QULATIS: A Quantum Error Correction Methodology toward Lattice Surgery

Yosuke Ueno[1, 2], Masaaki Kondo[2, 3], Masamitsu Tanaka[4], Yasunari Suzuki[5, 6], and Yutaka Tabuchi[7]

[1]Graduate School of Information Science and Technology, The University of Tokyo
[2]Faculty of Science and Technology, Keio University
[3]RIKEN Center for Computational Science
[4]Graduate School of Engineering, Nagoya University
[5]NTT Computer and Data Science Laboratories
[6]JST PRESTO
[7]RIKEN Center for Quantum Computing

ueno@hal.ipc.i.u-tokyo.ac.jp, kondo@acsl.ics.keio.ac.jp
masami_t@nagoya-u.jp, yasunari.suzuki.gz@hco.ntt.co.jp, yutaka.tabuchi@riken.jp

***Abstract*—Due to the high error rate of a qubit, detecting and correcting errors on it is essential for fault-tolerant quantum computing (FTQC). *Surface code* (SC) associated with its decoding algorithm is one of the most promising quantum error correction (QEC) methods because it has high fidelity and requires only nearest neighbor qubits connectivity. To realize FTQC, we need a decoder circuit capable of not only QEC in a 3-D lattice to deal with errors in measurement on ancillary qubits but also quantum operations on logically constructed qubits. Whereas several methods to perform logical operations on SC, such as *lattice surgery* (LS), are known, no practical decoders supporting them have been proposed yet.**

One of the most promising QC implementations today is made up of superconducting qubits that are located in a cryogenic environment. To reduce the hardware complexity of QC and latency of QEC, we are supposed to perform QEC in a cryogenic environment. Hence a power-efficient decoder is required due to the limited power budget inside a dilution refrigerator.

In this paper, we propose an online-QEC algorithm that supports LS with a practical decoder circuit, as well as a new FTQC architecture. We design a key building block of the proposed architecture with a hybrid of *SFQ*- and Cryo-CMOS-based digital circuits and evaluate it with a SPICE-level simulation. Each logic element includes about 2400 Josephson junctions, and power consumption is estimated to be 2.07 μW when operating with a 2 GHz clock frequency. We evaluate the decoder performance by a quantum error simulator for an essential operation of LS with code distances up to 11, and it achieves a 0.6% accuracy threshold. In an LS-based architecture further supporting a magic-state distillation protocol, which is expected to run for near-term universal quantum computing, we evaluate the QEC performance and power consumption of the architecture and show that it is practical to be operated in 4-K temperature region of a dilution refrigerator.

***Index Terms*—Quantum Computing, Quantum Error Correction, Single flux quantum (SFQ)**

I. Introduction

Quantum computers (QCs) are becoming an attractive computing paradigm as the number of implementable qubits increases, given they have the potential to solve several problems efficiently. Whereas more than 53 qubits are implemented in state-of-the-art QCs [1], [2], there remain several challenges to increase the number of qubits; for example, fragile qubits are a critical bottleneck for practical quantum computation. As quantum error correction (QEC) is the unique method to reduce the effective error rate of quantum gates in a scalable manner, it has been extensively studied so far [3]–[7]. Given a direct measurement on informational qubits at an error detection process in QEC is prohibited due to the nature of the quantum mechanics, ancillary qubits (ancilla bits) are equipped with the informational qubits to observe their error parity indirectly. The locations of erroneous qubits are inferred from measurement outcomes of the ancillary qubits. The procedure is called decoding. The decoding procedure is known to be a non-trivial task, and it requires a large amount of computational cost in classical computers. Thus, the QEC coding/decoding schemes should be optimized both in algorithmic and implementation perspectives.

One of the most promising QC implementations today is superconducting quantum circuits [1], [2]. They are operated in a cryogenic environment with an effective temperature of around ten milliKelvins to eliminate thermal noises in the device. While we place the qubits at the milliKelvin stage of a dilution refrigerator, the associated control electronics, including a QEC processing unit, are supposed to be located at a higher temperature stage or outside the cryostat. [8]. The spatial distance between the components demands many cables between different temperature stages, leading to the hardware complexity in wiring and latency in QEC. This point hinders scaling up QCs. Although the QEC processing unit right next to the qubit chip alleviates this problem [9], it is usually unrealistic under the restricted power budget in the lower temperature stages of a cryostat (*e.g.*, tens of μW or around 1 W in the millikelvin stage or the 4-K stage, respectively). Extraordinary low-power QEC is necessary. The same problem applies to other types of solid-state qubits that operate in a

cryogenic environment, *e.g.* spin qubits in donor silicon and MOS-type double quantum dots.

Surface code (SC) [6] is a promising candidate in practical QEC coding schemes. SC is implemented on a square grid of qubits and only requires interactions between geometrically adjacent physical qubits. These features simplify the hardware implementation and provide extendability and high reliability in QEC. It is known that the decoding capability only for the two-dimensional (2-D) plane of ancilla bits is insufficient to realize fault-tolerant quantum computing (FTQC). SC decoders are requested to perform 1) QEC on a three-dimensional (3-D) lattice to deal with measurement errors and 2) quantum gate operations among logically constructed qubits. Although most SC decoders for 2-D lattices are extendable to 3D, the computational complexity of decoding typically increases significantly. There have been several proposals to perform quantum gate operations with SC-based QEC [7], [10]. In particular, lattice surgery (LS) is considered as the current state-of-the-art method to perform logical operations among logical qubits, and most FTQC techniques have been proposed based on the combination of SC and LS [11]–[16].

Finding the most probable error locations and types in a qubit array can be solved using a minimum-weight perfect matching (MWPM) algorithm with approximations. The MWPM guarantees a polynomial-time solution; however, it requires a high computational cost. Several low-cost decoding algorithms and their implementations [17]–[21] have been proposed to reduce the computational burden. Most of them except for [21] are mainly discussed for 2-D cases, and all of them are designed for single logical qubit protection. To our best knowledge, no prior research has been reported on the low-cost decoder algorithm or architectural design of QEC capable of quantum operations among multiple logical qubits. Again, QEC processors having low latency and low power consumption are preferable for scalability.

In this paper, we propose *QULATIS*, a QUantum error correction methodology for LATtIce Surgery, and a decoder implementation using a hybrid of Cryo-CMOS and single flux quantum (SFQ) digital circuits. For a low latency case, references [20], [21] pointed out that ancilla-qubit measurement and decoding processes can be overlapped instead of waiting for completing measurements before a decoding phase. In other words, we can simultaneously execute the measurement and decoding processes. The overlapped ones are called *online-QEC*, and the conventional ones are referred to as *batch-QEC* [21]. When designing a decoder architecture with LS, the number of measurements required for 3-D SC decoding becomes much greater than that for single logical qubit protection. The vast amount of error information in batch-QEC makes QEC latency worse and degrades the error correction accuracy. Hence, we employ the online-QEC technique in our decoder architecture.

The contributions of this paper are summarized as follows:

- We propose the QULATIS concept and an SC decoding algorithm with LS support.
- We propose the QULATIS architecture to support FTQC and a hybrid of Cryo-CMOS and SFQ logic design based on QULATIS architecture.
- We evaluate the power consumption and area of a key building block of QULATIS architecture with a SPICE-level simulator.
- We evaluate the error correction performance of QULATIS by a quantum error simulator.

We evaluate the QEC performance using a typical magic-state distillation circuit [11] which frequently appears in practical FTQC processing [22], [23] and requires logical Pauli-X measurements on eight logical qubits. The evaluation results show that our FTQC architecture can perform sufficiently low-latency QEC for LS and achieve low-power consumption.

Although we focus on superconducting qubits in this paper, our method is applicable to any QC system with SC. Due to the scalability issues of wiring cables between cryogenic and room temperature environments, it is especially beneficial in implementing our decode process by an SFQ digital logic for solid-state qubits operated in a cryogenic environment.

II. Background

A. Quantum computation with superconducting qubits

Quantum computing has enabled exponential speed-ups of several computational tasks [24], [25], retrieval of rich information with small queries [26], and information-theoretic security with a simple proof [27]. Thus, many research groups have focused on developing useful QCs [1]. A unit of quantum data, called a qubit, can preserve not only binary information but also its phase. Arbitrary processing on qubits can be constructed as a combination of a universal set of quantum operations, such as Hadamard, CNOT, and T gates [28]. To extract information from quantum states, we need to measure each qubit state. A Pauli measurement is a class of binary measurement operations that are characterized by n-qubit Pauli operators. See [28] for a basic introduction.

Since the theory of QCs was established, many experimental efforts have been paid for exploring scalable and controllable quantum devices. A qubit implementation consisting of superconducting circuits [1], [2], [29] is one of the most promising candidates because of its scalability, controllability, and long lifetime. Currently, the lifetime of superconducting qubits is about 1 ms, and local physical operations, as well as Pauli measurements, are operated with below 1% physical error rate. Few tens of qubits can be fabricated on a superconducting chip. While these parameters are not enough for useful applications, they are expected to be improved rapidly.

B. Quantum error correction with surface codes

One of the fundamental obstacles to practical quantum computing is the high error rate of physical qubits. Thus, we need to protect the quantum information with quantum error correction (QEC) [3], [30] mechanisms. Surface code (SC) [6], [30] is one of the most promising QEC codes; SC can reduce logical errors effectively and be implemented simply with physical operations on geometrically adjacent

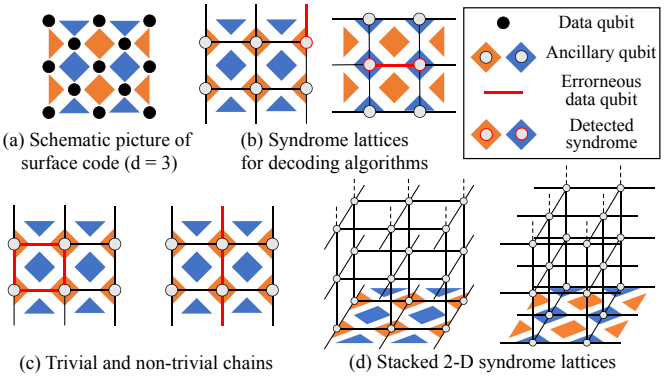


Fig. 1. (a) Schematic picture of surface code ($d = 3$). (b) Syndrome lattices generated from surface codes. (c) Examples of trivial and non-trivial chains of errors. (d) Stacked 2-D syndrome lattices.

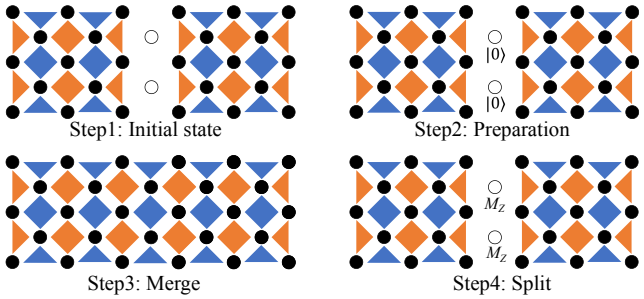


Fig. 2. Process of logical Pauli-XX measurements with lattice surgery.

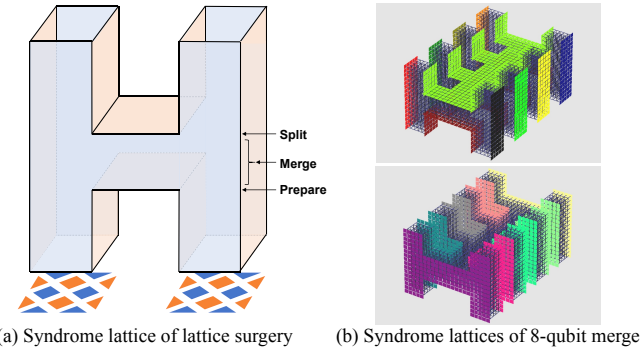
qubits located on a two-dimensional grid. SC consists of two types of physical qubits, data qubits and ancillary qubits. Data qubits are used to represent a logical qubit. Ancillary qubits are utilized to check the parity of errors on the neighboring data qubits. This parity check is called a stabilizer measurement, and its binary outcome is called a syndrome value.

To deal with both bit-flip (Pauli-X) and phase-flip (Pauli-Z) errors, QEC needs two types of stabilizer measurement, *i.e.*, X- and Z-stabilizer measurements. X- and Z-stabilizer measurements can detect the Pauli-Z and -X errors on the neighboring data qubits, respectively. Note that Pauli-Y errors are considered to be a combination of bit- and phase-flip errors because the Pauli operators X, Y, and Z have the following relationship: $Y = iXZ$. Fig. 1 (a) shows a schematic picture of the SC with code distance $d = 3$. Here, data qubits and ancillary qubits for X-(Z-) stabilizer measurements are represented as circles and blue (red) squares, respectively. X-(Z-)stabilizer measurements detect Pauli-Z(-X) errors when the corresponding parity of the neighboring qubits is odd.

If we assume a noise model where Pauli errors occur on data qubits probabilistically, the estimation of the most likely Pauli errors can be obtained by the following minimum-weight perfect matching (MWPM) problem [7]. Suppose a 2-D grid graph where its nodes and edges correspond to the syndrome values and the data qubits, respectively. An example is shown in Fig. 1 (b), where erroneous data qubits are represented as red edges, and detected syndromes are represented as nodes with red rims. The two lattices for X- and Z-stabilizer measurements have two distinct boundaries, which are called smooth and rough boundaries, respectively. We construct a weighted complete graph where its nodes correspond to the detected syndromes or boundaries, and the weights of its edges are determined from the noise model. We find a minimum weight perfect matching of the complete graph, and the estimated errors are represented as edges between each pair of syndromes of the matching. Whether the estimation of errors is successful or not is determined by the properties of the occurred and estimated errors. If we estimate errors as a perfect matching, the resultant error chain consists of topologically trivial and non-trivial chains, as shown in Fig. 1 (c). The odd number of non-trivial chains indicates the failure of error estimation. Even when ancillary qubits also suffer from noise, we can reliably estimate errors by extending the task to a 3-D lattice, *i.e.*, by considering a stacked 2-D snapshot as shown in Fig. 1 (d). A procedure to generate a 2-D snapshot is called a *code cycle*. Since the time of the code cycle must be much shorter than the coherence time, a practical decoding algorithm must be capable of decoding these sequentially captured 2-D snapshots. See [7] for the detail of this formalism.

C. Lattice surgery and gate teleportation with magic states

During the computation, we need to perform a universal set of logical gates, Hadamard, CNOT, and T-gates, on logical qubits. The implementation of logical Hadamard gates is straightforward. On the other hand, the implementation of logical CNOT and T-gates only with neighboring physical operations is not trivial. These two logical gates can be efficiently implemented with the techniques of lattice surgery (LS) and gate teleportation with magic states, respectively [11], [31]. LS provides a way to implement logical Pauli measurements on multiple logical qubits via the following merge-and-split operations. Note that logical CNOT operations can be achieved through multiple-qubit logical Pauli measurements and feedback of logical Pauli operations. The minimum example of the LS, a logical Pauli-XX measurement on two logical qubits, is shown in Fig. 2. In this figure, 1) we initialize all the sandwiched physical qubits to physical $|0\rangle$ states, 2) two surface codes are merged by performing another set of stabilizer measurements and repeating them for d code cycles, and 3) split it into two planes by performing the original stabilizer measurements and measure all the sandwiched physical qubits on Pauli-Z basis. These operations are known to be equivalent to a logical Pauli-XX measurement on the two logical qubits. The outcome of the logical Pauli measurement is calculated from the parity of the outcomes of Pauli-X stabilizer measurements in the first cycle of LS. Since this procedure merges two rough boundaries, this operation is called rough merge. The Pauli-ZZ measurement can also be achieved in a similar way by merging smooth boundaries. During the merge-and-split operations for rough boundaries, we obtain a stacked 2-D lattice to be decoded, as shown in Fig. 3 (a). This figure shows rough and smooth boundaries of



(a) Syndrome lattice of lattice surgery (b) Syndrome lattices of 8-qubit merge

Fig. 3. Rough and smooth boundaries of syndrome lattices.

stacked syndrome lattices as red and blue faces, respectively. As in the case of Fig. 1 (b), a Pauli error connecting a boundary to another distinct one with the same color is undetectable with stabilizer measurements and modifies the logical states or flips the result of the logical measurement. We keep d code cycles during the merge phase since the distance between two U-shaped red boundaries corresponds to the code cycles during the merge.

As for the logical T-gate, the operation is indirectly performed with the gate-teleportation technique by consuming a logical qubit prepared in the state $|A\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$, which is called the magic state [32]. While the direct preparation of high-fidelity $|A\rangle$ in the logical space is difficult, we can construct a clean magic state from several noisy magic states. This procedure is called magic-state distillation [32] and is expected to occupy a dominant part of the whole FTQC [22], [23]. A typical distillation process known as 15-to-1 magic-state distillation [32] requires the space of 24 surface-code patches and five repetitions of eight-qubit logical Pauli measurements to prepare a clean magic state, which is consumed in each application of a T-gate. An example of the stacked 2-D lattices for an eight-qubit logical Pauli measurement is shown in Fig. 3 (b). Since we need to consider several rough and smooth boundaries in a multi-qubit merge, each distinct rough and smooth boundary is rendered with different colors in the upper and lower figures, respectively. The blue cylinders filled between the boundaries correspond to the edges of stacked syndrome lattices explained in Fig. 1 (d). Thus, the implementation of the decoding algorithms should be capable of processing the decoding tasks for eight-qubit LS.

D. Single Flux Quantum logic

A single flux quantum (SFQ) logic is a digital circuit composed of superconductor devices. Information processing in SFQ circuits is performed with magnetic flux quanta stored in superconductor rings with Josephson junctions (JJs). In SFQ circuits, logical '1' and '0' are represented by the presence or absence of a single magnetic flux quantum ($\Phi_0 = 2.068 \times 10^{-15}$ Wb). JJs act as switching devices like transistors in ordinary CMOS circuits. An impulse-shaped voltage pulse, called an SFQ pulse, is generated only when an SFQ travels across a JJ. Several researchers have designed SFQ-based microprocessors and demonstrated ultra-high-speed, low-power operations with them [33]–[38]. Some researchers have demonstrated an 8-bit microprocessor composed of more than 10,000 JJs which operates at 50 GHz [36] and a gate-level-pipelined, bit-parallel multiplier that operates at up to 48 GHz and consume 5.6 mW [39], [40]. Since SFQ circuits consist of superconductor devices, they work only in a cryogenic environment around 4 kelvin. Because of this constraint, finding appropriate applications for SFQ is not a trivial task. Research on quantum computers is ongoing, and solid-state qubits based on superconductor devices are considered to be promising for quantum computers that operate in a cryogenic environment. In this study, we apply SFQ circuits to quantum computers, especially for rapid and efficient QEC.

III. Related Work

Since the performance and efficiency of a decoding process determine the achievable code distance in FTQC, huge efforts have been aimed at reducing its latency. The most popular solution is to solve the MWPM problem with a polynomial-time algorithm, that is, Edmonds' blossom algorithm [41]. Besides that, renormalization group decoders [42], machine-learning-based decoders [17], and union-find (UF) decoders [19] were proposed as potentially fast and near-optimal candidates. Most of these proposals were tested as software solutions, and practical implementation is not considered. Since the cycle of stabilizer measurements is expected to be about 1 μs [20], [23], the implementation of low latency decoding is critical for the demonstration of QCs.

There have been proposed several practical SC decoders. Table I summarizes a qualitative comparison between QULATIS and these decoders. An architecture based on a UF-based decoding algorithm [18] was recently proposed; they attracted much attention because of their accuracy and simplicity. Das *et al.* [18] stated that fully pipelined hardware helps speed up the decoding process. Cryogenic computing, such as SFQ or Cryo-CMOS, has actively been studied to design peripherals of QC controllers [9], [20], [43]. Holmes *et al.* [20] proposed an algorithm named AQEC where they devised a power-efficient and high-speed SFQ-based decoder for SC. Their implementation consists of multiple units corresponding to each data and ancillary qubit of an SC logical qubit to detect and correct errors by propagating simple signals between the units with a distributed processing scheme. Their implementation is capable of correcting Pauli errors on the data qubits. In 2021, we proposed the QECOOL algorithm, which is an extension of AQEC to deal with measurement errors of ancillary qubits [21]. We implemented an SFQ-based decoder which achieves lower power consumption than AQEC. To cope with the measurement error, we also proposed the concept of online-QEC, where the stabilizer measurements and decoding processes are performed simultaneously, in contrast to the conventional batch-QEC, where the decoding process is

TABLE I
QUALITATIVE COMPARISON AMONG THIS WORK AND OTHER APPROACHES

	MWPM [7], [31], [41]	UF [18], [19]	AQEC [20]	QECOOL [21]	QULATIS (This work)
Impleme-ntation	Software	Software FPGA [18]	SFQ	SFQ	SFQ
Cryogenic environment			✓	✓	✓
Measurement error	✓	✓		✓	✓
Online decoding				✓	✓
Lattice surgery	✓ [13]	(✓) (Software)[1)]			✓

done after all the measuring processes. We stated that online-QEC is preferred for realistic QEC in the previous paper [21].

Though these implementations are effective, all of them focus only on the QEC of a single logical qubit and do not support any logical operation. It is necessary to study whether a decoding algorithm that can support a set of universal logical gates can be implemented efficiently. In this paper, we develop a new algorithm capable of processing LS operations and construct an online-QEC architecture with it. We also show that our architecture can process the merge-and-split operations required in the magic-state distillation protocols with reasonable latency and power consumption.

It should be noted that a universal FTQC can also be achieved with other strategies than LS. For example, virtualized logical qubit, which is a brand-new superconducting qubit architecture [12], enables transversal logical-CNOT gates by achieving 2.5-D architecture using multi-mode cavities [44]. Note that these approaches do not conflict with our proposal, and they are expected to be even efficient when combined.

IV. SPIKE BASED ONLINE DECODING ALGORITHM

A. Base decoding algorithm

We first describe how to decode an SC lattice of a single logical qubit with measurement errors with the QULATIS algorithm based on QECOOL [21]. This algorithm finds another hot syndrome closest to a given hot syndrome by propagating signals between distributed decoding units.

We introduce a decoding unit named the *united line module* (ULM). It is associated with each horizontal or vertical line of ancilla qubits in an SC lattice, while the *Units* of QECOOL and the ancilla qubits are in one-to-one correspondence. For Z-error (X-error) correction, each ULM is vertically (horizontally) connected to 1 or 2 neighboring ULMs forming a columnwise (rowwise) ULM set and has a queue-like memory module (Mem) to store a 2-D array of ancilla qubit measurement values. Figure 4 shows the mapping of an X-stabilizer lattice to vertically aligned ULMs. We define a *Controller* module to orchestrate all the ULMs corresponding to a syndrome lattice of a logical qubit.

The QULATIS algorithm is shown in Algorithm 1. In the algorithm, *MeasureEachUnit* and *RestartUnit* correspond to the measurement phase and decoding phase, respectively.

[1)]Although we expect LS is implementable by modifying the UF algorithm, no implementation has been proposed so far.

Algorithm 1 QULATIS: Spike-based online QEC for 3-D Surface code

```
Controller:
start_loop:
for C = 1 to N_limit do
  for b = 0 to N_depth do
    resetFlag()
    shift = true
    for i = 0 to N_ULM do
      if m − b > th_v then
        giveToken(i)
        for x = 0 to N_mem do
          RestartUnit(b, x)
        end for
        while !getFinish()&&!Timeout()
      end if
      ULM(i).FlagToken = 1
      shift& = !isAllZero(
        ULM(i).Mem[0][:])
    end for
    if shift then
      SHIFTMEM()
      goto start_loop
    end if
  end for
end for

procedure SPIKE(flag)
  if flag == 1 then
    sendSpikeSouth();
  else
    sendSpikeNorth();
  end if
end procedure

procedure SHIFTMEM
  if m > 0 then
    for i = 0 to N_depth − 1 do
      Mem[i]=Mem[i+1]
    end for
    m = m − 1
  end if
end procedure

procedure CHECKMEM(b, Δ)
  for δ_t = 0 to Δ do
    δ_x = Δ − δ_t
    if x + δ_x == N_mem
        or x − δ_x == −1
        or b + δ_t == m then
      Return BoundaryFlag
    else
      t' = b + δ_t
      if mem[t'][x + δ_x] == 1 then
        x' = x + δ_x
        Return x'
      else
        if mem[t'][x − δ_x] == 1 then
          x' = x − δ_x
          Return x'
        end if
      end if
    end if
  end for
end procedure

RestartUnit(Input: b, x)
if Token == 1 then
  if Mem[b][x] == 1 then
    requestSpike(b, x)
    Δ = 0
    while true do
      if (S = getSpike()) != NULL then
        Dir = rotate(S)
        correctVertical(Dir, x)
        sendAcknowledge(Dir)
        Mem[b][x] = 0
      end if
      if ( x' = CHECKMEM(b, Δ))
          != NULL && Δ > 0 then
        correctHorizontal(x, x')
        Mem[b][x] = 0
        Mem[t'][x'] = 0
        sendController("Finish")
      end if
      Δ+ = 1
    end while
  else
    sendController("Finish")
  end if
else
  Δ = 0
  while true do
    if ( x' = CHECKMEM(b, Δ))
        != NULL then
      SPIKE(FlagToken)
      if getCorrect() then
        mem[t'][x'] = 0
        correctHorizontal(x, x')
        sendController("Finish")
      end if
    else
      if (S = getSpike()) != NULL then
        Dir = rotate(S)
        SPIKE(FlagToken)
        if getCorrect() then
          correctVertical(Dir, x)
          sendAcknowledge(Dir)
        end if
      end if
    end if
    Δ+ = 1
  end while
end if

MeasureEachUnit:
m = 0
while true do
  A = checkAncilla()
  for x = 0 to N_mem do
    if m == 0 then
      Mem[0][x] = A
    else
      Mem[m][x] = Mem[m-1][x] ⊕ A
    end if
  end for
  m = m + 1
  Sleep(Mcycle)
end while
```

Figure 5 (b) and (c) also show the flowcharts of *Controller* and *RestartUnit*, respectively.

Whenever each ancilla qubit is measured, its value (0 and 1 indicate trivial and hot, respectively) is pushed into the appropriate location in *Mem[t][x]* of the associated ULM (MeasureEachUnit in Algorithm 1). For example, as shown in Fig. 4, the top-left ancilla qubit illustrated as a blue inverted triangle located on the top-left corner under the lattice is associated with ULM0, and its first measurement value is stored in Mem[0][0] of ULM0. Similarly, the second measurement

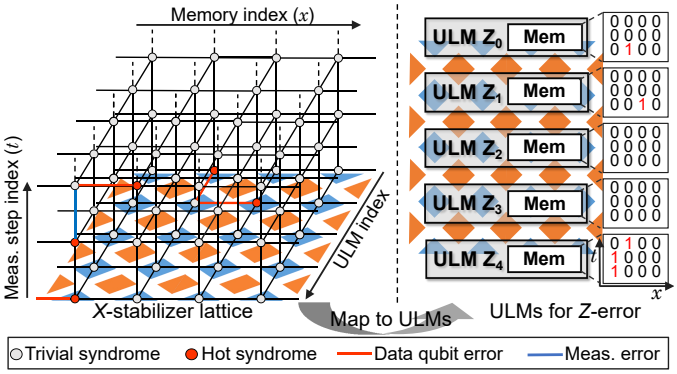


Fig. 4. Mapping an X-stabilizer lattice (left side) to ULMs (right side).

value of the bottom-right ancilla qubit is stored in Mem[1][3] of ULM4. When the values of Mem[0][:] (corresponding to the oldest measurements) of all the ULMs become 0, which means that no more decoding processes are required for the bottom layer of the lattice, each Mem is shifted by 1-bit (L.16 to 20 of Controller procedure and ShiftMem procedure in Algorithm 1).

In the decoding phase, our algorithm proceeds as follows. First, the algorithm determines the *root hot syndrome*, which is the starting point for finding a hot syndrome pair. Next, another hot syndrome is found by using two types of signals named *Spike* and *Acknowledge* between each ULM. Finally, the algorithm clears the paired hot syndromes, corrects the physical qubits between the hot syndromes, and searches the next root hot syndrome. These are visualized in Fig. 5 (a), and the details of each process are as follows.

(1) Determining the root hot syndrome: The Controller assigns *Token* to each ULM from the top (North), and a ULM with Token checks its own Mem[b][x] cycle-by-cycle, where x and b indicate the ancilla qubit position index and the base depth of the matching search, respectively (Steps 1 and 2 in Fig. 5 (a)). If the Mem[b][x] is 1, the corresponding syndrome value and its owner become the root hot syndrome and the *root ULM*, respectively.

(2) Spike signal propagation: The root ULM requests all the other ULMs to send a *Spike* signal toward itself. The ULMs, including the root, perform a cycle-by-cycle search for nearest '1' in Mem by using memory indices of b and x as a query point as follows (Steps 3 and 4 in the figure). First, the ULM checks Mem[b][x]. If it is not '1', in the next cycle, the ULM simultaneously references the elements of Mem whose Hamming distance is 1 from the query (*e.g.*, Mem[b][$x+1$], Mem[b][$x-1$] and Mem[$b+1$][x]). The above procedure is repeated cycle-by-cycle, increasing the Hamming distance until the ULM finds '1' or gets a Spike signal from other ULM or receives a *TimeOut* signal. After finding Mem[t'][x'] with value '1' or getting a Spike signal, the non-root ULM sends a Spike signal. The Spike direction is determined by the *FlagToken* indicating whether the Token has already been passed (see SPIKE procedure in Algorithm 1). The source of the Spike signal that first arrives at the root ULM is called the *target ULM*. If the root ULM finds Mem[t'][x'] with value '1' before getting a Spike or a Timeout, the propagation of Spike and Acknowledge signals is skipped, and the physical qubits are corrected while the root ULM also acts as the target ULM.

(3) Acknowledge signal propagation: If the first Spike comes to the root ULM, it sends an *Acknowledge* signal in the direction that the Spike came from (Step 5 in the figure). The Acknowledge signal is passed toward the target ULM. These two types of signals form a request-grant mechanism, which prevents more than two hot syndromes from being matched simultaneously.

(4) Correcting physical qubits: Each ULM that sends an Acknowledge signal corrects the x-th data qubit that is sandwiched between itself and the ULM to which the Acknowledge signal is sent (*correctVertical(Dir, x)*). The root ULM also clears the value of Mem[b][x] while sending Acknowledge signal. After getting an Acknowledge signal, the target ULM clears the value of Mem[t'][x'] and corrects the horizontally aligned data qubits that are between x and x' (*CorrectHorizontal(x, x')*).

If an error syndrome from one of the ULMs in the SC goes toward the outside, there is no proper matching pair among erroneous ancilla qubits. To deal with such a case, we use "out-of-bounds array access" of the root ULM in process 2) for boundary matching. If Mem[t][x] is the root hot syndrome, the root ULM attempts to check Mem[t][-1] (Mem[t][N_{mem}]) at the ($x+1$)-th (($N_{mem}-x$)-th) cycle if no Spike or TimeOut signals come before that (L.4 to 9 of CheckMem procedure in Algorithm 1). We consider this case to be left-boundary (right-boundary) matching, and the root ULM corrects all data qubits to the left (right) of x[2]. Boundaries are flexibly controlled by the contents of Mem, and this flexible boundary matching is the main difference between our algorithm and the previous decoders such as QECOOL [21] and AQEC [20]. This allows QULATIS to cope with LS processing as described in the next subsection.

This algorithm finds the closest hot syndrome to the root hot syndrome. However, it does not necessarily find a set of pairs that globally minimizes the total distance because the root hot syndrome is sequentially determined. To increase the possibility to find the global minimum, we limit the maximum number of hops to propagate a Spike in a single run of the above processes and increase it iteratively. The *Controller* procedure in Algorithm 1 implements this by setting a Timeout after restarting the ULMs. As a result, the QULATIS algorithm finds a matching similar to the greedy matching shown in Fig. 1. of Reference [45], and its performance is lower than that of the MWPM decoder.

As with the QECOOL algorithm, our QULATIS algorithm controls the online QEC process with the parameter th_v, which means the maximum length or threshold in the vertical direction to search for a pair of hot syndromes.

[2]To prioritize matching between hot syndromes, the timing for determining boundary matching needs to be adjusted in the implementation.

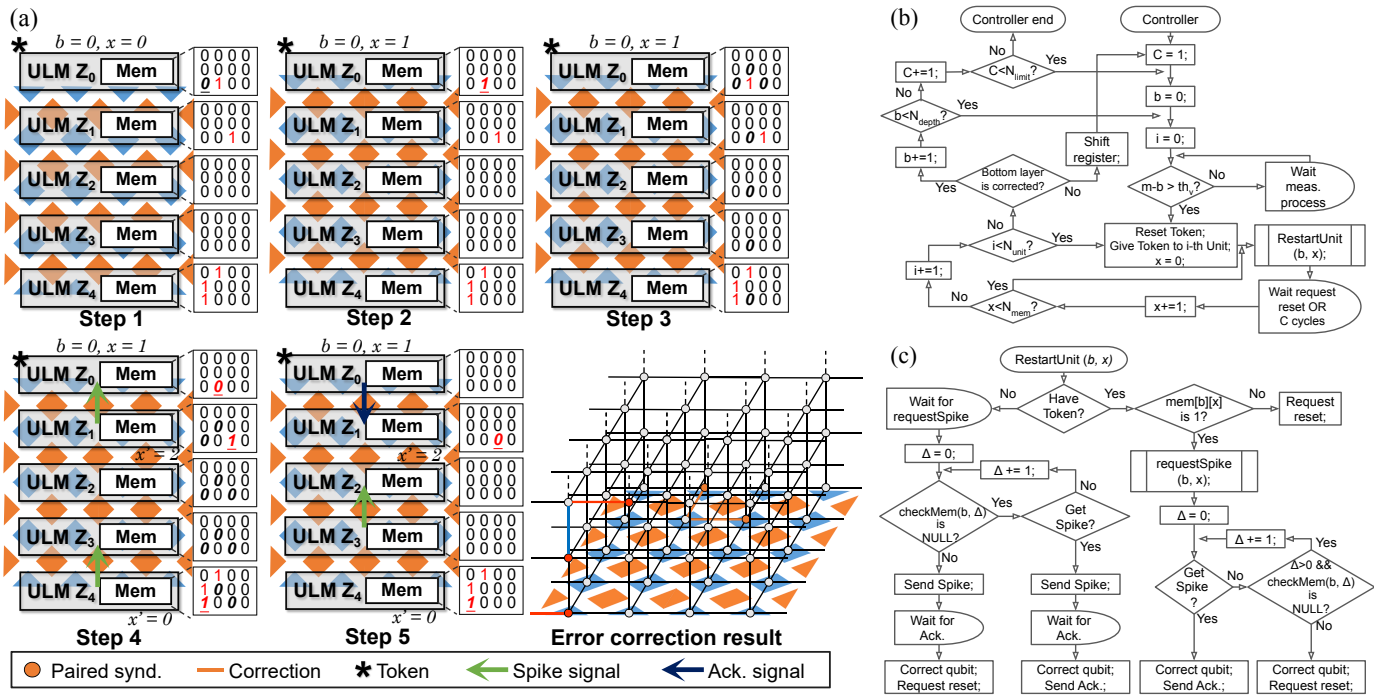


Fig. 5. (a) QULATIS process for error correction of single logical qubit. (b) Flowchart of Controller procedure. (c) Flowchart of RestartUnit procedure.

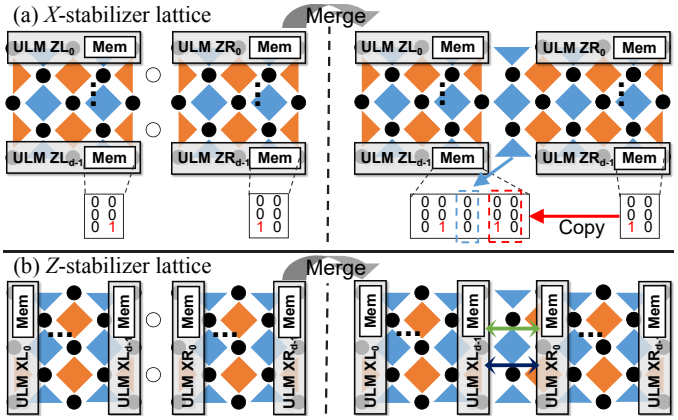


Fig. 6. The merge operation for X- and Z-stabilizer lattices with ULMs.

B. QULATIS for merge and split operations

Here, we describe how to use the QULATIS algorithm to perform the merge-and-split operations of the LS process. We consider the same situation as in Fig. 2, where two adjacent logical qubits are to be merged. Note that the previously proposed hardware decoders are not applicable to this operation because of their hardware limitation.

Figure 6 illustrates an overview of the merge operation for X- and Z-stabilizer lattices with ULMs, and Table II shows the parameters in Algorithm 1 that have different values for single logical qubit decoding and merge operation. The merge operation is realized on the QULATIS by setting the contents of each Mem appropriately, or by connecting two pairs of ULMs in series, and then executing Algorithm 1 with the parameters changed as Table II.

(a) X-stabilizer lattice: Before the merge operation, two sets of vertically aligned ULMs are responsible for Z-error correction of the left and right logical qubits, respectively. The merge operation is performed by copying the memory contents of the right ULMs to the corresponding left ULMs and operating the left set of ULMs as described in Algorithm 1. In other words, the ULMs treat the merged logical qubits as a single wide lattice by expanding the range of ancilla qubits managed by a single ULM. Note that additional ancilla qubits are placed between the logical qubits in a merge operation to connect their rough boundaries. The measurement values of the new ancilla qubits must be stored in the memory of the appropriate ULM. Therefore, the value of N_{mem} is doubled and added by one compared to the case of a single logical qubit.

(b) Z-stabilizer lattice: Two sets of ULMs arranged in a row correct an X-error on the left and right logical qubits before merge operation. A merge operation of the Z-stabilizer lattice can be achieved simply by connecting two sets of ULMs in series. Therefore, N_{ULM} has a value twice the code distance d. There are no additional ancilla qubits for Z-stabilizer measurements between the logical qubits, unlike the case of X-stabilizer, and N_{mem} has the same value as in the case of a single logical qubit.

V. QULATIS ARCHITECTURE FOR LATTICE SURGERY

In this section, we describe in detail our FTQC architecture supporting LS with the QULATIS algorithm for a 2-D grid of physical qubits with a 3-D SC decoding capability. First, we give an overview of our FTQC architecture. Second, we

TABLE II
PARAMETERS IN ALGORITHM 1 FOR CODE DISTANCE d

	Single logical qubit	Merge operation (X-stabilizer lattice)	Merge operation (Z-stabilizer lattice)
N_{mem}	$d-1$	$2(d-1)+1$	$d-1$
N_{ULM}	d	d	$2d$

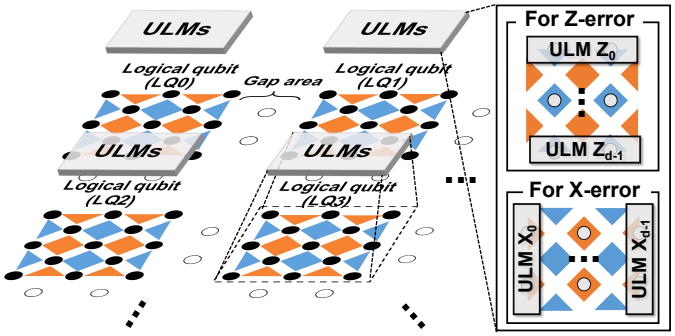


Fig. 7. Overview of our FTQC architecture with QULATIS and ULMs.

describe the detailed functions and design of the key building block of the architecture referred to as ULM. Finally, we implement it using SFQ digital circuits and evaluate its circuit characteristics.

A. Overview of the FTQC Architecture based on QULATIS

Figure 7 shows the overview of the proposed FTQC architecture based on the QULATIS algorithm. Our architecture consists of a 2-D grid of physical qubits, pairs of decoding units (ULMs), and a controller to orchestrate the ULMs. The detailed functionalities and design of the ULM are described in the following subsections. The array of physical qubits is divided into several logical qubits coordinated by SC with a predetermined code distance d with appropriate spacing. Each logical qubit has two sets of ULMs for X- and Z-error.

This architecture can perform not only an error correction of a single logical qubit but also a merge operation between logical qubits. While no merge operation is performed, the sets of ULMs operate independently in parallel; each is responsible for the error correction of the corresponding logical qubit. During a merge operation, the two sets of ULMs for logical qubits corresponding to the merge operation act cooperatively by copying memory contents from one to another or connecting two sets of ULMs in a series as described in Section IV-B. The other ULMs operate independently as usual.

The QULATIS architecture also supports Pauli measurement operations on more than three logical qubits by cooperating corresponding ULMs, as in the case of the merge operation of two logical qubits. For example, in the case of merging logical qubits No. 0 to No. 3 in Fig. 7, the process of Z-error correction can be realized by copying the memory contents from LQ1 to LQ0 and LQ3 to LQ2 and by connecting the ULM of LQ0 and LQ2 in a series.

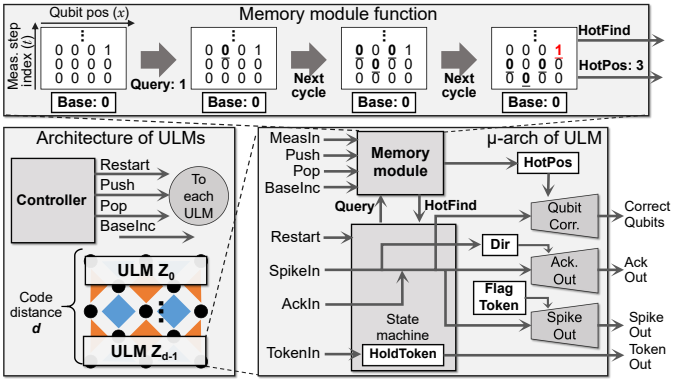


Fig. 8. Overview of the architecture of ULM and memory module function.

B. Microarchitecture of ULM

Figure 8 illustrates an overview of the hardware architecture of ULMs for detecting and correcting Z-errors for a single logical qubit. The ULMs are arranged in a column, and the SC boundaries are placed on the left and right sides of the ULMs. A single ULM manages a row of data qubits connected horizontally through the X-stabilizer, while data qubits connected vertically are managed by multiple ULMs. Note that in the hardware architecture for X-error correction, the ULMs' hardware blocks shown in Fig. 8 are rotated 90 degrees and arranged in a row.

There is one Controller in each group of ULMs for X- or Z-error correction for a logical qubit. It orchestrates the corresponding ULMs by passing them the Token and distributing the control signals. Whenever the ancilla bits are measured, the Controller sends the Push signal to all ULMs to store the measured values in their memory modules. If there are enough valid syndrome data in the memory module, the Controller starts the decoding process by sending Restart signal to all the ULMs. During the decoding phase, when the Token returns from the ULM to the Controller and the error correction process for the current layer has been completed, it sends a Pop signal to the ULMs to shift the memory contents; otherwise, it broadcasts the BaseInc signal to start the decoding process in one layer above the current layer.

The previous studies [20], [21], [46] used hardware modules, named "boundary module" or "boundary unit" to pair a hot syndrome with boundaries, and the modules cannot handle dynamic boundary changes required for LS procedures. On the other hand, a boundary is flexibly controlled in QULATIS by storing syndrome values of an SC patch into the memory appropriately, enabling the proposed circuit to merge and split logical qubits.

Each ULM is composed of the five components as below.

(1) State machine: The state machine controls the behavior of the ULM. The state transition is based on reset signals from the Controller, the Token, Spike signals from other ULMs, and referenced memory values.

(2) Memory module: This module has a two-dimensional memory that holds a time series of measurement values of

each ancilla qubit managed by the ULM. This module performs a cycle-by-cycle minimum Manhattan-distance search by reading the memory location spreading from the point-of-interest (query point) to around. If the memory value of '1' is found, a signal named "FindHot" is sent to the state machine, telling it to proceed with the matching process. It also sends information of the position of the corresponding ancilla qubit to the Qubit correction module to specify the data qubit for error correction. The query point is determined by a "Query" signal from the state machine that points to the position of an ancilla qubit and a "Base" register that indicates the starting layer of the lattice for matching.

(3) Spike out module: This module sends a Spike to another ULM, and its direction is determined by a 1-bit flag named "FlagToken".

(4) Acknowledge out module: This module sends an Acknowledge signal to one of adjacent ULMs from which it received the most recent Spike signal.

(5) Qubit correction module: This module corrects data qubits against observed errors. To correct appropriate qubits, it has a buffer to hold the position of the qubit to be corrected. The size of the buffer depends on the code distance d of the QEC architecture. It can generate two types of qubit correction signal, "CorrectHorizontal" and "CorrectVertical" to correct data qubits connected horizontally via the stabilizer and data qubits between two ULMs, respectively.

C. ULM implementation based on RSFQ logic

We designed the ULM hardware for $d = 9$ SC decoding except the memory module using the existing, well-developed cell library [47]. The library is with a conventional logic family of rapid single flux quantum (RSFQ) targeting a niobium nine-layer, 1.0-μm fabrication technology [48], [49] for 4-K temperature operation. The memory module is assumed to be a 64-kb hybrid Josephson-CMOS memory array proposed by Van Duzer *et al.* [50]. Its readout delay and power consumption at 1 GHz operation frequency are 400 ps and 13 mW, respectively. As we explain later, this memory device is sufficient for a memory module of QULATIS architecture.

Table III summarizes the SFQ logic gates used in this work. A JJ in an SFQ circuit is the basic switching element like a transistor in traditional circuits, and they are the main determinant of the power consumption and hardware cost. The table shows the number of JJs for each gate and the assumed bias current required for operation when the designed supply voltage is 2.5 mV. In RSFQ, since most of the power is consumed statically, almost independently of switching activity, it is calculated by multiplying the bias voltage and currents.

We used the Josephson simulator (JSIM) [51], which is a SPICE-level simulator, to verify the functionality of the designed ULM except for the memory module. Table IV shows the total number of JJs, total area, total bias current, and latency of each submodule of the ULM. Figure 9 shows the layout of our design. The designed circuit consists of 2412 JJs in total, and its area footprint is $0.8892\ \mathrm{mm}^2$.

TABLE III
SUMMARY OF SFQ LOGIC ELEMENTS

cell	JJs	Bias current (mA)	Area (μm^2)	Latency (ps)
splitter	3	0.300	900	4.3
merger	7	0.880	900	8.2
1:2 switch	33	3.464	8100	10.5
destructive readout (DRO)	7	0.689	1800	5.4
resettable DRO (RD)	11	0.901	900	6.0
dual-output DRO (D2)	12	0.943	900	6.5
XOR	11	1.068	3600	6.5

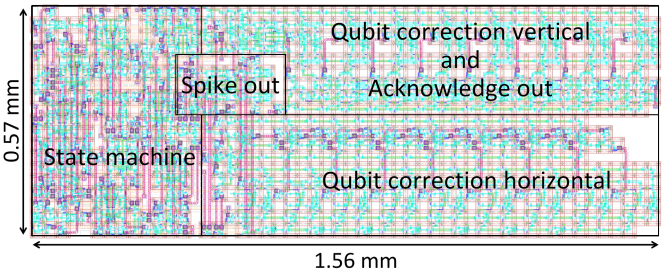


Fig. 9. Layout of the part of ULM designed based on RSFQ logic.

The latency of the logic part of ULM is 157.5 ps and, it is overlapped by the memory module readout. Therefore, the critical path of the entire ULM is the memory access path whose delay is 400 ps [50], resulting in a theoretical maximum operating frequency of 2.5 GHz. We pessimistically assume the proposed architecture operates at 2 GHz in the rest of this paper.

D. Power estimation with ERSFQ logic

To achieve a lower power decoder, we use the more energy-efficient technology, called the energy-efficient RSFQ (ERSFQ) [52], instead of RSFQ logic. In conventional RSFQ, most power is consumed by bias feed resistors statically. On the other hand, in ERSFQ, bias resistors are replaced with JJs, by which we can eliminate the large static power consumption in exchange for doubled dynamic power consumption by JJs. Because the JJ's dynamic energy consumption is roughly represented by bias-current $\times\Phi_0$ per switch, we estimated the power consumption with ERSFQ based on the RSFQ design and the power model of ERSFQ [53]. Here, the power of a ULM with ERSFQ can be estimated as follows: $P_{unit} = (\text{bias current}) \times (\text{frequency}) \times \Phi_0 \times 2$.

For the designed ULM for $d = 9$ SC decoding, the total bias current is 250.7 mA. If we suppose a 2 GHz clock frequency, the power consumption of a ULM in a 4-K environment is estimated as follows: $250.7_{[\mathrm{mA}]} \times 2_{[\mathrm{GHz}]} \times (2.068 \times 10^{-15})_{[\mathrm{Wb}]} \times 2 = \mathbf{2.07}_{[\mu\mathrm{W}]}$. By combining this circuit with the 64-kb memory module [50], we can construct a ULM and estimate the power consumption required for an architecture composed of ULMs.

TABLE IV
TOTAL NUMBER OF LOGIC ELEMENTS, NUMBER OF JJS, CIRCUIT AREA AND LATENCY OF EACH MODULE THAT CONSTITUTES THE PART OF THE ULM BASED ON THE AIST 10-KA/CM^2 ADP CELL LIBRARY [47].

cell	State machine	Qubit Correction vertical ($d = 9$)	Qubit Correction horizontal ($d = 9$)	Spike out	Other	Total ($d = 9$)
splitter	39	55	54	2	9	159
merger	12	8	15		3	38
1:2 switch	8	1		1		10
DRO		1				1
RD	8	15	9	2		34
D2		9	8			17
XOR			9			9
Wire	285	396	379	22	50	1132
Total JJs	721	765	778	77	71	2412
Total area (μm^2)	230400	276300	342000	40500		889200
Total bias current (mA)	74.2	79.2	82.3	7.03	7.94	250.7
Latency (ps)	65.2	83.4	73.3	18.4		157.5

VI. PERFORMANCE EVALUATION

A. Evaluation setup

In this section, we numerically evaluate the performance of the QULATIS algorithm and its architecture implementation. We assume a noise model where Pauli-X and -Z errors occur on each data and ancillary qubit independently with a uniform physical error probability p. Note that while the actual noise of qubits would be locally correlated and coherent, the surface codes are expected to correct these errors with a modest performance degradation [54], [55]. Thus, this evaluation captures the essential performance of our proposal. For a given physical error rate, we repetitively sample the error patterns, simulate the propagation of errors and decoding procedure, and evaluate the probability of logical failure. While a full simulation of quantum circuits requires the time exponentially scaling with the number of qubits, we can efficiently simulate the propagation of Pauli errors since we assume Pauli errors and QEC circuits consist of Clifford gates and Pauli measurements [56].

The latency of the MWPM decoder, which is implemented with Blossom V library [57], is measured by a PC using a single core of Intel Xeon Gold 6240R and 96 GB RAM. Note that the MWPM decoder latency contains only the time for solving MWPM problems, excluding graph construction. The latencies of QECOOL and QULATIS are measured by their total execution cycles multiplied by the clock cycle time of 0.5 ns, assuming 2 GHz clock frequency.

B. Error correction performance for a single logical qubit

First, we show the performance of the QULATIS algorithm for the idling operations for a single logical qubit during d code cycles assuming the existence of measurement errors and compare it with the existing decoding algorithms. To this end, we compare the batch execution of the QULATIS algorithm with the existing batch decoding algorithms with a checkmark in 'Measurement error' in Tab. I: MWPM, UF, and QECOOL.

Figure 10 shows the error correction performances of the four batch decoding algorithms. The threshold behavior is observed for all decoders. The threshold is about 5% for MWPM, 4% for UF, and 2% for QULATIS and QECOOL algorithms. Thus, instead of being compatible with online decoding, the threshold values of the online decoders are twice as small as those of the MWPM and UF.

TABLE V
COMPARISON OF BATCH-QECOOL AND QULATIS FOR THE NUMBER OF EXECUTION CYCLES PER LAYER DURING ERROR CORRECTION FOR SINGLE LOGICAL QUBIT.

		$p = 0.001$			$p = 0.01$		
Algorithm	d	Max.	Avg.	σ	Max.	Avg.	σ
Batch-QECOOL	5	105	5.75	4.12	148	12.2	12.8
	7	341	10.2	11.9	551	35.0	34.1
	9	727	18.3	25.7	1017	80.1	70.1
	11	1107	32.2	46.1	2098	154	129
Batch-QULATIS	5	90	2.54	3.28	112	7.15	10.2
	7	111	3.20	4.96	220	12.9	15.5
	9	164	4.24	6.93	259	20.9	22.4
	11	224	5.70	9.15	367	30.9	31.3

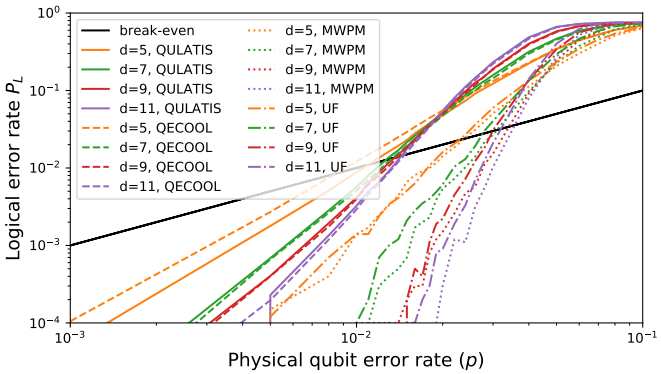


Fig. 10. Performance comparison of batch-QECOOL, batch-QULATIS, UF and MWPM for error correction of single logical qubit.

Comparing QULATIS with QECOOL, the performance of the QULATIS is slightly better than QECOOL at $d = 5$, and they are almost the same in the other cases. Table V lists the per-layer execution cycles of QECOOL and QULATIS for several p and d combinations. The execution cycles are not shown in the table since the MWPM decoder is not an online decoder. The per-layer execution cycles of QULATIS are less than those of QECOOL because of the reduction in unnecessary Token exchanges (L.9 of Controller procedure in Algorithm 1). This trend is prominent for larger d.

The trade-off between the accuracy and latency for decoding a single logical qubit is shown in Fig. 11 (a). Note that QECOOL and QULATIS are executed with batch-manner for a fair comparison with MWPM, which cannot work in an online manner. The physical error rate p is assumed to be 0.001, and the target latency is set to d μs, which is required for online decoding. While QULATIS has the lowest latency, even MWPM can decode a single logical qubit with enough lower latency than the target latency since the assumed p is fairly small. In addition, there is no significant difference in the accuracy for different decoders.

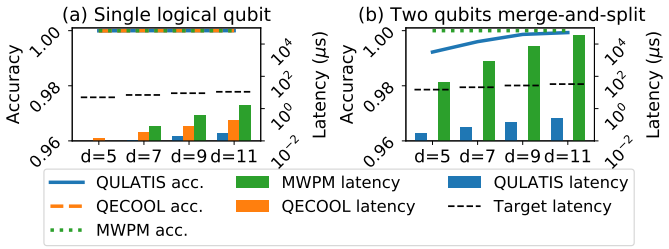


Fig. 11. Trade-off between accuracy and latency for decoding (a) single logical qubit, and (b) two qubits merge-and-split operations. Physical error rate p is assumed to be 0.001, and all the algorithm are executed with batch-manner.

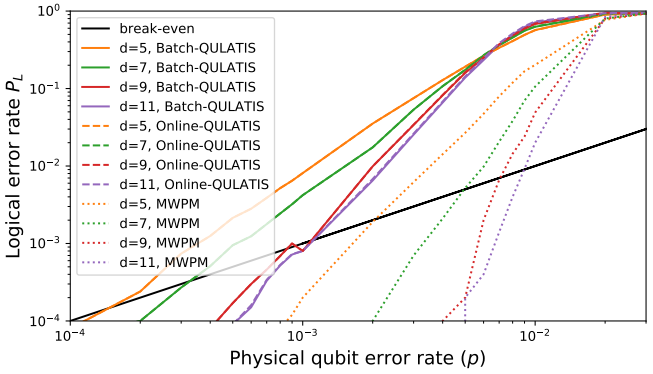


Fig. 12. Logical error rate performance of MWPM, batch- and online-QULATIS for merge-and-split operations.

C. Error correction performance for merge-and-split operations of two logical qubits

Next, we evaluate the error correction performance of the QULATIS algorithm for merge-and-split operations on two logical qubits described in Fig. 2. We performed the benchmark on the $3d$-cycle merge-and-split operations that consist of d-cycle split phase, d-cycle merge phase, and d-cycle split phase. To the best of our knowledge, no implementation of the online decoder that is compatible with LS operations has been known. Thus, we compare the performances of online-QULATIS algorithms with two decoders, the MWPM and batch-QULATIS decoders, for evaluating the performance penalty of the online execution. Note that since the QECOOL[3)]and UF do not support LS, no performance comparisons with them are provided in the following parts.

Figure 12 shows the error rate scaling for the QULATIS and MWPM. The MWPM and batch-QULATIS algorithms process the decoding lattices after the whole process. On the other hand, in the evaluation of online-QULATIS, we assume that the measurement process is performed every 1 μs, and if the algorithm cannot catch up with the cycle of stabilizer measurements, the trial is assumed to be a failure. Thus, the difference between online- and batch-QULATIS represents the penalty of the online execution. Here, we set $th_v = 3$, $N_{depth} = 7$, and the operating frequency to 2 GHz, which is the same as the existing work [21]. According to the figure, the performance degradation of the online-QULATIS compared with the batch-QULATIS is negligibly small. We observe the threshold value around $p = 0.6\%$ for the QULATIS and $p = 2\%$ for the MWPM.

Figure 11 (b) shows the trade-off between the accuracy and the latency for a merge-and-split operation of two logical qubits. Here, we assume the target latency is $3d$ μs and p is 0.001. Note that there are no results of QECOOL because it does not support that operation. The result shows that the latency of MWPM is larger than the target latency, even if the code distance is small. On the other hand, QULATIS runs fast enough for any code distances. Although the accuracy of QULATIS is lower than that of MWPW, their difference becomes very small as the code distance increases. We again emphasize that the MWPM algorithm is a software decoder and huge latency due to its batch processing manner is critical in practice even though it has high accuracy.

D. Error correction performance of online-QULATIS for 15-to-1 magic-state distillation

Finally, we evaluate the error correction performance of the online-QULATIS algorithm for magic-state distillation circuits. As described in Section II-C, the magic-state distillation circuit consists of five eight-qubit Pauli-X measurements on 16 logical qubits and consumes the space of 24 SC patches, and this subroutine is expected as one of the largest merge-and-split operations in the practical FTQC. To reduce the complexity of the performance evaluation, we numerically calculate the logical error probabilities for each of the five eight-qubit merge-and-split operations independently and evaluate the total logical error probability from them.

Figure 13 shows the error correction performance of the QULATIS algorithm for the magic-state distillation circuit. The architecture runs at 2 GHz, and the parameters for online-QEC are set as $th_v = 3$ and $N_{depth} = 7$. The figure shows that the performance improves as the code distance d increases up to $d = 7$. Thus, we can conclude that the online-QULATIS algorithm can suppress the logical errors according to the code distance, even for the magic-state distillation circuits. On the other hand, the dropping rate of the logical error rate according to the code distance becomes small at $d = 9$. This is because the size of the buffer storing the measurement values is too small for processing. By increasing the buffer size according to the required logical error rates, the QULATIS algorithm would work even for long code distances.

Table VI shows the number of execution cycles per layer of Online-QULATIS for several combinations of coding distance d and physical error rate p. The parameters for online processing are set to the same values as above. In this case,

[3)]If the QECOOL decoder is straightforwardly applied to merge-and-split operations, it cannot handle a hot syndrome at the sandwiched area during merging. The syndrome will be randomly matched to either the left or right boundary, causing a logical error with a 50% probability. As a result, physical errors at the gap vicinity contribute to a constant logical error rate, whereas the remaining part exhibits a lower logical error rate with increasing code distance.

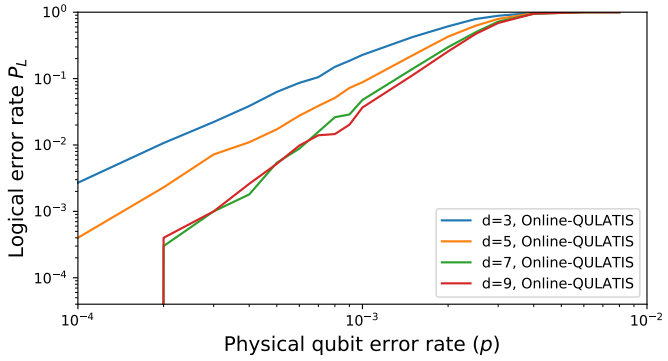


Fig. 13. Error correction performance of Online-QULATIS for magic-state distillation circuit.

TABLE VI
PER-LAYER EXECUTION CYCLES OF ONLINE-QULATIS FOR MAGIC-STATE DISTILLATION CIRCUIT.

	$p = 0.0001$			$p = 0.0005$			$p = 0.001$		
d	Max.	Avg.	σ	Max.	Avg.	σ	Max.	Avg.	σ
3	318	5.60	18.6	456	20.7	44.5	455	40.4	65.3
5	575	12.7	35.4	777	50.9	81.8	812	93.4	120
7	968	23.5	55.3	1205	83.5	127	1266	145	187
9	1539	37.9	79.4	1689	121	185	1913	212	274

the execution cycles of QULATIS are highly dependent on d and p, and the latency of QULATIS is suitable for online processing if the physical error rate p is sufficiently small for coding distances up to 9.

E. Power consumption and scalability of QULATIS

We estimate the number of protectable logical qubits with QULATIS architecture in a cryogenic environment in terms of power consumption. We assume that the code distance, the operating frequency, and N_{depth} are 9, 2 GHz, and 7, respectively. The contribution of the controllers to the power consumption becomes relatively small and negligible as d and the number of logical qubits increases. Each box of "ULMs" in Fig. 7 has 18 ULMs, and each ULM stores the measurement values of 8 ancillary qubits. Hence, the total amount of memory size of ULMs required for an SC patch is $7_{\text{[bit/qubit]}} \times 8_{\text{[qubits/ULM]}} \times 18_{\text{[ULMs/patch]}} = 1008_{\text{[bit/patch]}}$. Assuming that a single 64-kbit memory device [50] can be shared by $64_{\text{[kbit]}}/1008_{\text{[bit/patch]}} \approx 63$ patches, the total power consumption of ULMs required for 63 logical qubits is $2.07_{[\mu\text{W/ULM}]} \times 18_{\text{[ULMs/patch]}} \times 63_{\text{[patches]}} + 12_{\text{[mW/GHz]}} \times 2_{\text{[GHz]}} \approx 26.3$ mW. The power budget of the 4-K temperature region of a dilution refrigerator is supposed to be 1 W [8]. Thus, we expect that $1_{\text{[W]}}/(26.3/63)_{\text{[mW/patch]}} \approx$ **2395** logical qubits can be protected in a cryogenic environment with our architecture, which is comparable to QECOOL [21].

Next, we estimate the power consumption of the QULATIS architecture that supports a magic-state distillation circuit. From the results shown in Fig. 13, we assume that N_{depth} is 16, which is about twice as large as the conventional setup, and the other setup is the same as above. As shown in Fig. 16. of Ref. [11], a distillation circuit requires 24 patches of SC. The total amount of memory size required for the architecture supporting a distillation circuit is $16_{\text{[bit/qubit]}} \times 8_{\text{[qubits/ULM]}} \times 18_{\text{[ULMs/patch]}} \times 24_{\text{[patches]}} = 55296$ bit, which is lower than 64 kbit. Assuming that a single memory device [50] can be shared by all ULMs in the architecture, the total power consumption of the architecture is $2.07_{[\mu\text{W / ULM}]} \times 18_{\text{[ULMs / patch]}} \times 24_{\text{[patches]}} + 12_{\text{[mW/GHz]}} \times 2_{\text{[GHz]}} \approx 25$ mW. Thus, we expect that our architecture supports $1_{\text{[W]}}/25_{\text{[mW/distillation circuit]}} = 40$ magic-state distillation circuits with code distance $d = 9$ in such an environment.

VII. CONCLUSION

In this paper, we proposed QULATIS, a decoding algorithm and an FTQC architecture that supports lattice surgery. We designed a key building block of the architecture with superconducting circuits and evaluated the performance and power consumption of the circuit. We evaluated the error correction performance of the proposed architecture for a merge-and-split operation of two logical qubits and a magic-state distillation circuit in a quantum error simulator. The results show that our architecture has sufficient error correction performance to support the distillation circuit, and it is power-efficient enough to operate in the 4-K layer of a dilution refrigerator.

ACKNOWLEDGMENT

This work was supported by JST Mirai Program Grant Number JPMJMI17E1, JST CREST Grant Number JPMJCR18K1, JST PRESTO Grant Number JPMJPR1916, JST ERATO Grant Number JPMJER1601, MEXT Quantum Leap Flagship Program Grant Numbers JPMXS0120319794, JPMXS0118068682, JST Moonshot R&D Grant Number JPMJMS2061, and the ANRI fellowship.

REFERENCES

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. Bardin, R. Barends, R. Biswas, S. Boixo, F. Brandao, D. Buell, B. Burkett, Y. Chen, J. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. M. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. Harrigan, M. Hartmann, A. Ho, M. R. Hoffmann, T. Huang, T. Humble, S. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. Martinis, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, p. 505–510, 2019.

[2] M. Gong, S. Wang, C. Zha, M.-C. Chen, H.-L. Huang, Y. Wu, Q. Zhu, Y. Zhao, S. Li, S. Guo, H. Qian, Y. Ye, F. Chen, C. Ying, J. Yu, D. Fan, D. Wu, H. Su, H. Deng, H. Rong, K. Zhang, S. Cao, J. Lin, Y. Xu, L. Sun, C. Guo, N. Li, F. Liang, V. M. Bastidas, K. Nemoto, W. J. Munro, Y.-H. Huo, C.-Y. Lu, C.-Z. Peng, X. Zhu, and J.-W. Pan, "Quantum walks on a programmable two-dimensional 62-qubit superconducting processor," *Science*, vol. 372, no. 6545, pp. 948–952, 2021.

[3] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Physical review A*, vol. 54, no. 4, p. R2493, 1995.

[4] A. M. Steane, "Error correcting codes in quantum theory," *Phys. Rev. Lett.*, vol. 77, pp. 793–797, Jul 1996.

[5] A. Kitaev, "Fault-tolerant quantum computation by anyons," *Annals of Physics*, vol. 303, no. 1, pp. 2–30, 2003.

[6] S. B. Bravyi and A. Y. Kitaev, "Quantum codes on a lattice with boundary," *arXiv preprint quant-ph/9811052*, 1998.

[7] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, no. 3, p. 032324, 2012.

[8] J. M. Hornibrook, J. I. Colless, I. D. Conway Lamb, S. J. Pauka, H. Lu, A. C. Gossard, J. D. Watson, G. C. Gardner, S. Fallahi, M. J. Manfra, and D. J. Reilly, "Cryogenic Control Architecture for Large-Scale Quantum Computing," *Phys. Rev. Applied*, vol. 3, p. 024010, 2015.

[9] 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*, 2017, pp. 679–691.

[10] C. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, "Surface code quantum computing by lattice surgery," *New Journal of Physics*, vol. 14, no. 12, p. 123011, 2012.

[11] A. G. Fowler and C. Gidney, "Low overhead quantum computation using lattice surgery," *arXiv preprint arXiv:1808.06709*, 2018.

[12] C. Duckering, J. M. Baker, D. I. Schuster, and F. T. Chong, "Virtualized Logical Qubits: A 2.5 D Architecture for Error-Corrected Quantum Computing," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 173–185.

[13] C. Chamberland, K. Noh, P. Arrangoiz-Arriola, E. T. Campbell, C. T. Hann, J. Iverson, H. Putterman, T. C. Bohdanowicz, S. T. Flammia, A. Keller, G. Refael, J. Preskill, L. Jiang, A. H. Safavi-Naeini, O. Painter, and F. G. S. L. Brandão, "Building a fault-tolerant quantum computer using concatenated cat codes," *arXiv preprint arXiv:2012.04108*, 2020.

[14] H. Bombin, I. H. Kim, D. Litinski, N. Nickerson, M. Pant, F. Pastawski, S. Roberts, and T. Rudolph, "Interleaving: Modular architectures for fault-tolerant photonic quantum computing," *arXiv preprint arXiv:2103.08612*, 2021.

[15] J. E. Bourassa, R. N. Alexander, M. Vasmer, A. Patil, I. Tzitrin, T. Matsuura, D. Su, B. Q. Baragiola, S. Guha, G. Dauphinais *et al.*, "Blueprint for a scalable photonic fault-tolerant quantum computer," *Quantum*, vol. 5, p. 392, 2021.

[16] A. Erhard, H. P. Nautrup, M. Meth, L. Postler, R. Stricker, M. Stadler, V. Negnevitsky, M. Ringbauer, P. Schindler, H. J. Briegel, R. Blatt, N. Friis, and T. Monz, "Entangling logical qubits with lattice surgery," *Nature*, vol. 589, no. 7841, pp. 220–224, 2021.

[17] G. Torlai and R. G. Melko, "Neural decoder for topological codes," *Physical review letters*, vol. 119, no. 3, p. 030501, 2017.

[18] P. Das, C. A. Pattison, S. Manne, D. Carmean, K. Svore, M. Qureshi, and N. Delfosse, "A scalable decoder micro-architecture for fault-tolerant quantum computing," *arXiv preprint arXiv:2001.06598*, 2020.

[19] N. Delfosse and N. H. Nickerson, "Almost-linear time decoding algorithm for topological codes," *arXiv preprint arXiv:1709.06218*, 2017.

[20] 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*, 2020, p. 556–569.

[21] Y. Ueno, M. Kondo, M. Tanaka, Y. Suzuki, and Y. Tabuchi, "QECOOL: on-line quantum error correction with a superconducting decoder for surface code," in *Proceedings of the 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 451–456.

[22] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, "Encoding electronic spectra in quantum circuits with linear T complexity," *Physical Review X*, vol. 8, no. 4, p. 041015, 2018.

[23] C. Gidney and M. Ekerå, "How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits," *Quantum*, vol. 5, p. 433, 2021.

[24] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, p. 1484–1509, 1997.

[25] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Physical review letters*, vol. 103, no. 15, p. 150502, 2009.

[26] S. Arunachalam and R. de Wolf, "Guest column: A survey of quantum learning theory," *ACM SIGACT News*, vol. 48, no. 2, pp. 41–67, 2017.

[27] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical computer science*, vol. 560, pp. 7–11, 2014.

[28] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.

[29] J. Kelly, R. Barends, A. Fowler, A. Megrant, E. Jeffrey, T. White, D. Sank, J. Mutus, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, I. Hoi, C. Neill, P. O'Malley, C. Quintana, P. Roushan, A. Vainsencher, J. Wenner, A. Cleland, and J. Martinis, "State preservation by repetitive error detection in a superconducting quantum circuit," *Nature*, vol. 519, no. 7541, pp. 66–69, 2015.

[30] A. Y. Kitaev, "Quantum computations: algorithms and error correction," *Russian Mathematical Surveys*, vol. 52, no. 6, pp. 1191–1249, 1997.

[31] A. G. Fowler, A. C. Whiteside, and L. C. Hollenberg, "Towards practical classical processing for the surface code: timing analysis," *Physical Review A*, vol. 86, no. 4, p. 042313, 2012.

[32] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal clifford gates and noisy ancillas," *Phys. Rev. A*, vol. 71, p. 022316, Feb 2005.

[33] M. Dorojevets, P. Bunyk, and D. Zinoviev, "FLUX chip: design of a 20-GHz 16-bit ultrapipelined RSFQ processor prototype based on 1.75-μm LTS technology," *IEEE Transactions on Applied Superconductivity*, vol. 11, no. 1, pp. 326–332, 2001.

[34] P. Farrell, R. Clarke, M. Vesely, S. Shauck, B. Konigsburg, P. Tschirhart, and S. Rahman, "A superconducting 8-bit CPU design," in *Applied Superconductivity Conference*, 2018, pp. 1EOr1C–03.

[35] Y. Nobumori, T. Nishigai, K. Nakamiya, N. Yoshikawa, A. Fujimaki, H. Terai, and S. Yorozu, "Design and implementation of a fully asynchronous SFQ microprocessor: SCRAM2," *IEEE Transactions on Applied Superconductivity*, vol. 17, no. 2, pp. 478–481, 2007.

[36] R. Sato, R. Sato, Y. Hatanaka, Y. Ando, M. Tanaka, A. Fujimaki, K. Takagi, and N. Takagi, "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*, vol. 27, no. 4, p. 1300505, 2017.

[37] M. Tanaka, F. Matsuzaki, T. Kondo, N. Nakajima, Y. Yamanashi, A. Fujimaki, H. Hayakawa, N. Yoshikawa, H. Terai, and S. Yorozu, "A single-flux-quantum logic prototype microprocessor," in *2004 IEEE International Solid-State Circuits Conference*, 2004, pp. 298–529.

[38] X. Peng, Q. Xu, T. Kato, Y. Yamanashi, N. Yoshikawa, A. Fujimaki, N. Takagi, K. Takagi, and M. Hidaka, "High-speed demonstration of bit-serial floating-point adders and multipliers using single-flux-quantum circuits," *IEEE Transactions on Applied Superconductivity*, vol. 25, no. 3, pp. 1–6, 2015.

[39] I. Nagaoka, M. Tanaka, K. Sano, T. Yamashita, A. Fujimaki, and K. Inoue, "Demonstration of an energy-efficient, gate-level-pipelined 100 TOPS/W arithmetic logic unit based on low-voltage rapid single-flux-quantum logic," in *2019 IEEE International Superconductive Electronics Conference*, 2019, pp. 1–3.

[40] I. Nagaoka, M. Tanaka, I. Koji, and A. Fujimaki, "A 48 GHz 5.6 mW gate-level-pipelined multiplier using single-flux quantum logic," in *2019 IEEE International Solid-State Circuits Conference, ISSCC 2019*, ser. Digest of Technical Papers - IEEE International Solid-State Circuits Conference. United States: Institute of Electrical and Electronics Engineers Inc., 3 2019, pp. 460–462.

[41] J. Edmonds, "Paths, trees, and flowers," *Canadian Journal of mathematics*, vol. 17, no. 3, pp. 449–467, 1965.

[42] G. Duclos-Cianci and D. Poulin, "Fast decoders for topological quantum codes," *Physical review letters*, vol. 104, no. 5, p. 050504, 2010.

[43] 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*, 2017, p. 189–195.

[44] R. Naik, N. Leung, S. Chakram, P. Groszkowski, Y. Lu, N. Earnest, D. McKay, J. Koch, and D. Schuster, "Random access quantum information processors using multimode circuit quantum electrodynamics," *Nature communications*, vol. 8, no. 1, pp. 1–7, 2017.

[45] D. E. Drake and S. Hougardy, "A simple approximation algorithm for the weighted matching problem," *Information Processing Letters*, vol. 85, no. 4, pp. 211 – 213, 2003.

[46] A. Holmes, "Quantum and classical algorithms and optimizations enabling practical quantum computation," Ph.D. dissertation, The University of Chicago, 2020.

[47] Y. Yamanashi, T. Kainuma, N. Yoshikawa, I. Kataeva, H. Akaike, A. Fujimaki, M. Tanaka, N. Takagi, S. Nagasawa, and M. Hidaka, "100 GHz demonstrations based on the single-flux-quantum cell library for the 10 kA/cm^2 Nb multi-layer process," *IEICE Transactions on Electronics*, vol. 93, no. 4, pp. 440–444, 2010.

[48] S. Nagasawa, K. Hinode, T. Satoh, M. Hidaka, H. Akaike, A. Fujimaki, N. Yoshikawa, K. Takagi, and N. Takagi, "Nb 9-layer fabrication process for superconducting large-scale SFQ circuits and its process evaluation,"

IEICE Transactions on Electronics, vol. E97.C, no. 3, pp. 132–140, 2014.

[49] A. Fujimaki, M. Tanaka, R. Kasagi, K. Takagi, M. Okada, Y. Hayakawa, K. Takata, H. Akaike, N. Yoshikawa, S. Nagasawa, K. Takagi, and N. Takagi, "Large-scale integrated circuit design based on a Nb nine-layer structure for reconfigurable data-path processors," *IEICE Transactions on Electronics*, vol. E97.C, no. 3, pp. 157–165, 2014.

[50] T. Van Duzer, L. Zheng, S. R. Whiteley, H. Kim, J. Kim, X. Meng, and T. Ortlepp, "64-kb hybrid Josephson-CMOS 4 kelvin RAM with 400 ps access time and 12 mw read power," *IEEE Transactions on Applied Superconductivity*, vol. 23, no. 3, pp. 1 700 504–1 700 504, 2013.

[51] E. S. Fang and T. Van Duzer, "A Josephson integrated circuit simulator (JSIM) for superconductive electronics application," in *Extended Abstracts of 1989 International Superconductivity Electronics Conference*, 1989, pp. 407–410.

[52] D. E. Kirichenko, S. Sarwana, and A. F. Kirichenko, "Zero static power dissipation biasing of RSFQ circuits," *IEEE Transactions on Applied Superconductivity*, vol. 21, no. 3, pp. 776–779, 2011.

[53] O. A. Mukhanov, "Energy-efficient single flux quantum technology," *IEEE Transactions on Applied Superconductivity*, vol. 21, no. 3, pp. 760–769, 2011.

[54] S. Bravyi, M. Englbrecht, R. König, and N. Peard, "Correcting coherent errors with surface codes," *npj Quantum Information*, vol. 4, no. 1, pp. 1–6, 2018.

[55] A. G. Fowler and J. M. Martinis, "Quantifying the effects of local many-qubit errors and nonlocal two-qubit errors on the surface code," *Physical Review A*, vol. 89, no. 3, p. 032316, 2014.

[56] D. Gottesman, "The heisenberg representation of quantum computers," *arXiv preprint quant-ph/9807006*, 1998.

[57] V. Kolmogorov, "Blossom V: a new implementation of a minimum cost perfect matching algorithm," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 43–67, 2009.