Maximizing the Yield of Bucket Brigade Quantum Random Access Memory using Redundancy Repair

Dongmin Kim, Sovanmonynuth Heng, Sengthai Heng, and Youngsun Han[*]
Pukyong National University, Department of AI Convergence, Busan, South Korea
(Dated: May 24, 2024)

Quantum Random Access Memory (qRAM) is an essential computing element for running oracle-based quantum algorithms. qRAM exploits the principle of quantum superposition to access all data stored in the memory cell simultaneously and guarantees the superior performance of quantum algorithms. A qRAM memory cell comprises logical qubits encoded through quantum error correction technology for the successful operation of qRAM against various quantum noises. In addition to quantum noise, the low-technology nodes based on silicon technology can increase the qubit density and may introduce defective qubits. As qRAM comprises many qubits, its yield will be reduced by defective qubits; these qubits must be handled using QEC scheme. However, the QEC scheme requires numerous physical qubits, which burdens resource overhead. To resolve this overhead problem, we propose a quantum memory architecture that compensates for defective qubits by introducing redundant qubits. We also analyze the yield improvement offered by our proposed architecture by varying the ideal fabrication error rate from 0.5% to 1% for different numbers of logical qubits in the qRAM. In the qRAM comprising 1,024 logical qubits, eight redundant logical qubits improved the yield by 95.92% from that of qRAM not employing the redundant repair scheme.

I. INTRODUCTION

Quantum algorithms promise to solve specific problems that cannot be solved by classical algorithms within a reasonable time. Representative quantum algorithms are the Shor algorithm [1] and the Grover algorithm [2]. The Shor algorithm divides a huge composite number into two prime factors, and the Grover algorithm finds the desired data among a large dataset. Exploiting the "superposition" phenomenon of qubits, both algorithms simultaneously process all data combinations that can be represented by the qubits, thereby speeding up the calculations. Various quantum algorithms that utilize the advantages of qubit-based computation are currently being developed.

To guarantee that quantum superposition can exponentially increase the speed of quantum algorithms over their classical counterparts, we must load the superposed data into a quantum processor. Most of the representative quantum algorithms used in various fields assume that data can be loaded through an oracle, commonly referred to as a black box [3]. Such a black box must be fully supported with some degree of abstraction for oracle-based quantum algorithms. As a quantum oracle, Quantum Random Access Memory (qRAM) [3, 4] is required for achieving an exponential computational advantage of quantum algorithms over conventional algorithms [4, 5]. For this purpose, the qRAM must store data in a superposition state. qRAM already guarantees exponential speed improvement in tasks such as data processing [6, 7] and pattern recognition [8–11] and is required for other quantum algorithms such as quantum searching [12, 13], collision finding [14, 15], and element distinctness problems [16].

Promising qRAM architectures are broadly divided into Fanout and Bucket Brigade schemes [17], which adopt trapped ion qubits and qutrits (storage units of three-state information $|wait\rangle$, $|left\rangle$, and $|right\rangle$) as the basic unit, respectively. The two schemes differ mainly by the number of gates activated in their memory-addressing processes. For a single n-bit query string, Fanout qRAM activates $O(2^n)$ gates, whereas in Bucket Brigade qRAM activates $O(poly(n))$ gates [5, 17]. In the actual implementation of qRAM, gate activation is considered as physical transistor activation [5, 17]. However, as numerous transistor activations cannot be operated within a relatively short quantum coherence time, they deteriorate the accuracy of the results. Accordingly, the Fanout scheme is more susceptible to decoherence error than the Bucket Brigade scheme.

Various studies have shown that memory addressing is more error-resistant in the Bucket Brigade qRAM scheme than in the Fanout scheme [17]. Bucket Brigade also accelerates memory addressing by parallelizing the queries [18]. These studies focus on the robustness of Bucket Brigade qRAM with a small number of gate activations required for memory addressing. However, to improve the fault tolerance of qRAM, one must consider the robustness of the memory cell qubits comprising the qRAM [3, 19]. Although Bucket Brigade achieves more error-resistant memory addressing than the Fanout structure, if the memory cells are vulnerable to errors this memory addressing becomes meaningless. Nevertheless, the existing studies that aim at performing memory addressing fault tolerantly do not consider the occurrence of the errors in the memory cell qubits. Therefore, for fully fault-tolerant qRAM, the memory cell qubits should be error-resistant as well. This requires a quantum error

* youngsun@pknu.ac.kr

correction (QEC) scheme, which typically spreads data stored in a physical qubit across multiple highly entangled physical qubits. The QEC code scheme creates a "logical qubit", a virtual qubit holding the data used in the quantum algorithm [20–23].

QEC applied to a logical qubit protects the quantum state from general quantum noise. Moreover, QEC must handle defective physical qubits that may occur during the fabrication process. This is because the density of qubits has increased based on recent silicon technology and as mass production of physical qubits has become possible [24, 25]. It makes the probability of occurrence of defective qubits also increases so that many defective qRAMs are manufactured. As defective qRAMs lead to poor yield, reducing the number of defective logical qubits is essential for improving the qRAM yield. To this end, the QEC for encoding a logical qubit should cover a large number of physical qubits [26–29]. However, increasing the number of required logical qubits in qRAM increases the resource overhead because many additional physical qubits are required to support a higher degree of QEC.

In this paper, we propose a novel quantum memory architecture that improves the yield of qRAM while lowering the degree of QEC to minimize the physical qubit overhead. Our proposed qRAM adopts a redundant repair scheme based on the Bucket Brigade architecture. By employing redundant qubits, we minimize the number of additional physical qubits for the QEC scheme. Moreover, by lowering the degree of QEC, we reduce the number of physical qubits per logical qubit and repair the resulting fabrication error using additional redundant qubits. This scheme considerably reduces the number of physical qubits constituting one qRAM. Here we compare and analyze the yields for various error rates in fabricating physical qubits, various degrees of QEC applied to logical qubits, and the application of redundant repair technology. Through this analysis, we improve the yield to 95.92% using only eight redundant qubits supported by an additional 1.01% of all physical qubits. Our contributions are summarized below.

- We advocate circuit-level redundant repair for qRAMs, enabling fault recovery of defective logical qubits, and addressing fabrication error and QEC overhead.
- We propose a qRAM architecture with a redundant repair scheme. The proposed qRAM structure is based on the Bucket Brigade model [17], which is less vulnerable to noise than the Fanout structure. We introduce several elements for the redundant repair scheme and support our proposed qRAM with a quantum circuit model.
- We analyze the qRAM yield in our method while varying the fabrication error rate and number of logical qubits. Moreover, we analyze the impact of varying redundant qubit quantities on the qRAM yield. Furthermore, we evaluate the resource overhead by considering the necessary amount of physical qubits.

The remainder of this paper is organized as follows. In Section II we provide backgrounds of Bucket Brigade qRAM and quantum fabrication defects and motivate our research. In Section III we explain our proposed architecture and present the redundant repair algorithm. Section IV describes our circuit model implementation. The experimental setup and analysis of performance evaluation are given in Section V. In Section VI we discuss related works and conclude our work in Section VII.

II. BACKGROUND AND MOTIVATION

This section explains the Bucket Brigade qRAM architecture and quantum fabrication defects as well as the motivation of our study.

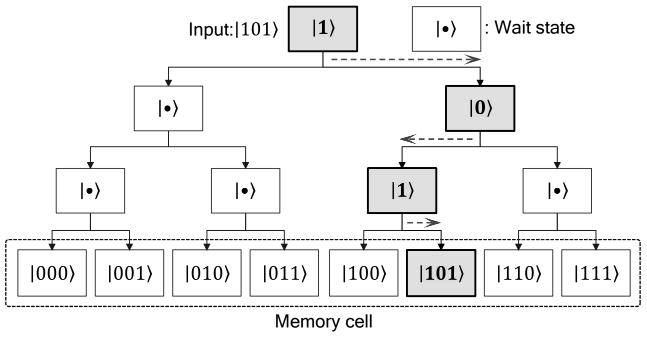
A. Bucket Brigade qRAM

Most representative quantum algorithms assume that the data of the algorithm are loaded into the quantum processor through a quantum oracle called a black box [3]. For example, the Grover and the Harrow-Hassidim-Lloyd (HHL) algorithms can achieve a significant speedup through quantum oracle [2, 30]. In fact, qRAM plays a crucial role in the quantum oracles that most representative quantum algorithms assume [3]. qRAM can store quantum information in either the classical representation ($|0\rangle$ or $|1\rangle$) or the quantum representation (an arbitrary superposition of $|0\rangle$ and $|1\rangle$). qRAM also allows querying superposition of address form [17, 31, 32], as shown in Equation (1):

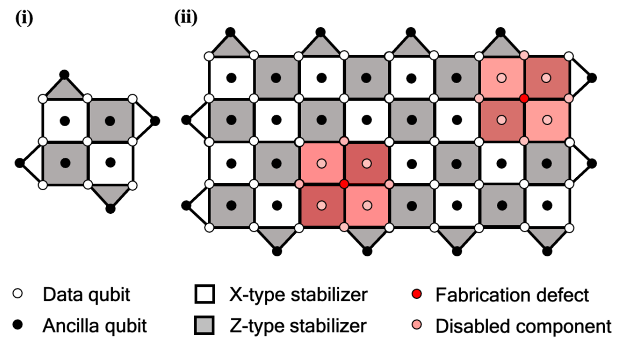
$$\sum_j \alpha_j |j\rangle|0\rangle \xrightarrow{\text{qRAM}} \sum_j \alpha_j |j\rangle \, |m_j\rangle \,, \tag{1}$$

where $\sum_j \alpha_j |j\rangle$ is a superposition of query addresses and $|m_j\rangle$ represents the content of the j-th memory location. qRAM can store the Grover algorithm [2], HHL algorithm [30], and other quantum algorithms [33–36] as classical information while allowing superposed queries, thus offering considerable speedup over classical algorithms.

To describe the qRAM algorithm, we present a qRAM architecture model based on the Bucket Brigade architecture. This architecture can support the superposition of addresses as input [37], but for better understanding, we show on the process of memory addressing using a specific input address in Figure 1a. This architecture can be routed in a three-level quantum-controlled 'qutrit' [17, 32, 35, 38] fashion. Each node of the binary tree is a 'trit' with three possible internal states: $|0\rangle$, $|1\rangle$, or $|\bullet\rangle$. A trit in state $|0\rangle$ or $|1\rangle$ acts as a switch that routes the incoming signal to the 'left' or 'right', respectively. A trit in the $|\bullet\rangle$ state does not propagate the



(a) Bucket Brigade qRAM architecture. The routing nodes are qutrits with three possible states: $|0\rangle$, $|1\rangle$, and $|\bullet\rangle$ states. The $|\bullet\rangle$ state is the "wait" state, indicating that no activation is performed on that node. Input state is $|101\rangle$, the routing path is routed sequentially to the memory cell colored gray.



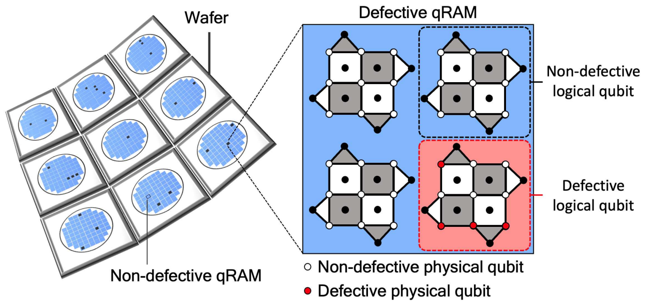
(b) Fabrication defects in the surface code. Red circles are defective physical qubits introduced by fabrication errors. Both X- and Z-type stabilizers, data, and ancilla qubits connected to defective physical qubits (highlighted in pink) are disabled components.

Fig. 1. Bucket Brigade qRAM architecture example (top) and occurrence of fabrication defects on surface code (bottom)

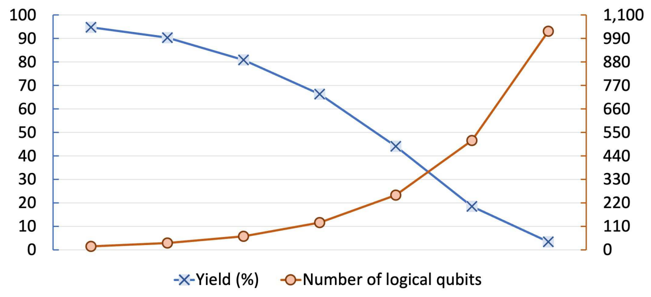
incoming signal. It continues until it reaches the *k-th* level of the tree. The gray routing path in Figure 1a is the path determined by the input state $|101\rangle$. Once the n qubits of the input register $|101\rangle$ have passed through the tree, n quantum switches are active, i.e., in the state $|0\rangle$ or $|1\rangle$; consequently, the output is $|101\rangle$ memory cell. Note that although this procedure requires the order of 2^n qutrits, only n qutrits are active in any run of the protocol [17, 31, 32, 38].

B. Quantum Fabrication Defects

Quantum noise is the major problem limiting the computational advantage of current quantum computers. As quantum information is very fragile when encoded in larger quantum systems, the quantum states must be protected by QEC [32, 39–41]. Besides detecting and


Wafer
Defective qRAM
Non-defective logical qubit
Defective logical qubit
Non-defective qRAM
○ Non-defective physical qubit
● Defective physical qubit

(a) qRAMs on the wafer with defective logical and physical qubits



(b) Relationship between the yield of qRAM and number of logical qubits

Fig. 2. Fabrication of qRAM with wafers (top) and the relationship between the number of logical qubits qRAM yield (bottom)

correcting errors, QEC provides an additional degree of freedom for controlling the complexity of logical qubit encoding. In practice, QEC observes the errors occurring on fault-tolerant circuits while attempting to control the logical qubits.

When constructing topological QEC codes, a lattice of physical qubits must be embedded on a manifold with a non-trivial topology such that the quantum information is encoded in the global degrees of freedom, as shown in Figure 1b(i) [20, 41–43]. However, the industrial production line of large-scale topological devices introduces defective physical qubits because manufacturing processes are inherently imperfect. We refer to such faults as quantum fabrication defects [44–46]. In the example of Figure 1b(ii), quantum fabrication defects are indicated as red-colored circles. If fabrication defects occur in the topology of the surface code lattice, the distances of the code and the quality of the encoded logical qubits are seriously reduced [40, 45, 47, 48].

Fabrication defects also create punctures in the qubit array, some of which may be very large. We cannot assume control over the qubits within each puncture. Some protocols are designed to reliably collect the syndrome data near fabrication defects and perform the computation over the remaining intact qubits of the lattice with high probability, assuming a suitably low error rate of the intact qubits [45–47]. However, if several defective physical qubits are sparsely distributed through the qubit array, this method is difficult to implement on a limited-size lattice. The lattice must then be expanded with additional qubits, which incurs a large resource overhead in terms of physical qubits.

C. Motivation

qRAM is an essential element in practical quantum computation. In addition, qRAM must contain sufficiently many qubits for theoretical quantum calculations using practical-scale quantum algorithms that utilize large amounts of data. Recent improvements in silicon technology have increased the density and hence the number of qubits that can be manufactured. Practical-scale quantum algorithms and quantum gains are expected in the future, but we believe that a large number of qubits will be infected by fabrication errors. In fact, numerous fabrication-induced defects have been reported in superconducting quantum dots, which are promising candidates for building scalable quantum computers [49–51].

Such defects have fatal consequences in qRAM. If the memory cells constituting the qRAM include defective qubits, the qRAM cannot support random address accesses, and the qRAM yield is greatly reduced. Figure 2a shows a qRAM fabricated with wafers, along with its logical and physical qubits. In the left upper panel of this figure, the non-defective qRAM cells (blue squares) are interspersed with defective qRAM cells (black squares). If one black square is magnified as shown in the upper right panel of Figure 2a, we observe (for simplicity) four logical qubits encoded with multiple physical qubits. Among the four logical qubits is one defective qubit containing several defective physical qubits. Figure 2b shows the simulated relationship between the yield and the number of logical qubits in the qRAM. The yield is calculated as Equation (2).

$$Yield = \left(1 - \frac{Number\ of\ defective\ qRAMs}{Number\ of\ fabricated\ qRAMs}\right) \times 100 \tag{2}$$

Our simulation was performed on 1,000 qRAMs. Each logical qubit in each qRAM was encoded with 17 physical qubits subjected to a 0.5% fabrication error rate. As shown in Figure 2b, the yield decreases with the increasing number of logical qubits in the qRAM.

Theoretically, the low qRAM yield can be mitigated with the QEC scheme. There is a lot of research that employs high-degree QEC for encoding a logical qubit, which improves the reliability and fault tolerance of logical qubits [52–54]. However, high-degree QEC incurs enormous resource overhead because it requires an exponentially increasing number of physical qubits. Given the limited resources for making physical qubits at present, a high-degree QEC scheme is expected to reduce the productivity and yield of qRAM. The number of physical qubits in a single qRAM is determined by Equation (3):

$$\begin{aligned} N_{PQ}(qRAM) = {} & N_{LQ}(qRAM) \\ & \times N_{PQ}(Logical\ Qubit), \end{aligned} \tag{3}$$

where $N_{PQ}(qRAM)$ and $N_{LQ}(qRAM)$ represent the numbers of physical and logical qubits in the qRAM, respectively, and $N_{PQ}(Logical\ Qubit)$ is the number of physical qubits required for implementing a single logical qubit. In other words, the number of physical qubits in the qRAM is the product of the number of logical qubits in the qRAM and the number of physical qubits per logical qubit. Therefore, we must improve the mass-production yield of qRAM while reducing the resource overhead, i.e., the number of required physical qubits per qRAM. For this purpose, we propose a redundant repair method using additional spare qubits in the Bucket Brigade qRAM structure. This method reduces the number of physical qubits required per qRAM by lowering the degree of QEC. It also maximizes the quality of the qRAM yield by replacing defective qubits with redundant qubits. *To the best of our knowledge*, our proposed redundant repair scheme in qRAM is the first approach to improve qRAM yield while minimizing resource overhead.

III. BUILT-IN SELF REPAIR FOR QRAM

In this section, we present our novel qRAM architecture based on the Bucket Brigade structure and our redundant repair algorithm that resolves quantum fabrication defects.

A. Overall Architecture

Figure 3 shows the overall architecture of our proposed qRAM. To represent defective and spare memory addresses, we use a fault address table (FAT), a table composed of faulty and spare addresses, and assume that FAT is provided by an automatic test equipment (ATE) external device. Classically, a redundant repair scheme on random access memory (RAM) does this method [55, 56]. The ATE tests the memory cell of qRAM, identifies faulty addresses, and allocates spare addresses accordingly. Afterward, ATE composes the FAT with both types of addresses and passes the FAT to the qRAM and the quantum oracle of redundant repair circuits. The redundant repair scheme consults the FAT and qRAM input address.

Currently, various studies are being conducted to detect quantum defects [49]. Based on those studies, we assumed an external device specialized for detecting quantum defects in qRAM will be supported in the future. Our work is to propose a way to hide defects through redundant repair in the presence of quantum defects.

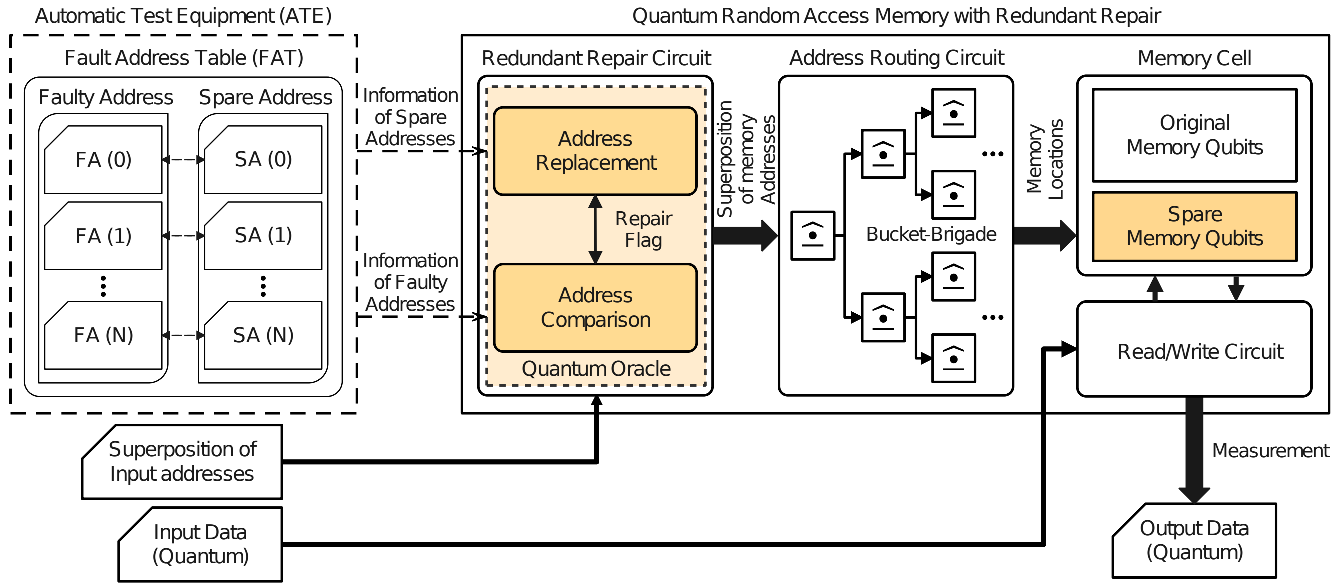


Fig. 3. Overall architecture of our proposed qRAM with redundant repair scheme and an external device ATE. All the data and states transferred to/from different parts are quantum. The ATE sends information of faulty addresses and spare addresses from the FAT to the quantum oracle of the redundant repair circuit. Based on two different types of address information, quantum oracle implements address comparison and address replacement parts, respectively. When the superposition of input addresses is given as input to qRAM, address comparison compares faulty addresses from FAT with each of the input addresses and sets the value of the repair flag. Checking the repair flag, the address replacement decides whether to replace or not replace the input address with the spare address. After replacing all faulty addresses, the redundant repair circuit passes the superposition of memory addresses to the address routing circuit for routing. Memory locations are then given to the memory cells, which communicate with read/write circuit data to read the memory cell data(Quantum) and write the input data(Quantum) to the memory cell.

Our proposed qRAM consists of four major components: the redundant repair circuit, address routing circuit, read/write circuit, and memory cells.

First, the redundant repair circuit of qRAM receives the information of faulty and spare addresses from ATE. Based on these addresses, redundant repair circuit support quantum oracle that consists of address replacement and address comparison modules. When the superposition of the input address is given to the quantum oracle of the redundant repair circuit, the quantum oracle compares each of the input addresses with the fault addresses by using the address comparison module. If the input and fault addresses match, the corresponding memory cell is defective. The repair flag is then activated and the address replacement module replaces the input address with spare address corresponding to the faulty address. If the input address does not match the faulty address, the repair flag is not activated and the address replacement module does not replace the input address. This process is repeated until all superposition of input addresses is checked. After all iterations, the superposition of memory addresses comes out as the output of the redundant repair circuit.

Second, the address routing circuit activates the routing nodes with the superposition of the memory address. The structure of the address routing circuit is based on Bucket Brigade qRAM, with the routing nodes configured as a binary tree. From the initial state $|\bullet\rangle$ of each routing node, specific routing nodes are activated by setting $|0\rangle$ or $|1\rangle$ based on the superposition of memory addresses. The redundant repair circuit provides the locations of original and spare memory qubits. Since memory addresses are in the superposed state, they address all memory cells with the same probability except defected memory cells.

Third, the read/write circuit reads the data in the memory cells or writes data to the memory cells. The read/write circuit can access the memory cells with memory location data. When reading from memory, the read/write circuit finds the memory locations of the memory cells and accesses them for measurement. After the measurement, the read/write circuit generates the output data. When writing to memory, the read/write circuit accesses the memory cells with the memory locations and writes data to those cells. At this time, the read/write circuit must be provided with input data.

Finally, in memory cells, there are original and spare memory qubits. By adopting spare memory qubits, we can achieve redundant repair for memory cells if faults occur on original memory cells. Based on the data of

Algorithm1 Redundant repair on qRAM

INPUT: FAT: Fault address table, SoI: Superposition of Input addresses
OUTPUT: SoM: Superposition of Memory addresses

```
1:  procedure REDUNDANTREPAIR($FAT$, $SoI$)
2:      for $i \leftarrow 1$ to $n$ do
3:          for $j \leftarrow 1$ to size of $SoI$ do
4:              if $SoI_j = FA_i$ then
5:                  $SoI_j \leftarrow SA_i$
6:              else
7:                  continue
8:              end if
9:          end for
10:     end for
11:     $SoM \leftarrow SoI$
12: end procedure
```

memory locations from the address routing circuit and data from the read/write circuit, qRAM reads or writes both types of qubits avoiding addressing defective ones.

B. Redundant Repair Algorithm

Algorithm 1 describes the operation of our proposed redundant repair scheme on qRAM when the superposition of input addresses is given. The algorithm returns a superposition of memory addresses (SoM) by address comparison and replacement using the FAT and the superposition of input addresses (SoI). As the FAT contains multiple pairs of faulty address (FA) and spare address (SA), we define the FAT as a set of FAs and SAs as following Equation (4).

$$FAT = \{(FA_i,\ SA_i) \mid i = 1, 2, ..., n\} \tag{4}$$

We assume n pairs of faulty and spare addresses in the FAT. The variable i in the Equation (4) is an index for comparing the input address with all faulty addresses in the FAT. The algorithm begins comparing the superposition of input addresses with all FAs(*Address Comparison*). If a specific SoI matches the first FA, the corresponding SA is used to replace the specific SoI(*Address Replacement*). Otherwise, comparison with SoI and the first FA continues. After finishing the comparison of SoI with the first FA, the second FA is compared with SoI in the same manner as well. This comparison and replacement continue until all FAs have been compared and all SAs have been replaced. After the address comparison and address replacement SoI is allocated to SoM and the algorithm returns SoM.

$$U_{FA}|IA\rangle = \begin{cases} |RFQ\rangle \otimes |SA\rangle, & \text{if } IA = FA \\ |RFQ\rangle \otimes |IA\rangle, & \text{otherwise} \end{cases} \tag{5}$$

Figure 4 is an example of the redundant repair algorithm with the Bucket Brigade qRAM. The FAT includes faulty addresses with their corresponding spare

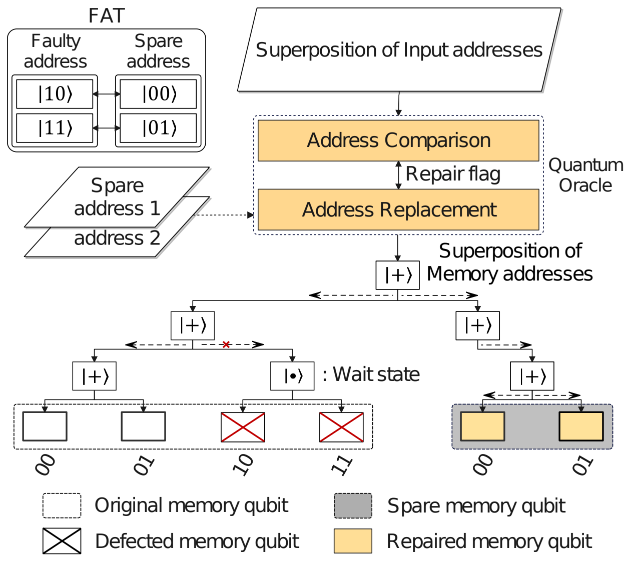


Fig. 4. Example of our proposed redundant repair algorithm. When the superposition of addresses including faulty addresses is given as input the quantum oracle does address comparison and address replacement based on FAT. The output of the quantum oracle is the superposition of addresses as well. For faulty addresses, the repair flag qubit is activated to route spare memory qubits. original memory qubits will be routed.

addresses. Since faulty addresses and spare addresses of FAT depend on the location of defects in memory cells, we use quantum oracle. We designed the quantum oracle as the Equation (5). The superposition of input addresses are entered into our quantum oracle. Each of the input addresses will be compared with faulty addresses in FAT by the address comparison module. If two addresses match, the repair flag address replacement module will replace the faulty address with the corresponding spare address based on FAT. This process is repeated in the quantum oracle until all faulty addresses are replaced. After all iterations, the superposition of memory addresses will be given into the Bucket Brigade structure. In the figure, we simply represent SoM as $|+\rangle$. There are no longer faulty addresses in SoM so it routes all memory cells including spare ones except $|10\rangle$ and $|11\rangle$ states.

IV. CIRCUIT-MODEL IMPLEMENTATION

In this section, we describe the quantum circuit model of our proposed qRAM architecture. As the redundant repair scheme in the proposed method uses redundant qubits, some components of our scheme mirror those of Bucket Brigade in [17].

Figure 5 shows the entire quantum circuit model of our proposed qRAM. We assume there are four original memory qubits and two spare memory qubits for the memory cells. We also employ two qubits each for the input ad-

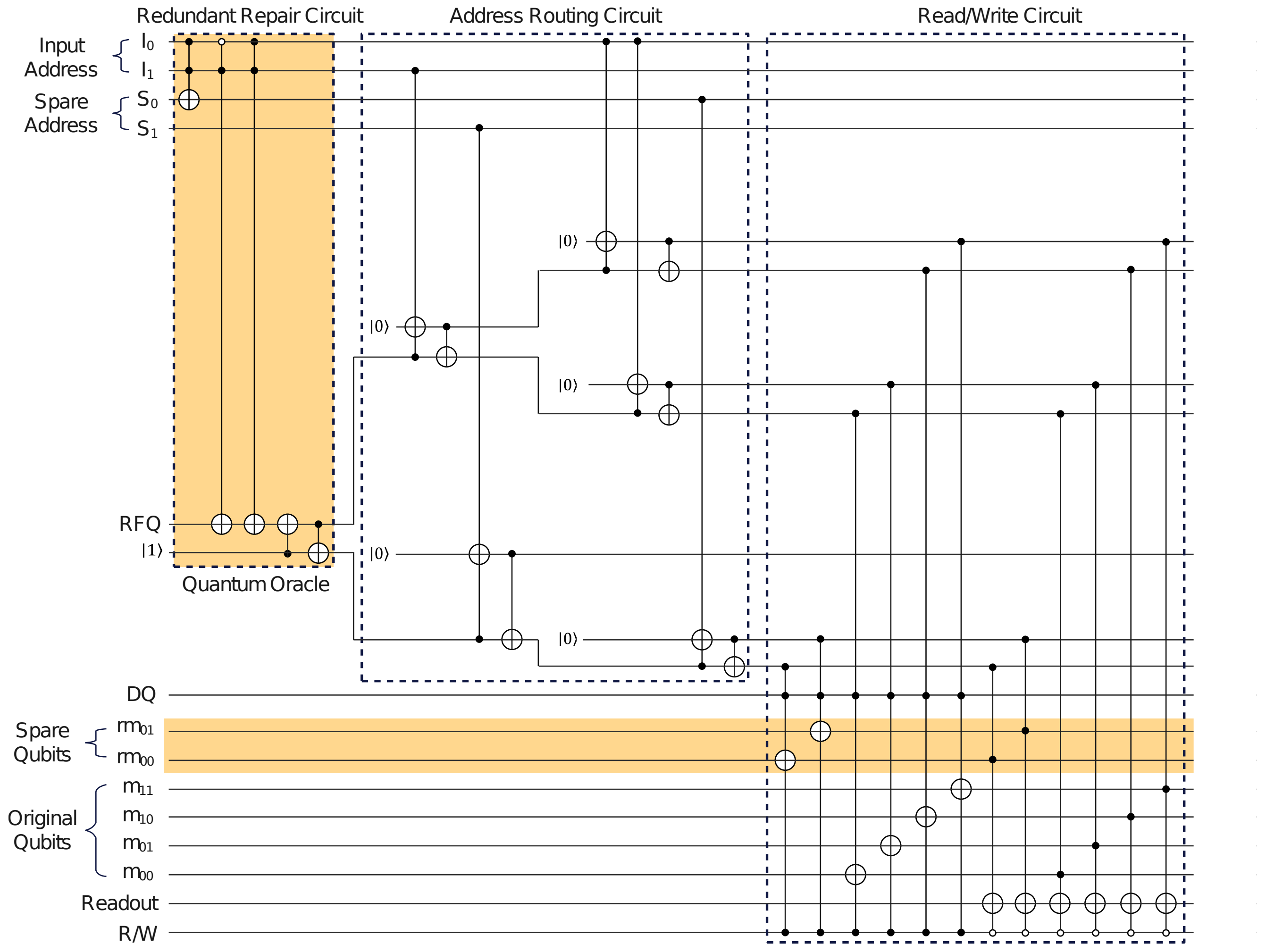


Fig. 5. Quantum circuit example of the proposed qRAM consisting of redundancy recovery circuit, address routing circuit, read/write circuit, and memory cell. This circuit is an example of an input address represented by two logical qubits. Parts highlighted with a yellow background are essential to support redundant repair schemes.

dress and spare address. Since qRAM must support when the input is given as a superposition state, we describe our proposed method assuming that the superposition of input addresses is given as input. The proposed quantum circuit consists of four parts: a redundant repair circuit, an address routing circuit, a read/write circuit, and memory cells.

The redundant repair circuit includes multiple positive and negative control gates to support quantum oracle implementation. The redundant repair circuit compares each address in the superposition state with the fault addresses embedded in our quantum oracle. All qubits for spare address are initialized with $|0\rangle$. For address comparison, check whether the input address is the same as one of the fault addresses. Based on the input address state and quantum oracle's quantum gates, qubits for the spare address are flipped. After address comparison, the RFQ qubit and additional $|1\rangle$ qubit are introduced for address replacement. The RFQ is activated by entangling with input address qubits through the control gates. An additional $|1\rangle$ qubit is entangled with the RFQ qubit through two $CNOT$ gates to support routing spare memory qubits depending on the RFQ state. When all input addresses are given as superposition states, the state of RFQ and $|1\rangle$ will have the same probability so that it supports routing of original and spare memory cells respectively.

The address routing circuit is configured as a binary tree to mimic the Bucket Brigade structure. Each routing node addresses both the original memory qubit and a spare memory qubit. The routing nodes on the upper binary tree address the original memory qubits, starting with the RFQ. Each node is entangled with the qubits of the input address through $Toffoli$ and $CNOT$ gates. Note that RFQ is the root node of the upper binary tree. Therefore, if the RFQ state is $|1\rangle$, the nodes for the upper binary tree are activated as $|1\rangle$ state. The lower binary tree is the route to spare memory qubits. Each routing node is entangled with the qubits of a spare address through $Toffoli$ and $CNOT$ gates. These nodes are activated if the root node initialized with $|1\rangle$ maintains the $|1\rangle$ state after the redundant repair circuit. Since the input address is given as the superposition state, RFQ and $|1\rangle$ states are also in the superposition state with the same probability. It derives every routing node will have $|1\rangle$ state at the end of the routing. However, based on FAT, routing nodes that address defected memory cells cannot have a $|1\rangle$ state.

The read/write circuit reads (writes) data from (to) memory. For this purpose, we entangle an R/W qubit with routing nodes, a DQ as the input data, and memory qubits through multi-controlled gates. When the R/W state is $|0\rangle$, the quantum circuit reads data from memory. The Readout qubit, initialized as $|0\rangle$, is entangled with routing nodes and memory cells. After reaching specific routing nodes and determining their memory cell states, based on probability, the Readout qubit state is flipped and measured. Conversely, for memory writing, the R/W qubit exists in the $|1\rangle$ state to activate multi-controlled NOT gates. After the address routing circuit, all routing nodes exist in the $|1\rangle$ state except defective ones. As the states of R/W and the specifically routed nodes are both $|1\rangle$, likewise based on probability, the NOT gate operates on the routed memory cells.

V. PERFORMANCE EVALUATION

This section evaluates the yield and resource overhead of our proposed method. We first explain the experimental setup and evaluation metrics of our simulation and then analyze the performance in terms of the evaluation metrics. We also present resource overhead data and a simulated result table for a deeper level of understanding.

A. Experimental Setup

We simulated the qRAM with a number of logical qubits from a minimum of 16 to a maximum of 1,024 while doubling its number of logical qubits. In addition, each logical qubit is constructed as a surface code lattice using $2d^2 - 1$ number of physical qubits. Here, d is the code distance of the surface code, and $\lfloor \frac{d-1}{2} \rfloor$ is the amount of error that can be corrected. With the code distance, we define a logical qubit as defective if it has more errors than it can correct. We also define a qRAM as defective if at least one logical qubit in the qRAM is defective. To vary the degree of QEC, we set d to 3, 5, 7, and 9. The d-dependent degree of QEC is denoted as *QEC d*.

For modeling the fabrication error occurrence to each physical qubit, we randomly injected errors with a binomial distribution at the fabrication error rate. The simulated fabrication error rate ranged from 0.5% to 1% in 0.1% increments. As the fabrication error rate increases, the occurrence probability of defective logical qubits increases as well. We also evaluate yield improvement by varying numbers of redundant qubits to 1, 2, 4, and 8.

The simulation results were evaluated regarding two metrics: yield and resource overhead of the qRAM. The yield was computed using Equation (2). To determine the resource overhead, we must consider the total number of physical qubits required for constructing the proposed qRAM. To check the total number of physical qubits, we divided the memory cell from the peripheral parts (namely, the addressing qubits and qubits of the routing nodes).

To ensure the reliability and validity of our results, we ran the experiment 10 times with a total of 1,000 qRAMs. The yields are reported as the averages of the 10 experimental yields. The simulation results were obtained using an in-house simulation program for this study.

B. Performance Analysis

1. Yield Improvement

Figure 6 plots the qRAM yields and number of physical qubits required for qRAMs with and without redundant repair qubits for different degrees of QEC. The *QEC 3, RR(8)* scheme in Figure 6 describes the QEC 3 scheme for logical qubits using eight additional redundant qubits as the spare memory qubits. A 0.5% fabrication error rate was applied to all physical qubits in each logical qubit. As shown in the figure, the yield of a qRAM with a given number of logical qubits increased with an increasing degree of QEC. In the case of 256 logical qubits, the yields at QECs of 3, 5, 7, and 9 were 44.08%, 60.0%, 67.73%, and 70.6%, respectively. Moreover, the yield of the qRAM reached 100% when employing eight redundant qubits with QEC 3. Meanwhile, the number of additional physical qubits required for the eight redundant qubits was 4.51% of the total number of physical qubits utilized in QEC 3. By using a negligible additional number of physical qubits, the yield of qRAM was greatly increased. This outcome holds great significance since the utilization of a low degree of QEC with redundant qubits can substantially reduce the number of additional physical qubits rather than using a higher degree of QEC without redundant qubits to increase the yield of qRAM.

Through the results of Figure 6, we demonstrated the average yield improvement according to the presence or absence of redundant repair. To obtain average yield improvement, we first calculated the average yields of QEC 3, QEC 5, QEC 7, and QEC 9 and differentiated them

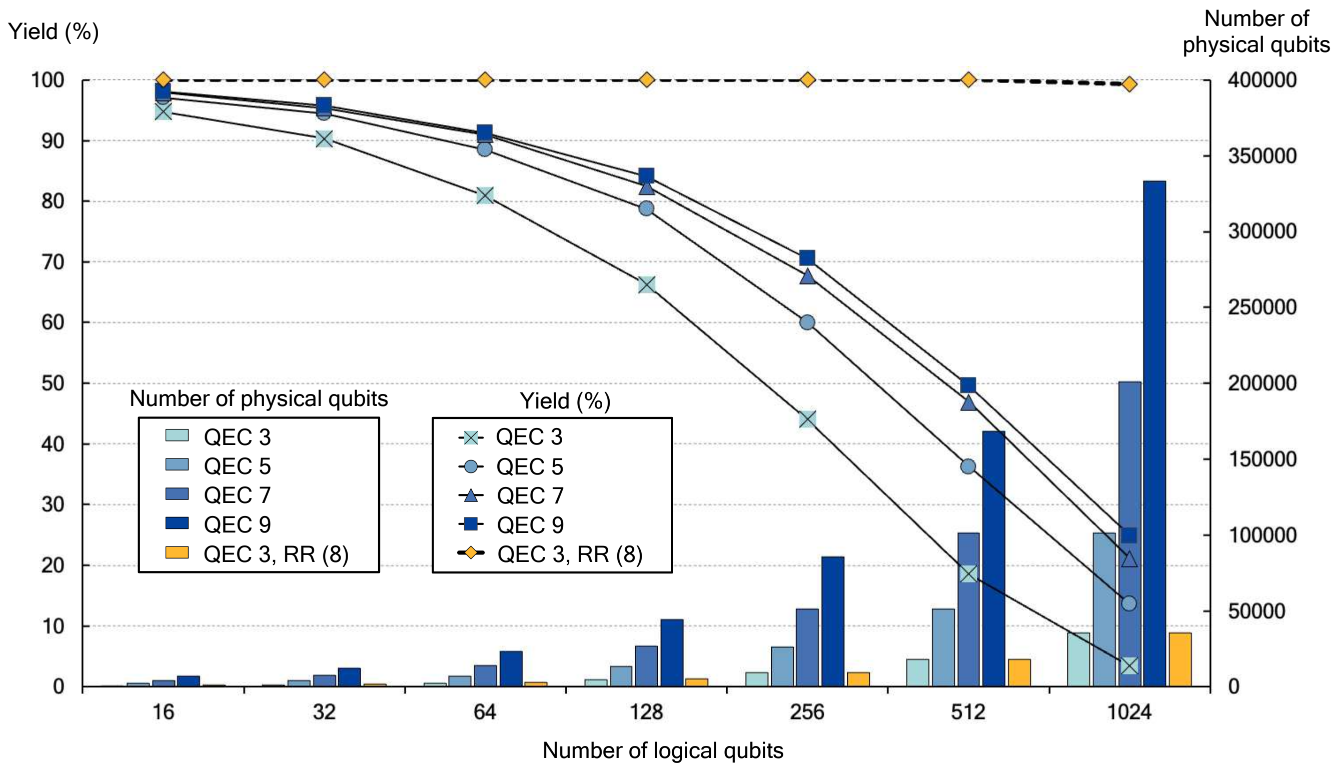


Fig. 6. Simulated yield (left y-axis) and number of physical qubits in the qRAM construction (right y-axis) versus number of logical qubits in qRAMs with different degrees of quantum error correction (QEC). The fabrication error rate was fixed at 0.5%. $RR(8)$ denotes using eight spare qubits for redundant repair.

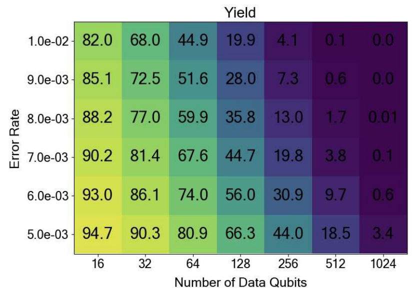
from the yield with the redundant repair. We defined this variation value as the average improvement of the yield for different numbers of logical qubits. Through this calculation, we illustrated the average yield improvement by 3.05%, 6.01%, 12.08%, 22.09%, 39.39%, 62.14%, and 83.59% for 16, 32, 64, 128, 256, 512, and 1,024 logical qubits, respectively.

Figure 7, we also compared the yield of qRAM by varying experimental parameters, namely, the fabrication error rate, number of logical qubits in the qRAM, and number of redundant logical qubits. For all five sub-figures, the x-axis represents the number of logical qubits, and the y-axis represents different degrees of fabrication error rate from 0.5% to 1%. To make it easier to distinguish visually, we use the colormap representation. The higher the yield, the closer to the yellow color, and the lower the yield, the closer to the indigo color. In all experiments, each logical qubit was encoded with QEC 3.

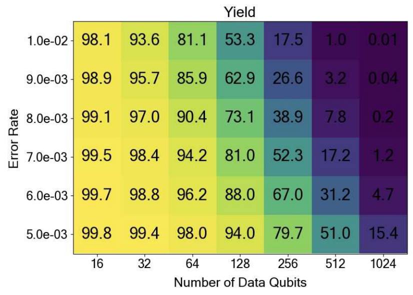
As shown in the figure, increasing the number of redundant qubits increased the yields of qRAMs with the same fabrication error rate and number of logical qubits. The same phenomenon was observed in qRAMs with small and large numbers of logical qubits. In the qRAM with 16 memory cells and no redundant qubits under a 1% fabrication error rate, the yield was 82.05%. In the same qRAM, one redundant qubit improved the yield to 98.18% (an approximate improvement of 16%). After increasing the number of logical qubits eight times (from 16 to 128) without changing the fabrication error rate, the yield reduced to 19.94% with no redundant qubits, but after adding 1, 2, 4, and 8 redundant qubits, the yield improved by 53.35%, 78.43%, 97.56%, and 100%, respectively. The same tendency was observed after doubling the number of logical qubits from 128 to 256. In the absence of redundant qubits, the yield reached only 4.12% but after adding 1, 2, 4, and 8 redundant qubits, the yield increased to 17.58%, 37.94%, 77.98%, and 99.42%, respectively. These results confirm that the qRAM yield can be significantly improved by employing a small number of redundant qubits relative to the number of logical qubits constituting the qRAM.

2. *Resource Overhead*

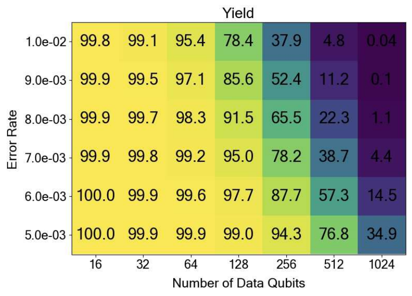
The proposed qRAM requires additional physical qubits to support the redundant qubits. Like the QEC scheme, this scheme introduces resource overheads on the number of physical qubits. Therefore, the number of additional physical qubits required in our method must be compared with that of QEC. To consider the resource overhead of our proposed qRAM architecture, we analyzed the total number of physical qubits constructed into the qRAM memory cells and peripherals. The number of physical qubits along the right axis of Figure 6 refers to



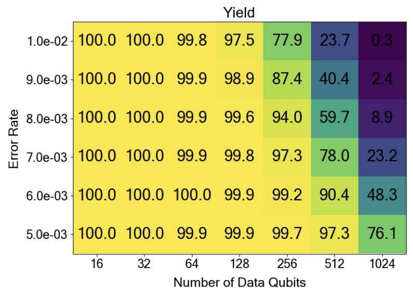
(a) No redundant qubits



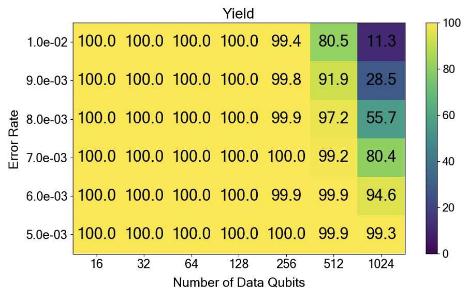
(b) One redundant qubit



(c) Two redundant qubits



(d) Four redundant qubits



(e) Eight redundant qubits

Fig. 7. qRAM yields according to manufacturing error rates, numbers of logical qubits, and numbers of redundant logical qubits. The yield is higher for colors closer to yellow and green, while the yield is lower for colors closer to blue and indigo. From (a) to (e), the number of redundant qubits is sequentially set as 0, 1, 2, 4, and 8, and the surface code distance of the logical qubits of all qRAMs is set to 3.

the number of qubits required for the memory cells and peripheral parts. For qRAMs with the same number of logical qubits, increasing the QEC degree dramatically increased the required number of physical qubits. When eight redundant qubits were added to a qRAM with a given number of logical qubits, the resource overhead was similar to the number of physical qubits required for QEC 3. In other words, the resource overhead (number of required physical qubits) for improving the yield was much lower in the proposed qRAM than in the conventional method of increasing the degree of QEC.

To analyze the resource overhead in more detail, we determined the numbers of physical qubits required for composing the memory cells and peripheral parts. The factor determining the number of physical qubits for a memory cell depends on the code distance. In terms of QEC degree, the number of required physical qubits N_{mem} is given by

$$N_{mem} = d \times (N + X), \tag{6}$$

where d, N, and X are the degree of QEC, number of original memory cells, and number of spare memory qubits, respectively.

Meanwhile, the required number of physical qubits for the peripheral part is based on a quantum circuit model. The peripheral part can be largely divided into three parts such as address qubits, routing nodes, and read/write part. Using address qubits, we first identified input and spare addresses. For example, suppose that there are N original memory qubits and X spare memory qubits to replace X faults. To represent the N original memory qubits, the input address requires $\log_2(N)$ qubits. A spare address requires an additional $\log_2(N)$ qubits because an arbitrary memory cell in qRAM must be addressed within the same time. Additionally, the routing nodes are the qubits needed for addressing, which require RFQ and the $|1\rangle$ qubits to support the redundant repair scheme. The proposed qRAM also requires original and spare memory qubits to support the address routing circuit composed of a binary tree. Addressing the original memory qubits requires $N-1$, whereas the required number of qubits for addressing the spare memory qubits depends on the number of faults. DQ, Readout, and R/W qubits are required for memory read and write. Therefore, the number of physical qubits for peripheral part N_{peri} in the circuit model is calculated as Equation (7).

$$N_{peri} = \begin{cases} 3log_2 N + N + 4, & \text{if } X \leq 1 + log_2 N \\ 2log_2 N + N + X + 3, & \text{if } X > 1 + log_2 N \end{cases} \tag{7}$$

Table I shows the detailed resource overheads obtained in all experiments. This table compares the numbers of physical qubits required for the memory cell and peripherals in qRAMs with different degrees of QEC, numbers of logical qubits, and numbers of redundant qubits. When no redundant repair scheme was applied, the overhead was compared in terms of the number of physical qubits for different numbers of redundant qubits (gray

Code Distance	Number of Logical Qubits	Number of Physical Qubits										Resource Overhead (%)							
		Original		RR1		RR2		RR4		RR8		RR1		RR2		RR4		RR8	
		Mem	Peri	Mem	Peri	Mem	Peri	Mem	Peri	Mem	Peri	Mem	Peri	Mem	Peri	Mem	Peri	Mem	Peri
QEC 3	16	272	374	289	544	306	544	340	544	408	595	6.25	45.45	12.5	45.5	25.0	45.5	50.0	59.1
	32	544	663	561	867	578	867	612	867	680	901	3.13	30.77	6.25	30.77	12.5	30.8	25.0	35.9
	64	1,088	1,224	1,105	1,462	1,122	1,462	1,156	1,462	1,224	1,479	1.56	19.44	3.13	19.44	6.25	19.44	12.5	20.8
	128	2,176	2,329	2,193	2,601	2,210	2,601	2,244	2,601	2,312	2,601	0.78	11.68	1.56	11.68	3.13	11.68	6.25	11.68
	256	4,352	4,522	4,369	4,828	4,386	4,828	4,420	4,828	4,488	4,828	0.39	6.77	0.78	6.77	1.56	6.77	3.13	6.77
	512	8,704	8,891	8,721	9,231	8,738	9,231	8,772	9,231	8,840	9,231	0.20	3.82	0.39	3.82	0.78	3.82	1.56	3.82
	1,024	17,408	17,612	17,425	17,986	17,442	17,986	17,476	17,986	17,544	17,986	**0.10**	**2.12**	**0.20**	**2.12**	**0.39**	**2.12**	**0.78**	**2.12**
QEC 5	16	784	1,078	833	1,568	882	1,568	980	1,568	1,176	1,715	6.25	45.45	12.5	45.5	25.0	45.5	50.0	59.1
	32	1,568	1,911	1,617	2,499	1,666	2,499	1,764	2,499	1,960	2,597	3.13	30.77	6.25	30.77	12.5	30.8	25.0	35.9
	64	3,136	3,528	3,185	4,214	3,234	4,214	3,332	4,214	3,528	4,263	1.56	19.44	3.13	19.44	6.25	19.44	12.5	20.8
	128	6,272	6,713	6,321	7,497	6,370	7,497	6,468	7,497	6,664	7,497	0.78	11.68	1.56	11.68	3.13	11.68	6.25	11.68
	256	12,544	13,034	12,593	13,916	12,642	13,916	12,740	13,916	12,936	13,916	0.39	6.77	0.78	6.77	1.56	6.77	3.13	6.77
	512	25,088	25,627	25,137	26,607	25,186	26,607	25,284	26,607	25,480	26,607	0.20	3.82	0.39	3.82	0.78	3.82	1.56	3.82
	1,024	50,176	50,764	50,225	51,842	50,274	51,842	50,372	51,842	50,568	51,842	0.10	2.12	0.20	2.12	0.39	2.12	0.78	2.12
QEC 7	16	1,552	2,134	1,649	3,104	1,746	3,104	1,940	3,104	2,328	3,395	6.25	45.45	12.5	45.5	25.0	45.5	50.0	59.1
	32	3,104	3,783	3,201	4,947	3,298	4,947	3,492	4,947	3,880	5,141	3.13	30.77	6.25	30.77	12.5	30.8	25.0	35.9
	64	6,208	6,984	6,305	8,342	6,402	8,342	6,596	8,342	6,984	8,439	1.56	19.44	3.13	19.44	6.25	19.44	12.5	20.8
	128	12,416	13,289	12,513	14,841	12,610	14,841	12,804	14,841	13,192	14,841	0.78	11.68	1.56	11.68	3.13	11.68	6.25	11.68
	256	24,832	25,802	24,929	27,548	25,026	27,548	25,220	27,548	25,608	27,548	0.39	6.77	0.78	6.77	1.56	6.77	3.13	6.77
	512	49,664	50,731	49,761	52,671	49,858	52,671	50,052	52,671	50,440	52,671	0.20	3.82	0.39	3.82	0.78	3.82	1.56	3.82
	1,024	99,328	100,492	99,425	102,626	99,522	102,626	99,716	102,626	100,104	102,626	0.10	2.12	0.20	2.12	0.39	2.12	0.78	2.12
QEC 9	16	2,576	3,542	2,737	5,152	2,898	5,152	3,220	5,152	3,864	5,635	6.25	45.45	12.5	45.5	25.0	45.5	50.0	59.1
	32	5,152	6,279	5,313	8,211	5,474	8,211	5,796	8,211	6,440	8,533	3.13	30.77	6.25	30.77	12.5	30.8	25.0	35.9
	64	10,304	11,592	10,465	13,846	10,626	13,846	10,948	13,846	11,592	14,007	1.56	19.44	3.13	19.44	19.44	12.66	12.5	20.8
	128	20,608	22,057	20,769	24,633	20,930	24,633	21,252	24,633	21,896	24,633	0.78	11.68	1.56	11.68	3.13	11.68	6.25	11.68
	256	41,216	42,826	41,377	45,724	41,538	45,724	41,860	45,724	42,504	45,724	0.39	6.77	0.78	6.77	1.56	6.77	3.13	6.77
	512	82,432	84,203	82,593	87,423	82,754	87,423	83,076	87,423	83,720	87,423	0.20	3.82	0.39	3.82	0.78	3.82	1.56	3.82
	1,024	164,864	166,796	165,025	170,338	165,186	170,338	165,508	170,338	166,152	170,338	0.10	2.12	0.20	2.12	0.39	2.12	0.78	2.12

Table. I. Simulated numbers of physical (memory and periphery) qubits and resource overheads (%) in the absence of a repair scheme (Original) and with 1, 2, 4, 8 redundant repair qubits (RR1, RR2, RR4, and RR8) respectively.

cells in the table). For example, in the qRAM with 1,024 logical qubits, the number of additional physical qubits required to support eight redundant qubits incurred memory and peripheral overheads of 0.78% and 2.12%, respectively, relative to the total number of physical qubits. In numerical terms, the requirement is 35,530 physical qubits, versus 35,020 qubits in the absence of repair application. Moreover, even if the degree of QEC in a qRAM increases up to QEC 9 with the same number of logical qubits, only 1.45% of 331,660 physical qubits, i.e., 4,830 additional qubits were used for the redundant repair scheme. Thus, even with a high degree of QEC, we confirmed the resource overhead of our scheme is still low. Still, it is worth noting the increment of resource overhead with the higher degree of QEC. As the logical qubits of qRAM and redundant qubits use the same code distance, a higher degree of QEC will increase resource overhead for redundant qubits as well.

VI. RELATED WORKS

Practical implementations of universal quantum computing require the mitigation of quantum errors. Recent studies [44, 46, 57, 58] have focused on suppressing the two well-known noise channels, i.e., the depolarizing error and fabrication defect.

Vovrosh et al. [44] proposed a simple but effective error-mitigation technique for depolarizing error channels. As an error model ansatz, they assumed a deep quantum circuit with global depolarizing error channels. They extracted the error-free results from the noisy data in this error model. Error mitigation was first demonstrated in entanglement measurements and then in real-time dynamics of confinement in quantum spin chains. Their error-mitigation technique was deemed applicable to broader settings and in numerical simulations of more general tasks using a realistic error model. Despite its advantages, this protocol incurs a high computational overhead because the system size grows exponentially with the number of randomized unitaries and random measurements.

Tang et al. [46] proposed a defect-tolerant surface code topology with resistance to spare fabrication defects, which can be practically implemented on universal quantum computing. The authors stated that when a disk is folded into N layers, a defective physical qubit can be replaced by a working physical qubit on the same unit cell from a different layer. This technique uses one additional layer in the real physical qubit topology to store the spare fabrication errors that replace the defective physical qubits. Tang et al. also demonstrated the robustness of the surface code topology by fine-tuning the operation of the logical measure qubits. When coupling two physical qubits into one logical measure qubit, they bridged the two layers of the physical units via two schemes based on superconducting flux qubits and Xmon qubits on the circuit level. Although this technique effectively replaces a defective physical qubit, the additional physical topology incurs a large overhead.

Finally, the robustness of qRAM has recently been investigated. Giovannetti et al. [59] developed the Bucket Brigade structure, which improves the error robustness by reducing the number of gate activations in memory addressing from those of the Fanout structure. More robust qRAMs based on the Bucket Brigade structure have since been developed. For example, optimization studies of query parallelization have improved the speed of querying, and a resource estimation study has implemented a Bucket Brigade qRAM in a fault-tolerant quantum circuit model [18, 31, 32]. However, these studies focus on the errors in memory addresses and memory-addressing processes such as routing nodes and quantum gates. They aim to reduce qRAM errors by quickly minimizing the errors sourced from quantum gates and routing to each addressed node. *To the best of our knowledge*, our work is the first to consider the fabrication defects in memory cells used for read and write operations after routing. Such research is essential for improving the reliability of qRAM and realizing fully fault-tolerant quantum computing.

VII. CONCLUSION

qRAM is an essential component of quantum computers and an effective qRAM would realize the full computational benefits of quantum algorithms. Various studies have attempted to improve the fault tolerance of the Bucket Brigade architecture, which is robust against errors because its circuit model uses fewer quantum gates requiring activation than the existing Fanout model. The Bucket Brigade architecture also parallelizes queries for fast handling of quantum memory. Although studies for improving the fault tolerance of the Bucket Brigade circuitry are underway, errors in the qubits of memory cells have been neglected.

Here we considered the error-proneness of qubits consisting of actual memory cells for fully fault-tolerant qRAM. Additionally, we proposed a redundant repair scheme for qRAM that reduces the number of physical qubits required in the existing QEC method by assigning spare logical qubits. After implementing this scheme, the qRAM yield increased to 99.35% while the proportion of additional physical qubits was only 1.01% of all physical qubits. In addition, in qRAMs with 16, 32, 64, 128, 256, 512, and 1,024 logical qubits, the yield improved by 3.05%, 6.01%, 12.08%, 22.09%, 39.39%, 62.14%, and 83.59%, respectively. Our experimental results confirmed that our redundant repair scheme improves the yield and reduces the resource overheads of qRAM. The proposed scheme will greatly enhance the efficiency of future qRAM involving a large number of mass-produced qubits.

We presented a qRAM architecture for redundancy repair, utilizing superconducting qubit technology, and showed quantum circuit model implementation. Currently, numerous studies are actively exploring various qRAM designs, each based on different qubit technologies [5, 14, 59]. In our approach, the qRAM is specifically designed to accommodate the repair of defective qubits by using redundant qubits. Therefore, the proposed method can be applied regardless of the qubit technologies used for qRAM if redundant qubits are used to replace defective qubits.

ACKNOWLEDGMENTS

This work was supported by Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [No. 2020-0-00014, A Technology Development of Quantum OS for Fault-tolerant Logical Qubit Computing Environment]

[1] P. W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in *Proceedings 35th annual symposium on foundations of computer science* (IEEE, 1994) pp. 124–134.

[2] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (ACM, 1996) pp. 212–219.

[3] C. T. Hann, *Practicality of Quantum Random Access Memory*, Ph.D. thesis, Yale University (2021).

[4] J. Bang, A. Dutta, S.-W. Lee, and J. Kim, Optimal usage of quantum random access memory in quantum machine learning, Physical Review A **99**, 012326 (2019).

[5] V. Giovannetti, S. Lloyd, and L. Maccone, Architectures for a quantum random access memory, Physical Review A **78**, 052310 (2008).

[6] T. M. De Veras, I. C. De Araujo, D. K. Park, and A. J. Da Silva, Circuit-based quantum random access memory for classical data with continuous amplitudes, IEEE Transactions on Computers **70**, 2125 (2020).

[7] M. Zidan, A.-H. Abdel-Aty, A. Khalil, M. Abdel-Aty, and H. Eleuch, A novel efficient quantum random access

memory, IEEE Access **9**, 151775 (2021).

[8] K. K. Soni and A. Rasool, Quantum-based exact pattern matching algorithms for biological sequences, ETRI Journal **43**, 483 (2021).

[9] C. A. Trugenberger, Probabilistic quantum memories, Physical Review Letters **87**, 067901 (2001).

[10] G. Schaller and R. Schützhold, Quantum algorithm for optical-template recognition with noise filtering, Physical Review A **74**, 012303 (2006).

[11] R. Schützhold, Pattern recognition on a quantum computer, Physical Review A **67**, 062311 (2003).

[12] M. Zajac and U. Störl, Towards quantum-based search for industrial data-driven services, in *2022 IEEE International Conference on Quantum Software (QSW)* (IEEE, 2022) pp. 38–40.

[13] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. (Cambridge University Press, 2010).

[14] D. K. Park, F. Petruccione, and J.-K. K. Rhee, Circuit-based quantum random access memory for classical data, Scientific reports **9**, 3949 (2019).

[15] A. M. Childs, A. W. Harrow, and P. Wocjan, Weak fourier-schur sampling, the hidden subgroup problem, and the quantum collision problem, in *STACS 2007: 24th Annual Symposium on Theoretical Aspects of Computer Science, Aachen, Germany, February 22-24, 2007. Proceedings 24* (Springer, 2007) pp. 598–609.

[16] H. Buhrman, C. Durr, M. Heiligman, P. Hoyer, F. Magniez, M. Santha, and R. De Wolf, Quantum algorithms for element distinctness, in *Proceedings 16th Annual IEEE Conference on Computational Complexity* (IEEE, 2001) pp. 131–137.

[17] S. Arunachalam, V. Gheorghiu, T. Jochym-O'Connor, M. Mosca, and P. V. Srinivasan, On the robustness of bucket brigade quantum ram, New Journal of Physics **17**, 123010 (2015).

[18] A. Paler, O. Oumarou, and R. Basmadjian, Parallelizing the queries in a bucket-brigade quantum random access memory, Physical Review A **102**, 032608 (2020).

[19] K. C. Chen, W. Dai, C. Errando-Herranz, S. Lloyd, and D. Englund, Scalable and high-fidelity quantum random access memory in spin-photon networks, PRX Quantum **2**, 030319 (2021).

[20] Google Quantum AI, R. Acharya, I. Aleiner, R. Allen, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, J. Atalaya, R. Babbush, D. Bacon, J. C. Bardin, J. Basso, A. Bengtsson, S. Boixo, G. Bortoli, A. Bourassa, J. Bovaird, L. Brill, M. Broughton, B. B. Buckley, D. A. Buell, T. Burger, B. Burkett, N. Bushnell, Y. Chen, Z. Chen, B. Chiaro, J. Cogan, R. Collins, P. Conner, W. Courtney, A. L. Crook, B. Curtin, D. M. Debroy, A. Del Toro Barba, S. Demura, A. Dunsworth, D. Eppens, C. Erickson, L. Faoro, E. Farhi, R. Fatemi, L. Flores Burgos, E. Forati, A. G. Fowler, B. Foxen, W. Giang, C. Gidney, D. Gilboa, M. Giustina, A. Grajales Dau, J. A. Gross, S. Habegger, M. C. Hamilton, M. P. Harrigan, S. D. Harrington, O. Higgott, J. Hilton, M. Hoffmann, S. Hong, T. Huang, A. Huff, W. J. Huggins, L. B. Ioffe, S. V. Isakov, J. Iveland, E. Jeffrey, Z. Jiang, C. Jones, P. Juhas, D. Kafri, K. Kechedzhi, J. Kelly, T. Khattar, M. Khezri, M. Kieferová, S. Kim, A. Kitaev, P. V. Klimov, A. R. Klots, A. N. Korotkov, F. Kostritsa, J. M. Kreikebaum, D. Landhuis, P. Laptev, K.-M. Lau, L. Laws, J. Lee, K. Lee, B. J. Lester, A. Lill, W. Liu, A. Locharla, E. Lucero, F. D. Malone, J. Marshall, O. Martin, J. R. McClean, T. McCourt, M. McEwen, A. Megrant, B. Meurer Costa, X. Mi, K. C. Miao, M. Mohseni, S. Montazeri, A. Morvan, E. Mount, W. Mruczkiewicz, O. Naaman, M. Neeley, C. Neill, A. Nersisyan, H. Neven, M. Newman, J. H. Ng, A. Nguyen, M. Nguyen, M. Y. Niu, T. E. O'Brien, A. Opremcak, J. Platt, A. Petukhov, R. Potter, L. P. Pryadko, C. Quintana, P. Roushan, N. C. Rubin, N. Saei, D. Sank, K. Sankaragomathi, K. J. Satzinger, H. F. Schurkus, C. Schuster, M. J. Shearn, A. Shorter, V. Shvarts, J. Skruzny, V. Smelyanskiy, W. C. Smith, G. Sterling, D. Strain, M. Szalay, A. Torres, G. Vidal, B. Villalonga, C. Vollgraff Heidweiller, T. White, C. Xing, Z. J. Yao, P. Yeh, J. Yoo, G. Young, A. Zalcman, Y. Zhang, and N. Zhu, Suppressing quantum errors by scaling a surface code logical qubit, Nature **614**, 676 (2023).

[21] M. Abobeih, Y. Wang, J. Randall, S. Loenen, C. Bradley, M. Markham, D. Twitchen, B. Terhal, and T. Taminiau, Fault-tolerant operation of a logical qubit in a diamond quantum processor, Nature **606**, 884 (2022).

[22] F. Hua, Y. Chen, Y. Jin, C. Zhang, A. Hayes, Y. Zhang, and E. Z. Zhang, Autobraid: A framework for enabling efficient surface code communication in quantum computing, in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture* (2021) pp. 925–936.

[23] A. Molavi, A. Xu, M. Diges, L. Pick, S. Tannu, and A. Albarghouthi, Qubit mapping and routing via maxsat, in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)* (IEEE, 2022) pp. 1078–1091.

[24] D. Rotta, F. Sebastiano, E. Charbon, and E. Prati, Quantum information density scaling and qubit operation time constraints of cmos silicon-based quantum computer architectures, npj Quantum Information **3**, 26 (2017).

[25] A. Zwerver, T. Krähenmann, T. Watson, L. Lampert, H. C. George, R. Pillarisetty, S. Bojarski, P. Amin, S. Amitonov, J. Boter, *et al.*, Qubits made by advanced semiconductor manufacturing, Nature Electronics **5**, 184 (2022).

[26] K. N. Smith, G. S. Ravi, J. M. Baker, and F. T. Chong, Scaling superconducting quantum computers with chiplet architectures, in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)* (IEEE, 2022) pp. 1092–1109.

[27] S. Maurya and S. Tannu, Compaqt: Compressed waveform memory architecture for scalable qubit control, in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)* (IEEE, 2022) pp. 1059–1077.

[28] R. W. Heeres, P. Reinhold, N. Ofek, L. Frunzio, L. Jiang, M. H. Devoret, and R. J. Schoelkopf, Implementing a universal gate set on a logical qubit encoded in an oscillator, Nature communications **8**, 94 (2017).

[29] Y.-H. Luo, M.-C. Chen, M. Erhard, H.-S. Zhong, D. Wu, H.-Y. Tang, Q. Zhao, X.-L. Wang, K. Fujii, L. Li, *et al.*, Quantum teleportation of physical qubits into logical code spaces, Proceedings of the National Academy of Sciences **118**, e2026250118 (2021).

[30] B. Duan, J. Yuan, C.-H. Yu, J. Huang, and C.-Y. Hsieh, A survey on HHL algorithm: From theory to application in quantum machine learning, Physics Letters A **384**, 126595 (2020).

[31] O. Di Matteo, V. Gheorghiu, and M. Mosca, Fault-tolerant resource estimation of quantum random-access memories, IEEE Transactions on Quantum Engineering **1**, 1 (2020).

[32] C. T. Hann, G. Lee, S. Girvin, and L. Jiang, Resilience of quantum random access memory to generic noise, PRX Quantum **2**, 020311 (2021).

[33] A. J., A. Adedoyin, J. Ambrosiano, P. Anisimov, W. Casper, G. Chennupati, C. Coffrin, H. Djidjev, D. Gunter, S. Karra, N. Lemons, S. Lin, A. Malyzhenkov, D. Mascarenas, S. Mniszewski, B. Nadiga, D. O'malley, D. Oyen, S. Pakin, L. Prasad, R. Roberts, P. Romero, N. Santhi, N. Sinitsyn, P. J. Swart, J. G. Wendelberger, B. Yoon, R. Zamora, W. Zhu, S. Eidenbenz, A. Bärtschi, P. J. Coles, M. Vuffray, and A. Y. Lokhov, Quantum algorithm implementations for beginners, ACM Transactions on Quantum Computing **3** (2022).

[34] A. Montanaro, Quantum algorithms: an overview, npj Quantum Information **2**, 1 (2016).

[35] N. Goss, A. Morvan, B. Marinelli, B. K. Mitchell, L. B. Nguyen, R. K. Naik, L. Chen, C. Jünger, J. M. Kreikebaum, D. I. Santiago, *et al.*, High-fidelity qutrit entangling gates for superconducting circuits, Nature Communications **13**, 7481 (2022).

[36] F. Leymann and J. Barzen, The bitter truth about gate-based quantum algorithms in the nisq era, Quantum Science and Technology **5**, 044007 (2020).

[37] K. Phalak, A. Chatterjee, and S. Ghosh, Quantum Random Access Memory for Dummies, Sensors **23**, 10.3390/s23177462 (2023).

[38] R. Asaka, K. Sakai, and R. Yahagi, Two-level quantum walkers on directed graphs. ii. application to quantum random access memory, Physical Review A **107**, 022416 (2023).

[39] L. Egan, D. M. Debroy, C. Noel, A. Risinger, D. Zhu, D. Biswas, M. Newman, M. Li, K. R. Brown, M. Cetina, *et al.*, Fault-tolerant control of an error-corrected qubit, Nature **598**, 281 (2021).

[40] Y. Suzuki, T. Sugiyama, T. Arai, W. Liao, K. Inoue, and T. Tanimoto, Q3de: A fault-tolerant quantum computer architecture for multi-bit burst errors by cosmic rays, in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)* (IEEE, 2022) pp. 1110–1125.

[41] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, *et al.*, Realizing repeated quantum error correction in a distance-three surface code, Nature **605**, 669 (2022).

[42] Z. Ni, S. Li, X. Deng, Y. Cai, L. Zhang, W. Wang, Z.-B. Yang, H. Yu, F. Yan, S. Liu, *et al.*, Beating the break-even point with a discrete-variable-encoded logical qubit, Nature , 1 (2023).

[43] A. Jayashankar and P. Mandayam, Quantum error correction: Noise-adapted techniques and applications, Journal of the Indian Institute of Science , 1 (2022).

[44] J. Vovrosh, K. E. Khosla, S. Greenaway, C. Self, M. S. Kim, and J. Knolle, Simple mitigation of global depolarizing errors in quantum simulations, Physical Review E **104**, 035309 (2021).

[45] J. M. Auger, H. Anwar, M. Gimeno-Segovia, T. M. Stace, and D. E. Browne, Fault-tolerance thresholds for the surface code with fabrication errors, Physical Review A **96**, 042316 (2017).

[46] Y.-C. Tang and G.-X. Miao, Robust surface code topology against sparse fabrication defects in a superconducting-qubit array, Physical Review A **93**, 032322 (2016).

[47] S. Nagayama, A. G. Fowler, D. Horsman, S. J. Devitt, and R. Van Meter, Surface code error correction on a defective lattice, New Journal of Physics **19**, 023050 (2017).

[48] C. Piveteau, D. Sutter, S. Bravyi, J. M. Gambetta, and K. Temme, Error mitigation for universal gates on encoded qubits, Physical review letters **127**, 200505 (2021).

[49] A. Bilmes, A. Megrant, P. Klimov, G. Weiss, J. M. Martinis, A. V. Ustinov, and J. Lisenfeld, Resolving the positions of defects in superconducting quantum bits, Scientific reports **10**, 3090 (2020).

[50] R. Barends, J. Kelly, A. Megrant, D. Sank, E. Jeffrey, Y. Chen, Y. Yin, B. Chiaro, J. Mutus, C. Neill, *et al.*, Coherent josephson qubit suitable for scalable quantum integrated circuits, Physical review letters **111**, 080502 (2013).

[51] J. Verjauw, R. Acharya, J. Van Damme, T. Ivanov, D. P. Lozano, F. Mohiyaddin, D. Wan, J. Jussot, A. Vadiraj, M. Mongillo, *et al.*, Path toward manufacturable superconducting qubits with relaxation times exceeding 0.1 ms, npj Quantum Information **8**, 93 (2022).

[52] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Physical Review A **86**, 032324 (2012).

[53] J. P. Bonilla Ataides, D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, The XZZX surface code, Nature Communications **12**, 2172 (2021).

[54] 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 *2021 58th ACM/IEEE Design Automation Conference (DAC)* (2021) pp. 451–456.

[55] B. H. Fang, *Embedded Memory BIST for Systems-on-a-Chip*, Master's thesis, McMaster University (2003).

[56] V. Sridhar and M. R. Prasad, Built-in self-repair (bisr) technique widely used to repair embedded random access memories (rams), International Journal of Computer Science Engineering (IJCSE) **1**, 42 (2012).

[57] D. Qin, X. Xu, and Y. Li, An overview of quantum error mitigation formulas, Chinese Physics B (2022).

[58] A. Strikis, S. C. Benjamin, and B. J. Brown, Quantum computing is scalable on a planar array of qubits with fabrication defects, Phys. Rev. Appl. **19**, 064081 (2023).

[59] V. Giovannetti, S. Lloyd, and L. Maccone, Quantum random access memory, Physical review letters **100**, 160501 (2008).