ISCTE IUL
REPOSITÓRIO
INSTITUTO UNIVERSITÁRIO DE LISBOA

Repositório ISCTE-IUL

Deposited in *Repositório ISCTE-IUL*:
2026-10-12

Deposited version:
Accepted Version

Peer-review status of attached file:
Peer-reviewed

Citation for published item:
Ferraz, O., Coutinho, B. C., Falcão, G., Gomes, M., Monteiro, F. A. & Silva, V. (2025). GPU-accelerated syndrome decoding for quantum LDPC codes below the 63 μs latency threshold. In Michael B. Matthews (Ed.), 2025 59th Asilomar Conference on Signals, Systems, and Computers (pp. 1878?1882). IEEE. https://doi.org/10.1109/IEEECONF67917.2025.11443687

Further information on publisher's website:
10.1109/IEEECONF67917.2025.11443687

Publisher's copyright statement:
This is the peer reviewed version of the following article: Ferraz, O., Coutinho, B. C., Falcão, G., Gomes, M., Monteiro, F. A. & Silva, V. (2025). GPU-accelerated syndrome decoding for quantum LDPC codes below the 63 μs latency threshold. In Michael B. Matthews (Ed.), 2025 59th Asilomar Conference on Signals, Systems, and Computers (pp. 1878?1882). IEEE. https://doi.org/10.1109/IEEECONF67917.2025.11443687, which has been published in final form at https://dx.doi.org/10.1109/IEEECONF67917.2025.11443687. This article may be used for non-commercial purposes in accordance with the Publisher's Terms and Conditions for self-archiving.

Use policy

Creative Commons CC BY 4.0
The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a link is made to the metadata record in the Repository
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

Serviços de Informação e Documentação, Instituto Universitário de Lisboa (ISCTE-IUL)
Av. das Forças Armadas, Edifício II, 1649-026 Lisboa Portugal
Phone: +(351) 217 903 024 | e-mail: administrador.repositorio@iscte-iul.pt
https://repositorio.iscte-iul.pt

GPU-Accelerated Syndrome Decoding for Quantum LDPC Codes below the 63 µs Latency Threshold

Oscar Ferraz[1,2], Bruno Coutinho[2], Gabriel Falcao[1,2], Marco Gomes[1,2], Francisco A. Monteiro[2,3], Vitor Silva[1,2]
[1]University of Coimbra, Department of Electrical and Computer Engineering, Coimbra, Portugal
[2]Instituto de Telecomunicações, Portugal
[3]ISCTE – Instituto Universitário de Lisboa, Lisbon, Portugal
Email: {oscar.ferraz, gff, marco, vitor}@co.it.pt, {bruno.coutinho, francisco.monteiro}@lx.it.pt

***Abstract*—This paper presents a GPU-accelerated decoder for quantum low-density parity-check (QLDPC) codes that achieves sub-63 μs latency, below the surface code decoder's real-time threshold demonstrated on Google's Willow quantum processor. While surface codes have demonstrated below-threshold performance, the encoding rates approach zero as code distances increase, posing challenges for scalability. Recently proposed QLDPC codes, such as those by Panteleev and Kalachev, offer constant-rate encoding and asymptotic goodness but introduce higher decoding complexity. To address such limitation, this work presents a parallelized belief propagation decoder leveraging syndrome information on commodity GPU hardware. Parallelism was exploited to maximize performance within the limits of target latency, allowing decoding latencies under 26 μs for [[784, 24, 24]] codes and as low as 16.9 μs for smaller codes, meeting the tight timing constraints of superconducting qubit cycles. These results show that real-time, scalable decoding of asymptotically good quantum codes is achievable using widely available commodity hardware, advancing the feasibility of fault-tolerant quantum computation beyond surface codes.**

I. Introduction

Quantum error correction (QEC) is a fundamental requirement for building of fault-tolerant quantum computers due to the delicate nature of quantum information and the relatively high noise rates in physical quantum devices [1]–[4]. Superconducting qubits, such as those used in Google's Willow processor, experience error rates in the order of 10^{-3} to 10^{-2} per gate [5] (without QEC), far above what is tolerable without active error correction [2]. In 2024, Google demonstrated the first instance of fault-tolerant quantum computation using a distance-5 surface code and achieved an average decoder latency of 63 µs for real-time syndrome processing over a million cycles [2].

However, while surface codes are well-suited to current hardware due to their locality and high thresholds, the encoding rate (ratio of logical to physical qubits) asymptotically approaches zero as the code distance increases [6], making them suboptimal for large-scale quantum computation. This leads to substantial physical qubit overheads in large-scale systems.

This work is funded by national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., and, when eligible, co-funded by EU funds under project/support UID/50008/2025 – Instituto de Telecomunicações, with DOI identifier https://doi.org/10.54499/UID/50008/2025.

A promising alternative lies in quantum low-density parity-check (QLDPC) codes such as those by Panteleev and Kalachev [7], which are "good" in the information-theoretic sense, maintaining constant encoding rates and linear minimum distance scaling [8]. Moreover, recent constructions [9] prove the existence of QLDPC codes with linear-time decoders, opening new avenues for practical and scalable quantum error correction.

Yet, the decoding of QLDPC codes is more intricate than that of surface codes. Their larger block lengths, sparse parity-check matrices, and often nonlocal stabilizers impose significant computational burdens on classical decoding hardware when aiming to meet sub-63 μs latency targets [2].

A key distinction explored in this work lies in the fundamental difference between classical and quantum low-density parity-check (LDPC) decoding. Classical LDPC decoding algorithms assume access to the noisy codeword, as is typical in communication systems [10], [11], allowing direct use of soft information (such as log-likelihood ratios (LLRs)) to iteratively update messages in a factor graph. In contrast, QEC must operate under the constraint that directly measuring the quantum state collapses its superposition, thereby destroying the encoded information.

To circumvent this, QLDPC decoding relies on syndrome-based decoding [12], [13], where stabilizer measurements are used to extract error syndromes, with the help of ancilla qubits, without disturbing the underlying quantum state. This enables error detection and correction while preserving quantum coherence.

Unlike classical decoders, which can observe and process the entire corrupted codeword, quantum decoders must infer the most likely error pattern solely from the measured syndrome, which indicates which stabilizer constraints have been violated. This imposes a fundamentally more challenging inference problem and alters the decoding dynamics [14]. Moreover, the syndrome extraction process in quantum hardware is itself subject to noise, further complicating the decoding task and motivating the use of soft-syndrome information to support probabilistic reasoning and improve convergence under uncertainty [13].

This work presents a graphics processing unit (GPU)-accelerated, belief-propagation-based decoder for QLDPC codes that leverages syndrome inputs and achieves latencies

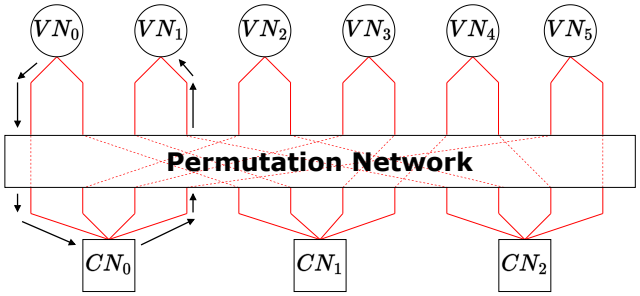


Fig. 1: LDPC Tanner graph based on (1). The number of rows of $\mathbf{H}$ corresponds to the number of check nodes (CNs) (M), while the number of columns represents the number of variable nodes (VNs) (N). The non-zero elements of $\mathbf{H}$ dictate the connections between VNs and CNs. After initialization, the messages are propagated from VNs to the connected CNs and back in an iterative manner until the parity-check equation is verified or until a maximum number of iterations has been reached, as described in Algorithm 1 [14].

Algorithm 1: Min-sum algorithm (MSA) for syndrome-based LDPC decoding

Input: syndrome ($\mathbf{s}$), parity-check matrix ($\mathbf{H}$), max. number of iterations ($\mathbf{I_{max}}$)
Initialization: $q_{m,n}^{(0)} = \gamma_n = 1$, $k = 1$
repeat
  Check node processing:
  for each check node m, each $n \in S_v(m)$ **do**
    $$r_{m,n}^{(k)} = \left(\alpha s_m \prod_{j \in S_v(m)\setminus n} \left(\operatorname{sign}\left(q_{m,j}^{(k-1)}\right)\right)\right) \min_{j \in S_v(m)\setminus n} \left(\left|q_{m,j}^{(k-1)}\right|\right)$$
  end for
  Variable node processing:
  for each variable node n, each $m \in S_c(n)$ **do**
    $$q_{m,n}^{(k)} = \gamma_n + \sum_{i \in S_c(n)\setminus m} r_{i,n}^{(k)}$$
  end for
  A posteriori calculation & Tentative decision:
  for each variable node n **do**
    $$Q_n^{(k)} = \gamma_n + \sum_{i \in S_c(n)} r_{i,n}^{(k)}$$
    $$e_n^{(k)} = \begin{cases} 1 & \text{if } Q_n^{(k)} < 0 \\ 0 & \text{if } Q_n^{(k)} \geq 0 \end{cases}$$
  end for
until $k < I_{max}$ and $\left(\mathbf{e^{(k-1)}}\mathbf{H}^T\right) \neq \mathbf{s}$

below the 63 μs threshold. Implemented on GPU platforms, the decoder exploits the fine-grained parallelism, shared memory, and reduced-precision arithmetic features of desktop and system on a chip (SoC) GPUs. The decoder implementation is based on codes structurally similar to [7], [9], [15]. While different quantum hardware platforms may impose distinct timing constraints, this work specifically targets the 63 μs latency threshold demonstrated in Google's Willow processor [2], with the goal of providing a viable solution for real-time fault-tolerant quantum error correction under such conditions.

This paper proposes the following contributions:

- The first GPU-based QLDPC decoder implementation, to the best of our knowledge.
- An efficient mapping of the QLDPC decoder to minimize latency and maximize throughput in GPU systems, exploiting the architectures' advantages and avoiding its drawbacks and limitations.

II. GPU-based QLDPC Decoder

The syndrome-based LDPC decoding process does not rely on direct access to the transmitted codeword, as in classical communication systems, but instead infers the most likely error configuration based solely on the syndrome vector. This is particularly relevant in the context of quantum LDPC decoding, where the syndrome, extracted via stabilizer measurements, provides indirect information about which qubits were affected by errors [16].

An LDPC code is defined by a parity-check matrix $\mathbf{H}$, which specifies the structure of a bipartite Tanner graph composed of CNs and VNs. The number of CNs and VNs are given by M and N, respectively, and their interconnections are determined by the non-zero entries in $\mathbf{H}$. For example, the following matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}, \tag{1}$$

defines a simple LDPC code with six VNs and three CNs. The sets $S_v(m)$ and $S_c(n)$ identify which VNs are connected to each CN, and vice versa. This code has a CN degree of four ($d_c = 4$) and a VN degree of two ($d_v = 2$), meaning each CN is connected to four VNs and each VN to two CNs. The associated Tanner graph is illustrated in Fig. 1.

In syndrome-based decoding, the decoder has no access to the original transmitted codeword or the channel likelihoods. Instead, it receives a syndrome vector $\mathbf{s} = \mathbf{H} \cdot \mathbf{e^T}$, where $\mathbf{e}$ is the unknown error pattern affecting the true codeword, and all the calculations are assumed to be carried out in $GF(2)$ [17], [18]. The goal is to find the most probable $\mathbf{e}$ such that the observed syndrome is satisfied, i.e., $\mathbf{H} \cdot \mathbf{e^T} = \mathbf{s}$

In classical channel-based LDPC decoding, the decoder typically operates on a received word $\mathbf{y} = \mathbf{c} + \mathbf{e}$, where $\mathbf{c}$ is the transmitted codeword. The decoder uses channel LLRs derived from $\mathbf{y}$ to estimate $\mathbf{c}$. The syndrome can also be computed directly from $\mathbf{y}$ as $\mathbf{s} = \mathbf{H} \cdot \mathbf{y^T}$, since $\mathbf{H} \cdot \mathbf{c^T} = \mathbf{0}$ by definition of $\mathbf{c}$ being a codeword. This yields the same result as $\mathbf{s} = \mathbf{H} \cdot (\mathbf{c} + \mathbf{e})^T = \mathbf{H} \cdot \mathbf{e}^T$ ($\mathbf{s}$, $\mathbf{e}$, $\mathbf{y}$ and $\mathbf{c}$ are defined as row-vectors).

However, in QLDPC decoding, the actual received word $\mathbf{y}$ is not observed, since the decoder only has access to the syndrome $\mathbf{s}$. Therefore, decoding must proceed without channel likelihoods. In this setting, soft-syndrome decoding extends traditional binary syndrome decoding by associating analog reliability values to each syndrome bit [14]. These can be derived from repeated quantum measurements, circuit-level noise simulations, or confidence estimates in the extraction circuit [19]. The decoder then uses this soft information to prioritize more likely error patterns and accelerate convergence.

In QEC, particularly under the stabilizer formalism, QLDPC codes are constructed by combining two classical binary LDPC codes to separately detect and correct *bit-flip* (X) and *phase-flip* (Z) errors [20]. These two types of errors are associated with the non-commuting Pauli operators and must be corrected independently to preserve quantum information. The

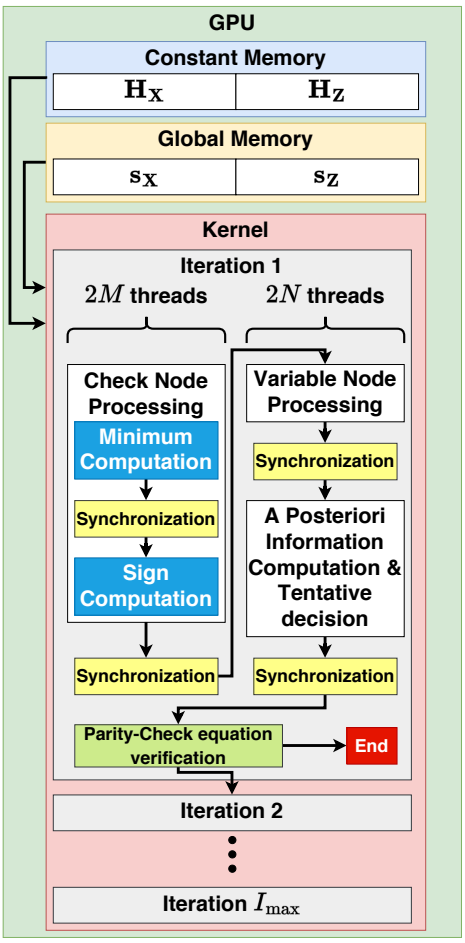


Fig. 2: Proposed GPU-based QLDPC decoder. The QLDPC parity-check matrices $\mathbf{H_X}$ and $\mathbf{H_Z}$ and stored in the compressed format in constant memory for faster accesss. The remaining data, such as the syndromes $\mathbf{s_X}$ and $\mathbf{s_Z}$, is stored in global memory. Each decoding iteration has four synchronization barriers and the CN processing is further divided into the minimum and sign computations. The CN processing is executed with $2M$ threads (equal to the size of $\mathbf{s_X} + \mathbf{s_Z}$) while the VN processing, a posteriori information, and tentative decision are processed with $2N$ threads. Both X and Z components are concatenated in memory and decoded in the same kernel, reducing latency and increasing throughput.

codes are defined by two parity-check matrices, $\mathbf{H_X} \in GF(2)$ and $\mathbf{H_Z} \in GF(2)$, where each row of $\mathbf{H_X}$ (respectively, $\mathbf{H_Z}$) corresponds to an X-type (respectively, Z-type) stabilizer generator acting on the n physical qubits [6].

To define a valid Calderbank-Shor-Steane (CSS)-type quantum code [21], [22], the stabilizer generators must commute, which imposes the commutativity condition $\mathbf{H_X H_Z^T} = \mathbf{0}$, which ensures that the X and Z stabilizer generators commute, a necessary property for the stabilizer group to form a valid quantum code space. This condition guarantees that the simultaneous eigenstates of all stabilizers define a proper code subspace.

During quantum error correction, syndromes are measured independently for X and Z errors using their respective stabilizers $\mathbf{s_Z} = \mathbf{H_X e_Z}^T$ and $\mathbf{s_X} = \mathbf{H_Z e_X}^T$, where $\mathbf{e_X}$, $\mathbf{e_Z}$ are binary vectors representing the X and Z error patterns, respectively. Two independent syndrome decoders are then used to estimate error vectors $\hat{\mathbf{e}}_\mathbf{X}$ and $\hat{\mathbf{e}}_\mathbf{Z}$ such that $\mathbf{H_Z}\hat{\mathbf{e}}_\mathbf{X}^T = \mathbf{s_X}$ and $\mathbf{H_X}\hat{\mathbf{e}}_\mathbf{Z}^T = \mathbf{s_Z}$.

To perform this decoding in practice, classical LDPC decoding algorithms can be applied separately to the $\mathbf{s_X}$ and $\mathbf{s_Z}$ syndrome equations, treating each as an independent binary decoding problem over $GF(2)$. Given the sparse nature of the parity-check matrices $\mathbf{H_X}$ and $\mathbf{H_Z}$, iterative message-passing algorithms are well-suited for this task, enabling efficient approximation of the most likely error patterns.

Among the various LDPC decoding algorithms, this work adopts the min-sum algorithm (MSA) [23] for its low complexity and good performance in sparse-graph decoding [24]. Algorithm 1 [14] outlines the decoding procedure, where each CN computes a new message based on incoming VN messages, and each VN aggregates information to update its belief of certainty of a given bit. This algorithm also features a scaling factor $\alpha \in (0, 1)$ in the CN processing step to mitigate the overestimation of message magnitudes and improve decoding accuracy compared to the standard formulation.

The proposed GPU-based QLDPC decoder implementation uses a single kernel to execute both X and Z components. As illustrated in Fig. 2, each decoding iteration is executed within one GPU kernel and comprises multiple computational stages mapped onto distinct threads. The process begins with the *CN processing*, which includes the computation of message signs and magnitudes, followed by *VN processing*, *a posteriori information computation*, and *tentative decision*. Constant memory allows to exploit faster read times and is used to store parity-check matrices $\mathbf{H_X}$ and $\mathbf{H_Z}$ which are static structures. After each CN and VN update (and inside the CN processing), synchronization barriers ensure consistent memory states across threads before proceeding to the next operation or iteration. The decoding loop proceeds until a stopping criterion is met (e.g., maximum number of iterations or valid syndrome check). This architecture exploits fine-grained parallelism by assigning individual CN and VN operations to CUDA threads, achieving high throughput.

III. Experimental Results

A. Experimental apparatus

The preliminary results use binary QLDPC codes [[72, 12, 6]], [[108, 8, 10]], [[144, 12, 12]], [[288, 12, 18]], and [[784, 24, 24]] (the code parameter [[n, k, d]] encodes k logical qubits into n data qubits offering a code distance d [15]), as provided in [15] running ten decoding iterations without early termination. In this work, three decoder versions were developed: 32-bit FP, 32-bit integer, and 8-bit integer precisions. In the 8-bit integer variant, the α factor is replaced by a clipping function to guarantee convergence. Shared memory is used to store $r_{m,n}$, while constant memory holds static structures (parity-check matrices) and global memory stores the remaining data. The devices used are the Nvidia Jetson TX2, Jetson AGX Orin, RTX 3090 and RTX 4090.

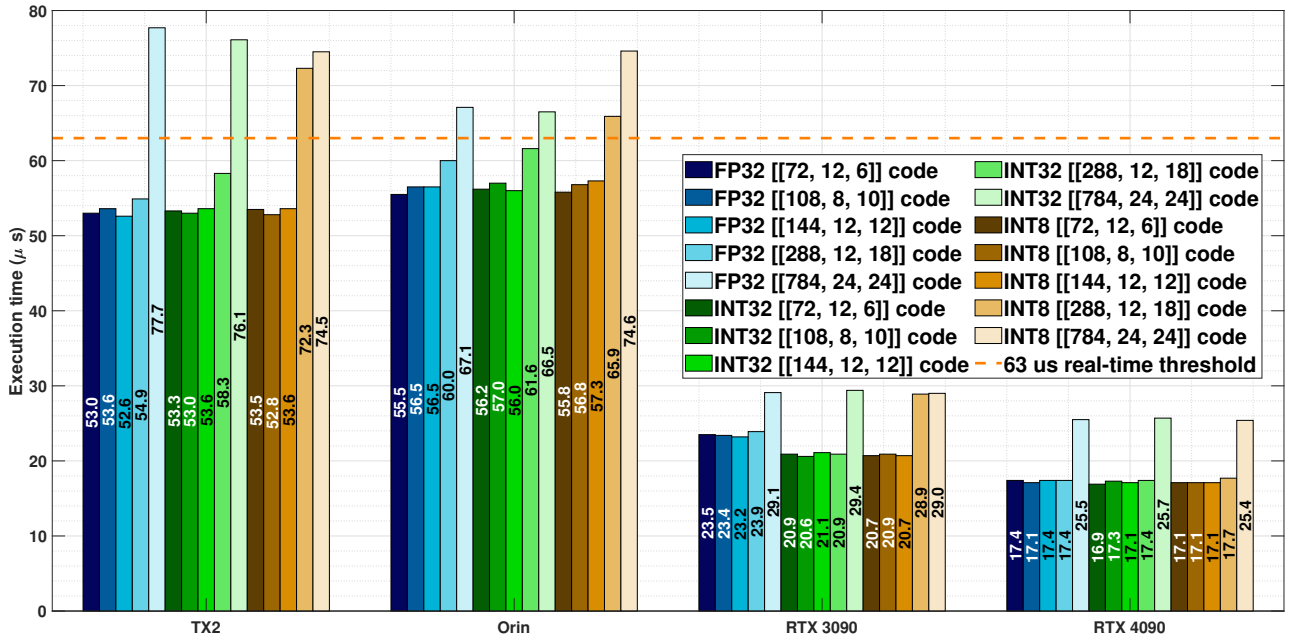


Fig. 3: Decoding performance for the QLDPC codes [[72, 12, 6]], [[108, 8, 10]], [[144, 12, 12]], [[288, 12, 18]], and [[784, 24, 24]] provided in [15] with implementations using 32-bit floating-point (FP), 32-bit fixed point and 8-bit fixed precisions. The bars represent the decoding time considering data transfers between host-to-device and device-to-host and kernel execution time for ten decoding iterations in four different GPUs. The dashed line represent the 63 μs real-time threshold [2].

B. QLDPC decoder results

The performance evaluation focuses on measuring the end-to-end decoding latency across different QLDPC codes and GPU platforms with data transfers included. Fig. 3 presents the average decoding latency obtained across the four GPU platforms for all benchmarked QLDPC codes and numerical precisions. The results reveal clear performance trends related to device class, code parameters, and arithmetic precision.

On the Jetson TX2, most of the decoding configurations remain bellow the 63 μs real-time threshold. For the smaller codes, [[72, 12, 6]], [[108, 8, 10]], [[144, 12, 12]], and [[288, 12, 18]], the FP32 latency tightly clusters between 53.0 μs and 54.9 μs, with INT32 and INT8 offering nearly identical behavior, hovering between 52.8 μs and 58.3 μs. The INT8 decoder for the [[288, 12, 18]] reaches 72.3 μs due to the clipping function computation. These values suggest that the message-passing workload fits reasonably well within the TX2's capabilities when code sizes are modest. However, once the largest [[784, 24, 24]] code is considered, a dramatic increase is observed: FP32 jumps to 77.7 μs, INT32 rises to 76.1 μs, and INT8 reaches 74.5 μs. Even the most efficient configuration therefore exceeds the real-time threshold by more than9 μs, highlighting the TX2's limited suitability for large-scale QLDPC decoding where Tanner graphs become considerably denser and computationally demanding.

The Jetson AGX Orin exhibits similar performance. For the smaller and mid-sized codes, the FP32 runtime falls between 55.5 μs and 60.0 μs, while INT32 and INT8 approximate similar values, generally between 55.8 μs and 65.9 μs. However, the Orin still struggles significantly on [[784, 24, 24]]. For this largest code, FP32 executes in 67.1 μs, while INT32 and INT8 complete in 65.5 μs and 74.6 μs respectively. These results show that, despite its higher compute density compared to the TX2, the Orin remains insufficient for real-time decoding of large QLDPC codes.

A pronounced shift occurs when moving to the RTX 3090, which successfully decodes every code within the required time window regardless of precision. Execution times across all configurations lie between 20.6 μs and 29.4 μs, demonstrating a comfortable margin relative to the 63 μs constraint. For the largest [[784, 24, 24]] code, the FP32 runtime is 29.1 μs, whereas INT32 and INT8 achieve 29.4 μs and 29.0 μs, respectively. These values illustrate not only substantial speedup compared to embedded platforms, but also a nearly negligible variation across arithmetic precisions. This indicates that the kernel execution is compute-bound and that the RTX 3090's architectural resources are well matched to the fine-grained parallelism inherent to the QLDPC decoding algorithm.

The RTX 4090 further amplifies these gains, delivering the lowest latencies across the entire dataset. All tested codes decode within 16.9 μs to 25.7 μs. For the [[784, 24, 24]] code, FP32 completes in 25.5 μs, while INT32 and INT8 reach 25.7 μs and 25.4 μs, respectively. This substantial additional margin reflects the 4090's much larger number of streaming multiprocessors, higher memory bandwidth, and improved warp scheduling efficiency. As a result, even the most demanding decoding workload becomes trivial in comparison to the device's available computational throughput.

Another important trend observed across all devices is that the reduction from FP32 to INT32 or INT8 provides only similar or incremental improvements rather than large gains. On desktop GPUs, the three precisions differ by less than a microsecond in many cases, indicating that the kernel

is dominated by parallel arithmetic operations and memory access patterns that are not significantly impacted by value representation. On the Jetson devices, INT8 yields slightly lower runtimes for the largest codes, but not enough to change the overall performance classification of the platform.

A deeper inspection of the latency components reveals that kernel execution becomes the dominant contributor as the code length increases. For small block lengths, host-device and device-host transfers represent between 15% and 25% of total latency, but for larger codes this drops to roughly 5% to 10%, as the arithmetic cost of CN and VN processing grows significantly. This reinforces the conclusion that architectural compute density, not memory transfers, becomes the main determinant of scalability.

Interestingly, Jetson platforms, despite their lower transfer overhead, do not outperform desktop GPUs on small codes, indicating that raw computational capability is the primary factor affecting performance even at modest code sizes. Meanwhile, the RTX GPUs clearly benefit from their higher clock frequency and advanced memory systems, which allow efficient exploitation of the parallelism and the reduced latency required by the QLDPC decoding algorithms.

These results demonstrate that commodity, off-the-shelf desktop GPUs are fully capable of meeting and exceeding real-time QLDPC decoding requirements, while embedded platforms currently fall short of the strict quantum-cycle timing constraints. Most importantly, the obtained latencies show that even complex, asymptotically good QLDPC codes, traditionally viewed as too computationally heavy for real-time operation, can be decoded within the 63 μs window using widely available GPU hardware.

IV. Conclusion

This work presented a GPU-accelerated, syndrome-based decoder for quantum LDPC codes capable of meeting the 63 μs real-time constraint demonstrated by Google's Willow processor. By exploiting GPU-oriented optimizations, such as constant memory, shared-memory, reduced-precision arithmetic, and a fused X/Z decoding kernel, the implementation efficiently maps the QLDPC message-passing workload onto massively parallel hardware.

The results show a clear divide between embedded and desktop GPUs. Jetson-class devices cannot meet real-time deadlines for larger codes, while commodity desktop GPUs, such as the RTX 3090 and RTX 4090 successfully decode all tested codes, including the large $[[784, 24, 24]]$ code, with latencies well below the 63 μs threshold. This demonstrates that real-time decoding of asymptotically good QLDPC codes is already feasible using widely available hardware.

Future work will explore mixed-precision pipelines, and evaluations across other quantum hardware platforms, as well as hybrid GPU–accelerator solutions. Overall, the proposed decoder provides a practical and scalable path toward real-time QLDPC decoding in future fault-tolerant quantum computing systems.

References

[1] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.

[2] R. Acharya, D. A. Abanin, L. Aghababaie-Beni, I. Aleiner, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, N. Astrakhantsev *et al.*, "Quantum error correction below the surface code threshold," *Nature*, 2024.

[3] L. Hanzo, Z. Babar, Z. Cai, D. Chandra, I. B. Djordjevic, B. Koczor, S. X. Ng, M. Razavi, and O. Simeone, "Quantum Information Processing, Sensing, and Communications: Their Myths, Realities, and Futures," *Proceedings of the IEEE*, 2025.

[4] D. Cruz, F. A. Monteiro, and B. C. Coutinho, "Quantum error correction via noise guessing decoding," *IEEE Access*, vol. 11, pp. 119 446–119 461, 2023.

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

[6] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Reports on Progress in Physics*, vol. 76, no. 7, p. 076001, 2013.

[7] P. Panteleev and G. Kalachev, "Quantum LDPC codes with almost linear minimum distance," *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 213–229, 2021.

[8] J. Coffey and R. Goodman, "Any code of which we cannot think is good," *IEEE Transactions on Information Theory*, vol. 36, no. 6, pp. 1453–1461, 1990.

[9] I. Dinur, M.-H. Hsieh, T.-C. Lin, and T. Vidick, "Good quantum LDPC codes with linear time decoders," in *Proceedings of the 55th annual ACM symposium on theory of computing*, 2023, pp. 905–918.

[10] Y. Ren, H. Harb, Y. Shen, A. Balatsoukas-Stimming, and A. Burg, "A generalized adjusted min-sum decoder for 5G LDPC codes: Algorithm and implementation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2024.

[11] O. Ferraz, S. Subramaniyan, R. Chinthala, J. Andrade, J. R. Cavallaro, S. K. Nandy, V. Silva, X. Zhang, M. Purnaprajna, and G. Falcao, "A survey on high-throughput non-binary ldpc decoders: Asic, fpga, and gpu architectures," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 524–556, 2021.

[12] J. Valls, F. Garcia-Herrero, N. Raveendran, and B. Vasić, "Syndrome-based min-sum vs OSD-0 decoders: FPGA implementation and analysis for quantum LDPC codes," *IEEE Access*, vol. 9, pp. 138 734–138 743, 2021.

[13] H. Yao, W. A. Laban, C. Häger, A. G. i Amat, and H. D. Pfister, "Belief propagation decoding of quantum LDPC codes with guided decimation," in *2024 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2024, pp. 2478–2483.

[14] N. Raveendran, J. Valls, A. K. Pradhan, N. Rengaswamy, F. Garcia-Herrero, and B. Vasić, "Soft syndrome iterative decoding of quantum LDPC codes and hardware architectures," *EPJ Quantum Technology*, vol. 10, no. 1, p. 45, 2023.

[15] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, "High-threshold and low-overhead fault-tolerant quantum memory," *Nature*, vol. 627, no. 8005, pp. 778–782, 2024.

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

[17] R. A. Carrasco and M. Johnston, *Non-binary error control coding for wireless communication and data storage*. John Wiley & Sons, 2008.

[18] T. K. Moon, *Error correction coding: mathematical methods and algorithms*. John Wiley & Sons, 2020.

[19] L. Pecorari, S. Jandura, G. K. Brennen, and G. Pupillo, "High-rate quantum LDPC codes for long-range-connected neutral atom registers," *Nature Communications*, vol. 16, no. 1, p. 1111, 2025.

[20] 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.

[21] N. Rengaswamy, R. Calderbank, M. Newman, and H. D. Pfister, "On optimality of CSS codes for transversal T," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 2, pp. 499–514, 2020.

[22] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Physical Review A*, vol. 54, no. 2, p. 1098, 1996.

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

[24] O. Ferraz, Y. Falevoz, V. Silva, and G. Falcao, "Unlocking the Potential of LDPC Decoders with PiM Acceleration," in *2023 57th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2023, pp. 1579–1583.