Optimization of Protograph LDPC Codes based on High-Level Energy Models

Mohamed Yaoumi[1], François Leduc-Primeau[2], Elsa Dupraz[1], Frederic Guilloud[1]
[1] IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238, France.
[2] Department of Electrical Engineering, Polytechnique Montreal, QC, Canada.

***Abstract*—This paper considers the optimization of the energy consumption of LDPC decoders. For a given protograph, two models are introduced to approximate the energy consumption of a quantized Min-Sum decoder. The first model takes into account the number of operations performed by the decoder, the second model considers the number of bits that are written into memory. An optimization problem is then formulated to minimize the decoder energy consumption with respect to the protograph and the number of iterations, while satisfying a given performance criterion. Finally, an optimization method based on Differential Evolution is proposed.**

I. Introduction

Low-density parity-check (LDPC) [1], [2] are known to be capacity-approaching error-correction codes, and they were retained in the 5G standardization process. In addition to the decoding performance and the transmission power, the decoding energy consumption can be considered as a design criterion which was taken into account only recently [3]. In this case, the objective is to find the best compromise between decoding performance and energy consumption.

In [4], two decoding energy consumption models are introduced. The first model considers the energy consumed in each variable and check node for message computation. The second model evaluates the energy consumed by wires in the decoder. In both models, energy consumption depends on the code length and on the number of edges connected to each variable and check node. However, in [4], the energy required by access to memory is not taken into account. Recently, [5] introduced an energy minimization method for Finite Alphabet Iterative Decoders (FAIDs) of LDPC Codes. FAIDs process discrete messages and the goal of [5] is to minimize the size of message alphabets in order to save energy while maintaining a good level of performance. The approach of [5] reduces both memory and wire energy consumption.

While previous works [4] and [5] optimize the decoder, the code itself can play an important role in energy consumption. Therefore, [6], [7] seek to minimize the decoding complexity for a target decoding performance by a numerical optimization of the code rate and irregular degree distributions. Both works [6], [7] assume an infinite codeword length, and [6] considers the Gallager B decoder while [7] studies the Sum-Product decoder.

Alternatively, in this paper, we consider quantized Min-Sum decoders for their easy hardware implementation [8]. While previous works [4]–[7] study LDPC codes constructed from regular and irregular degree distributions, we consider codes constructed from protographs [9], as they allow for the design of hardware-friendly quasy-cyclic LDPC codes with good performance [10]. Our objective is to design protographs that minimize the decoder energy consumption for a target decoding performance. Since not only the decoder energy consumption but also its performance depend on the codeword length, our method relies on the approach of [11] to evaluate the finite-length decoder performance.

We introduce two models that estimate the energy consumption of a quantized Min-Sum decoder. The first model uses the average number of operations required for decoding a codeword as a proxy for energy consumption. The second model considers the total number of bits that must be written in memory during the decoding process. We then formulate an optimization problem that corresponds to minimizing the decoder energy consumption while satisfying a given performance criterion. In this formulation, the energy consumption is minimized with respect to the code and decoder parameters (protograph, number of iterations, etc.) that participate to the energy models. We then propose an optimization method to solve this problem, using the Differential Evolution [12] genetic algorithm.

The rest of the article is organized as follows. In Section II, we review LDPC codes construction and decoding. In Section III, we present the finite-length density evolution method. In Section IV, we describe the two energy estimation models. In Section V, we present the optimization problem. Simulation results are shown in Section VI.

II. LDPC Codes

We assume that each codeword bit is transmitted over an additive white Gaussian noise (AWGN) channel using binary phase-shift keying (BPSK) modulation. The i-th received value y_i is thus given by $y_i = x_i + b_i$ where b_i are independent centered Gaussian random variables with variance σ^2 and where $x_i \in \{-1, 1\}$ is the $i-$th modulated coded bit. The channel signal-to-noise ratio (SNR) is given by $\xi = 1/\sigma^2$.

A. LDPC Code Construction

We consider an LDPC code construction from a protograph [13]. A protograph is specified by a matrix $\boldsymbol{S}$ of size $m \times n$ whose elements indicate the number of edges connecting the respective variable and check nodes of the Tanner graph [14] associated with $\boldsymbol{S}$. Let d_{v_i} be the total number of edges connected to a variable node of type $i \in \{1, \cdots, n\}$ (variable node degree) and d_{c_j} the total number of edges connected to a check node of type $j \in \{1, \cdots, m\}$ (check node degree).

A length-N LDPC code of rate R can be constructed from a protograph by applying a "copy-and-permute" operation on the protograph. The protograph is copied Z times, where $Z = N/n$ is called the lifting factor. The parity check matrix $\boldsymbol{H}$ (which will be assumed full-rank hereafter) is then obtained by interleaving the edges. The degree distribution of the LDPC code is the one of the protograph, provided by the entries in $\boldsymbol{S}$.

Hereafter, a two-steps lifting procedure described in [10] will be applied to come up with quasy-cyclic LDPC codes where the amount of short cycles is minimized. The quasi-cyclic nature of LDPC codes will allow to design hardware-friendly decoder implementations.

B. Quantized Min-Sum Decoder

In the quantized Min-Sum decoder [15]–[17], we use messages between $-Q$ and Q, with a quantization step-size s. The quantization function is given by

$$\Delta(x) = \operatorname{sgn}(x) \min\left(Q, s\left\lfloor \frac{|x|}{s} + \frac{1}{2}\right\rfloor\right), \quad (1)$$

where $\operatorname{sgn}(x) = 1$ if $x \geq 0$ and $\operatorname{sgn}(x) = -1$ if $x < 0$. For implementation efficiency, we usually choose $Q = 2^{q-1} - 1$, where q is the number of bits used to represent messages. In the log-likelihood ratio (LLR) domain, the received value y becomes

$$r_i = \alpha \log\left(\frac{\mathbb{P}(x_i = 1|y_i)}{\mathbb{P}(x_i = -1|y_i)}\right) = \frac{2\alpha y_i}{\sigma^2}, \quad (2)$$

where α is a scaling parameter to be optimized later on.

In the Min-Sum decoder, the message sent from variable v_i to check c_j is denoted β_i^ℓ at iteration ℓ and is calculated by:

$$\beta_i^{(\ell)} = \Delta(r_i) + \sum_{j \in \mathcal{N}_{v_i}} \gamma_{j \to i}^{(\ell-1)}, \quad (3)$$

where $\mathcal{N}_{v_i}$ is the set of all check nodes connected to variable node v_i, and $\gamma_{j \to i}^{(\ell)}$ is the message sent back from the check c_j to the variable v_i at iteration ℓ. Note that unlike commonly described, the messages sent by a variable to its neighboring check nodes are equal. The contribution of each check message will be taken into account in the check node update, as described in the following. Messages $\gamma_{j \to i}^{(\ell)}$ are calculated according to:

$$\beta_{i \to j}^{(\ell)} = \beta_i^{(\ell)} - \gamma_{j \to i}^{(\ell-1)} \quad (4)$$

$$\gamma_{j \to i}^{(\ell)} = \left(\prod_{i' \in \mathcal{N}_{c_j} \setminus \{i\}} \operatorname{sgn}\left(\beta_{i' \to j}^{(\ell)}\right)\right) \times \max\left[\min_{j' \in \mathcal{N}_{c_j} \setminus \{i\}} \left|\beta_{i' \to j}^{(\ell)}\right| - \lambda, 0\right], \quad (5)$$

where λ is an offset parameter to be optimized later on, $\mathcal{N}_{c_j \setminus \{i\}}$ is the set of all the variable nodes connected to check node c_j except v_i.

C. Serial and Parallel Scheduling

Message passing decoders can be implemented with different types of scheduling [18]. In flooding or parallel scheduling all the variable nodes (or check nodes) update their edges simultaneously. Alternatively, in *serial-C* scheduling, the check nodes are updated in a serial manner. After each check node update, all the variable nodes connected to it are updated. Serial-C scheduling enables to reduce the size of the circuit and requires fewer decoding iterations [18]. In this article, the decoder follows the serial-C schedule.

III. Finite-Length Performance

Density evolution [19], [20] is a standard tool to evaluate the performance of an LDPC decoder under asymptotic conditions. For a given SNR, density evolution provides the decoder error probability p_{e_∞} averaged over the ensemble of codes described by protograph $\boldsymbol{S}$.

With density evolution, the error probability p_{e_∞} is evaluated assuming an infinite codeword length. In order to predict the finite-length performance of the quantized Min-Sum decoder described in Section II-B, we rely on the method proposed in [11]. In this method, in order to evaluate the decoder error probability $p_{e_N}(\xi)$ at SNR ξ for a codeword length N, we use the following equation:

$$p_{e_N}(\xi) = \int_0^{\frac{1}{2}} p_{e_\infty}(x)\, \phi_{\mathcal{N}}\left(x; p_0, \frac{p_0(1-p_0)}{N}\right) dx \quad (6)$$

In this expression, $p_0 = \frac{1}{2} - \frac{1}{2}\operatorname{erf}\left(\sqrt{\xi/2}\right)$, and $p_{e_\infty}(x)$ is the error probability evaluated with standard density evolution at SNR value $2(\operatorname{erf}^{-1}(1-2x))^2$. The function $\phi_{\mathcal{N}}(x; \mu, \sigma^2)$ is the probability density function of a Gaussian random variable with mean μ and variance σ^2.

This method takes into account the variability in the channel at finite-length. It does not take into account the effect of cycles in the code parity check matrix, which can also affect the finite-length performance of the decoder. This method is therefore well suited for codes from moderate to long length N.

The same method [11] can be used to estimate the average number of iterations for the decoder to achieve a target error probability p_e for a given SNR ξ. At length

N, the number of iterations $L_N(\xi, p_e)$ for a decoder to achieve p_e can be evaluated by:

$$L_N(\xi, p_e) = \int_0^{\frac{1}{2}} L_\infty(x, p_e)\, \phi_N\left(x;\, p_0, \frac{p_0(1-p_0)}{N}\right) dx \quad (7)$$

where $L_\infty(x, p_e)$ is the number of required iterations estimated by standard density evolution to achieve the target error probability p_e and p_0 is as defined above. If the decoder cannot reach the error probability p_e for the considered SNR, $L_N(\xi, p_e)$ is set to $+\infty$ by convention.

IV. Energy Models

We now introduce two energy models for the quantized Min-Sum decoder. The complexity energy model evaluates the total number of decoding operations. The memory energy model considers the total number of bits written into memory during the decoding. Before providing the energy models, we first analyze the memory use and number of operations performed by the decoder.

A. Memory Analysis

For a check node c with degree d_c, we must store one sign bit for every output message, and two minimum values of $q-1$ bits each. Thus the total number of stored bits is d_c+2q-2. In a variable node v of degree d_v, only $\beta_i^{(\ell)}$ has to be saved in memory. In order to avoid any saturation error when storing the sum, we must be able to represent any sum of d_v+1 messages. Thus storing a sum requires $q+q_s$ bits, with $q_s = \lceil \log_2(d_v+1) \rceil$. Since d_v varies, we define $q_s = \lceil \log_2(d_{v,\max}+1) \rceil$, where $d_{v,\max} = \max\limits_{i\in\{1,\dots,n\}}(d_{v_i})$.

B. Complexity Analysis

Due to the serial-C scheduling, a variable node is updated each time one of its neighboring check nodes is updated. Considering that variable node v_i is connected to check node c_j that is being updated, we first compute (4), and once the check node has been updated, finish the variable node update with

$$\beta_i^{(\ell)} \leftarrow \beta_{i\to j}^{(\ell-1)} + \gamma_{j\to i}^{(\ell-1)},$$

requiring $2d_{v_i}$ additions during one iteration, each applied to inputs of $q+q_s$ bits.

For the check node update, the processing of the sign in (5) requires $(2d_{c_j}-1)$ 2-input exclusive-OR (XOR-2) operations. Finally, we assume that the calculation of the two minimum values of (5) is performed using a merge-sort circuit architecture [17]. This circuit requires $\lfloor \frac{d_{c_j}}{2} \rfloor + 2(\lceil \frac{d_{c_j}}{2} \rceil - 1)$ comparisons, and all the comparisons are performed on inputs of $q-1$ bits.

C. Energy Models

In order to derive the complexity energy model, we denote by E_{add}, E_{xor}, E_{comp}, the elementary energy consumption of a 1-bit addition, an XOR-2 operation, and a 1-bit comparison, respectively. Consider an LDPC code of length N, rate R, and constructed from a protograph $\boldsymbol{S}$. For a target SNR ξ and bit error rate (BER) p_e, the complexity energy model is given by:

$$\begin{aligned} E_c = \frac{L_N(\xi,p_e)N}{n}\Bigg(2(q+q_s)E_{\text{add}}\sum_{i=1}^{n} d_{v_i} \\ + (1-R)\Bigg(E_{\text{xor}}\sum_{j=1}^{m}\left(2d_{c_j}-1\right) \\ + E_{\text{comp}}(q-1)\left(\lfloor \frac{d_{c_j}}{2} \rfloor + 2\left(\lceil \frac{d_{c_j}}{2} \rceil - 1\right)\right)\Bigg)\Bigg) \end{aligned} \quad (8)$$

where $L_N(\xi, p_e)$ (7) is the number of iterations needed by the decoder to achieve the performance target.

The number of operations performed by check nodes with d_{c_j} even and by check nodes with d_{c_j} odd only differ by a constant $\frac{1}{2}$. We can thus approximate E_c with the worst case where all the d_{c_j} are odd. If we also assume that a comparison has the same complexity as an addition, i.e. $E_{\text{comp}} = E_{\text{add}}$, the complexity energy model simplifies to:

$$\begin{aligned} E_c = L_N(\xi,p_e)N\Bigg(2E_{\text{add}}(q+q_s)\tilde{d}_v + E_{\text{xor}}(2\tilde{d}_c-1) \\ + \frac{3}{2}E_{\text{add}}(q-1)(\tilde{d}_c-1)\Bigg), \end{aligned} \quad (9)$$

where $\tilde{d}_v = \frac{1}{n}\sum_{i=1}^{n} d_{v_i}$ and $\tilde{d}_c = \frac{1}{m}\sum_{j=1}^{m} d_{c_j}$ are respectively the average variable and check node degrees of the code. Note that for protographs, we have $\tilde{d}_v = (1-R)\tilde{d}_c$.

In order to derive the memory energy model, we denote by E_{bit} the elementary energy consumption for writing one bit in memory. For an LDPC code of length N and rate R constructed from protograph $\boldsymbol{S}$, the memory energy model is given by

$$\begin{aligned} E_m = \frac{L_N(\xi,p_e)N}{n}E_{\text{bit}}\left(\sum_{i=1}^{n} d_{v_i}\,(q+q_s)\right. \\ \left. + (1-R)\left(\sum_{j=1}^{m}\left(2q+d_{c_j}-2\right)\right)\right), \end{aligned} \quad (10)$$

which can be simplified to

$$E_m = L_N(\xi,p_e)E_{\text{bit}}N\Big((q+q_s)\tilde{d}_v + (1-R)(\tilde{d}_c+2q-2)\Big). \quad (11)$$

The two energy models E_c and E_m depend on the SNR and BER targets through the average number of iterations L_N. In addition, only the average degrees $\tilde{d}_v$ and $\tilde{d}_c$ of the protograph $\boldsymbol{S}$ explicitly appear in these energy models. The protograph $\boldsymbol{S}$ however also has an influence on the number L_N of iterations, see (7).

V. Energy Optimization

In this section, we first formulate the decoder energy optimization problem. We then present an optimization method to solve this problem.

A. Optimization Problem

We now want to minimize the decoder energy consumption while maintaining a certain level of decoding performance. In order to simplify the optimization problem, we first assume that the code rate R, the codeword length N, the number q of quantization bits, and the dimensions m, n of the protograph are fixed. Then, in order to specify the decoding performance, we set a target SNR ξ^* and a target error probability p_e to be achieved at that SNR. Once these parameters are set, we formulate the optimization problem as

$$\min_{\boldsymbol{S},L} E(\boldsymbol{S}, L) \quad \text{s.t.} \quad p_{e,\text{opt}}(\xi^*) < p_{e,\max} \tag{12}$$

where

$$p_{e,\text{opt}}(\xi^*) = \min_{\alpha,\lambda} p_{e_N}(\xi^*)$$

and $p_{e_N}(\xi^*)$ is defined in (6). Note that $p_{e_N}(\xi^*)$ also depends on $\boldsymbol{S}$, and L, which is not explicitly stated here in order to simplify the notation. In (12), the energy function E can be given either by the complexity energy model (8), by the memory energy model (10), or by a weighted combination of both.

B. Optimization Method

In order to solve the optimization problem (12), we use a genetic algorithm called Differential Evolution [12]. This algorithm was initially introduced for non-linear and non-differentiable continuous space functions. However, in our optimization problem, the protograph coefficients and the number of iterations are discrete parameters. Thefore, we modified the original algorithm as described in the following.

Denote by $\mathcal{F}$ the function to be minimized. The Differential Evolution algorithm first generates randomly a population g_1 of size W of matrices $\boldsymbol{S}_1^{(1)}, \cdots, \boldsymbol{S}_W^{(1)}$, each of size $m \times n$. In order to generate a new population g_{i+1} of W matrices from the previous population g_i, Differential Evolution relies on two functions called Mutation and Crossover. These two functions realize W random combinations $\boldsymbol{V}_1^{(i+1)}, \cdots, \boldsymbol{V}_W^{(i+1)}$ of the matrices $\boldsymbol{S}_1^{(i)}, \cdots, \boldsymbol{S}_W^{(i)}$ of the population g_i. The population g_{i+1} is then constructed from the following selection rule: $\forall k \in \{1, \cdots, W\}$,

$$\boldsymbol{S}_k^{(i+1)} = \begin{cases} \boldsymbol{V}_k^{(i+1)} & \text{if } \mathcal{F}(\boldsymbol{V}_k^{(i+1)}) < \mathcal{F}(\boldsymbol{S}_k^{(i)}) \\ \boldsymbol{S}_k^{(i)} & \text{otherwise.} \end{cases} \tag{13}$$

In other words, a newly generated matrix $\boldsymbol{V}_k^{(i+1)}$ is included into the population only if it decreases the optimization criterion.

To properly adjust the algorithm to generate discrete protographs, the following changes are required. First, the populations g_i only contain protographs, and do not include the number of iterations. Second, when applying the Mutation and Crossover operations, the components of each vector of the population are rounded to the closest integer values, and forced to be between 0 and a given value $S_{\max}$. Then, for a protograph to be included into a population, it is necessary that $d_{v,\min} = \min_i(d_{v_i}) > 1$ and $d_{c,\min} = \min_i(d_{c_i}) > 1$ in order to avoid degree 0 and degree 1 nodes. In particular, we eliminate degree 1 variable nodes that could show good performance under density evolution, but a bad minimum distance, which would affect the code performance at finite length [21].

Before applying the selection step (13), we should check whether the protograph $\boldsymbol{V}_k^{(i+1)}$ verifies the constraint $p_{e,\text{opt}}(\xi^*) < p_{e,\max}$. However, computing $p_{e,\text{opt}}(\xi^*)$ is computationally expensive, because of the integral in (6). This is why we introduce a second SNR value ξ^{**} and only verify that

$$\min_{\alpha,\lambda} p_{e_\infty}(\xi^{**}) < p_{e,\max}.$$

The minimum over α and λ is computed by numerical minimization of $p_{e_\infty}(\xi^{**})$. If protograph $V_k^{(i+1)}$ satisfies the above constraint, we then compute the minimum number of iterations $L_N^*(\xi^*, p_{e,\text{opt}}(\xi^*))$ (see (7)) that allow to achieve $p_{e,\text{opt}}(\xi^*)$ for $\boldsymbol{V}_k^{(i+1)}$, if it exists. Finally, we apply the selection step with the function $E(\boldsymbol{V}_k^{(i+1)}, L_N^*)$.

VI. Simulation

For the protograph construction, we used a code rate $R = 0.5$, a protograph size $n = 4$, $m = 2$, $S_{\max} = 6$, and $p_{e,\max} = 10^{-9}$ is set as the maximum error probability at an SNR $\xi^{**} = 1.22$dB (at infinite code length) and $\xi^* = 1.45$dB .

For illustrative purposes, we substitute the energy constants with rough estimates. Based on the estimate of 0.1pJ for a 32-bit addition reported in [22], we set $E_{\text{add}} = 3.13$ fJ, and since a 1-bit adder contains two XOR-2 gates, $E_{\text{xor}} = 1.56$ fJ. We base the storage energy on the estimate of 10 pJ for a 64-bit access from an 8KB cache, yielding an average of $E_{\text{bit}} = 0.156$ pJ. These values do not affect the optimization result.

Table I shows examples of generated protographs using the proposed optimization method. The protograph S_0 is generated without the energy criterion, while the protograph S_m is generated based on the energy memory model, and S_c based on the complexity model. The energy evaluated using the memory model is denoted E_m, and E_c is the energy evaluated using the complexity model. As we can see, S_0 achieves a better SNR threshold, but S_c and S_m respect the SNR threshold criterion and consume less energy based on both energy models.

TABLE I: Infinite-length thresholds and finite-length energy scores of the protographs for $\xi^* = 1.45dB$ and $N = 10^4$.

Protograph	SNR	E_c	E_m
$S_m = \begin{bmatrix} 3 & 2 & 1 & 2 \\ 0 & 1 & 1 & 4 \end{bmatrix}$	1.21 dB	20.1 nJ	523 nJ
$S_c = \begin{bmatrix} 0 & 1 & 2 & 5 \\ 2 & 2 & 0 & 2 \end{bmatrix}$	1.20 dB	19.7 nJ	533 nJ
$S_0 = \begin{bmatrix} 2 & 1 & 3 & 2 \\ 5 & 1 & 1 & 0 \end{bmatrix}$	1.15 dB	33.5 nJ	883 nJ

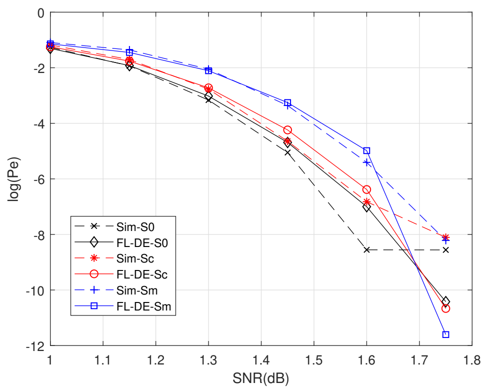

Fig. 1: Bit error rate of codes generated from protographs with energy criteria (S_c, S_m) and without the energy criteria (S0).

Figure 1 provides the BER of codes of length $N = 10000$ generated from each protograph, evaluated using Monte-Carlo simulation of the quantized decoder (Sim) as well as with the finite-length density evolution (FL-DE) method. At the target SNR $\xi^* = 1.45$dB, protograph S_c reduces E_c by 41% and S_m reduces E_m also by 41%, compared to the threshold-optimized protograph S_0. However, they also exhibit higher BER, especially in the case of the code constructed from S_m. Future work will improve the optimization method to allow finding solutions with any desired tradeoff of energy and BER performance.

VII. Conclusion

In this paper, we introduced two models to evaluate the energy consumption of quantized Min-Sum LDPC decoders. We then proposed an optimization method to minimize the energy consumption with respect to the protograph and the number of iterations, while satisfying a given decoding performance constraint. Future works will include other parameters in the optimization such as the quantization alphabets and the lifting factors.

Acknowledgements

This work was supported by grant ANR-17-CE40-0020 of the French National Research Agency (ANR).

References

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

[2] D. J. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics letters*, vol. 32, no. 18, pp. 1645–1646, 1996.

[3] K. Ganesan, P. Grover, J. Rabaey, and A. Goldsmith, "On the total power capacity of regular-LDPC codes with iterative message-passing decoders," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 375–396, 2016.

[4] ——, "Towards approaching total-power-capacity: Transmit and decoding power minimization for LDPC codes," *arXiv preprint arXiv: 1504.01019, 2015*, 2015.

[5] T. T. Nguyen-Ly, K. Le, V. Savin, D. Declercq, F. Ghaffari, and O. Boncalo, "Non-surjective finite alphabet iterative decoders," in *IEEE Int. Conf. on Communications (ICC)*, 2016, pp. 1–6.

[6] W. Yu, M. Ardakani, B. Smith, and F. Kschischang, "Complexity-optimized low-density parity-check codes for gallager decoding algorithm B," in *2005 Int. Symp. on Information Theory (ISIT)*, 2005, pp. 1488–1492.

[7] B. Smith, M. Ardakani, W. Yu, and F. R. Kschischang, "Design of irregular LDPC codes with optimized performance-complexity tradeoff," *IEEE Trans. on Communications*, vol. 58, no. 2, 2010.

[8] M. Karkooti and J. R. Cavallaro, "Semi-parallel reconfigurable architectures for real-time LDPC decoding," in *Int. Conf. on Information Technology: Coding and Computing*, 2004, pp. 579–585.

[9] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," *IPN progress report*, vol. 42, no. 154, pp. 42–154, 2003.

[10] D. G. Mitchell, R. Smarandache, and D. J. Costello, "Quasi-cyclic LDPC codes based on pre-lifted protographs," *IEEE Trans. on Information Theory*, vol. 60, no. 10, pp. 5856–5874, 2014.

[11] F. Leduc-Primeau and W. J. Gross, "Finite-length quasi-synchronous LDPC decoders," in *9th Int. Symp. on Turbo Codes and Iterative Information Processing (ISTC)*, 2016, pp. 325–329.

[12] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[13] Y. Fang, G. Bi, Y. L. Guan, and F. C. Lau, "A survey on protograph LDPC codes and their applications," *IEEE Commun. Surveys & Tutorials*, vol. 17, no. 4, pp. 1989–2016, 2015.

[14] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[15] A. Balatsoukas-Stimming and A. Burg, "Density evolution for min-sum decoding of LDPC codes under unreliable message storage," *IEEE Commun. Lett.*, vol. 18, no. 5, pp. 849–852, 2014.

[16] C. K. Ngassa, V. Savin, E. Dupraz, and D. Declercq, "Density evolution and functional threshold for the noisy min-sum decoder," *IEEE Trans. on Commun.*, vol. 63, no. 5, pp. 1497–1509, 2015.

[17] F. Leduc-Primeau, F. R. Kschischang, and W. J. Gross, "Modeling and energy optimization of LDPC decoder circuits with timing violations," *IEEE Trans. on Commun.*, vol. 66, no. 3, pp. 932–946, 2018.

[18] E. Sharon, S. Litsyn, and J. Goldberger, "Efficient serial message-passing schedules for LDPC decoding," *IEEE Transactions on Information Theory*, vol. 53, no. 11, pp. 4076–4091, 2007.

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

[20] T. Richardson, R. Urbanke *et al.*, "Multi-edge type LDPC codes," in *Workshop honoring Prof. Bob McEliece on his 60th birthday, California Institute of Technology, Pasadena, California*, 2002, pp. 24–25.

[21] D. Divsalar, S. Dolinar, C. R. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 6, pp. 876–888, 2009.

[22] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE Int. Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb 2014, pp. 10–14.