2023 12th International Symposium on Topics in Coding (ISTC) | 979-8-3503-2611-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISTC57237.2023.10273490

Turbo-XZ Algorithm: Low-Latency Decoders for Quantum LDPC Codes

Nithin Raveendran*, Emmanuel Boutillon‡, Bane Vasić*
* Department of Electrical and Computer Engineering,
Center for Quantum Networks,
The University of Arizona, Tucson, AZ, 85721.
Email: nithin@arizona.edu, vasic@ece.arizona.edu
‡Lab-STICC, UMR 6285 CNRS, Université Bretagne Sud, 56321 Lorient Cedex, France.
Email: emmanuel.boutillon@univ-ubs.fr

***Abstract*—We propose a low latency hardware-friendly decoding framework for Calderbank-Shor-Steane (CSS) quantum low-density parity-check (QLDPC) codes under the depolarizing noise model. With a given latency constraint, the proposed decoder, referred to generally as the Turbo-XZ decoding algorithm utilizes the correlation of Pauli X and Z errors. In this framework, we introduce early stopping and switching decoders to meet latency constraints and improve error correction performance for different decoders including the bit-flip (BF), fixed BF (proposed hardware-friendly variant of BF), and normalized min-sum algorithm (nMSA). This decoding framework allows various trade-offs in terms of latency, complexity, and decoding performance which are discussed briefly. Simulation results show that the BF-Turbo-XZ decoder performs close to (and beyond in some cases) the nMSA version with lower complexity and latency. Our proposed fixed BF approach reduces complexity with minimal performance degradation. For example with a generalized bicycle code, nMSA performs better for higher depolarizing values ($p > 0.02$) at a higher cost, while low-complexity BF-Turbo-XZ decoders are better at low depolarizing values.**

I. Introduction

Real-time decoding plays an essential role in achieving fault-tolerant quantum computing, as quantum systems are inherently noisy and susceptible to errors. Quantum error correction (QEC) codes and decoder implementations must be designed with low latency such that the measured syndrome is processed faster than or at the rate it is received. The expected latency requirements for real-time decoders depend on the qubit platform of implementation, ranging from the order of 1 μs for a surface code using superconducting transmon qubits to the order of 1 ms for silicon spin-qubits and even beyond 100 ms for ion traps [1]. For instance, Google's Quantum AI team's [2], demonstration needed error correction cycles of 921 nanoseconds. In another small-scale surface code experiment, researchers perform error correction in every 1.1 microseconds [3].

The shift from QEC using surface codes to general quantum low-density parity-check (QLDPC) [4], [5] codes necessitates long-range qubit connections [6], which can increase noise through delays and swaps, potentially further limiting the already restricted latency budget for specialized decoding hardware implementations. Nevertheless, QLDPC codes have been studied extensively in recent years mainly due to their superiority in the code minimum distance and code rate scaling asymptotically [7], [8]. There has been significant progress in designing efficient decoding algorithms for such 'asymptotically good' codes [9]. For the finite-length regime with these sparse graphical codes, although iterative decoders are a natural choice for decoding, they exhibit poor error correction performance [10], [11]. A syndrome-based belief propagation (BP) algorithm in its original form has been observed to perform significantly worse for QLDPC codes than their classical counterparts. The presence of unavoidable cycles in their Tanner graph, stabilizer splitting symmetric errors, and harmful trapping sets [12] typically make BP-based decoders fail in the waterfall regime (high to moderate depolarizing probability) and more so in the error floor regime (low depolarizing probability). Various decoding approaches have been pursued in attempts to improve the overall logical error rate curves in comparison to BP using heuristics approaches [11], neural-network trained BP decoders [13]–[15], exploiting the correlation between X and Z errors [16], [17] and soft syndrome information [18], refined BP [19], and numerous post-processing techniques of increasing complexity such as Bit-Flip (BF) [20], Small Set Flip (SSF) [21], Stabilizer Inactivation (SI) [22], and Ordered Statistics Decoder (OSD) [23], [24]. Such post-processing (PP) steps can be deployed after one or multiple rounds of BP decoder failure. Their main drawback is that the current implementation of these PP steps significantly adds up both in terms of the complexity (polynomial complexity in codelength) and the latency of error correction cycles (not suitable for parallel hardware implementation).

In this paper, we restrict ourselves to the problem of a given fixed, but low latency. The goal is to perform error correction rounds under a given timing budget by reducing the latency of decoding and hardware implementation complexity. It is well known that Bit-Flip (BF) algorithms are far less complex than BP-based algorithms like the normalized min-sum algorithm (nMSA). Choosing BF decoder for a fixed number of decoding iterations as the baseline for improving performance and complexity, we first develop strategies to exploit the correlation among Pauli X and Z errors considering the depolarizing channel. Since we operate in a fixed latency setting defined by decoding clock cycles, every decoding iteration and the PP step can be used to exploit the correlation information. The proposed BF-turbo-XZ decoding framework operates by switching the decoder graph and the decoded error vector when successful. We also aim to reduce the hardware implementation complexity by proposing *Fixed BF* decoder by choosing optimized fixed thresholds. Combining all these approaches in the decoder diversity scheme (combining strengths of multiple decoders) helps to development of 3 algorithms, sharing the same number of decoding iterations but with different trade-off performance complexity.

II. Preliminaries

A. Stabilizer Formalism

The n-qubit Pauli group can be denoted as $\mathcal{P}_n = j^l\{\mathrm{I}, \mathrm{X}, \mathrm{Y}, \mathrm{Z}\}^{\otimes n}$, with $0 \leq l \leq 3$, where $\otimes n$ denotes the n-fold tensor product, and X, Y, and Z are the Pauli matrices. I is the 2×2 identity matrix and $j = \sqrt{-1}$. Let $S = \langle S_1, S_2, ..., S_m \rangle$, $-\mathrm{I} \notin S$, be an Abelian subgroup of $\mathcal{P}_n$ with m generators S_i, $1 \leq i \leq m$. A (n, k) quantum stabilizer code [25] is defined as a 2^k-dimensional subspace $\mathcal{C}$ of the Hilbert space $(\mathbb{C}^2)^{\otimes n}$ that satisfies the condition that $S_i \left|\psi\right\rangle = \left|\psi\right\rangle$ for all i, for any $\left|\psi\right\rangle$ in $\mathcal{C}$. The projective measurement outcomes of these stabilizer generators correspond to their respective eigenvalues, which are referred to as the syndrome. The syndrome is used to determine whether an error has occurred and to identify its location and type. We focus on the quantum depolarizing channel (memoryless Pauli channel), which is characterized by the depolarizing probability p, such that the error on one qubit is independent of the error on other qubits. A Pauli error on a qubit i, denoted by E_i, takes values from the set $\{\mathrm{I}, \mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$ with probability $\mathbb{P}(E_i = \mathrm{X}) = \mathbb{P}(E_i = \mathrm{Y}) = \mathbb{P}(E_i = \mathrm{Z}) = p/3, \mathbb{P}(E_i = \mathrm{I}) = 1 - p$. Pauli error vector on n qubits can be expressed as a binary error vector of length $2n$ with the following mapping from Pauli elements to binary tuples: $\mathrm{I} \rightarrow (0, 0), \mathrm{X} \rightarrow (1, 0), \mathrm{Z} \rightarrow (0, 1), \mathrm{Y} \rightarrow (1, 1)$. Hence, we have the corresponding binary error vector $\mathbf{e} = [\mathbf{e}_\mathrm{x}, \mathbf{e}_\mathrm{z}]$ of length $2n$.

This mapping also gives a pair of binary matrix representations of the stabilizer generators referred to as the parity check matrices. Similar to the Pauli representation, the stabilizers also commute with respect to the *symplectic inner product* in binary representation. Among the stabilizer codes, *Calderbank-Shor-Steane (CSS)* codes [11] are generated via purely X-type ($(\mathbf{a}, \mathbf{0})$ in binary form) and purely Z-type ($(\mathbf{0}, \mathbf{b})$ in binary form) operators. This constraint enables CSS code construction using two compatible classical codes with binary parity-check matrices $\mathbf{H}_\mathrm{x}$ and $\mathbf{H}_\mathrm{z}$ such that $\mathbf{H}_\mathrm{x}{\mathbf{H}_\mathrm{z}}^\mathrm{T} = \mathbf{0}$ (the all-zeros matrix). The binary syndromes are measured as $\mathbf{s}_\mathrm{z} = \mathbf{H}_\mathrm{z}{\mathbf{e}_\mathrm{x}}^\mathrm{T} \pmod{\mathbf{2}}$ and $\mathbf{s}_\mathrm{x} = \mathbf{H}_\mathrm{x}{\mathbf{e}_\mathrm{z}}^\mathrm{T} \pmod{\mathbf{2}}$. The focus of our paper is on QLDPC codes from the class of CSS codes.

B. Quantum LDPC Codes

Sparse quantum stabilizer codes — *quantum low-density parity-check (QLDPC)* codes — have both $\mathbf{H}_\mathrm{x}$ and $\mathbf{H}_\mathrm{z}$ with a low density of ones in comparison to zeros. Decoders utilize the bipartite graphical representation of such sparse parity check matrices, commonly referred to as Tanner graphs. The syndrome decoding task of obtaining the error estimate given the measured syndrome is efficiently performed through syndrome-based iterative algorithms, such as BP, over the respective Tanner graphs. The Tanner graph for $\mathbf{H}_\mathrm{x}$ (or $\mathbf{H}_\mathrm{z}$) is denoted as $\mathcal{G}_\mathrm{x} = (V_\mathrm{x}, C_\mathrm{x}, F_\mathrm{x})$ (or $\mathcal{G}_\mathrm{z} = (V_\mathrm{z}, C_\mathrm{z}, F_\mathrm{z})$), where V and C represent the variable and check nodes, respectively, and F represents the edges. The subscripts corresponding to the X and Z graphs are omitted in the following unless necessary.

Since our focus is not only on performance improvement but also on maintaining latency requirements, we want to identify and improve decoders with low complexity - such as the BF decoder which is known for its low complexity on the iterative decoder hierarchy.

III. Bit-Flipping Syndrome Decoders

Given a parity check matrix $\mathbf{H}$ and measured syndrome $\mathbf{s}$, a syndrome BF decoder iteratively flips the error bits to find an error pattern $\mathbf{f}$ with syndrome $\hat{\mathbf{s}} = \mathbf{H}\mathbf{f}^\mathrm{T} \pmod{\mathbf{2}}$ that matches with $\mathbf{s}$. The syndrome BF algorithm is outlined in Algorithm 1. In the BF decoder, the error pattern estimate is initialized to $\mathbf{f}_{(0)} = \mathbf{0}$, with the subscript denoting iteration $\ell = 0$. Over iterations, the goal is to reduce the number of unsatisfied ($\neq$ $\mathbf{s}$ measured syndrome values) parity checks. The number of unsatisfied neighboring checks is often referred to as the 'energy' of the corresponding variable node as in the classical gradient descent bit-flipping algorithms [26]. At each iteration, we flip the error bit value(s) at the variable node(s) that meets a threshold $\boldsymbol{\Theta}$ defined as in Eq. (2). For the BF decoder, the threshold $\boldsymbol{\Theta}$ is the maximum value of the energy function $\mathcal{E}$, i.e., $\boldsymbol{\Theta} = \max(\mathcal{E})$. The BF decoder requires the computation of the maximum value of $\mathcal{E}$, a vector of length n varying for every iteration. The decoding process is declared a failure if the syndrome matching condition is not met after a maximum number of iterations, $\ell_{\max}$. When the decoder is successful with output $\mathbf{f}$, we declare that a logical frame error occurred if $\mathbf{f} + \mathbf{e} \pmod{\mathbf{2}} \notin \mathrm{RowSpace}(\mathbf{H})$.

A. Iteration Dependent Thresholds

In the proposed hardware-friendly variant - *Fixed BF* decoder, the threshold $\boldsymbol{\Theta}$ is a predefined sequence that depends only on the iteration number ℓ. An error bit $f(i)$ is flipped in the ℓ^{th} iteration if

$$\mathcal{E}(i) \geq \theta(\ell), \tag{3}$$

where $0 \leq \theta(\ell) \leq \max(d_v)$, where d_v is the variable node degree. The computation of the energy function $\mathcal{E}$ remains the same. This approach is adaptable to the specific use of a sequence of threshold values chosen to gradually correct the dominant errors stemming from trapping sets as shown in [27]. Furthermore, the fixed BF approach is more hardware friendly than the BF decoder since there is no more the necessity to determine the maximum energy among n variables, an operation that increases both the critical path and the hardware complexity.

Algorithm 1 Syndrome-based Bit-Flipping Algorithm

```
1:  **Input:** Measured syndrome $\mathrm{s}$, parity check matrix $\mathbf{H}$.
2:  **Output:** Estimated error vector $\mathbf{f}$
3:  **Initialization:** $\mathbf{f}_{(0)} = \mathbf{0}, \hat{\mathrm{s}} = \mathbf{0}, \ell = 0$.
4:  *Algorithm:*
5:  **while** $\ell \le \ell_{\max}$ **and** $\hat{\mathrm{s}} \ne \mathrm{s}$ **do**
6:      $\ell = \ell + 1$;
7:      Unsatisfied checks: $\delta\mathbf{s} = \mathbf{s} + \hat{\mathrm{s}} (\mathrm{mod}\ \mathbf{2})$;
        Energy $\mathcal{E} = \mathbf{H}^{\mathrm{T}} \delta \mathrm{s}$;    (1)
        Threshold: $\boldsymbol{\Theta} = \begin{cases} \max(\mathcal{E}), & \text{for BF,} \\ \theta(\ell) & \text{for Fixed BF.} \end{cases}$    (2)
8:      **for** $i = 1$ to $n$ **do**
9:          **if** $\mathcal{E}(i) \ge \boldsymbol{\Theta}$ **then**
10:             $f_\ell(i) = 1 - f_\ell(i)$;
11:         **end if**
12:     **end for**
13:     $\hat{\mathrm{s}} = \mathbf{H}{\mathbf{f}_\ell}^{\mathrm{T}} (\mathrm{mod}\ \mathbf{2})$;
14: **end while**
15: **return** $\mathbf{f} = \mathbf{f}_\ell$
```

B. Turbo-XZ Decoder Framework

The conventional approach of decoding $\mathbf{e}_{\mathrm{x}}$ and $\mathbf{e}_{\mathrm{z}}$ error patterns separately is sub-optimal in the depolarizing channel model we considered. When there is a Y error, knowledge of $\mathbf{e}_{\mathrm{x}}$ (or $\mathbf{e}_{\mathrm{z}}$) helps in decoding the other. This correlation between $\mathbf{e}_{\mathrm{x}}$ and $\mathbf{e}_{\mathrm{z}}$ is lost when we decode them separately using $\mathbf{H}_{\mathrm{z}}$ and $\mathbf{H}_{\mathrm{x}}$ parity check matrices, respectively. In [16], [17], XZ correlation is exploited by adjusting the initial likelihood vector used in the BP decoder which failed for the event when the other BP decoder succeeds. Based on the corrected error pattern, conditional log-likelihood ratios (LLRs) are calculated to adjust the initialization step. Alternatively, one can decode using the quaternary Tanner graph with either a quaternary BP [17] or refined BP [19] to exploit these correlations.

In this paper, we propose a novel approach that exploits the hard decision output from one of the BF decoders to synthesize a new error pattern. In the following, the proposed algorithm is illustrated by Fig. 1. In this figure, the syndromes are represented by diamonds that behave like classical variables, except that their value remains constant during the iterative decoding process. A satisfied check node is represented by a white square (dark square for unsatisfied check node). A white/black circle denotes a zero/one error bit. Let us partition $\mathbf{e}_{\mathrm{x}}$ into the set of X only errors $\mathbf{e}_{\mathrm{x}\bar{\mathrm{y}}}$ and Y only errors $\mathbf{e}_{\mathrm{y}}$, i.e., $\mathbf{e}_{\mathrm{x}} = \mathbf{e}_{\mathrm{x}\bar{\mathrm{y}}} + \mathbf{e}_{\mathrm{y}}$. Similarly, $\mathbf{e}_{\mathrm{z}}$ can be expressed as $\mathbf{e}_{\mathrm{z}} = \mathbf{e}_{\mathrm{z}\bar{\mathrm{y}}} + \mathbf{e}_{\mathrm{y}}$. Thus, one can note that the modulo-2 sum of vectors $\mathbf{e}_{\mathrm{z}}$ and $\mathbf{e}_{\mathrm{x}}$ gives $\mathbf{e}_{\mathrm{z}} + \mathbf{e}_{\mathrm{x}} = \mathbf{e}_{\mathrm{z}\bar{\mathrm{y}}} + \mathbf{e}_{\mathrm{x}\bar{\mathrm{y}}}$, i.e., a new synthetic error vector $\mathbf{e}_{\mathrm{z+x}}$.

Let us assume, without loss of generality, that the decoding of $\mathbf{e}_{\mathrm{z}}$ was unsuccessful (see Fig. 1.a) but $\mathbf{e}_{\mathrm{x}}$ was correctly decoded (Fig. 1.b and .c). It is possible to compute the syndrome given by the error vector $\mathbf{e}_{\mathrm{z+x}}$ using $\mathbf{H}_{\mathrm{x}}$ as $\mathrm{s}_{\mathrm{z+x}} = \mathrm{s}_{\mathrm{z}} + \mathbf{H}_{\mathrm{x}} \mathbf{e}_{\mathrm{x}}^{\mathbf{T}}$ (see Fig: 1.d). From this syndrome and $\mathbf{H}_{\mathrm{x}}$ as the parity check matrix, a decoding process can then retrieve the error vector $\mathbf{e}_{\mathrm{z+x}}$ (see Fig. 1.e, (initial conditions) and Fig. 1.f (final decoding result)). Then, the missing error vector $\mathbf{e}_{\mathrm{z}}$ is inferred as $\mathbf{e}_{\mathrm{z}} = \mathbf{e}_{\mathrm{z+x}} + \mathbf{e}_{\mathrm{x}}$, thus completing the decoding process (see 1.g). Note that, in this figure, the steps e) and f) are exposed to explain the rationale of the algorithm. In practice, step g) can be obtained directly from the initial condition of step d).

The proposed mechanism can be efficiently exploited in a latency constraint application, as shown in Fig. 1.h. The idea is to consider two decoders in parallel, each decoder being able to process both $\mathbf{H}_{\mathrm{x}}$ and $\mathbf{H}_{\mathrm{z}}$ matrices. In the beginning, as shown in Fig. 1.h the first decoder tries to decode $\mathbf{e}_{\mathrm{z}}$ from $\mathbf{H}_{\mathrm{x}}$ and s_{x} (see 1.a) while the second tries to decode $\mathbf{e}_{\mathrm{x}}$ from $\mathbf{H}_{\mathrm{z}}$ and s_{x} (see 1.b)). As soon as one decoder succeeds (second decoder in Fig. 1.c), the decoder is reconfigured to process the other matrix and uses its remaining time to try to decode $\mathbf{e}_{\mathrm{z+x}}$. Once one of the two decoders succeeds, both decoders stop and output the result. In this way, the time latency is guaranteed, and the decoders can take profit from the diversity added by the $\mathbf{e}_{\mathrm{z+x}}$ decoding attempt.

This method is very general and can be applied with any decoding algorithm.

IV. Analysis of Decoder Performance

We chose the family of generalized bicycle (GB) codes that were proposed in [23] with the two parity check matrices as $\mathbf{H}_{\mathrm{x}} = [\mathbf{A}\ \mathbf{B}]$ and $\mathbf{H}_{\mathrm{z}} = [\mathbf{B}^{\mathrm{T}}\ \mathbf{A}^{\mathrm{T}}]$ where $\mathbf{A}$ and $\mathbf{B}$ are square circulant matrices. For the GB $A1$ code [23] of codelength $n = 254$, variable and check degree, $(d_v, d_c) = (5, 10)$, the logical/frame error rate (FER) versus depolarizing channel noise probability (p) for different BF decoding strategies are plotted in Fig. 2. The turbo-XZ decoder switching approaches consistently deliver performance enhancements exceeding a tenfold improvement at low error rates for their respective decoder. The nMSA, with the best normalization value = 0.6, operates with full-precision messages and hence is far more complex both in terms of hardware and decoding clock cycles than the simple BF decoders given in Algorithm 1. We discuss in Section V some of these aspects pertaining to the hardware implementation and latency. Note that we plotted all the curves with the low latency (limited to $\ell_{\max} = 14$ iterations) constraint. Fig. 3 demonstrates the FER improvement over iterations within the allowed number of iterations $\ell_{\max} = 14$. FER improvement is shown for BF, Fixed BF, and nMSA curves. Specifically for the floating point nMSA, allowing more decoding iterations/latency can further improve the FER curves with the turbo-decoder switch approach. Note that the slope of the blue dashed line (nMSA + turbo decoder switch) is non-zero and steeper compared to BF decoders and nMSA, respectively.

Also, the fixed BF with iteration-dependent thresholds [27] when combined with the turbo-XZ decoder approach makes an excellent choice of a decoder in terms of reduced hardware complexity and latency. The threshold $\boldsymbol{\Theta}$ for the fixed BF decoder is a sequence of length 14 defined as $\boldsymbol{\Theta} =$

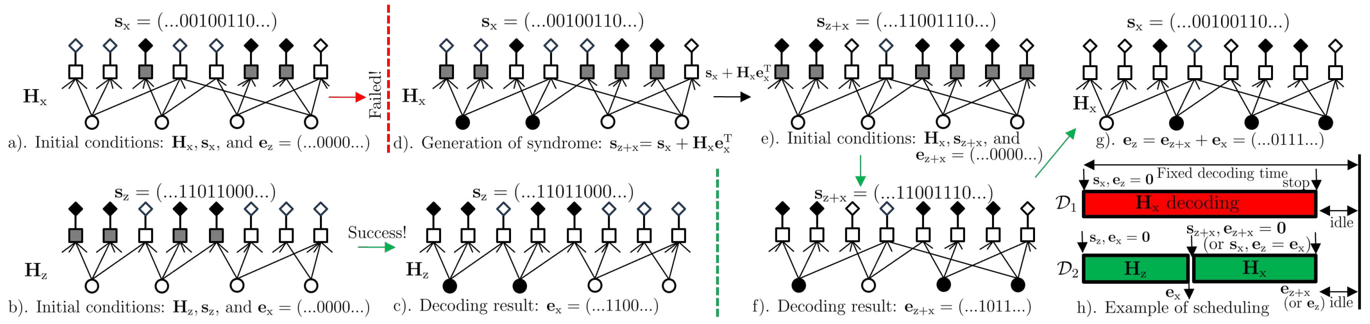
$\mathbf{s}_x = (\ldots00100110\ldots)$
$\mathbf{H}_x$
a). Initial conditions: $\mathbf{H}_x, \mathbf{s}_x$, and $\mathbf{e}_z = (\ldots0000\ldots)$
Failed!
d). Generation of syndrome: $\mathbf{s}_{z+x} = \mathbf{s}_x + \mathbf{H}_x\mathbf{e}_x^T$
$\mathbf{s}_x + \mathbf{H}_x\mathbf{e}_x^T$
$\mathbf{s}_{z+x} = (\ldots11001110\ldots)$
e). Initial conditions: $\mathbf{H}_x, \mathbf{s}_{z+x}$, and $\mathbf{e}_{z+x} = (\ldots0000\ldots)$
g). $\mathbf{e}_z = \mathbf{e}_{z+x} + \mathbf{e}_x = (\ldots0111\ldots)$
$\mathbf{s}_z = (\ldots11011000\ldots)$
$\mathbf{H}_z$
b). Initial conditions: $\mathbf{H}_z, \mathbf{s}_z$, and $\mathbf{e}_x = (\ldots0000\ldots)$
Success!
c). Decoding result: $\mathbf{e}_x = (\ldots1100\ldots)$
f). Decoding result: $\mathbf{e}_{z+x} = (\ldots1011\ldots)$
Fixed decoding time
$\mathbf{s}_x, \mathbf{e}_z = \mathbf{0}$
stop
$\mathcal{D}_1$
$\mathbf{H}_x$ decoding
idle
$\mathbf{s}_z, \mathbf{e}_x = \mathbf{0}$
$\mathbf{s}_{z+x}, \mathbf{e}_{z+x} = \mathbf{0}$
(or $\mathbf{s}_x, \mathbf{e}_z = \mathbf{e}_x$)
$\mathcal{D}_2$
$\mathbf{H}_z$
$\mathbf{H}_x$
$\mathbf{e}_x$
$\mathbf{e}_{z+x}$ (or $\mathbf{e}_z$)
idle
h). Example of scheduling

Fig. 1: Illustration of turbo-XZ decoder scheduling: from a) to g), an example of turbo-XZ decoding process, h) Fixed latency scheduling of the turbo-XZ process with two decoders. Red/green color indicates failure/success of the respective decoder.

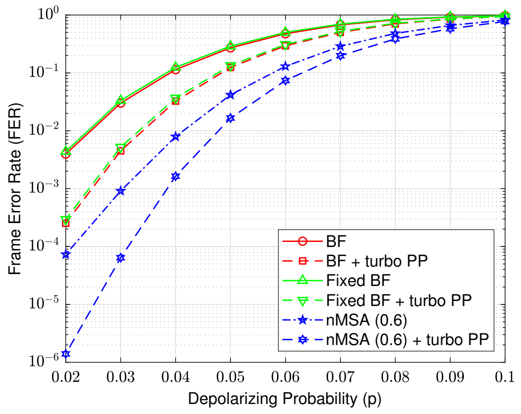

Fig. 2: Decoding using BF decoders with and without the turbo-XZ decoder for generalized bicycle-QLDPC code - A1 code [23]. This improvement is also illustrated with the floating-point normalized min-sum algorithm. Observe that fixed BF decoders achieve similar decoding performance as the BF strategy with lower hardware complexity at the same latency constraints.

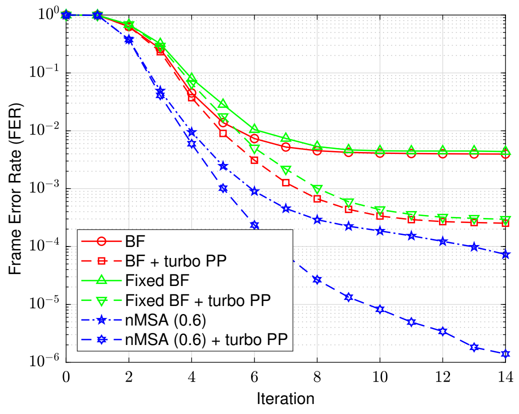

Fig. 3: Comparing FER vs. iteration for various BF decoders with and without the turbo-XZ schemes for QLDPC code GB - A1 code in [23] for a depolarizing probability $p = 0.02$.

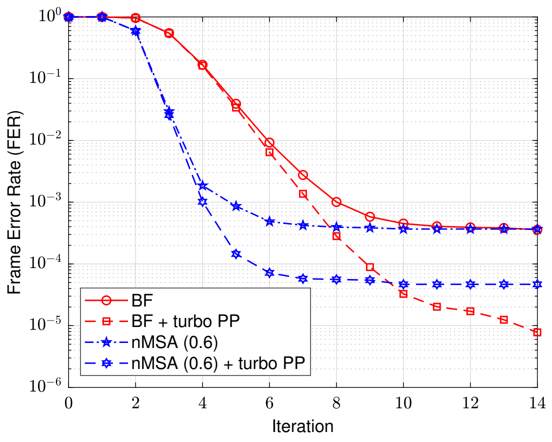

Fig. 4: FER vs. iteration for BF and nMSA decoder with and without the turbo-XZ decoding for the LP Tanner code.

$(4, 3, 3, 2, 3, 2, 2, 3, 3, 3, 4, 4, 2, 2)$. This sequence of thresholds has been obtained by an exhaustive search of the first 7 terms that give the best performance with 7 decoding iterations. The last 7 threshold values are obtained in a similar way - with 14 decoding iterations fixing the first 7 best-obtained thresholds. We also propose such specific decoders as a PP approach for tackling trapping set errors and in preliminary experiments, deploying threshold-optimized fixed BF decoders after a normalized min-sum decoder allows us to decrease the $\log_{10}$(FER) by a factor of 0.79 at a depolarizing $p = 0.02$ (Extended analysis is omitted due to lack of space).

In Fig. 4, we compare simulation results using a longer code of length $n = 1054$ from a different code family, the Lifted Product (LP) QLDPC code constructed using a classical quasi-cyclic LDPC code [18], [28]. At low error rates (for $p = 0.01$), the BF with turbo-XZ decoding shows lower FER than even a floating point nMSA with turbo-XZ.

V. Hardware Implementation

BF decoders are known to be far less complex than classical BP-based algorithms like the nMSA. In fact, in the latter case, a check node of degree d_c (variable node of degree d_v) sends d_c (d_v) distinct soft messages to its variables (checks) neighbors. In a BF algorithm, each node broadcast only a single bit to all its neighbors. Furthermore, in terms of hardware, it allows the implementation of one decoding iteration in a single clock cycle. Moreover, the use of a predefined sequence of threshold $\theta(\ell)$ suppresses the need for maximum finder among the energies of the variable node to get the value of $\max(\mathcal{E})$, thus further reducing the critical path of the design. Assuming a classical VLSI circuit design, the clock frequency can be set to 1 GHz, giving an overall decoding latency of only 14 ns. Assuming a required time latency of 1.4 μs, the factor 100 in time latency can be exploited to reduce by one or two orders of magnitude the power dissipation of the chip by using non-conventional hardware processing like adiabatic computation [29] or sub-threshold computation [30]. This reduction of power dissipation can help to place the decoder closer to the core of the quantum computer, thus reducing the interactions, and thus, perturbations, with the external world.

VI. Conclusion and Future Directions

In this work, we presented a turbo-XZ decoding framework as a promising solution that leverages the correlation information from decoding Pauli X and Z errors appropriately for fixed and low latency decoding of QLDPC codes. The proposed decoder shows orders of magnitude improvement for different codes we simulated at the same latency as without the turbo-XZ approach. By choosing the thresholds appropriately for handling trapping sets, we will extend the fixed BF approach as a low-latency post-processing approach to lower the error floors of QLDPC codes and explore general noise models. The need for efficient and low-latency quantum error correction algorithms will only grow as quantum computing platforms and technologies continue to advance. As such, the development of hardware-friendly error correction solutions like these for QLDPC codes will be crucial for future implementations.

VII. Acknowledgments

The work of the University of Arizona team is supported by the NSF under grants CIF-1855879, CIF-2106189, CCF-2100013, ECCS/CCSS-2027844, ECCS/CCSS-2052751, and ERC-194158, and in part by Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration and funded through JPL's Strategic University Research Partnerships (SURP) program. Bane Vasić has disclosed an outside interest in his startup company Codelucida to The University of Arizona. Conflicts of interest resulting from this interest are being managed by The University of Arizona in accordance with its policies. The authors would like to thank Valentin Savin for insightful discussions.

References

[1] F. Battistel, C. Chamberland, K. Johar, R. W. J. Overwater, F. Sebastiano, L. Skoric, Y. Ueno, and M. Usman, "Real-time decoding for fault-tolerant quantum computing: Progress, challenges and outlook," *arXiv preprint arXiv:2303.00054*, 2023.

[2] G. Q. AI, "Suppressing quantum errors by scaling a surface code logical qubit," *Nature*, vol. 614, no. 7949, pp. 676–681, 2023.

[3] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann *et al.*, "Realizing repeated quantum error correction in a distance-three surface code," *Nature*, vol. 605, no. 7911, pp. 669–674, 2022.

[4] D. J. C. MacKay, G. Mitchison, and P. L. McFadden, "Sparse-graph codes for quantum error correction," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2315–2330, Oct. 2004.

[5] N. P. Breuckmann and J. N. Eberhardt, "Quantum low-density parity-check codes," *PRX Quantum*, vol. 2, no. 4, p. 040101, Oct. 2021.

[6] N. Baspin and A. Krishna, "Quantifying nonlocality: How outperforming local quantum codes is expensive," *Phys. Rev. Lett.*, vol. 129, p. 050505, Jul 2022.

[7] P. Panteleev and G. Kalachev, "Asymptotically good quantum and locally testable classical LDPC codes," *arXiv preprint arXiv:2111.03654*, 2021.

[8] A. Leverrier and G. Zémor, "Quantum Tanner codes," in *2022 IEEE 63rd Ann. Symp. on Foundations of Comp. Science*. IEEE, 2022, pp. 872–883.

[9] ——, "Decoding quantum Tanner codes," *IEEE Trans. Inf. Theory*, 2023.

[10] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," *Quantum Inform. and Computation*, vol. 8, no. 10, pp. 987–1000, Nov. 2008.

[11] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "Fifteen years of quantum LDPC coding and improved decoding strategies," *IEEE Access*, vol. 3, pp. 2492–2519, 2015.

[12] N. Raveendran and B. Vasić, "Trapping sets of quantum LDPC codes," *Quantum*, vol. 5, p. 562, Oct. 2021.

[13] Y. H. Liu and D. Poulin, "Neural belief-propagation decoders for quantum error-correcting codes," *Phys. Rev. Lett.*, vol. 122, p. 200501, May 2019.

[14] X. Xiao, N. Raveendran, and B. Vasić, "Neural-net decoding of quantum LDPC codes with straight-through estimators," in *Proc. Inf. Theory and Applications Workshop*, 2019.

[15] S. Miao, A. Schnerring, H. Li, and L. Schmalen, "Neural belief propagation decoding of quantum LDPC codes using overcomplete check matrices," *arXiv preprint arXiv:2212.10245*, 2023.

[16] N. Delfosse and J.-P. Tillich, "A decoding algorithm for CSS codes using the X/Z correlations," in *IEEE Intl. Symp. on Inform. Theory*, 2014, pp. 1071–1075.

[17] A. Rigby, J. Olivier, and P. Jarvis, "Modified belief propagation decoders for quantum low-density parity-check codes," *Phys. Rev. A*, vol. 100, p. 012330, Jul. 2019.

[18] N. Raveendran, N. Rengaswamy, A. K. Pradhan, and B. Vasić, "Soft syndrome decoding of quantum LDPC codes to correct of data and syndrome errors," in *IEEE Intl. Conf. on Quantum Computing and Engineering*, Sep. 2022.

[19] K.-Y. Kuo and C.-Y. Lai, "Refined belief propagation decoding of sparse-graph quantum codes," *IEEE J. Selected Areas in Inf. Theory*, vol. 1, no. 2, pp. 487–498, 2020.

[20] T. R. Scruby and K. Nemoto, "Local probabilistic decoding of a quantum code," *arXiv preprint arXiv:2212.06985*, 2023.

[21] A. Grospellier, L. Grouè s, A. Krishna, and A. Leverrier, "Combining hard and soft decoders for hypergraph product codes," *Quantum*, vol. 5, p. 432, Apr. 2021.

[22] J. D. Crest, M. Mhalla, and V. Savin, "Stabilizer inactivation for message-passing decoding of quantum LDPC codes," in *2022 IEEE Inf. Theory Workshop (ITW)*, 2022, pp. 488–493.

[23] P. Panteleev and G. Kalachev, "Degenerate quantum LDPC codes with good finite length performance," *Quantum*, vol. 5, p. 585, Nov. 2021.

[24] J. Roffe, D. R. White, S. Burton, and E. Campbell, "Decoding across the quantum low-density parity-check code landscape," *Phys. Rev. Res.*, vol. 2, Dec 2020.

[25] D. Gottesman, "Stabilizer codes and quantum error correction," Ph.D. dissertation, California Institute of Technology, 1997.

[26] 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, June 2010.

[27] E. Boutillon, C. Winstead, and F. Ghaffari, "The syndrome bit flipping algorithm for LDPC codes," *IEEE Commun. Lett.*, Accepted, May 2023.

[28] R. Tanner, D. Sridhara, A. Sridharan, T. Fuja, and J. Costello, D.J., "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. on Inf. Theory*, vol. 50, no. 12, pp. 2966–2984, Dec. 2004.

[29] H. Fanet, *Front Matter*. John Wiley & Sons, Ltd, 2016.

[30] A. Wang, B. Highsmith, Calhoun, and A. P. Chandrakasan, *Sub-Threshold Design for Ultra Low-Power Systems*. Springer, 2006.