of values is 0 to $d_v + 1$ for the inversion function. Considering that only $d_v + 2$ values are assigned to thousands of bits and d_v is small in most cases, the number of bits with maximum value in each iteration may be very large. Therefore, in the PGDBF algorithm, a random perturbation is introduced to the decoding. Only a randomly selected subset of the bits which have the maximum energy value is flipped in each iteration. The random fraction is fixed to a pre-defined value p_0 and $0 < p_0 < 1$.

III. The Tabu-List Aided PGDBF Algorithm

Trapping sets are the main cause of the error-floor phenomenon [10], [11]. Fig. 1 shows two states of a trapping set. Each VN is marked with a circle and each CN is marked with a square. The values of the black ones are 1 and the values of the white ones are 0. The values of the five VNs received from BSC channel are 1, 0, 1, 0, and 1, respectively. According to Eq. (2), the energy values of the VNs in Fig. 1(a) are 2, 2, 2, 2, and 1, respectively. Regardless of the random perturbation, the first four VNs are flipped. After a flip, Fig. 1(b) is obtained where the energy values of the VNs are 4, 4, 4, 4, and 3, respectively. Hence, the first four VNs should be flipped again.

Based on the above observation, we find that the decoding will be locked in this trapping set without taking into account the random perturbation. After adding the random perturbation to the decoding, the regularity of the decoding is broken so that the probability of the decoding escaping from the trapping set is increased. This is the reason that the PGDBF algorithm is superior to the GDBF algorithm in performance. However, the PGDBF algorithm is not effective enough. To illustrate this problem, we monitor the decoding process of the PGDBF algorithm and record the locations of flipped bits in each iteration, which are shown in Fig. 2. A $d_v = 4$, $d_c = 8$, $R = 0.5$, $N = 1296$ (dv4R050N1296) code is used and the maximum number of iterations T_{max} is set to 300. Only the results in the first 100 iterations are shown for clarity.

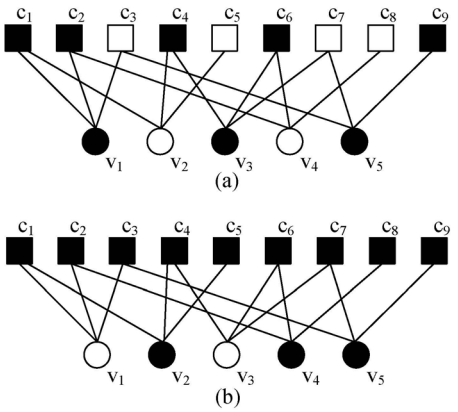

Fig. 1. Two states of a trapping set.

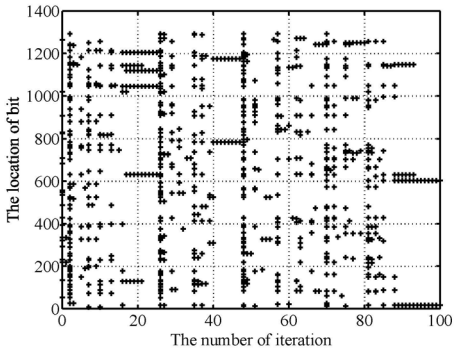

Fig. 2. The locations of flipped bits in each iteration.

As can be seen, some bits are flipped repeatedly in several iterations. This means that the PGDBF algorithm needs several iterations to help the decoding escape from a trapping set. To improve the effectiveness of the decoding, a tabu-list is defined and the bits which are flipped in the current iteration will be added to this list. Only the bits which are not in the tabu-list and have the maximum energy value are flipped with probability p_0. The T-PGDBF algorithm is described in Alg. 1, where $\boldsymbol{c}^{(t)}$ denotes the estimated codeword in the t-th iteration and $t \in [0, T_{max}]$. $\boldsymbol{R}$ is the random sequence and $\Pr(R_i = 1) = p_0$. l_i indicates whether the i-th bit is in the tabu-list or not.

IV. Simulation Results

The decoding performance of the T-PGDBF algorithm is illustrated in Fig. 3 and compared to the GDBF, PGDBF, and MS algorithms. The MS algorithm is a layered version with double precision floating-point arithmetic. Moreover, an offset factor is used to improve the performance of it [12]. All numerical results are obtained based on Monte Carlo methods.

Algorithm 1: Tabu-list aided PGDBF Algorithm

input : $\boldsymbol{r} = (r_0, r_1, \cdots, r_{N-1})$
initialize: $\boldsymbol{c}^{(0)} = \boldsymbol{r}$, $t = 0$, $(l_0, l_1, \cdots, l_{N-1}) = \mathbf{0}$
for $t = 0$ **to** $T_{max} - 1$ **do**
  for $j = 0$ **to** $M - 1$ **do**
    $s_j = \oplus_{i \in \mathcal{N}_c(j)} c_i^{(t)}$
  if $\boldsymbol{s} = \mathbf{0}$ **then**
    break
  $\Lambda_{max} = 0$
  for $i = 0$ **to** $N - 1$ **do**
    $\Lambda_i(\boldsymbol{c}) = c_i \oplus r_i + \sum_{j \in \mathcal{N}_v(i)} s_j$
    if $\Lambda_{max} < \Lambda_i$ *and* $l_i = 0$ **then**
      $\Lambda_{max} = \Lambda_i$
  Generate the binary random sequence $\boldsymbol{R}$
  for $i = 0$ **to** $N - 1$ **do**
    if $\Lambda_i = \Lambda_{max}$, $R_i = 1$, *and* $l_i = 0$ **then**
      $c_i^{(t+1)} = 1 - c_i^{(t)}$
      $l_i = 1$
    else $l_i = 0$
output : $\boldsymbol{c}^{(t)}$

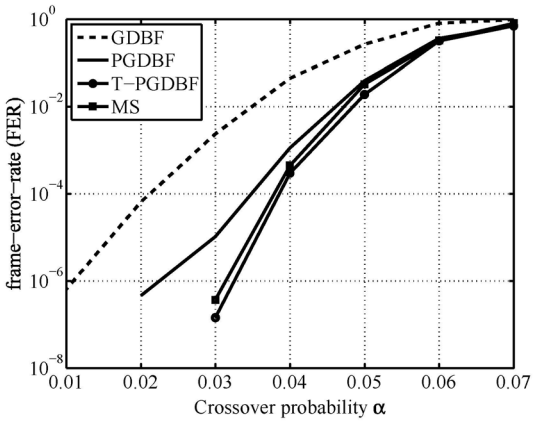

Fig. 3. The simulation results for the dv4R050N1296 LDPC code.

The dv4R050N1296 LDPC code is considered to evaluate the performance of the T-PGDBF algorithm without loss of generality. T_{max} is set to 20 for the MS algorithm, otherwise $T_{max} = 300$. In our simulations, the parameter p_0 is set to 0.9 for the PGDBF and T-PGDBF algorithms, which is proved to be the optimal value [8].

TABLE I
THE AVERAGE NUMBER OF ITERATIONS PER FRAME

Crossover Probability α	0.07	0.06	0.05	0.04	0.03
PGDBF	240.6	155.3	46.4	16.1	8.5
TPGDBF	246.7	161.7	47.0	18.8	10.3

As shown in Fig. 3, the performance of the T-PGDBF algorithm is far better than that of the PGDBF algorithm. It should be noted that the error-floor phenomenon appearing in the PGDBF algorithm has not been observed yet in the T-PGDBF algorithm. Table I shows the average numbers of iterations per frame of the PGDBF and T-PGDBF algorithms in each crossover probability point. Since adding the tabu-list to the decoding reduces the average convergence speed, the T-PGDBF algorithm needs slightly more iterations than the PGDBF algorithm. Fig. 3 also shows that the performance of the T-PGDBF algorithm is slightly better than that of the MS algorithm, which is a commonly used soft-decision algorithm. Therefore, the T-PGDBF algorithm bridges the performance gap between hard-decision and soft-decision algorithms. Due to the low-complexity and high-performance property, it is a very competitive algorithm for decoding LDPC codes.

V. HARDWARE ARCHITECTURE

A. Top-Level Architecture

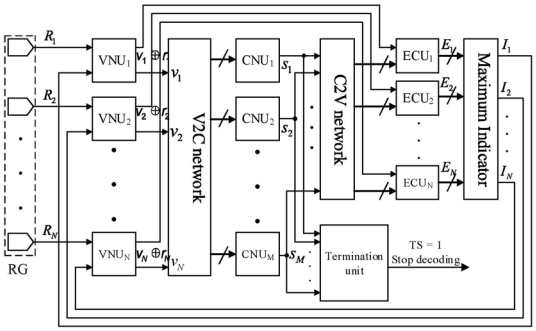

Fig. 4. The top-level architecture of the T-PGDBF decoder.

The top-level architecture of the T-PGDBF decoder is shown in Fig. 4, which is similar to that of the PGDBF decoder [13] due to the similar decoding process. At the initialization of the decoding, the value of the received codeword $\boldsymbol{r}$ is assigned to the estimated codeword $\boldsymbol{v}$ in VN units (VNUs). Then, M CN units (CNUs) are used to calculate the syndrome vector $\boldsymbol{s}$. Next, in order to minimize the latency, the Leading-zero Counting Topology (LCT) [14] is employed to find the bits which have the maximum energy value. The energy values are calculated in N Energy Calculation Units (ECUs) and the maximum indicator produces the indicator value I_i of each bit. For the bits which have the maximum energy value, $I_i = 1$, otherwise $I_i = 0$. The random sequence is pre-generated and saved to avoid using the complex random generator. In order to cover the performance loss caused by the fixed $\boldsymbol{R}$, the random sequence is cyclically shifted once in each iteration. Finally, the bits which satisfy the conditions described in Alg. 1 will be flipped in VNUs. In each iteration, the termination unit will perform an OR operation among the whole syndrome vector to verify whether the current estimated codeword is valid or not. By employing the tree structure, the latency of this unit is reduced from $M - 1$ to $\lceil \log_2 M \rceil$. When the valid codeword

is obtained or the maximum number of iterations is reached, $TS = 1$ and the decoder is stopped.

B. Analysis of Additional Hardware Resources

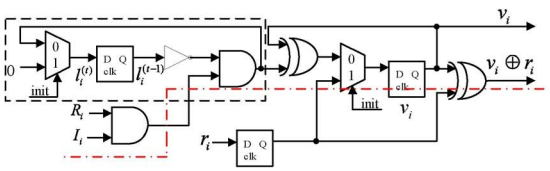

Fig. 5. The architecture of the VNU.

The additional hardware resources used to implement the tabu-list are described in this sub-section. Fig. 5 shows the architecture of the i-th VNU, in which the signal *init* is only triggered once to initialize l_i and v_i. Since the bits which are in the tabu-list are not allowed to flip in the current iteration, the value of vector $\boldsymbol{l}^{(t-1)}$ should be saved to identify them. For the bits which are in the tabu-list, $l_i^{(t-1)} = 1$ and thus the second input of the XOR gate is 0. Hence, the values of these bits can not be changed. Compared to the VNU presented in [13], our design contains the circuit in the dashed box in addition. The red line in Fig. 5 is part of the critical path of the decoder. As can be seen, only the delay of an AND gate is added in extra.

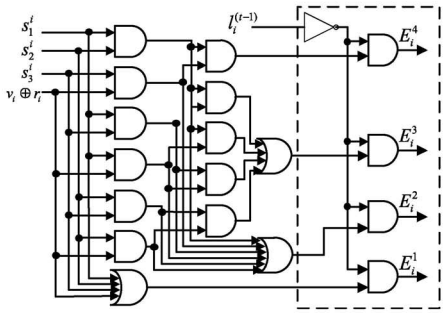

Fig. 6. The architecture of the ECU.

Fig. 6 shows the ECU which is designed for the code with $d_v = 3$. s_1^i, s_2^i, and s_3^i are the syndromes connected to the i-th VN. The energy value is expressed in one-hot format, i.e., E_i is represented by $d_v + 1$ bits $E_i^{d_v+1}...E_i^2E_i^1$ and $E_i^k = 1$ if and only if $E_i = k$. In the T-PGDBF algorithm, the bits in the tabu-list should not participate in the process of finding the maximum energy value. Therefore, the energy values of these bits are set to zero by using the vector $\boldsymbol{l}^{(t-1)}$. As shown in Fig. 6, the delay of an AND gate and the circuit in the dashed box are added to complete this operation.

C. Synthesis Results

The hardware architecture is described by Verilog language and synthesized under the TSMC 90nm CMOS technology using the Cadence RTL compiler. Synthesis and comparison results are shown in Table II, where a $d_v = 3$, $d_c = 6$, $R = 0.5$, $N = 1296$ (dv3R050N1296) code is used. When compared to the PGDBF decoder, the T-PGDBF decoder requires 10.6% extra area and 8.7% extra power. Moreover, the T-PGDBF decoder operates at a lower frequency, which is 90.5% of that of the PGDBF decoder. However, since the T-PGDBF decoder has a far better decoding performance than that of the PGDBF decoder, this trade-off is still very worthy.

TABLE II
THE SYNTHESIS RESULTS FOR THE DV3R050N1296 CODE

	T-PGDBF	PGDBF [8]	MS [9]
Technology	90nm	90nm	65nm
Total Area(mm^2)	0.406	0.367	1.38†
Power(mW)	56.3	51.8	-
Frequency(MHz)	323	357	130.4†
Average Iteration Number	6.10	5.15	1.29
Cycles per Iteration	1	1	6
Throughput (Gbps)	68.6	89.8	21.8†
Area Efficiency (Gbps/mm^2)	169.0	244.7	15.8†

†These are the scaled results under the TSMC 90nm CMOS technology.

To make a complete comparison, the synthesis results of the MS decoder are also shown in Table II. As a soft-decision decoder which has a similar performance to the T-PGDBF decoder, the MS decoder requires 3.4 times larger area and offers a significantly lower frequency. When simulated at the same noise level, $\alpha = 0.01$, the throughput of the T-PGDBF decoder is 3.1 times larger than that of the MS decoder. Due to the less area and higher throughput, the area efficiency of the T-PGDBF is about a dozen times higher than that of the MS decoder. Therefore, the T-PGDBF decoder is very competitive when compared to the soft-decision decoders.

VI. CONCLUSION

In this work, we introduce an algorithm named Tabu-list aided PGDBF algorithm. In order to improve the effectiveness of the LDPC decoding, a tabu-list is designed by observing the structure of trapping sets. The bits which are flipped in the current iteration will be added to the tabu-list to prevent them being flipped repeatedly. Simulation results show that the performance of the proposed T-PGDBF algorithm is much better than the state-of-the-art hard-decision algorithms and comparable to the MS algorithm in BSC. We also implement the T-PGDBF algorithm and present the synthesis results. Compared to the PGDBF decoder, only slightly increased hardware resources are needed. Moreover, the T-PGDBF decoder offers 3.1 times higher throughput than the MS decoder. Therefore, the proposed T-PGDBF decoder has great potential in practical applications due to its high-performance and high-throughput property.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 61604068 and in part by the Fundamental Research Funds for the Central Universities under Grant 021014380065. (Corresponding authors: Jun Lin; Zhongfeng Wang.)

REFERENCES

[1] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

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

[3] V. V. Zyablov and M. S. Pinsker, "Estimation of the error-correction complexity for gallager low-density codes," *Problem Inf. Transmiss.*, vol. 11, no. 1, pp. 18–28, Jul. 1975.

[4] B. Unal, A. Akouglu, 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*, vol. 65, no. 9, pp. 3074–3084, Sept. 2018.

[5] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, "Gradient descent bit flipping algorithms for decoding LDPC codes," *IEEE Trans. Commun.*, vol. 58, no. 6, pp. 1610–1614, Jun. 2010.

[6] P. O. Vontobel and R. Koetter, "Towards low-complexity linear programming decoding," in *proc. 4th Int. Symp. Turbo Codes Related Topics*, Munich, Germany, Apr. 2006.

[7] 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.

[8] 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.

[9] J. Sha, Z. Wang, M. Gao, and L. Li, "Multi-Gb/s LDPC code design and implementation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 2, pp. 262–268, Feb. 2009.

[10] T. Richardson, "Error floors of LDPC codes," in *Proc. Annual Allerton Conference on Communication, Control, and Computing*, Oct. 2003, pp. 1426–1435.

[11] H. Cui, J. Lin, and Z. Wang, "An efficient post-processor for lowering the error floor of LDPC codes," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, 2018, [Online: DOI: 10.1109/TCSII.2018.2849504].

[12] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[13] 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. 2016.

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