Vol.2022-QS-7 No.16
2022/10/28

WIT-Greedy: Hardware System Design of Weighted ITerative Greedy Decoder for Surface Code

廖　望[1,a)]　鈴木 泰成[2,4,b)]　谷本 輝夫[3,4,c)]　上野 洋典[1,d)]　徳永 裕己[2,e)]

Wang LIAO[1,a)]　Yasunari Suzuki[2,4,b)]　Teruo Tanimoto[3,4,c)]　Yosuke Ueno[1,d)]
Yuuki Tokunaga[2,e)]

Abstract: Large error rates of quantum bits (qubits) are one of the main difficulties in the development of quantum computing. Performing quantum error correction (QEC) with surface codes is considered the most promising approach to reduce the error rates of qubits effectively. To perform error correction, we need an error-decoding unit, which estimates errors in the noisy physical qubits repetitively, to create a robust logical qubit. While complicated graph-matching problems must be solved within a strict time restriction for the error decoding, several hardware implementations that satisfy the restriction at a large code distance have been proposed.
However, the existing decoder designs are still challenging in reducing the logical error rate. This is because they assume that the error rates of physical qubits are uniform while they have large variations in practice. According to our numerical simulation based on the most advanced quantum chip, neglecting the non-uniform error properties of a real quantum chip in the decoding process induces significant degradation of the logical error rate and spoils the benefit of QEC. To take the non-uniformity into account, decoders need to solve matching problems on a weighted graph, but they are difficult to solve using the existing designs without exceeding the time limit of decoding. Therefore, a decoder that can treat both the non-uniform physical error rates and the large surface code is strongly demanded.
In this paper, we propose a hardware design of decoding units for the surface code that can treat the non-identical error properties with small latency at a large code distance. The key idea of our design is 1) constructing a look-up table for calculating the shortest paths between nodes in a weighted graph and 2) enabling parallel processing during decoding. The implementation results in field programmable gate array (FPGA) indicate that our design scales up to code distance 11 within a microsecond-level delay, which is comparable to the existing state-of-the-art designs, while our design can treat non-identical errors.

1. Introduction

Quantum computing is an emerging computing paradigm for potential practical use since it enables us to use properties of quantum matters, such as superpositions and entanglements, for faster processing in several applications [1], [2], [3]. On the other hand, since the lifetime of computational elements, called a quantum bit (qubit), is limited up to hundreds of microseconds [4] due to heating and calibration noise [5], the reliability of current quantum devices is not enough for their practical applications [6]. To cope with this problem, we can construct a logical qubit with noisy physical qubits using quantum error-correcting (QEC) codes. By checking the error parities with a sufficiently faster period than the lifetime of qubits repetitively and by increasing the code distance, i.e., using more physical qubits in the code, the occurred errors can be reliably estimated and corrected, and error rates of logical qubits can be suppressed to

[1] 東京大学, the University of Tokyo
[2] NTT コンピュータ& データサイエンス研究所,
NTT Computer and Data Science Laboratories
[3] 九州大学, Kyushu University
[4] JST さきがけ, JST, PRESTO
[a)] liao@qi.t.u-tokyo.ac.jp
[b)] yasunari.suzuki.gz@hco.ntt.co.jp
[c)] tteruo@kyudai.jp
[d)] ueno@hal.ipc.i.u-tokyo.ac.jp
[e)] yuuki.tokunaga.bf@hco.ntt.co.jp

an arbitrarily small value [7], [8]. This error-estimation task is called a decoding process. Currently, surface codes [9] are believed to be the most promising QEC codes. While the surface codes require decoders to solve complicated graph-matching problems within a strict time restriction, several hardware implementations that satisfy the restriction at a large code distance have been proposed.

The existing state-of-the-art decoders are, however, still challenging in reducing the logical error rate in a real quantum chip since the physical error rates are not uniform in practice due to the manufacturing variation of the chip. According to our simulation results based on the noise model expected from the most advanced quantum chips [4], neglecting the non-uniform error properties induces significant degradation of the logical error rate and spoils the benefit of QEC. Unfortunately, the current state-of-the-art designs of scalable decoding units assume uniform physical error rates [10], [11], [12], and they cannot consider the error variations without exceeding the time budget of decoding. This is because we needs to repetitively find the shortest paths between nodes in a weighted graph to estimate the most probable error patterns for qubits with non-uniform error rates. Therefore, a decoder that can treat both the non-uniform physical error rates and the large surface code is strongly demanded.

In this paper, we propose a hardware design of decoding units, which we named Weight-aware ITerative Greedy (WIT Greedy) decoder, that can treat the non-identical error properties for surface codes with high scalability. The key idea of our design is 1) constructing a look-up table for quickly calculating the shortest paths between nodes with small latency and 2) enabling parallel processing during decoding. We evaluate the latency and overhead of the proposed decoding units using the implementation in the FPGA and show that our design scales up to code distance 11 within a microsecond-level delay. This performance is comparable to the existing state-of-the-art designs, while our decoders can treat non-identical errors properties.

2. Background and related works

2.1 Background of surface code decoding

The topological structure of physical qubits in a logical qubit encoded by the surface code is shown in Fig. 1. In the logical qubit, data qubits are used for computing, and the ancillary qubits are used for detecting Pauli-X or Z errors in the nearest neighboring data qubits. The size of

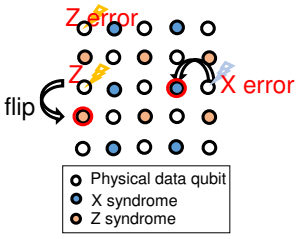

Fig. 1 Topological structure of surface code. This example shows the case of $d = 3$.

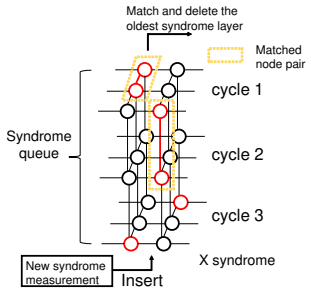

Fig. 2 Decoding process equals to matching problem

a surface-code lattice d corresponds to the encoding level called a code distance. The error detection is performed by repetitive parity checks, and the measurement value of ancillary qubits will flip if an odd number of errors occur simultaneously within the detection range. The parity measured by ancillary qubits is called a syndrome value, and the period for all the parity checks is called a code cycle.

At each code cycle, a set of X- and Z-syndrome measurements will be performed. Since these processes are symmetric, we only explain X-syndrome measurements. Figure 2 shows the decoding process of the X-syndrome queue storing the syndrome values shaped as a three-dimensional lattice. Each node of the lattice in Fig. 2 is a syndrome measurement value, and each layer stands for the set of measurement values in every code cycle. The red node represents the syndrome value of the odd parity, which we call an active syndrome node. The values will be recorded in a unit called a syndrome queue for decoding. The edges in the syndrome layer and the vertical edges across the layers represent Pauli-X errors in the data and ancillary qubit, respectively. We can represent an arbitrary error pattern with a set of edges connecting the active syndrome nodes, and the estimation of one of the most probable error patterns can be rephrased as the task of finding a minimum-weight perfect matching (MWPM) on a fully-connected graph. The nodes of the graph correspond to active syndromes, and the weight of its edges is the length of the shortest path between two

active nodes on the lattice, where the weight of lattice edges is calculated from the error rates of the corresponding qubit. Due to measurement errors, decoding is usually performed in a d-cycle window to gain the benefit of a temporal code distance.

To sequentially estimate the errors, at each cycle, the matching for the oldest layer, i.e., the oldest code cycle, will be determined, and the record of the matched active syndrome nodes will be deleted from the syndrome queue. The latency for decoding errors in a code cycle should be sufficiently shorter than the lifetime of qubits. Since the lifetime of superconducting qubits, which are known to be one of the state-of-the-art qubits, is currently up to hundreds of micro-seconds [4], the time restriction of the decoding units to process each cycle should be within about a micro-second [11], [13].

2.2 Related works

Edmonds' blossom algorithm [14] can provide the exact solution of an MWPM problem, and the decoder implementing the algorithm is called an MWPM decoder. Several MWPM decoders and their variants have been proposed theoretically with software implementation. However, the software implementation of a surface code decoder typically processes in milliseconds [16] and cannot satisfy the 1 μs time restriction. Moreover, only embedded systems are feasible to be applied near the quantum chip for a decoding task due to the transmission delay and the wiring complexity. Thus, fast hardware implementations have been intensively studied recently. Table 1 summarizes the existing proposals on various hardware implementations. We compare these studies with our proposal from two perspectives.

Scalability. There are two approaches to solving the MWPM problem satisfying the strict time restriction. The first approach is a data-driven model of the accurate MWPM solution by Poulami *et al.* [15] and Overwater *et al.* [13]. In these studies, they construct a look-up table (LUT) in which the entry of address is the syndrome value vector, and the data of each address is the matching results. Therefore, the matching problem can be solved within the latency of memory access. The dominant factor limiting its scalability is the LUT size. Since the entry size of the table grows exponentially by $2^{d\times d\times d}$ with code distance d, the available code distance is limited up to $d = 5$ even with the compression or the use of machine learning, while larger code distances are required for practical applications.

The second approach is using the fast approximated algorithms. Union-find-based and modified greedy algorithms have been used as approximated algorithms, and they also show a rapid decrease in the logical error rate as a code distance increases. Compared with the union-find-based approaches, those based on greedy algorithms are more lightweight and easy to scale up. The previous studies on greedy-based decoders mainly focus on the implementation with single-flux-quantum (SFQ) circuits, which has a limited logic resource but can work at the gigahertz-level clock frequency in a cryogenic environment [11], [12]. In contrast, the current matured embedded devices such as an FPGA have far more logic resources but slow working clocks compared with the SFQ devices and, thus, require the adjustment in design.

Weighted matching. We define weighted matching as the task of error estimation with the consideration of non-identical error rates, i.e., the task of solving the MWPM problem for a weighted graph. Similar to the variation in the threshold voltage of transistors in classical VLSI circuits, the error rates of qubits are not uniform due to manufacturing variation. The threshold-voltage variation of transistors can be solved by the worst-case design principle. However, the variation in error rates of qubits cannot be worked around in the same manner because only variation itself affects the probability of possible error patterns.

In previous studies, the data-driven MWPM decoder can implement weighted matching since there is no obvious extra cost while creating the LUT of matching results. However, the scalability of these approaches is not enough for practical applications. On the other hand, the scalable implementations based on approximated algorithms ignored the variations in error rates. In this paper, we propose a scalable design of greedy-based decoders that can take the non-uniform error properties into account.

3. Effect of weighted decoding

3.1 Greedy decoding of surface code

The MWPM decoder calculates the exact solution to MWPM problems, but it is difficult to satisfy the time restriction of a code cycle. We use a greedy algorithm [11], [12] as an alternative approach since it shows scalability and small latency. While there are several variants of greedy algorithms, an algorithm that matches pairs of nodes in ascending order with respect to their path weight is the most promising [12], which we call the *iterative greedy algorithm* throughout this paper. Although the

Table 1 Comparison of hardware implementation of surface code decoders.

Decoding algorithm	d_{max}	Latency	Measurement error?	Decoding window length of code cycle	Target device	Scalability	Weighted matching?
Look-up table [15]	5	42 ns	Y	3	FPGA	Hard	Y
Neural network [13]	5	194 ns	N	1	FPGA	Hard	Y
Union-find [10]	11	325 ns	Y	11	FPGA	Easy	N
Greedy [11]	9	162.72 ps	N	1	SFQ	Easy	N
Greedy [12]	13	215 ps	Y	3	SFQ	Easy	N
Greedy (Our proposal)	11	370 ns	Y	11	FPGA	Easy	Y

solution from the iterative greedy algorithm may be a local optimum, the matching problem can be solved much faster than the MWPM decoder. In this paper, we will propose a weighted variant of the iterative greedy algorithm and its low-overhead implementation, which can treat the variations of errors, i.e., solve the MWPM problem on weighted graphs.

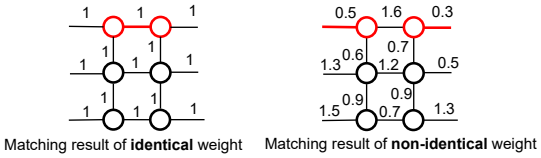

Fig. 3 Comparison between matching result by identical and non-identical weights.

3.2 Noise modeling of error variations

While most of the existing hardware implementation assumes that the error rates of qubits are uniform, the error properties of a state-of-the-art quantum chip, IBM Eagle [4], indicate that the lifetimes of qubits have large variances. We found that the distribution of two types of lifetimes known as T1 and T2 can be well fitted with a normal distribution, of which the average and deviation are 97.0 ± 24.5 μs for T1, and 93.6 ± 47.3 μs for T2. Throughout this paper, we assume that T1 and T2 of each qubit are independently sampled from the fitted normal distribution. Note that if unphysical lifetimes are sampled, we truncated them to physical values (e.g. a negative lifetime is truncated to small positive value).

We expect the mean value of the lifetimes will be improved in the future. Along with that, there are two possible assumptions on the variance of the qubit lifetime. In an optimistic scenario, the deviation is kept to be constant, and the error variations become relatively negligible in the future. In a pessimistic scenario, the ratio of deviation to the mean value is kept to be constant. While the actual scaling of error variations depends on the type of device improvement, we expect the pessimistic one is possible when unexpected two-level systems cause the decay [17], and the lifetime improvement comes from the reduction of their density.

3.3 Benefit of weighted decoding

We show a working example of how the consideration of weighted matching improves the error-estimation accuracy. Figure 3 show the matching results with and without considering the variation of error rates, i.e., the non-uniform weights of lattice edges. If we correct the qubits using the matching pattern in accordance with identical weights on the left side of the figure but the actual pattern is on the right side of the figure, a non-trivial loop, i.e., an unexpected logical Pauli-X error, will occur inside the logical qubits. In contrast, we can see that such logical error can be avoided by the estimation of non-identical weights.

To quantify the degradation in logical error rates, we performed the Monte Carlo simulation of one logical qubit holding its state for d cycles, where d equals the code distance of the logical qubit. Figure 4 shows the logical error rate as the function of the improvement ratio of the lifetimes. Here, the physical Pauli-X and Z error rates are calculated from T1 and T2 sampled from the fitted distribution of the IBM Q Eagle chip, assuming the error map is Pauli twirled[18]. When we swept the values of lifetimes, we assumed that the standard deviation of lifetimes scales linearly to the improvement ratio. We sampled 3×10^2 error-variation patterns at each improvement ratio for each code distance, and a logical error rate is calculated with 10^4 samples for each sampled error-variation pattern. The marker stands for the logical error rate averaged for error variation patterns, and the error bar shows a 95 % range of their logical error rates. Note that we slightly shift the positions of markers for visibility. From this figure, we can observe that the average logical error rate with unweighted decoders slowly drops with the im-

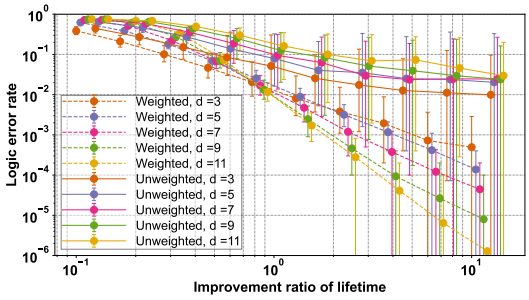

Fig. 4 Logical error rates with identical and non-identical weights in terms of the improvement ratio of qubit lifetimes.

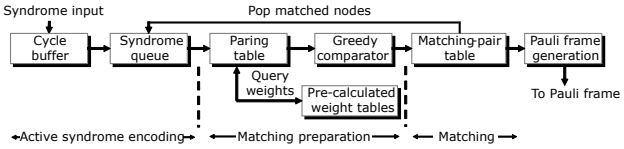

Fig. 5 Overall framework of the decoder system.

provement of lifetimes and shows no improvement by code scaling up. On the other hand, a clear drop in logical error rates is observed with the weighted decoding. Therefore, the algorithm considering non-identical weights enables significant improvement in logical error rates and is necessary for ensuring the error-rate reduction of QEC, and decoders for weighted matching are vital for realizing reliable quantum computation.

4. Design of weighted decoder system

Figure 5 shows an overview of our framework of the decoder system. The framework is designed for either Pauli-X and Z error estimation, and we only show the design for Z errors. The input of the system is a set of syndrome values, which is shaped as $d \times (d-1)$. After encoding active syndrome nodes and pushing them to the syndrome queue, the active nodes of the oldest code cycle are matched using the iterative greedy algorithm. The information of the matched pairs of nodes is sent to the unit called Pauli frame [19] for error correction and also sent to the syndrome queue to erase the matched active nodes.

In order to solve weighted matching problems within a code cycle, we designed the decoder system based on two tables: the pairing table and the pre-calculated weight table. The pairing table consists of registers that store all the pairing patterns of active nodes to enable parallel processing in matching. The pre-calculated weight table stores the cost of the shortest paths between all the lattice nodes in a succinct form using embedded memories. We will introduce the detailed designs in Section 4.2.

4.1 Active syndrome encoding

This part receives the syndrome measurement values as input and determines the location of the active syndrome nodes by encoding them with position information for the matching process. The position is represented by three integers (x, y, z), where x, y, and z stand for the column index, row index, and code cycle number, respectively. We divide the encoding process of a syndrome layer at each code cycle into that for a row syndrome-node vector. Figure 6 shows the example design of the row encoding unit for surface codes of $d = 5$ within one clock.

After encoding the active syndrome nodes, we merge them incrementally into the upper level, as shown in Fig. 7. In our design, three levels of buffers of rows, cycles, and the overall syndrome queue, are utilized to prevent the overflow. The syndrome queue controls the data flow of merging the new active nodes, output node list for the decoding process and erasing the matched node.

4.2 Matching preparation with weight table

While we can find the shortest path between two nodes directly from their positions on a lattice as a Manhattan distance under the assumption of the identical weighting [10], [12], we cannot use this strategy to support the weighted matching. Since the process of finding the shortest path is time-consuming and repeated, we prepare the table that can be queried the least weights between an arbitrary pair of nodes in advance, as shown in Fig. 5. Since the error probabilities of each qubit is expected to be constant during the single computing process, the weights of pairs between $(x, y, z) - (x', y', z')$ and $(x, y, z + c) - (x', y', z' + c)$ are assumed to be the same for any c of the Z-direction. Therefore, the entry of the address of the weight table uses the value for the relative position for the Z-direction. With this technique, the size of entry can be suppressed to the least successive unique node pair combination in the order of d^5. The data width of each entry is the bit width for the pair weight and a single bit of matching direction for Pauli frame generation. The actual size of the weight table will be discussed in Section 5. In practical parameter regions, the table size is sufficiently small, and we can build several tables using BRAMs (Block RAM) in FPGA to enable parallel read-in operation.

Figure 8 shows the procedure to update the pairing table by querying the weight tables for their weights of the shortest path. In this procedure, we generate the address information of each node pair in accordance with the co-

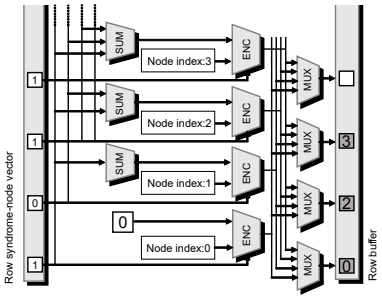

Fig. 6 Design of encoding unit for active syndrome.

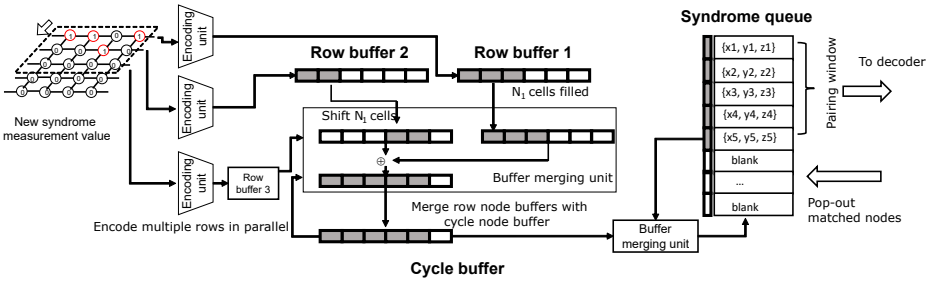

Fig. 7 Data flow in active syndrome encoding module.

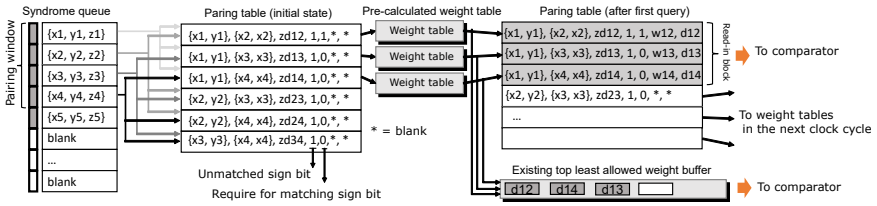

Fig. 8 Data flow in matching preparation.

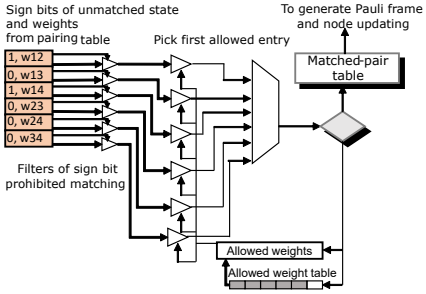

Fig. 9 Parallel iter. greedy comparator.

ordinate of the active node, as shown on the left side of Fig 8. During the process of querying the weight tables, we record several top minimum values among the read-in weights to the allowed weight table for accelerating the matching process discussed in the next section.

4.3 Matching using iterative greedy

Figure 9 shows the comparator design of the iterative greedy algorithm. The comparator takes the weights of the node pairs as input. At the beginning of the matching process, the comparator will initialize a counter with the range from 0 to $2^{w_{\text{weight}}} - 1$, which represents the current allowed weight in matching. During each clock cycle, the comparator selects the first node pair of the allowed weight from the unmatched pairs in the pairing table. In return, all the pairs related to the matched nodes will be marked as matched, and the next matching cycle begins. The matching process will end if the nodes of the oldest cycle are all matched.

The best performance of the greedy core is one pairing per clock cycle. If the allowed weight does not hit any unmatched pair, a penalty of one clock cycle will be imposed to increment the weight counter by one. Since the non-identical weights of all the possible error patterns may be sparse in some regions, the time penalty probably occurs successively in such a region of values. To address this issue, the allowed weight table mentioned in the last section is used for loading the record values to the weight counter in ascending order. When the pointer of the allow weight table reaches the last element, the weight counter will return to being incremented by one at each cycle.

4.4 Pauli frame generation

We extract the correction information from the matched pairs in each cycle and store them in a unit called a Pauli frame [19]. The Pauli frame generation will work in parallel with the iterative greedy comparator since every matched pair contains the independent information for the correction of qubit errors. Note that only the matched pair related to the active nodes in the oldest code cycle would be used for the Pauli frame generation. Then, the matched pair will feed back to the syndrome queue to erase relevant active nodes.

5. Overhead with scalability

Scalability is vital for the design of QEC decoders since we need to increase the code distance to reduce the logical error rate. In this session, we discuss the area and time overhead of our proposal in terms of the scale up of the surface codes.

5.1 Area overhead

The area overhead mainly originates from the weight table and resource utilization of logic, and it grows as the code distance increases. The entry of the weight table consists of an integer and real-value parts. While the size

Table 2 Overhead of weight table size with code scaling up

d	Address size	Data bit width	Table size (bit)
5	1.0 k	7	7.3 k
7	6.3 k	7	44.2 k
9	23.6 k	8	189.2 k
11	67.1 k	8	0.5 M

of the integer part is fixed as ceil($\log_2 d$), the size of real values for representing weights can be tuned with its precision. While rough precision reduces the size of the table, it may sacrifice the logical error rates. To evaluate the effect of the truncation, we performed the Monte Carlo simulation with 2×10^5 samples and evaluated the degradation of logical error rates for several bit widths. As a result, we found that the decimal-bit width of three is the roughest precision showing the performance comparable to that without the truncation. Thus, we created the weight table with the fixed point with three decimal bits. The size of the weight tables for several code distances is shown in Table 2. We assume ten parallel read-in operations, thus at least five times the BRAM resource consumption with dual ports.

The resource utilization of logic is mainly dominated by the active node encoding and greedy comparator. For the encoding part, we mainly consider the total buffer size of the syndrome queue. The buffer size of the syndrome queue should be large enough to prevent the overflow of active syndrome nodes and thus determined by the code distance d and the error rates of the physical qubits. We estimate the sufficient buffer size to suppress the overflow probability lower than 10^{-15}. Here, we assumed the occurrence probability of active syndrome nodes is equivalent to six times of physical error rate for simplicity, since each syndrome node has six connected edges and each of them flips with the physical error rate. For greedy comparator, its logic resource utilization is mainly determined by the pairing window size for decoding logic in Fig. 8. While the small pairing window reduces the overhead, it may result in local-optimal estimation since the pairing window sometimes cannot include all the active nodes. We choose the size of the pairing window so that the probability with which the window cannot contain all the active nodes in recent d cycles is below 10^{-3}. Based on the above strategies, we evaluate the resource utilization of decoding units using the implementation of Xilinx Vivado in Out-Of-Content mode with the default optimization options and the target clock frequency of 100 MHz. The results shown in Table 3 indicate that the current system design

Table 3 Resource utilization overhead for decoding logic

d	Pairing window size	Syndrome queue size	LUT	FF	DSP	BRAM
5	9	15	5.5k	1.6k	10	2.5
7	12	21	11.0k	2.4k	10	10
9	15	29	16.2k	3.4k	10	35
11	20	39	22.6k	4.8k	10	126

Table 4 Decoding cycle in RTL test bench.

d	Decoding cylces			Buffer overflow times
	min	average	max	
5	18	19	71	0
7	19	22	97	0
9	24	29	105	0
11	33	37	103	0

can treat surface codes with $d = 11$ using the mid-class FPGAs such as the Xilinx Zynq Ultrascale+ series.

5.2 Decoding latency

Latency overhead is another vital factor in the evaluation of the implementation. As our target clock is 100 MHz, all the steps of the decoding process required per code cycle must be finished within 100 clock cycles. Since the physical errors occur randomly, we cannot give a static formula to calculate the decoding delay. Instead, we performed the Monte Carlo simulation with the RTL design of the proposed decoder. We run the simulation for 2×10^4 code cycles for each condition. For simplicity, we assumed that each syndrome node becomes active with a six-times larger probability of the identical qubit error rate of 0.1 % in the RTL simulation.

Table 4 shows the average decoding cycle for each condition. We find that the configuration of 0.1% physical error rate and $d = 11$ is reasonable to finish within 1 μs time restriction on average. Since the syndrome queue buffer size still has a large timing slack, our framework can fit a higher physical error rate with performance degradation by increasing the queue size.

6. Conclusion

In this paper, we showed that error-decoding procedure for surface code with taking non-identical error properties into account has a significant improvement in the reduction of logical error rates. We proposed and implemented a hardware design of a weighted decoder system in an FPGA. For satisfying strict time restriction, we used the weight table to access the most probable error pattern for syndrome node pairs. The numerical evaluation based on

our implementation shows that our design scales up to code distance 11 within a microsecond-level delay.

Acknowledgements

This work was supported by JST Moonshot R&D Grant Number JPMJMS2061 and JPMJMS2067, JST PRESTO Grant Number JPMJPR1916 and JPMJPR2015. This work was also supported by JSPS KAKENHI Grant Number JP21J10882 and JP21K17721.

References

[1] Shor, P. W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, *SIAM Review*, Vol. 41, No. 2, pp. 303–332 (online), DOI: 10.1137/s0036144598347011 (1999).

[2] Grover, L. K.: Quantum Mechanics Helps in Searching for a Needle in a Haystack, *Physical Review Letters*, Vol. 79, No. 2, pp. 325–328 (online), DOI: 10.1103/physrevlett.79.325 (1997).

[3] Babbush, R., Gidney, C., Berry, D. W., Wiebe, N., McClean, J., Paler, A., Fowler, A. and Neven, H.: Encoding Electronic Spectra in Quantum Circuits with Linear T Complexity, *Physical Review X*, Vol. 8, No. 4 (online), DOI: 10.1103/physrevx.8.041015 (2018).

[4] Chow, J., Dial, O. and Gambetta, J.: IBM Quantum breaks the 100-qubit processor barrier, *IBM Research Blog* (2021).

[5] Krantz, P., Kjaergaard, M., Yan, F., Orlando, T. P., Gustavsson, S. and Oliver, W. D.: A quantum engineer's guide to superconducting qubits, *Applied Physics Reviews*, Vol. 6, No. 2, p. 021318 (online), DOI: 10.1063/1.5089550 (2019).

[6] Ladd, T. D., Jelezko, F., Laflamme, R., Nakamura, Y., Monroe, C. and O'Brien, J. L.: Quantum computers, *Nature*, Vol. 464, No. 7285, pp. 45–53 (online), DOI: 10.1038/nature08812 (2010).

[7] Shor, P. W.: Scheme for reducing decoherence in quantum computer memory, *Phys. Rev. A*, Vol. 52, pp. R2493–R2496 (online), DOI: 10.1103/PhysRevA.52.R2493 (1995).

[8] Gottesman, D.: Stabilizer Codes and Quantum Error Correction (1997).

[9] Fowler, A. G., Mariantoni, M., Martinis, J. M. and Cleland, A. N.: Surface codes: Towards practical large-scale quantum computation, *Physical Review A*, Vol. 86, No. 3 (online), DOI: 10.1103/physreva.86.032324 (2012).

[10] Das, P., Pattison, C. A., Manne, S., Carmean, D. M., Svore, K. M., Qureshi, M. and Delfosse, N.: AFS: Accurate, Fast, and Scalable Error-Decoding for Fault-Tolerant Quantum Computers, *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 259–273 (online), DOI: 10.1109/HPCA53966.2022.00027 (2022).

[11] Holmes, A., Jokar, M. R., Pasandi, G., Ding, Y., Pedram, M. and Chong, F. T.: NISQ+: Boosting quantum computing power by approximating quantum error correction, *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pp. 556–569 (online), DOI: 10.1109/ISCA45697.2020.00053 (2020).

[12] Ueno, Y., Kondo, M., Tanaka, M., Suzuki, Y. and Tabuchi, Y.: QECOOL: On-Line Quantum Error Correction with a Superconducting Decoder for Surface Code, *2021 58th ACM/IEEE Design Automation Conference (DAC)*, IEEE, (online), DOI: 10.1109/dac18074.2021.9586326 (2021).

[13] Overwater, R., Babaie, M. and Sebastiano, F.: Neural-Network Decoders for Quantum Error Correction using Surface Codes:A Space Exploration of the Hardware Cost-Performance Trade-Offs (2022).

[14] Edmonds, J.: Paths, Trees, and Flowers, *Canadian Journal of Mathematics*, Vol. 17, pp. 449–467 (online), DOI: 10.4153/cjm-1965-045-4 (1965).

[15] Das, P., Locharla, A. and Jones, C.: LILLIPUT: A Lightweight Low-Latency Lookup-Table Decoder for near-Term Quantum Error Correction, *Proceedings of the 27th ASPLOS*, ASPLOS 2022, New York, NY, USA, ACM, p. 541–553 (online), DOI: 10.1145/3503222.3507707 (2022).

[16] Higgott, O.: PyMatching: A Python package for decoding quantum codes with minimum-weight perfect matching, *arXiv preprint arXiv:2105.13082* (2021).

[17] Lisenfeld, J., Bilmes, A., Megrant, A., Barends, R., Kelly, J., Klimov, P., Weiss, G., Martinis, J. M. and Ustinov, A. V.: Electric field spectroscopy of material defects in transmon qubits, *npj Quantum Information*, Vol. 5, No. 1 (online), DOI: 10.1038/s41534-019-0224-1 (2019).

[18] iOlius, A. d., Martinez, J. E., Fuentes, P., Crespo, P. M. and Garcia-Frias, J.: Performance of surface codes in realistic quantum hardware (2022).

[19] Riesebos, L., Fu, X., Varsamopoulos, S., Almudever, C. G. and Bertels, K.: Pauli Frames for Quantum Computer Architectures, *Proceedings of the 54th Annual Design Automation Conference (DAC)*, ACM, (online), DOI: 10.1145/3061639.3062300 (2017).