Low-Latency GNN for quantum error correction

Alessio Cicero
supervised by Pedro Trancoso
co-supervised by Anton Frisk Kockum, Mats Granath
ciceroa@chalmers.se
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden

Abstract

Real-time decoding of quantum error correction codes is one of the important steps in achieving fault-tolerant quantum computing. The error rate of physical qubits is currently too high to allow the execution of complex applications. To overcome this issue, multiple physical qubits are grouped into a logical qubit. The logical qubit error rate is dependent on the ability to decode and correct errors in the multiple physical qubits encoding one logical qubit. Although a wide range of encoding approaches is possible, the surface code is the most common encoding algorithm. Decoding must be done in real-time to avoid an increasing backlog of errors. Different decoding algorithms have been developed, such as minimum weight perfect matching, union-find and neural networks. Our work focuses on the latter approach, proposing an implementation of a graph neural network (GNN), able to decode the surface code up to code distance $d = 7$. We present CPU-, GPU- and FPGA-accelerated implementations and compare them in terms of latency, accuracy and power. The FPGA implementation, the only hardware-accelerated solution that can achieve real-time decoding, is additionally compared to the HDL implementation obtained with the GNNBuilder framework.

CCS Concepts

• **Computer systems organization → Quantum computing**.

Keywords

Quantum error correction, Surface code, GNN

ACM Reference Format:
Alessio Cicero, supervised by Pedro Trancoso, and co-supervised by Anton Frisk Kockum, Mats Granath. 2025. Low-Latency GNN for quantum error correction. In *Proceedings of 19th ACM International Conference on Distributed and Event-based Systems (DEBS'25).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Problem Statement

Quantum computers have the potential to speed up the resolution of complex problems in multiple fields, such as developing new chemical compounds [17] or evaluating the physical properties of new materials [4]. While classical computers base their computations on bits, whose values can only be 0 or 1, quantum computers are based on qubits. A qubit value is defined in a two-dimensional complex vector space spanned by the basis vectors $|0\rangle$ and $|1\rangle$, as a probability of being either zero or one:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{1}$$

with α and β being probability amplitudes, such that $|\alpha|^2 + |\beta|^2 = 1$. This principle of superposition, together with the related principle of entanglement for multiple qubits allows to map more easily multiple optimisation problems and NP problems as quantum circuits, and therefore speed up their execution compared to the classical approach [18].

The maximum complexity of the problem we can solve is limited by the size of the quantum circuits that the quantum computer can run. The two main factors affecting this are the number of qubits and their lifetime. While the former is constrained mainly by the manufacturing capabilities, the latter is limited by the maximum time after which the qubit value is no longer reliable, as it has collapsed to either 0 or 1. This time is defined as the lifetime of the qubit, and it is technology-dependent. To improve this, there is a need to have quantum error correction [10].

As in the classical world, there are multiple quantum error correction codes, the most common one is the surface code [11]. Multiple physical qubits are used to form a single logical qubit, with an overall logical error rate lower than the individual qubits' physical error rate. The physical qubits are measured, a graph is built out of the readings, and it must be decoded before the next measurement, in the case of surface code for superconducting qubits [12], this takes 1 µs [1]. This introduces a heavy constraint on the available decoding time, which needs to be respected to have real-time quantum computer quantum error correction.

2 Methodology

The methodology followed is experimental research. We need to evaluate quantifiable metrics such as latency, resource utilisation and accuracy to compare different implementations with the state of the art. We are designing experiments such as parametric explorations of different architectures, we compare them by the previously described metrics, and our methodology focuses on reproducibility, measurement and comparative analysis.

It is the most suitable method as we can isolate the effect of our design choice from the variation in the experiment set-up, by keeping the dataset and model constant across implementations, using the same toolchain and target frequency, and using the same way of measuring the latency. The inference latency is considered

DEBS'25, Gothenburg, Sweden

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/10.1145/nnnnnnn.nnnnnnn

as the time to process a single graph, from input to output, and the resources considered for the utilisation are the logic element, DSP blocks and on-chip memory consumption reported by the synthesis tool. The set-up time of the decoder, such as weight loading is excluded as long as it is input dependant, and therefore it only needs to be executed once at start-up time.

3 Research setting

As mentioned previously, the biggest constraint for a real-time quantum error correction is the maximum decoding time. To avoid a backlog of decoded syndromes, it is necessary to process each error correction cycle before the next measurement is available. Using a high-performance computer or considering unlimited computational resources is not possible, as the decoding should be done physically close to the quantum chip, to avoid having a too long delay in the physical communication between the chip and the decoder, which sometimes can even be longer than the total time allowed to execute the decoding itself. The main research question is finding a scalable, run-time, as accurate as possible quantum error correction code and decoder.

My research focuses on investigating with a top-down approach the combination of code and decoder, and the optimal implementation in hardware of the latter. Starting from the top level, there are multiple quantum error correction codes, with varying accuracy, complexity, and scalability. Each code can be decoded with multiple types of algorithms, each one of them also with varying degree of accuracy, complexity, and scalability. Lastly, their implementation in hardware needs to achieve the wanted latency and this is done through a design space exploration of possible hardware optimizations. The scalability of the decoder determines the maximum size of the code or the number of logical qubits that can be decoded in real-time. Increasing the size of the code leads to a lower logical error rate, if the physical error rate of the single qubit is lower than the code-dependent threshold [1]. To summarise, our research is based on achieving the best accuracy, thanks to the most accurate algorithm that can decode the biggest code distance graph while running within the maximum latency.

4 Related work

The two most used quantum error correction codes are the surface code [11] and low-density parity-check codes [6]. Surface code based decoders use a variety of algorithms, including minimum weight perfect matching (MWPM) implemented as Blossom Algorithm [14], union find (UF) [8], belief propagation (BP) [19], and neural networks [15].

The most commonly used for hardware implementation is MWPM [3, 7, 23], with the current state of the art being the work Promatch [3]. They are able to decode surface codes of code distance up to $d = 13$ while still being under the 1 µs threshold, and fitting the hardware decoder in a single FPGA. While they are able to achieve the wanted latency, the accuracy of the MWPM approach is still worse than the one obtainable from a neural-network approach.

Another main line of works is on the UF decoding, which sacrifices even more accuracy, as the UF algorithm is even less accurate compared to MWPM, to have a simpler algorithm and therefore faster to execute. The state-of-the-art work implementing the UF decoder on FPGA is [16]; they are able to decode surface codes up to $d = 51$ while still achieving the real-time maximum latency.

5 Research approach

To achieve the optimal decoder, it is necessary to explore the broad design space of the quantum error correction code, the decoder algorithm, and the decoder implementation.

5.1 Quantum error correction code

An initial exploration was done of the possible error correction codes for a quantum circuit:

- Repetition codes [20]: the simplest one, but it can only correct for one type of error (bit or phase), so is not a proper QEC code
- Surface code [1, 9]: most commonly used, they have a better scaling, but require a large qubit overhead
- Low-density parity check code [6]: has a lower overhead, so it allows a better ratio of logical qubits to physical qubits, but requires even more complex algorithms to be decoded, therefore the latency increases

In our first work, the chosen quantum error correction code is the surface code, as it presents a good trade-off between accuracy, scalability, and complexity of the decoder.

5.2 Decoding algorithm

Also in the case of the decoding algorithm, there is a broad choice of possible approaches; the most common are reported here:

- Union find decoder [8]: it is one of the simplest decoding algorithms, but it also leads to a lower accuracy
- Minimum weight perfect matching [14]: it is the most commonly used, has better accuracy than union find, but also higher complexity
- Belief Propagation [19]: more complex than MWPM, it has seen more use for the LDPC codes as the MWPM cannot decode quantum error correction codes as complex
- Neural-network-based decoders [5, 15, 21, 22]: depending on the implementation, can be more or less complex, and with more or less accuracy

We decided to pick the last type of decoder, starting from the GNN implemented in [15], which can decode the surface code up to at least distance 7 with better accuracy compared to heuristic algorithms such as MWPM and UF [15], while still keeping the complexity limited. The comparison in terms of latency and accuracy between the software implementation of the GNN and a software implementation of MWPM [13] is shown in Fig. 1 and in Fig. 2.

5.3 GNN hardware implementation

The software implementation of the GNN has a latency in the order of hundreds of μs, therefore it is orders of magnitudes slower than the wanted threshold. A first hardware implementation has been done through the framework GNNBuilder [2], which can map a GNN composed of multiple type of layers to an FPGA. This first implementation shows good results, but it is still an order of magnitude slower compared to the threshold latency, therefore a custom design is required to achieve the 1 µs. The single-layer

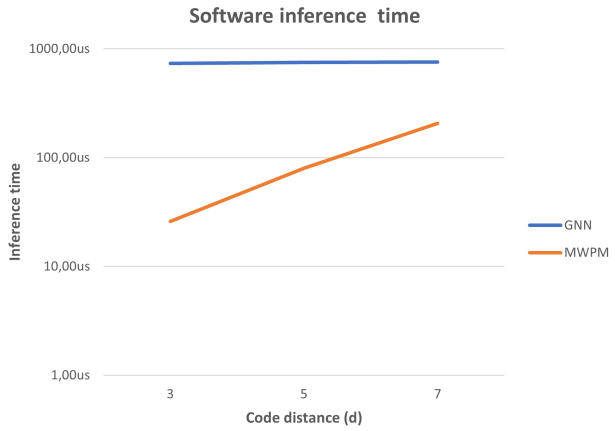

Figure 1: Comparison of the inference time between the software implementation of MWPM [13] and the software implementation of the GNN [15]

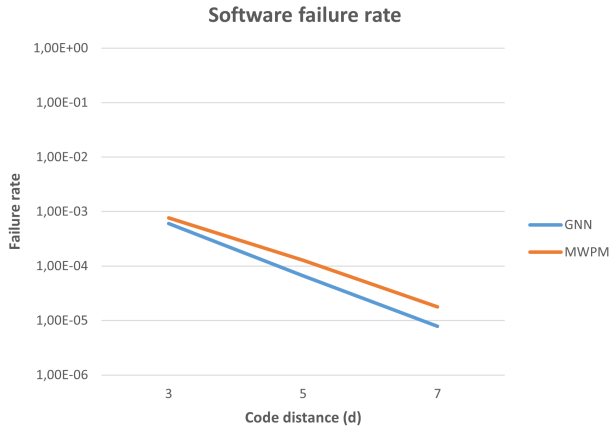

Figure 2: Comparison of the logical failure rate of the software implementation of MWPM [13] and the software implementation of the GNN [15]

execution has a latency of the order of 50 µs, as shown in Fig. 3, therefore we plan to find the best trade-off between hardware re-use and overall latency to achieve the wanted threshold latency.

6 Evaluation plan

The custom hardware implementation of the GNN will be evaluated with the same metrics as those used for the preliminary studies on the decoding algorithm and the GNNBuilder FPGA implementation. Its accuracy will be compared to the GNN software version and the MWPM software version, to validate that the achieved hardware is better in terms of accuracy compared to the state-of-art implementations. Then the latency decrease will be evaluated compared to the wanted threshold of 1 µs, to the software implementation of the GNN and to a hardware implementation of the GNN

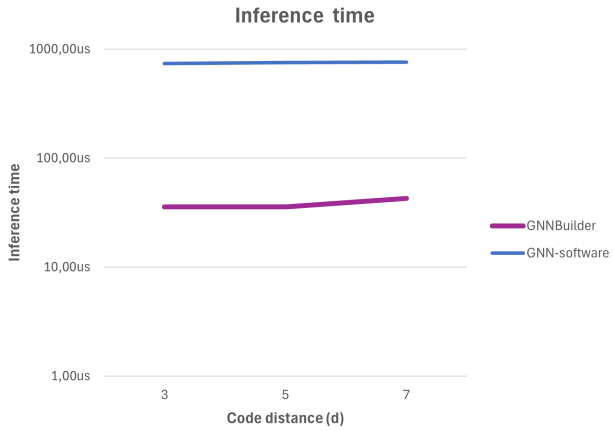

Figure 3: Comparison of the inference time of the software implementation of the GNN in blue [15] and the hardware implementation generated by GNNBuilder [2]

done with a GNN hardware-implementation generation framework, GNNBuilder [2]. This will eventually validate if the ad-hoc implementation can achieve better results than the state-of-art GNN hardware generation tool, and if a GNN decoder is the right path to achieving the most accurate real-time error correction

7 Conclusions and reflections

Quantum computer research is rapidly developing, with an evolving focus on achieving fault-tolerant quantum computers. Quantum error correction is one of the big challenges that needs to be solved before this is possible, and multiple different approaches are being researched to achieve this. While a lot of different quantum error correction codes are being evaluated, surface codes are still the most promising to be run in real-time. GNN decoders are promising approaches to achieve a better accuracy, which leads to a lower logical error rate and therefore requires a smaller number of physical qubits to achieve the same overall logical error rate, allowing for better scalability. The first step of my work was to implement the GNN with the GNNBuilder framework, which can fit the GNN in a single FPGA but it is not able to execute it within the wanted maximum latency. Therefore an ad-hoc hardware implementation is required to achieve the wanted performances, and will lead to a more accurate quantum error correction system. Future work will be based on these results, and if this approach is confirmed as promising as expected, GNN will continue to be the most fit decoders for the surface code. If this is not the case, a further exploration of error correction codes and decoders will be necessary to find the best set-up which will allow a scalable, accurate, quantum error correction device.

Acknowledgments

We acknowledge support from the Swedish Foundation for Strategic Research (grant number FUS21-0063).

References

[1] 2023. Suppressing quantum errors by scaling a surface code logical qubit. *Nature* 614, 7949 (2023), 676–681.

[2] Stefan Abi-Karam and Cong Hao. 2023. Gnnbuilder: An automated framework for generic graph neural network accelerator generation, simulation, and optimization. In *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 212–218.

[3] Narges Alavisamani, Suhas Vittal, Ramin Ayanzadeh, Poulami Das, and Moinuddin Qureshi. 2024. Promatch: Extending the Reach of Real-Time Quantum Error Correction with Adaptive Predecoding. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*. 818–833.

[4] Bela Bauer, Sergey Bravyi, Mario Motta, and Garnet Kin-Lic Chan. 2020. Quantum Algorithms for Quantum Chemistry and Quantum Materials Science. *Chemical Reviews* 120, 22 (2020), 12685–12717. doi:10.1021/acs.chemrev.9b00829 arXiv:https://doi.org/10.1021/acs.chemrev.9b00829

[5] Johannes Bausch, Andrew W Senior, Francisco JH Heras, Thomas Edlich, Alex Davies, Michael Newman, Cody Jones, Kevin Satzinger, Murphy Yuezhen Niu, Sam Blackwell, et al. 2023. Learning to decode the surface code with a recurrent, transformer-based neural network. *arXiv preprint arXiv:2310.05900* (2023).

[6] Nikolas P Breuckmann and Jens Niklas Eberhardt. 2021. Quantum low-density parity-check codes. *Prx Quantum* 2, 4 (2021), 040101.

[7] Poulami Das, Aditya Locharla, and Cody Jones. 2022. Lilliput: a lightweight low-latency lookup-table decoder for near-term quantum error correction. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 541–553.

[8] Nicolas Delfosse and Naomi H Nickerson. 2021. Almost-linear time decoding algorithm for topological codes. *Quantum* 5 (2021), 595.

[9] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. 2002. Topological quantum memory. *J. Math. Phys.* 43, 9 (2002), 4452–4505.

[10] Simon J Devitt, William J Munro, and Kae Nemoto. 2013. Quantum error correction for beginners. *Reports on Progress in Physics* 76, 7 (2013), 076001.

[11] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. 2012. Surface codes: Towards practical large-scale quantum computation. *Physical Review A—Atomic, Molecular, and Optical Physics* 86, 3 (2012), 032324.

[12] Jay M Gambetta, Jerry M Chow, and Matthias Steffen. 2017. Building logical qubits in a superconducting quantum computing system. *npj quantum information* 3, 1 (2017), 2.

[13] Oscar Higgott. 2022. Pymatching: A python package for decoding quantum codes with minimum-weight perfect matching. *ACM Transactions on Quantum Computing* 3, 3 (2022), 1–16.

[14] Vladimir Kolmogorov. 2009. Blossom V: a new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation* 1 (2009), 43–67.

[15] Moritz Lange, Pontus Havström, Basudha Srivastava, Valdemar Bergentall, Karl Hammar, Olivia Heuts, Evert van Nieuwenburg, and Mats Granath. 2023. Data-driven decoding of quantum error correcting codes using graph neural networks. *arXiv preprint arXiv:2307.01241* (2023).

[16] Namitha Liyanage, Yue Wu, Siona Tagare, and Lin Zhong. 2024. FPGA-based distributed union-find decoder for surface codes. *IEEE Transactions on Quantum Engineering* (2024).

[17] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan. 2020. Quantum computational chemistry. *Reviews of Modern Physics* 92 (Mar 2020), 015003. Issue 1. doi:10.1103/RevModPhys.92.015003

[18] Michael A. Nielsen and Isaac L. Chuang. 2023. *Quantum Computation and Quantum Information*. Cambridge University Press.

[19] Josias Old and Manuel Rispler. 2023. Generalized belief propagation algorithms for decoding of surface codes. *Quantum* 7 (2023), 1037.

[20] Peter W Shor. 1995. Scheme for reducing decoherence in quantum computer memory. *Physical review A* 52, 4 (1995), R2493.

[21] Boris M Varbanov, Marc Serra-Peralta, David Byfield, and Barbara M Terhal. 2025. Neural network decoder for near-term surface-code experiments. *Physical Review Research* 7, 1 (2025), 013029.

[22] Savvas Varsamopoulos, Koen Bertels, and Carmen G Almudever. 2020. Decoding surface code with a distributed neural network–based decoder. *Quantum Machine Intelligence* 2 (2020), 1–12.

[23] Suhas Vittal, Poulami Das, and Moinuddin Qureshi. 2023. Astrea: Accurate quantum error-decoding via practical minimum-weight perfect-matching. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*. 1–16.