# Spiking Neural Belief Propagation Decoder for Short Block Length LDPC Codes

Alexander von Bank, *Graduate Student Member, IEEE*, Eike-Manuel Edelmann, *Graduate Student Member, IEEE*, Sisi Miao, *Student Member, IEEE*, Jonathan Mandelbaum, *Graduate Student Member, IEEE*, and Laurent Schmalen, *Fellow, IEEE*

***Abstract*— Spiking neural networks (SNNs) are neural networks that enable energy-efficient signal processing due to their event-based nature. This letter proposes a novel decoding algorithm for low-density parity-check (LDPC) codes that integrates SNNs into belief propagation (BP) decoding by approximating the check node update equations using SNNs. For the (273,191) and (1023,781) finite-geometry LDPC code, the proposed decoder outperforms sum-product decoder at high signal-to-noise ratios (SNRs). The decoder achieves a similar bit error rate to normalized sum-product decoding with successive relaxation. Furthermore, the novel decoding operates without requiring knowledge of the SNR, making it robust to SNR mismatch.**

***Index Terms*— Spiking neural networks, channel codes, decoder, message passing, belief propagation decoding, LDPC codes, neuromorphic, event-based computing.**

## I. Introduction

NEUROMORPHIC computing has gained much attention in recent years as CPUs are approaching their physical limits of operation speed and efficiency [1]. Spiking neural networks (SNNs) mimic the behavior of the human brain, which is a highly efficient computational device that consumes only 20 W of power to solve several complex tasks concurrently [2]. Hence, implementing SNNs on neuromorphic hardware is particularly interesting since it promises energy-efficient signal processing due to its event-based computing [3]. Furthermore, SNNs can be implemented on various types of neuromorphic hardware, enabling computation in the analog, digital, or photonic domain [1], [4], [5]. An SNN-based equalizer was emulated on an FPGA and compared to benchmark equalizers in [6], demonstrating the superior energy efficiency of SNNs.

Low-density parity-check (LDPC) codes are forward error-correcting codes included in various modern communication standards, e.g., 5G and WLAN, due to their outstanding error correction performance with iterative message-passing decoding, typically known as belief propagation (BP) decoding. In particular, close-to-capacity performance is reported when decoding LDPC codes using the sum-product algorithm (SPA). However, the SPA requires the evaluation of computationally intensive check node updates [7]. Therefore, complexity-reduced variants like normalized min-sum (NMS) [8] or differential decoding with binary message passing (DD-BMP) [9] have been introduced.

It is known that introducing memory to decoding via successive relaxation (SR) enhances its performance for the SPA and the min-sum (MS) algorithm [10]. As pointed out in [11], DD-BMP is also related to SR due to the memory of the decoder. Hence, decoding with memory can reduce the gap between complexity-reduced BP and plain SPA. Furthermore, SPA with SR can outperform classical SPA. In addition, a threshold to suppress small values can also help to improve MS decoding, as the offset MS decoder shows [12].

A system implementing the BP algorithm with SNNs has the potential to provide energy-efficient decoding. Therefore, BP was implemented with SNNs in [13] by connecting multiple SNNs and exchanging messages via spikes. The deviation of SNN-based BP messages compared to the correct messages was measured for a small graph with six nodes. However, no practical channel code was evaluated in [13].

In this letter, we propose a novel BP decoder that incorporates SNNs. In contrast to [13], the novel system uses simple SNNs rather than liquid and readout pools. Furthermore, we rely on real-valued/graded spikes, compared to the encoding to a pulse rate of [13]. We optimized SNNs to replace the complex SPA check node update with energy-efficient SNN operations. Two decoding variants are introduced. The simplified variant requires SNNs consisting of only a single neuron. While both outperform NMS and DD-BMP decoding in Monte Carlo simulations, in high signal-to-noise ratio (SNR) regimes, the proposed decoder outperforms SPA, and the simplified variant achieves SPA-like performance. To the best of our knowledge, this is the first work that uses SNNs to improve the energy efficiency of BP for channel decoding of practically relevant channel codes. The implementation of the proposed SNN-based BP decoder and additional results are available at [14].

## II. Spiking Neural Networks

SNNs consist of multiple layers of spiking neurons. Like classical neural networks, the layers are connected by linear layers. Spiking neurons can be interconnected within a layer, resulting in recurrent connections. Compared to classical neural networks, SNNs exhibit two significant differences: First, they exchange information in short pulses,

Received 15 October 2024; accepted 2 November 2024. Date of publication 6 November 2024; date of current version 10 January 2025. This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 101001899). Parts of this work were carried out in the framework of the CELTIC-NEXT project AI-NET-ANTILLAS (C2019/3-3) (grant agreement 16KIS1316) and within the project Open6GHub (grant agreement 16KISK010) funded by the German Federal Ministry of Education and Research (BMBF). The associate editor coordinating the review of this letter and approving it for publication was W. Liu. *(Corresponding author: Alexander von Bank.)*

The authors are with the Communications Engineering Laboratory, Karlsruhe Institute of Technology, 76187 Karlsruhe, Germany (e-mail: alexander.bank@kit.edu; eike-manuel.edelmann@kit.edu; sisi.miao@kit.edu; jonathan.mandelbaum@kit.edu; laurent.schmalen@kit.edu).

Digital Object Identifier 10.1109/LCOMM.2024.3492711

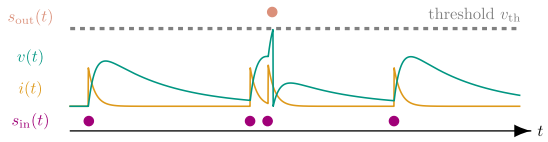



Fig. 1. Exemplary SNN dynamics. Input spikes $s_{\text{in}}(t)$ (illustrated by dots) generate a synaptic current $i(t)$, which charges the membrane potential $v(t)$. If $v(t)$ exceeds the firing threshold $v_{\text{th}}$, an output spike $s_{\text{out}}(t)$ is generated.

so-called *spikes*. Second, spiking neurons are state-dependent and exhibit temporal dynamics, mimicking the biological neuron [15]. The *membrane potential* $v(t)$ of a spiking neuron describes its state. An incoming spike induces a *synaptic current* $i(t)$, which charges $v(t)$. Concurrently, $v(t)$ decreases over time. If $v(t)$ exceeds the threshold potential $v_{\text{th}}$ of the neuron, an output spike is generated and the membrane potential is reset to the *resting potential* $v_{\text{r}}$. The leaky integrate-and-fire (LIF) model is a simple yet biologically plausible and, therefore, common spiking neuron model. Its dynamics in differential form are [15]

$$\frac{\mathrm{d}i(t)}{\mathrm{d}t} = -\frac{i(t)}{\tau_{\text{s}}} + \sum_j w_j s_{\text{in},j}(t), \tag{1}$$

$$\frac{\mathrm{d}v(t)}{\mathrm{d}t} = -\frac{((v(t)-v_{\text{r}})+i(t))}{\tau_{\text{m}}} + s_{\text{out}}(t)\,(v_{\text{r}}-v_{\text{th}})\,. \tag{2}$$

Hereby, $\tau_{\text{s}} \in \mathbb{R}^+$ denotes the time constant of the synaptic current, $w_j \in \mathbb{R}$ the weight applied to the $j$-th incoming sequence of spikes $s_{\text{in},j}(t) \in \mathbb{R}$, $\tau_{\text{m}} \in \mathbb{R}^+$ the time constant of the membrane potential, and $s_{\text{out}}(t) \in \{0,1\}$ the output spike sequence generated by $s_{\text{out}}(t) = \Theta(v(t)-v_{\text{th}})$, where $\Theta(\cdot)$ denotes the Heaviside function. The input spike sequence $s_{\text{in},j}(t)$ can be either sent from downstream layers (feed-forward) or spiking neurons within the same layer (recurrent). Fig. 1 provides an example of the dynamics of a LIF neuron. The leaky integrate (LI) neuron is obtained by deactivating the spiking functionality. Hence, LI neurons act as memory with leakage. For simulation and training, we utilize the PyTorch-based framework *Norse* [16].

## III. Belief Propagation Decoding

We consider the decoding of binary $(N,k)$ LDPC codes, which map $k$ data bits onto $N$ code bits. LDPC codes are defined as the null space of a sparse parity-check matrix (PCM) $\boldsymbol{H} \in \mathbb{F}_2^{M\times N}$. BP decoding is conducted on the Tanner graph, a bipartite graph associated with the respective PCM [17, pp. 51-59]. A Tanner graph consists of two types of nodes. First, the check nodes (CNs) $\mathsf{c}_j$, with $j \in \{1,2,\ldots,M\}$, which correspond to the single parity checks, i.e., the rows of the PCM. The second type of nodes are variable nodes (VNs) $\mathsf{v}_i$, with $i \in \{1,2,\ldots,N\}$, where each node is associated with a code bit, i.e., a column of the PCM. An edge between a CN $\mathsf{c}_j$ and a VN $\mathsf{v}_i$ in the Tanner graph exists iff $H_{j,i} = 1$. A regular $(d_{\text{v}}, d_{\text{c}})$ LDPC code possesses $d_{\text{c}}$ non-zero entries per row and $d_{\text{v}}$ non-zero entries per column in its PCM. The check node degree and variable node degree are denoted by $d_{\text{c}}$ and $d_{\text{v}}$, respectively.

Assume that a random codeword $\boldsymbol{x}$ is binary phase shift keying (BPSK) modulated and transmitted over a binary-input additive white Gaussian noise (AWGN) channel, i.e., $y_i = (-1)^{x_i} + n_i$, with $n_i \sim \mathcal{N}(0,\sigma^2)$.

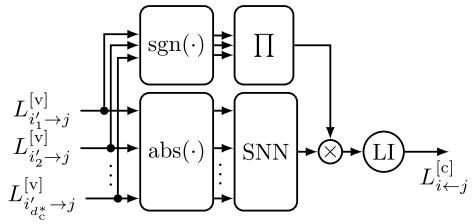



Fig. 2. Setup of an Elena-SNN SCNU calculating the message $L^{[\text{c}]}_{i\leftarrow j}$ based on the messages $L^{[\text{v}]}_{i'\to j}$, $i' \in \mathcal{M}(j)\setminus\{i\} =: \{i_1,\ldots,i_{d_{\text{c}}^*}\}$ with $d_{\text{c}}^* = d_{\text{c}} - 1$. The upper and lower branches realize (5) and (6), respectively. The LI neuron implements the memory.

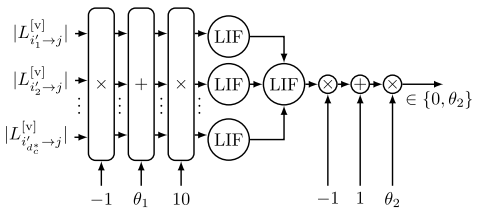



Fig. 3. Setup of the SNN block in Fig. 2 implementing (8) using $d_{\text{c}}$ LIF-neurons. If an input value is below $\theta_1$, the connected neuron is charged to emit a spike. The upstream neuron combines all incoming spikes and forwards them, leading to a zero output. If all input values are larger than $\theta_1$, no neuron is charged, leading to $\theta_2$ as output.

The bit-wise log-likelihood ratio (LLR) $L_i$ at the channel output w.r.t the $i$-th code bit is defined as

$$L_i = \log\left(\frac{P(Y_i = y_i | X_i = 0)}{P(Y_i = y_i | X_i = 1)}\right). \tag{3}$$

Next, we consider the SPA, in which messages are LLRs that are iteratively updated in the nodes and passed along the edges of the Tanner graph. A BP iteration with a flooding schedule includes the parallel updating of all CN messages and then all VN messages. The variable-to-check-node messages are initialized with $L^{[\text{v}]}_{i\to j} = L_i$, where $L^{[\text{v}]}_{i\to j}$ is the variable-to-check-node message sent from $\mathsf{v}_i$ to $\mathsf{c}_j$. First, the check-to-variable-node messages $L^{[\text{c}]}_{i\leftarrow j}$ are evaluated. To this end, we simplify the update equation [17, (2.17)] by splitting it into the absolute value $\alpha^{[\text{c}]}_{i\leftarrow j} = \left|L^{[\text{c}]}_{i\leftarrow j}\right|$ and the sign $\beta^{[\text{c}]}_{i\leftarrow j} = \operatorname{sign}\left(L^{[\text{c}]}_{i\leftarrow j}\right)$, i.e.,

$$L^{[\text{c}]}_{i\leftarrow j} = \alpha^{[\text{c}]}_{i\leftarrow j}\ \beta^{[\text{c}]}_{i\leftarrow j}, \quad \forall i \in \mathcal{M}(j), \tag{4}$$

$$\alpha^{[\text{c}]}_{i\leftarrow j} = 2\cdot\tanh^{-1}\left(\prod_{i'\in\mathcal{M}(j)\setminus\{i\}} \tanh\left(\frac{|L^{[\text{v}]}_{i'\to j}|}{2}\right)\right), \tag{5}$$

$$\beta^{[\text{c}]}_{i\leftarrow j} = \prod_{i'\in\mathcal{M}(j)\setminus\{i\}} \operatorname{sign}\left(L^{[\text{v}]}_{i'\to j}\right), \tag{6}$$

where $\mathcal{M}(j) = \{i : H_{j,i} = 1\}$ is the set of indices of VNs connected to CN $j$.

Next, we introduce the VN update calculating the variable-to-check-node message sent from $\mathsf{v}_i$ to $\mathsf{c}_j$ using

$$L^{[\text{v}]}_{i\to j} = L_i + \sum_{j'\in\mathcal{N}(i)\setminus\{j\}} L^{[\text{c}]}_{i\leftarrow j'}, \quad \forall j \in \mathcal{N}(i)\,, \tag{7}$$

where $\mathcal{N}(i) = \{j : H_{j,i} = 1\}$ is the set of indices of neighboring CNs of $\mathsf{v}_i$.

Note that both the CN and the VN update follow the extrinsic principle, i.e., when updating the message to VN

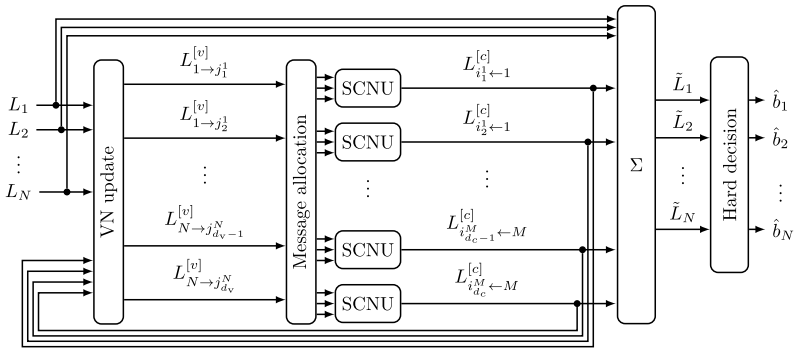



Fig. 4. The architecture of the proposed decoder. The SCNU contains the SNN. For $i \in \{1,\dots,N\}$ and $j \in \{1,\dots,M\}$, we use the notation $\mathcal{N}(i) := \{j_1^i,\dots,j_{d_v}^i\}$ and $\mathcal{M}(j) := \{i_1^j,\dots,i_{d_v}^j\}$, respectively.

$\mathsf{v}_i$ (CN $\mathsf{c}_j$), the message originating from $\mathsf{v}_i$ (CN $\mathsf{c}_j$) is excluded from the update equation. After reaching a predefined maximum number of iterations, the bit-wise output LLRs $\tilde{L}_i$ are calculated using $\tilde{L}_i = L_i + \sum_{j\in\mathcal{N}(i)} L_{i\leftarrow j}^{[\mathrm{c}]}$. Finally, the decoded bits $\hat{b}_i$ are obtained by applying a hard decision to $\tilde{L}_i$, where $\hat{b}_i = 0$ if $\tilde{L}_i > 0$ and $\hat{b}_i = 1$ otherwise. For a more detailed explanation of message-passing algorithms, the interested reader is referred to [17, Ch. 2].

## IV. Spiking Neural Belief Propagation Decoding

Equation (5) constitutes a significant part of the complexity of the SPA [7]. Therefore, an efficient replacement of the CN update using, e.g., SNNs, can improve the energy efficiency of the decoder. Hence, we propose a novel decoder named Enlarge-Likelihood-Each-Notable-Amplitude Spiking-Neural-Network decoder, referred to as *Elena-SNN*. Elena-SNN uses a so-called SNN CN update (SCNU) substitute for integrating SNNs into the CN update equations (4)-(6).

Fig. 2 depicts the setup of an SCNU calculating the message $L_{i\leftarrow j}^{[\mathrm{c}]}$ given the messages $L_{i'\to j}^{[\mathrm{v}]}$, $i' \in \mathcal{M}(j)\setminus\{i\}$. The incoming $L_{i'\to j}^{[\mathrm{v}]}$ are split into their sign and absolute value; both are processed, combined, and afterward integrated over time by an LI neuron, which acts as memory. Therefore, $\tau_m$ and $\tau_s$ of the LI neuron control the behavior of the SCNU memory. The upper branch performs (6), which can be implemented using XOR operations. The lower branch realizes (5), where the computationally complex operations of (5) are replaced by the SNN, shown in Fig. 3. Inspired by the offset MS algorithm [12], which approximates (5) by $\alpha_{i\leftarrow j}^{[\mathrm{c}]} \approx \max\left\{\min_{i'\in\mathcal{M}(j)\setminus\{i\}} \left|L_{i'\to j}^{[\mathrm{v}]}\right| - \theta_1, 0\right\}$, we further simplify the approach by

$$\alpha_{i\leftarrow j}^{[\mathrm{c}]} \approx \begin{cases} \theta_2 & \text{if } \min\limits_{i'\in\mathcal{M}(j)\setminus\{i\}} \left|L_{i'\to j}^{[\mathrm{v}]}\right| > \theta_1\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{8}$$

Like the offset MS algorithm, (8) returns zero if the input values are below the threshold $\theta_1 \in \mathbb{R}^+$. In contrast to the offset MS algorithm, which returns the biased minimum value, (8) returns a fixed value $\theta_2 \in \mathbb{R}^+$ if the threshold $\theta_1$ is exceeded. Hence, we approximate the term $\min_{i'\in\mathcal{M}(j)\setminus\{i\}} |L_{i'\to j}^{[\mathrm{v}]}| - \theta_1$ with $\theta_2$. Fig. 3 visualizes the implementation of (8) using LIF spiking neurons. To implement (8), we emit zero if at least one input value $|L_{i'\to j}^{[\mathrm{v}]}|$ is below the threshold $\theta_1$, and

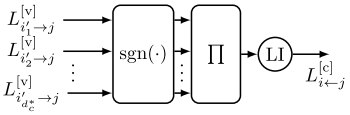
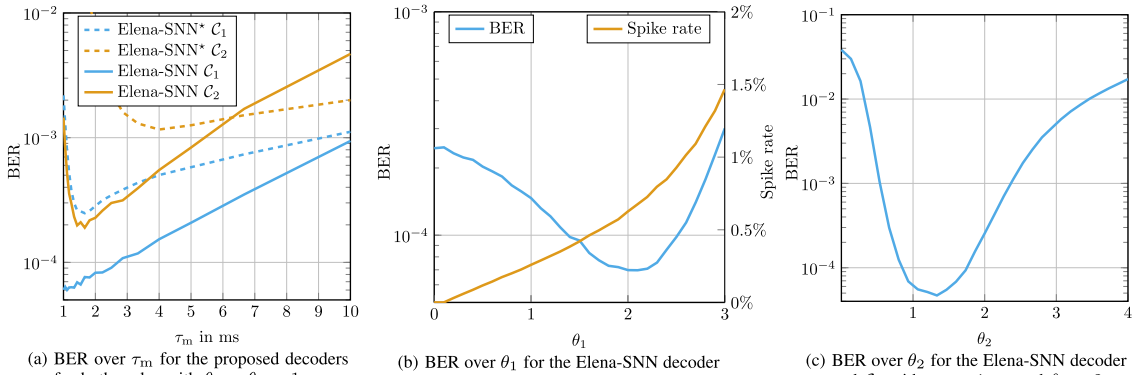



Fig. 5. Simplified Elena-SNN⋆ SCNU. The multiplication of the signs and the LI neuron act similarly to DD-BMP.

$\theta_2$ otherwise. Therefore, the signs of all inputs $|L_{i'\to j}^{[\mathrm{v}]}|$ are inverted, the bias $\theta_1$ is added, the signal is amplified and integrated by LIF neurons. If $(\theta_1 - |L_{i'\to j}^{[\mathrm{v}]}|) > 0$, the connected LIF neuron is charged, an output spike is generated and passed to the combining LIF neuron. If $(\theta_1 - |L_{i'\to j}^{[\mathrm{v}]}|) < 0$, the connected LIF neurons are discharged, and no output spike is generated. Hence, the multiplications, the adder, and the LIF neurons implement the inverted sign function; if a value is below $\theta_1$, a spike is emitted. The following combining LIF neuron fires if any of the previous neurons emit a spike. If the combining LIF neuron emits a spike, the output turns zero. Hence, if the minimum absolute of the input values is above $\theta_1$, $\theta_2$ is emitted. If at least one value is below $\theta_1$, zero is emitted.

Fig. 4 depicts the overall decoder architecture, integrating the SCNUs. All SCNUs share the same parameters. First, for initialization, all $L_{i\leftarrow j}^{[\mathrm{c}]}$ are set to zero, and the VNs are updated according to (7). Next, we start the iterative decoding process. The update of each CN consists in allocating the variable-to-check-node messages $L_{i\to j}^{[\mathrm{v}]}$ to the respective SCNU. Assuming a regular LDPC code, there exist $M \cdot d_\mathrm{c}$ SCNUs, each performing an extrinsic CN update. Hence, in the allocation, all variable-to-check node messages $L_{i'\to j}^{[\mathrm{v}]}$ with $i' \in \mathcal{M}(j)\setminus\{i\}$ participating in the update of $L_{i\leftarrow j}^{[\mathrm{c}]}$ are routed to the respective SCNU, which implements (4). Afterward, the VNs are updated, and the process is iteratively repeated until the hard decision maps the output LLRs to binary values. After that, the membrane potentials and synaptic currents of all neurons are reset to zero. The memory is, therefore, also reset. It is important to note that the decoder uses a flooding schedule, i.e., all node updates are performed in parallel.

Each SCNU of the Elena-SNN decoder uses $d_\mathrm{c}$ LIF spiking neurons and a single LI neuron. We furthermore propose a simplified version of Elena-SNN, called *Elena-SNN⋆*. Inspired by the DD-BMP, which only takes the sign of each $L_{i\to j}^{[\mathrm{v}]}$ into account, the number of neurons per SCNU can be reduced to a single LI neuron by omitting the lower path of Fig. 2. Its simplified SCNU can be seen in Fig. 5.




(a) BER over $\tau_\mathrm{m}$ for the proposed decoders for both codes with $\theta_1 = \theta_2 = 1$.

(b) BER over $\theta_1$ for the Elena-SNN decoder and $\mathcal{C}_1$ with $\tau_\mathrm{m} = 1\,\mathrm{ms}$ and $\theta_2 = 1$.

(c) BER over $\theta_2$ for the Elena-SNN decoder and $\mathcal{C}_1$ with $\tau_\mathrm{m} = 1\,\mathrm{ms}$ and $\theta_1 = 2$.

Fig. 6. Impact of the parameters of the decoders on the performance. Both decoders are run for 20 iterations. $\mathcal{C}_1$ is run at $\frac{E_\mathrm{s}}{N_0} = 3.5\,\mathrm{dB}$ and $\mathcal{C}_2$ at $\frac{E_\mathrm{s}}{N_0} = 3.4\,\mathrm{dB}$.

## V. Results and Discussion

We implemented and evaluated the two proposed decoders for two different finite-geometry (FG) LDPC codes, the (273,191) and the (1023,781) code [7], in the following called $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. Both FG LDPC codes are regular overcomplete codes, $\mathcal{C}_1$ has $d_\mathrm{c} = 17$, $d_\mathrm{v} = 17$, whereas $\mathcal{C}_2$ has $d_\mathrm{c} = 32$, $d_\mathrm{v} = 32$. FG LDPC codes together with BP decoding are known to typically possess very low error floors, which are of interest for, e.g., storage and wireline applications. Currently, due to the highly irregular structure of the 5G LDPC code and the presence of low-degree VNs, Elena-SNN does not yet achieve competitive performance. Adapting Elena-SNN to perform better with such codes is part of our ongoing investigation.

For $y_i$, which is the received value corresponding to the $i$-th code bit of a codeword, the bit-wise LLRs are $L_i = y_i \cdot L_\mathrm{c}$, where $L_\mathrm{c}$ is the channel reliability parameter. Usually, for an AWGN channel, $L_\mathrm{c} = 4\frac{E_\mathrm{s}}{N_0} = \frac{2}{\sigma^2}$ and therefore needs to be adapted to the channel SNR. However, depending on the code, we optimize both decoders at a fixed $\frac{E_\mathrm{s}}{N_0}$ corresponding to a BER of approximately $10^{-4}$. For evaluation, the value of $L_\mathrm{c}$ is maintained, regardless of the channel SNR. To compare against the fixed $L_\mathrm{c}$, the Elena-SNN-$L_\mathrm{c}$ decoder adjusts $L_\mathrm{c}$ to match the channel SNR. We choose $\frac{E_\mathrm{s}}{N_0} = 3.5\,\mathrm{dB}$ for $\mathcal{C}_1$ and $\frac{E_\mathrm{s}}{N_0} = 3.4\,\mathrm{dB}$ for $\mathcal{C}_2$. For 20 decoding iterations, the proposed decoders are compared to the following benchmark decoders: SPA, DD-BMP, MS, NMS, and NMS with SR decoding (SR-NMS). Further increasing the decoding iterations did not yield a significant improvement in the decoding performance. For all benchmark decoders, $L_\mathrm{c}$ matches the actual $E_\mathrm{s}/N_0$.

### A. Parameters

For all LIF spiking neurons of the Elena-SNN decoder, we fix $\tau_\mathrm{m} = 1\,\mathrm{ms}$, $\tau_\mathrm{s} = 1\,\mathrm{ms}$ and $v_\mathrm{th} = 1$. Furthermore, we fix $\tau_\mathrm{s} = 1\,\mathrm{ms}$ of the LI neurons. Hence, the remaining parameters subject to optimization are $\tau_\mathrm{m}$ of the LI neurons as well as $\theta_1$ and $\theta_2$. For the Elena-SNN⋆ decoder, $\tau_\mathrm{s} = 1\,\mathrm{ms}$ of the LI neurons is also fixed. Hence, the set of tuneable parameters is reduced to $\tau_\mathrm{m}$ of the LI neuron.

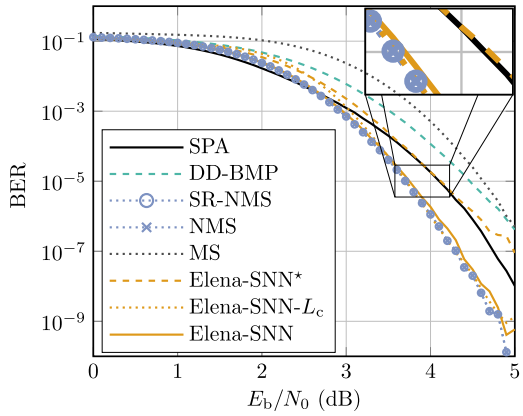
The membrane time constants $\tau_\mathrm{m}$ for the Elena-SNN decoder and Elena-SNN⋆ decoder are determined by a simple line search with $\theta_1 = \theta_2 = 1$. With the obtained $\tau_\mathrm{m}$, another line search yields $\theta_1$. The obtained $\tau_\mathrm{m}$ and $\theta_1$ are fixed to determine $\theta_2$ in a final line search. Fig. 6(a) shows the impact of $\tau_\mathrm{m}$ on the performance of both decoders for both codes. For Elena-SNN, small $\tau_\mathrm{m}$ yield the best performance, reducing the averaging effect of the LI neuron. Elena-SNN⋆ performs best at larger $\tau_\mathrm{m}$, demonstrating the need for averaging by the LI neuron. For Elena-SNN and $\mathcal{C}_1$, Fig. 6(b) shows the effect of the bias $\theta_1$ on the BER and the spike rate of the SNN, where the spike rate is the ratio of spikes generated by all LIF spiking neurons to possible spikes of the discrete SNN simulation. Up to a value of $\theta_1 \approx 2$, an increase in $\theta_1$ leads to a decreasing BER, while the spike rate increases. Further increasing $\theta_1$ yields both a higher BER and a higher spike rate. For $\mathcal{C}_2$, a similar behavior was observed. As Fig. 6(c) shows, the output amplitude $\theta_2$ significantly impacts the BER. Again, for $\mathcal{C}_2$, a similar behavior was observed. For the Elena-SNN decoder we have chosen ($\tau_\mathrm{m} = 1\,\mathrm{ms}$, $\theta_1 = 2$, $\theta_2 = 1.4$) for $\mathcal{C}_1$ and ($\tau_\mathrm{m} = 1.667\,\mathrm{ms}$, $\theta_1 = 1.6$, $\theta_2 = 1$) for $\mathcal{C}_2$. For the Elena-SNN⋆ decoder we have chosen $\tau_\mathrm{m} = 1.639\,\mathrm{ms}$ for $\mathcal{C}_1$ and $\tau_\mathrm{m} = 4.167\,\mathrm{ms}$ for $\mathcal{C}_2$.

### B. Evaluation

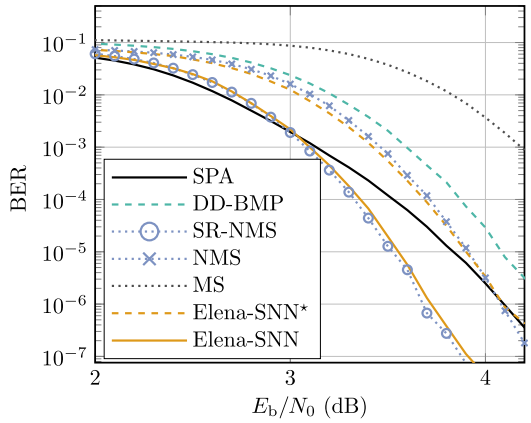
Fig. 7 compares the different decoders for $\mathcal{C}_1$ and $\mathcal{C}_2$. As expected, the SPA performs best in the low SNR regime for both codes. For higher SNR, the SR-NMS decoder performs best. Comparing the SR-NMS and the NMS decoders reveals that SR decreases the BER for $\mathcal{C}_2$. The Elena-SNN⋆ outperforms DD-BMP for both codes. Elena-SNN performs slightly worse than the SR-NMS decoder. At a BER of $10^{-5}$, Elena-SNN has a gap of $0.046$ dB to the SR-NMS decoder for $\mathcal{C}_1$. Elena-SNN yields an SNR gain of $0.35\,\mathrm{dB}$ compared to SPA. For $\mathcal{C}_2$, Elena-SNN achieves a gap of $0.03\,\mathrm{dB}$ to SR-NMS and a gain of $0.28\,\mathrm{dB}$ over SPA. The Elena-SNN decoder exhibits an error floor for $E_\mathrm{b}/N_0$ greater than $4.9\,\mathrm{dB}$, as Fig. 7(a) shows, resulting in a BER of approximately $10^{-9}$. The error floor results from the low resolution of the approximation in (8); future work will investigate an approximation with higher resolution and dynamic range. For the $\mathcal{C}_2$ code, we simulated $10^8$ codewords at an $E_\mathrm{b}/N_0$ of $4.48\,\mathrm{dB}$ and did not observe an error floor. Also, the constant $L_\mathrm{c}$ enables decoding without continuous SNR measurement, with improvement at low SNRs and only a slight penalty for high SNRs, as Fig. 7(a) shows.

### C. Discussion

We want to emphasize three significant points: First, Elena-SNN outperforms the SPA in high SNR regimes. Compared




(a) BER curve for the (273,191) FG LDPC code




(b) BER curve for the (1023,781) FG LDPC code.

Fig. 7. BER curves of the Elena-SNN , Elena-SNN⋆ and reference decoders.

to Elena-SNN, the simple Elena-SNN⋆ decoder reduces the computational complexity at the cost of decoding performance. It can only reach SPA performance; however, it solely requires the multiplication of signs and a single LI neuron. Second, Elena-SNN and Elena-SNN⋆ were optimized for a fixed $E_s/N_0$ and $L_c$ value. The results show that both approaches can generalize over a broad SNR regime. Thus, the set of applied parameters solely depends on the code, not the channel quality. Third, in high SNR regimes, the variable-to-check-node messages $L^{[v]}_{i\to j}$ tend to have large absolute values. Hence, the LIF spiking neurons of the Elena-SNN decoder are not sufficiently charged to generate an output spike. Therefore, at $E_b/N_0 = 3.5$ dB, a spike rate of $0.6\%$ is achieved, whereas, at $E_b/N_0 = 0$ dB, the spike rate is increased to $9\%$ at 0dB. Hence, depending on the SNR, the number of emitted spikes varies for the code $\mathcal{C}_1$ between 557 spikes per codeword and 8354 spikes per codeword between 0 dB and 3.5 dB.

## VI. Conclusion

In this work, we introduced Elena-SNN, a novel channel decoder for LDPC codes, which overcomes the complexity of the SPA by replacing its computationally complex calculations with an SNN. We furthermore introduced a simplified version of the Elena-SNN decoder, namely Elena-SNN⋆, which significantly reduces the number of neurons. The number of parameters that need to be optimized are three parameters for the Elena-SNN decoder and one parameter for the Elena-SNN⋆ decoder. For the (273,191) and the (1023,781) FG LDPC codes, BPSK, and an AWGN channel, we have shown that both decoders outperform complexity-reduced versions of the SPA, e.g., MS and DD-BMP. In high SNR regimes, the Elena-SNN decoder outperforms SPA. Both decoders were optimized at a fixed $E_s/N_0$ value and have been shown to generalize over the whole range of evaluated $E_s/N_0$ values. Hence, the decoder works without knowledge of the $E_s/N_0$ of the system, which is an advantage compared to the SPA.

Future work will test both proposed decoders on neuromorphic hardware and compare their complexity with the benchmark decoders. Furthermore, we will evaluate the proposed decoders for different types of LDPC codes. We are aware that the SNN input of the proposed decoders is real-valued. Thus, the neuromorphic hardware needs to support real-valued input, like, e.g., Intel Loihi [4].

## References


[1] T. F. de Lima, B. J. Shastri, A. N. Tait, M. A. Nahmias, and P. R. Prucnal, "Progress in neuromorphic photonics," *Nanophotonics*, vol. 6, no. 3, pp. 577–599, Mar. 2017.

[2] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, Nov. 2019.

[3] Y. S. Yang and Y. Kim, "Recent trend of neuromorphic computing hardware: Intel's neuromorphic system perspective," in *Proc. Int. SoC Design Conf. (ISOCC)*, Yeosu, South Korea, Oct. 2020, pp. 218–219.

[4] Intel. (2021). *Taking Neuromorphic Computing To the Next Level With Loihi 2*. Accessed: May 22, 2024. [Online]. Available: https://download.intel.com/newsroom/2021/new-technologies/neuromorphic-computing-loihi-2-brief.pdf

[5] C. Pehle et al., "The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity," *Frontiers Neurosci.*, vol. 16, Feb. 2022.

[6] M. Moursi, J. Ney, B. Hammoud, and N. Wehn, "Efficient FPGA implementation of an optimized SNN-based DFE for optical communications," in *Proc. IEEE Middle East Conf. Commun. Netw. (MECOM)*, Abu Dhabi, United Arab Emirates, Nov. 2024, pp. 1–6.

[7] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inf. Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001.

[8] F. Angarita, J. Valls, V. Almenar, and V. Torres, "Reduced-complexity min-sum algorithm for decoding LDPC codes with low error-floor," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 7, pp. 2150–2158, Jul. 2014.

[9] N. Mobini, A. H. Banihashemi, and S. Hemati, "A differential binary message-passing LDPC decoder," *IEEE Trans. Commun.*, vol. 57, no. 9, pp. 2518–2523, Sep. 2009.

[10] H. Xiao, S. Tolouei, and A. H. Banihashemi, "Successive relaxation for decoding of LDPC codes," in *Proc. 24th Biennial Symp. Commun.*, Kingston, ON, Canada, Jun. 2008, pp. 107–110.

[11] E. Janulewicz and A. H. Banihashemi, "Performance analysis of iterative decoding algorithms with memory over memoryless channels," *IEEE Trans. Commun.*, vol. 60, no. 12, pp. 3556–3566, Dec. 2012.

[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] A. Steimer, W. Maass, and R. Douglas, "Belief propagation in networks of spiking neurons," *Neural Comput.*, vol. 21, no. 9, pp. 2502–2523, Sep. 2009.

[14] A. von Bank. (2024). *Enlarge-Likelihood-Each-Notable-Amplitude Spiking-Neural-Network (ELENA-SNN) Decoder*. [Online]. Available: https://github.com/kit-cel/ELENA

[15] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Process. Mag.*, vol. 36, no. 6, pp. 51–63, Nov. 2019.

[16] C. Pehle and J. E. Pedersen. (Jan. 2021). *Norse—A Deep Learning Library for Spiking Neural Networks*. [Online]. Available: https://norse.ai/docs/

[17] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2008.