QECOOL: On-Line Quantum Error Correction with a Superconducting Decoder for Surface Code

Yosuke Ueno¹, Masaaki Kondo¹, Masamitsu Tanaka², Yasunari Suzuki³, and Yutaka Tabuchi⁴, (a)

¹Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan
²RIKEN Center for Computational Science, Hyogo, Japan
³Graduate School of Engineering, Nagoya University, Aichi, Japan
⁴NTT Secure Platform Laboratories, Tokyo, Japan

Presently the author is with RIKEN Center for Emergent Matter Science.

Abstract—Due to the low error tolerance of a qubit, detecting and correcting errors on it is essential for fault-tolerant quantum computing. Surface code (SC) associated with its decoding algorithm is one of the most promising quantum error correction (QEC) methods. QEC needs to be very power-efficient since the power budget is limited inside of a dilution refrigerator for superconducting qubits by which one of the most successful quantum computers (QCs) is built. In this paper, we propose an online-QEC algorithm and its hardware implementation with SFQ-based superconducting digital circuits. We design a key building block of the proposed hardware with an SFQ cell library and evaluate it by the SPICE-level simulation. Each logic element is composed of about 3000 Josephson junctions and power consumption is about 2.78 µW when operating with 2 GHz clock frequency which meets the required decoding speed. Our decoder is simulated on a quantum error simulator for code distances 5 to 13 and achieves a 1.0% accuracy threshold.

Index Terms—Quantum Error Correction, SFQ logic

I. INTRODUCTION

Quantum computers (QCs) are becoming an attractive computing paradigm, as the number of implementable qubits increases. A QC with the largest number of qubits today has 53 qubits [1] while real-world problems typically need more qubits. For example, a 256-bit RSA cipher with Shor’s algorithm [18] requires about thousands of qubits. One of the challenges towards a rapid increase in the number of qubits is its high fragility of the quantum state due to decoherence and other noises, which necessitates error tolerance mechanisms. A topological error-correcting code and associated quantum error correction (QEC) mechanism are widely studied to detect/correct errors whose error rate is equal to or even higher than that of informational qubits. Ancilla bits should be measured multiple times, and stacking each result temporarily creates a 3-D SC lattice, which can be decoded by ordinary decoding algorithms for the 2-D plane with slight modifications. However, computational and circuit complexity greatly increases.

In this paper, we propose a power-efficient Quantum Error Correction by On-Line decoding algorithm (QECOOL) for 3-D lattice and its hardware implementation using superconducting digital circuits with single flux quantum (SFQ). Most of the prior decoders for 3-D matching problems or direct extension of 2-D algorithms require a completed 3-D SC, which means the decoding process is performed after all the measurements are done for one QEC step. We refer to this as batch-QEC. Instead of waiting for all the measurements, QECOOL starts a QEC process for every obtained SC plane with several consecutive accumulated SC planes. We call this online-QEC. Since online-QEC can inherently achieve a shorter QEC cycle than batch-QEC, the number of qubit errors on a single decoding process is expected to be small, leading to higher error correction performance. QECOOL is implemented with a specialized hardware logic with a spike signal-based matching architecture implemented by SFQ logic to realize the online-QEC with ultra-low-power and low-latency. Since the semiconductor process technology of SFQ is not scalable as that of CMOS due to the necessity of constructing superconducting rings, a large number of memory blocks are not implementable. QECOOL adopts a distributed architecture approach and area-efficient design with small registers.

The contributions of this paper are summarized as follows:

- We propose QECOOL, an on-line decoding algorithm for SC on a 3-D lattice that has significantly lower complexity than the standard MWPM and previously proposed decoders at the cost of slight degradation of error correction performance.
- We design a power- and area-efficient SFQ decoder based on the above algorithm.
- We evaluate the power consumption and area of the design with a SPICE-based circuit simulator and show that our implementation can work at the 4-K layer with around 2,500 logical qubits.
- We evaluate the error correction performance of QECOOL with a quantum error simulator.

(a) Presently the author is with RIKEN Center for Emergent Matter Science.
A practical architecture for the union-find decoder is first proposed by Das et al. [2]. In their work, the fully pipelined hardware implementation contributed to the higher speed of the decoding process.

Using cryogenic computing as peripherals of QCs to perform controls, including QEC, has been actively studied [11], [19], [20]. Holmes et al. [11] designed a novel algorithm with SFQ to find quantum errors online for near-term quantum systems. In their work, multiple pairs of two flipped ancilla bits can be found in parallel to reduce the latency of the decoding process. They implemented a mechanism that makes “agreement” among ancilla bits to find pairs. Our base decoder algorithm is inspired by their mechanism but differs in the fact that QECool does not need the agreement mechanism to simplify the design. Moreover, none of the prior work addresses online-QEC on 3-D lattice for faster QEC cycle.

A. Spike-based on-line quantum error correction

We first describe how to decode a SC on 2-D structures in QECool, which is inspired by the greedy algorithm of minimum-weight perfect matching problems [5]. The purpose of the decoder is to find pairs of erroneous ancilla bit locations such that the total Manhattan distance of all the pairs is minimized. We introduce a module named Unit associated with each ancilla bit. Each Unit is connected to neighboring four Units forming a 2-D grid structure as same as X or Z stabilizers. We also introduce a Controller module to orchestrate all the Units in a logical qubit. QECool algorithm is shown in Algorithm 1. Though the algorithm shows online-QEC on a 3-D lattice, it can be considered as an algorithm for a 2-D plane if $N_{\text{depth}}$, the number of depth in the vertical direction (V-direction), and $h_{iv}$, the maximum length or threshold in the V-direction to search for matchings, are set to 1 and -1, respectively. Each Unit has a unique row and column IDs originated by the Unit at the top-left corner. When each ancilla bit is measured, its value (0 and 1 indicate consistent and inconsistent for connected qubits, respectively) is stored into Reg on the associated Unit (MeasureEachUnit in Algorithm 1). In a QEC phase, the Controller assigns a Token to each Unit from the top-left (northwestern) corner. The decoding process shown in the “RestartUnit” procedure of Algorithm 1 is summarized as follows. Note that this procedure is supposed to be done in parallel in all the Units.

1) If a Unit gets the Token, it checks its own Reg. If it is 1, the Unit becomes the sink Unit and requests all the other Units to send a Spike toward the sink if their Reg is also 1, and then waits for the first Spike to come. If not, the Token is passed to the next Unit through the Controller.

2) When the Spike request comes to a non-sink Unit, it checks its own Reg. If Reg is 1, it initiates a Spike toward the sink Unit (Fig. 2(a)). The direction to send is determined locally by comparing its row ID and currentRow set by the Controller, and also FlagToken which indicates whether the Token is already passed in the past (see SPIKE procedure in Algorithm 1).

3) For a non-sink Unit with Reg value is 0, it may get a Spike from one of its neighboring Units. In this case, it passes the Spike to one of the other neighboring Units so that the Spike reaches the sink Unit. The direction to pass is determined in the same way as Step 2). It also calculates and saves the relative direction of the coming Spike by rotating 180 degrees of the incoming port’s direction. This direction is used to generate an error syndrome in the following Step 4).

4) If the first Spike comes to the sink Unit, it generates a correct signal to an informational qubit on the coming Spike direction. It also sends Syndrome signal to that direction to correct errors.
Algorithm 1 Spike-based on-line QEC for 3-D Surface code

```
1: MeasureEachUnit:
2: m = 0
3: while true do
4: A = checkAncilla()
5: if m == 0 then
6: Reg[0] = A
7: else
8: Reg[m] = Reg[m-1] ⊕ A
9: end if
10: m = m + 1
11: Sleep(Mcycle)
12: end while

1: Controller:
2: start_loop
3: for C = 1 to N
4: for b = 0 to N
5: shift = true
6: for i = 0 to N
7: currentRow = i
8: for j = 0 to N
9: if m - b > th then
10: giveToken(i,j)
11: RestartUnit(b)
12: while !getFinish() && !Timeout()
13: end if
14: if (m = 1) then
15: if Cond = (Unit(i,j).Reg[0]) then
16: end for
17: end if
18: sendResetFlag()
19: if shift then
20: SHIFTReg()
21: goto start_loop
22: end if
23: end for
24: end for
25: procedure SPIKE(row, flag)
26: if row == currentRow then
27: if flag = 1 then
28: sendSpokeEast();
29: else
30: sendSpokeWest();
31: end if
32: else
33: sendSpokeNorth();
34: end if
35: end procedure
```

This algorithm ensures to find the closest Unit pairs whose Reg values are both 1 by sending a Syndrome signal to the initiator of the first coming Spike. However, this does not necessarily set a find of pairs that minimize the total distance since the sink node is sequentially allocated to Units. Therefore, we limit the maximum number of hops to propagate a Spike in a single run of the above steps and increase it iteratively. The Controller procedure in Algorithm 1 implements this by setting a timeout after giving the Token to a Unit.

If an error syndrome appears from one of the Units in the SC toward the outside (Fig. 3(c)), there is no proper matching pair between erroneous ancilla bits. To deal with such a case, we introduce additional Units called Boundary Units at the edges of the SC. The boundary Units never get a Token from the Controller, but always send a Spike upon a requestSpike() call.

To deal with matching problems on a 3-D lattice, results of multiple rounds of measurement for an ancilla bit are stored in each Unit instead of actually having Units for every 3-D lattice points. Reg is extended to be an array to hold several measurement results. For every new measurement, the value of the ancilla bit is XORed with the latest result. Each Unit finds the value of 1 in the current bit position, it sends a Spike to propagate a Spike in a single run of the above steps and increase it iteratively. The Controller procedure in Algorithm 1 implements this by setting a timeout after giving the Token to a Unit.

If an error syndrome appears from one of the Units in the SC toward the outside (Fig. 3(c)), there is no proper matching pair between erroneous ancilla bits. To deal with such a case, we introduce additional Units called Boundary Units at the edges of the SC. The.

```
1: procedure SHIFTReg
2: if m > 0 then
3: for i = 0 to N
4: Reg[i] = Reg[i+1]
5: end for
6: m = m - 1
7: end if
8: end procedure
```

 Boundary Units never get a Token from the Controller, but always send a Spike upon a requestSpike() call.

To deal with matching problems on a 3-D lattice, results of multiple rounds of measurement for an ancilla bit are stored in each Unit instead of actually having Units for every 3-D lattice points. Reg is extended to be an array to hold several measurement results. For every new measurement, the value of the ancilla bit is XORed with the latest result stored in Reg (MeasureEachUnit process in Algorithm 1).

```
1: RestartUnit(Input: b)
2: if Token == 1 then
3: FlagToken = 1
4: for i = 0 to N
5: Reg[i] = A
6: Reg[0] = A
7: if Reg[i] == 1 then
8: if (S = getSpoke()) != NULL then
9: Dir = rotate(S)
10: correctQubit(Dir)
11: end if
12: sendController("Finish")
13: end if
14: do
15: if t != b & Reg[t] == 1 then
16: sendController("Finish")
17: end if
18: else
19: else
20: for j = 0 to N
21: if Reg[j] == 1 then
22: SPIKE(self.row,FlagToken)
23: end if
24: end if
25: end if
26: while !getFinish() && !Timeout()}
27: end if
28: procedure SPIKE(self.row,FlagToken)
29: SPIKE(self.row,FlagToken)
30: if getCorrect() then
31: correctQubit(Dir)
32: end if
33: SPIKE(self.row,FlagToken)
34: end if
35: end if
36: if (S = getSpoke()) != NULL then
37: end if
38: end if
39: end procedure
```

**B. Online-QEC for faster error correction cycle**

Figure 3 compares the concepts of batch and online-QEC for a distance-3 SC with measurement errors. In batch-QEC, an error correction phase can be scheduled after three measurements are processed. On the contrary, an error correction phase is associated with each measurement step in online-QEC. This helps correct qubit errors as soon as they occur, which reduces the number of qubit errors in a single QEC phase and makes the matching process easy. However, QEC with only a single SC plane should be avoided due to the occurrence of ancilla bit measurement errors. Hence, the Controller waits for several measurements have done before starting a QEC phase. This is ensured by having at least \( t_h \) measurement results in Reg for each Unit (L.9 in Controller code in Algorithm 1). The threshold value of \( t_h \) controls a trade-off between the error-to-correction latency and error correction performance. We evaluate the appropriate \( t_h \) in the next Subsection.

In the Spike generation, multiple bits in a Reg are sought step-by-step from the oldest to the newest measurement. Once a non-sink Unit finds the value of 1 in the current bit position, it sends a Spike to the sink Unit. This process is repeated by changing the base depth...
which points to the start position of the Reg for seeking. Otherwise, the matching process is the same as the 2-D case.

C. Necessary vertical depth for online-QEC

We evaluate how the vertical depth threshold \( t_{\text{th}} \) affects the error correction performance of QECOOL. We perform a spike-based QEC shown in Algorithm 1 with the batch-QEC manner by setting \( N_{\text{depth}} \) and \( t_{\text{th}} \) to \( d \) and \( d-1 \), respectively, indicating that the Controller process is executed after \( d \) times measurement. We call this method as batch-QECOOL.

The error-correcting performance is evaluated numerically by a quantum error simulator. We simulate the QECOOL algorithm with both informational and ancilla qubit errors using the phenomenological noise model \( [4] \). We show logical X error rates versus physical Pauli-X error rate as the performance indicated \( [4] \). We assume the error probabilities of data and ancilla qubits are equal. Note that the same experimental setup is used in Section III-B.

Figure 4(a) shows the log scale plots of physical qubit error rate and logical X error rate for Pauli-X errors. The solid black line is the break-even (physical error rate = logical error rate) value. The other solid lines show the results for batch-QECOOL, whereas the dashed lines represent cases for MWPM \( [7] \).

We use threshold values to evaluate the error correction performance of our algorithm. The threshold value, \( p_{\text{th}} \), is defined for each decoding algorithm and represents the value of physical error rate \( p \) that satisfies the following properties; if the physical error rate \( p \) is less than \( p_{\text{th}} \), the logical error rate \( p_L \) decreases as the code distance \( d \) increases. It can be defined as the value of \( p \) at the intersection of \( p \) and \( p_L \) plotted for several code distances \( d \). The higher the \( p_{\text{th}} \), the better the decoding algorithm. \( p_{\text{th}} \) of batch-QECOOL can be obtained from Fig. 4(a) and it is around \( p = 0.015 \), while that of MWPM is 0.03.

Figure 4(b) shows the proportion of matchings that propagate through three or more planes in the vertical (temporal) direction on a 3-D lattice. While three or more propagation in the vertical direction happens many times in larger physical error rate \( p \), it is negligible for relatively smaller \( p \), especially when \( p \) is less than \( p_{\text{th}} \). In general, QEC decoders require that the physical error rate \( p \) is smaller than their \( p_{\text{th}} \) for realistic error correction. This indicates having three SC planes (or Reg of three entries in QECOOL) for online-QEC is almost satisfactory. Thus, we assume \( t_{\text{th}} = 3 \) in the following sections.

2) Under the assumption that Pauli \( X \), \( Y \), and \( Z \) errors occur stochastically, \( Y \) errors can be considered as a simultaneous \( X \) and \( Z \) error. Therefore, even if \( X \) and \( Z \) errors are corrected independently, all errors can be decoded correctly. Thus, we show only the case of \( X \) error.

![Fig. 4](image_url)

**Fig. 4.** (a) Error rate scaling for the MWPM decoder and batch-QECOOL. We can observe the threshold value of QECOOL at approximately 1.5% physical error rate. (b) The proportion of matchings that propagate through three or more planes in the vertical direction.

IV. HARDWARE IMPLEMENTATION

A. Hardware architecture

Figure 5 shows the overview of the hardware architecture for \( X \) error detection for a single distance-\( d \) logical qubit \( [7] \). Each hardware Unit corresponds to the “Unit” of Algorithm 1, and \( d \times (d-1) \) Units are aligned in a 2-D grid pattern. There is one Controller per one logical qubit to orchestrate the Units by distributing \( \text{Push}, \text{Pop}, \) and \( \text{Restart} \) signal. Note that this architecture can be easily extended to any code distance \( d \) thanks to its distributed nature.

Each Unit has a register (Reg) that stores measured values of the corresponding ancilla bit. It works as a queue; whenever the measurement process is performed, the Controller sends the Push signal to all the Units and the measured value is stored at the end of Reg. When the first bit entry of Reg (corresponding to the oldest measurement) of all the Units becomes 0, the error correction process for the current layer is completed. In this case, the Controller broadcasts the Pop signal, which makes the values in the Reg being one-bit shifted. Based on the results of Section III-C, we assume \( t_{\text{th}} = 3 \) which means the Reg needs to store at least three measurement values. In this paper, we set the size of a Reg to 7-bit with some margin.

A “Row Master” module is attached to each row of the 2-D Unit array and gives the Token to the first Unit of the row. It always checks Reg values in all the Units in the row, and if none of the bits in Reg is 1, it avoids giving the Token to the row to avoid meaningless Token passing on non-erroneous Units which helps reduce the time taken for QEC. In this case, it just passes the Token to the next Row Master. It is also responsible to send \( \text{CurrentRow} \) signal to all the Units of the row.

Two Boundary Units are located on each side of the Unit array. Each of them is connected to \( d \) Units on a horizontal edge. One Boundary Unit can be shared by all the Units on each edge. Distributing a Spike to all the Units does not affect QEC’s correctness since only the first Spike that reaches the Sink Unit is valid in the algorithm.

B. Unit implementation

In each Unit, there are five components as described below.

- **State machine**: A state machine controls the behavior of the Unit. The state transition happens based on Restart signal, coming Token, and coming Spike. This module has two register memories; HoldToken and Reg, and their values affect state transition.

- **Prioritization module**: As a Unit may get multiple Spikes from different directions simultaneously, we need to prioritize the input from a specific direction. This module selects one Spike from them by the predefined priority. We use the race logic concept, which utilizes the relative propagation time of signals. We put an

---

2) The identical hardware applies to \( Z \) error detection.
appropriate signal delay in each direction for coming Spike so that a Spike from the highest priority direction must come faster than the others.

- **Spike out module**: This module sends a Spike to the appropriate direction based on values of the CurrentRow and the FlagToken.
- **Syndrome out module**: This module sends a Syndrome signal to the direction indicated by the Dir register that stores the opposite direction of the coming Spike. It also generates a correction signal to the associated data qubit.
- **BasePointer module**: This module controls which bit position in Reg to be read out based on the value of the base register. The read value from Reg is used to decide whether the Unit should send a Spike.

C. **SFQ logic gates**

We designed the QECOOL hardware based on an RSFQ cell library [22] developed for a niobium nine-layer, 1.0-µm fabrication technology [9], [15]. Table I summarizes the SFQ logic gates used in this work. Since the essential element of SFQ that affects power consumption and hardware cost is Josephson junction (JJ), the Table shows the number of JJs for each gate and assumed the bias current required for operation. The operating temperature and designed supply voltage are 4-K and 2.5 mV, respectively. In RSFQ, since most of the power is consumed statically almost independent of switching activities, it is calculated by multiplying the bias voltage and currents.

| cell             | JJs | Bias current (mA) | Area (µm²) | Latency (ps) |
|------------------|-----|-------------------|------------|--------------|
| splitter         | 4   | 0.880             | 900        | 4.3          |
| merger           | 7   | 0.880             | 900        | 8.2          |
| 1.2 switch       | 33  | 3.464             | 8100       | 10.5         |
| destructive readout (DRO) | 6   | 0.720             | 900        | 5.1          |
| nondestructive readout (NDRO) | 11  | 1.112             | 1800       | 6.4          |
| resettable DRO (RD) | 11  | 0.900             | 1800       | 6.0          |
| dual-output DRO (D2) | 12  | 0.944             | 1800       | 6.8          |

We used the Josephson simulator (JSIM) [9], a SPICE-level simulator, to verify the functionality of the designed Unit with 7-bit Reg and evaluate its latency. Table II shows the total number of JJs, total area, total bias current, and latency of each module. Figure 6 shows the layout of the QECOOL Unit. A Unit consists of 3177 JJs in total, and its area footprint is 1.274 mm². The maximum delay of the designed circuit is 215 ps, results in the maximum operating frequency of about 5 GHz. It can operate fast enough to achieve the required QEC latency described later in Section V-A. The power consumption of a Unit is $330 J_{\text{cell}} \times 2.5 \mu W/\text{cell} = 840 J_{\text{cell}}$ including the wiring power, if implemented with RSFQ logic.

![Layout of the designed QECOOL Unit](image)

**Fig. 6.** The layout of the designed QECOOL Unit

**Fig. 7.** Physical vs. logical error rate plot for the QECOOL algorithm operating at several frequencies.

V. EVALUATIONS OF THE ONLINE-QECOOL ALGORITHM

A. Execution cycles

| $p$ | $\sigma$ | $\alpha$ | $\beta$ | $\gamma$ |
|-----|----------|----------|---------|----------|
| 0.001 | Max   | Avg    | Max   | Avg    |
| 0.005 | Max   | Avg    | Max   | Avg    |
| 0.01  | Max   | Avg    | Max   | Avg    |

Table III shows the number of execution cycles per layer for QECOOL for several combinations of coding distance $d$ and physical error rate $p$. It is observed that the execution cycles of QECOOL are highly dependent on $d$ and $p$.

Since it is reported that measuring ancilla bit takes about $1 \mu s$ [10], QEC is fast enough if the process for one layer is finished within $1 \mu s$.

B. Error correction performance

We use the same error model as in Subsection III-C to evaluate the performance of QECOOL for online-QEC. The measurement process is assumed to be performed once every $1 \mu s$. Each Unit has 7-bit Reg, and the Algorithm 1 is set to 3. If Reg overflows because of the slow QEC performance, the trial is considered as a failure.

Figures 7(a) and (b) indicate that the slower frequency causes buffer overflow of Unit, which affects the error correction performance at larger code distance $d$. Only in Figure 7(c), we can observe the $p_{th}$ of QECOOL at approximately $p = 1.0\%$, which is slightly smaller than that of batch-QECOOL.

C. Power estimation with ERSFQ logic

To put the more decoder units in a dilution refrigerator, their power consumption should be much lower, and the RSFQ technology is not
feasible. Instead, we need to use the ERSFQ [13] technology where the static power consumption is eliminated. Power consumption in ERSFQ circuits is only consumed by dynamic power twice as high as that of RSFQ. Though ERSFQ is slower than RSFQ, this is not a problem for our hardware design since our target clock frequency is much lower than the maximum frequency in RSFQ.

Based on the RSFQ design and the power model of ERSFQ [14], we estimate the power consumption when ERSFQ is applied. Here, the power of a Unit with ERSFQ can be estimated as follows:

\[ P_{\text{Unit}} = \text{(bias current)} \times (\text{frequency}) \times \Phi_0 \times 2 \]

We use flux quantum \( \Phi_0 \) of \( 2.068 \times 10^{-15} \text{ Wb} \). The reason why it is multiplied by 2 is to represent twice of dynamic power in ERSFQ.

For a proposed Unit with 7-bit Reg, the total bias current is 336 mA. If we suppose 2 GHz clock frequency, the power consumption of a Unit at a 4-K environment is estimated as follows:

\[ 336 \ times 10^{-3} A \times (2 \times 10^{12} \times 2) = 2.78 \times 10^{-9} \text{ W} / \text{Unit} \]

\[ \text{D. Comparison to existing decoders} \]

Table IV shows a brief comparison of MWPM and recent prominent hardware-efficient decoding algorithms. While MWPM and union-find (UF) have a higher threshold, they are designed to operate in a room-temperature environment as they are implemented by software or an FPGA device.

AQEC [11] and our decoder are designed to operate in a cryogenic environment and have higher scalability than others, though these have a slightly lower threshold. We summarize a detailed comparison of AQEC and QECool in Table V. We assume the coding distance \( d \) of 9, the longest one evaluated in the AQEC paper, and the physical error rate (for both data and ancilla qubits) \( p \) of 0.001, which is 0.1 times the \( p_{tu} \) of our decoder. The power budget of the 4-K temperature region of dilution refrigerators is supposed to be 1 W [12], and the number of protectable qubits is estimated in terms of the decoders’ power consumption. Applying the same arguments as Subsection III-C, we assume that extending AQEC to 3-D requires 7 times the modules needed for 2-D processing. In addition, we optimistically consider AQEC’s latency for 2-D processing as its per layer latency when extended to 3-D. The Table shows that QECool is superior to AQEC in other aspects than latency, which is not a major disadvantage since our decoder can finish one layer process within 1 \( \mu s \), the interval of the measurement process [10]. AQEC can lower its operating frequency to reduce its power consumption by taking advantage of its low latency, however, our method still has lower power consumption, even taking this into account. Note that the \( p_{tu} \) of AQEC for the 3-D case is unknown, and it is expected to be slightly smaller than ours as the \( p_{tu} \) trends of 2-D to 3-D in Table IV indicate 70-80% reduction in 3-D cases.

### References

[1] F. Arute et al. Quantum supremacy using a programmable superconducting processor. Nature, 574(7779):505–510, 2019.

[2] P. Das et al. A scalable decoder micro-architecture for fault-tolerant quantum computing. arXiv preprint arXiv:2001.06598, 2020.

[3] N. Delfosse and N. H. Nickerson. Almost-linear time decoding algorithm for the surface code. In *Proceedings of the ACM/IEEE 47th Annual Symposium on Computer Architecture*, page 407–410, 1989.

[4] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *Proceedings of the ACM/IEEE 47th Annual Symposium on Computer Architecture*, page 414–419, 1999.

[5] A. G. Fowler. Minimum weight perfect matching of fault-tolerant topological quantum error correction in average O(1) parallel time. *Quantum Inf. Comput.*, 15(1–2):145–158, 2015.

[6] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86:032324, 2012.

[7] A. Fujimaki et al. Large-scale integrated circuit design based on a Nb nine-layer structure for reconfigurable data-path processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(1):776–779, 2003.

[8] C. Gidney and M. Ekerer. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. *arXiv: Quantum Physics*, 2019.

[9] A. Holmes, M. R. Jokar, G. Pasandi, Y. Ding, M. Pedram, and F. T. Chong. NISQ+: Boosting quantum computing power by approximating quantum error correction. *In Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture*, page 556–569, 2020.

[10] J. M. Hornbork et al. Cryogenic control architecture for large-scale quantum computing. *Phys. Rev. Applied*, 3:024010, 2015.

[11] D. E. Kirichenko, S. Sarwana, and A. F. Kirichenko. Zero static power dissipation biasing of RSFQ circuits. *IEEE Transactions on Applied Superconductivity*, 21(3):776–779, 2011.

[12] O. A. Mukhanov. Energy-efficient single flux quantum technology. *IEEE Transactions on Applied Superconductivity*, 21(3):760–769, 2011.

[13] S. Nagasawa et al. Nb 9-layer fabrication process for superconducting large-scale SFQ circuits and its process evaluation. *IEEE Transactions on Electronics*, E97.C(3):132–140, 2014.

[14] R. Sato et al. High-speed operation of random-access-memory-embedded microprocessor with minimal instruction set architecture based on rapid single-flux-quantum logic. *IEEE Transactions on Applied Superconductivity*, 27(4):1300505, 2017.

[15] P. W. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 54(4):R2493, 1996.

[16] P. W. Shor. Polynomial-time algorithms for primality factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[17] S. S. Tannu, D. M. Carmean, and M. K. Qureshi. Cryogenic-dram based memory system for scalable quantum computers: A feasibility study. In *Proceedings of the International Symposium on Memory Systems*, pages 189–195, 2017.

[18] S. S. Tannu, Z. A. Myers, P. J. Nair, D. M. Carmean, and M. K. Qureshi. Taming the instruction bandwidth of quantum computers via hardware-managed error correction. In *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 679–691, 2017.

[19] S. Varastamopoulos, K. Bertels, and C. G. Almudever. Comparing neural network based decoders for the surface code. *IEEE Transactions on Computers*, 69(2):300–311, 2019.

[20] Y. Yamanashi et al. 100 GHz demonstrations based on the single-flux-quantum cell library for the 10 kA/cm² Nb multi-layer process. *IEEE Transactions on Electronics*, 93(4):440–444, 2010.