Reduction of Resources for a Fault-tolerant qRAM using Pieceable Bucket-Brigade Schemes

Bernard Ousmane Sane
Graduate School of Media and Governance, Keio University, Japan
bernard@sfc.wide.ad.jp

Praveen Balaji
Department of Physics, 1110 West Green Street Urbana, IL 61801-3003, USA
pbalaji4@illinois.edu

Michal Hajdušek
Graduate School of Media and Governance, Keio University, Japan
michal@sfc.wide.ad.jp

Liang Jiang
Pritzker School of Molecular Engineering The University of Chicago, USA
liang.jiang@uchicago.edu

Rodney Van Meter
Faculty of Environment and Information Studies, Keio University, Japan
rdv@sfc.wide.ad.jp

2024 IEEE International Conference on Quantum Computing and Engineering (QCE) | 979-8-3315-4137-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/QCE60285.2024.00153

***Abstract*—Quantum random access memory (qRAM) is valuable for applying quantum computers to a broad range of problems and for successfully implementing large-scale quantum computation systems. Despite its importance, its adoption may be restricted by the fact that the decomposition of the qRAM function contains non-transversal gates, which cannot be executed transversally on any quantum error correction (QEC) code. To overcome these limitations, existing works employ techniques like magic state distillation. However, distillation is time- and qubit-intensive. This paper presents a fault-tolerant, resource-efficient implementation of qRAM based on pieceable fault tolerance. By breaking the logical CSWAP and Toffoli gate into simpler, easier-to-execute but non-transversal components, we first show that bucket-brigade qRAM can be implemented in polynomial depth without the need for ancillary magic states, further gate decomposition, or other gates. Afterward, we demonstrate the resource efficiency of this method of attaining fault tolerance in quantum operations. Based on resource estimation results, fault-tolerant bucket brigades using the pieceable technique perform better than fault-tolerant bucket brigades using magic state distillation with respect to the number of physical qubits, time, and cost.**

***Index Terms*—Quantum Random-Access Memory, Quantum computer, Quantum error correction**

I. Introduction

The ability to load extensive classical data sets into quantum processors will be crucial for applying quantum computers to various problems and, ultimately, for the success of large-scale quantum computation as an industry. It is a significant challenge to create a quantum superposition of a data set without incurring prohibitive cost at run time. The $O(N)$ quantum cost to convert N unstructured classical data elements (each with a key, or address) to a superposition is inescapable. We face the choice of paying this cost in hardware, software, or a combination of the two.

Quantum Random Access Memory (qRAM), the quantum equivalent of classical random access memory (RAM), has been proposed as a general solution to this problem. qRAM differs from classical RAM in that it utilizes qubits instead of classical bits as addresses. The use of qRAM as a quantum oracle facilitates efficient implementation of a wide variety of quantum algorithms such as machine learning, chemistry, etc. [2]–[8].

Various qRAM architectures have been proposed to efficiently query classical data on a quantum computer, choosing different points in the hardware/software tradeoff [1], [9]–[14]. Variants exist, but for the purposes of this paper we will assume qRAM designs that execute the unitary operation

$$\sum_j \alpha_j |j\rangle |0\rangle \longrightarrow \sum_j \alpha_j |j\rangle |b_j\rangle , \tag{1}$$

where j is the memory address and b_j is the classical value at that address. The data value register $|0\rangle \longrightarrow |b_j\rangle$ we will refer to as the *data bus* or *bus qubit*.

If it can be built, potentially the most valuable form of qRAM is a design with logarithmic circuit depth (access time) $\Theta(\log(N))$. Thinking in physical terms, the bus qubit must be distributed to individual memory cells ("downstreaming") where a small portion of the quantum amplitude will be acted upon separately, then reassembled ("upstreaming") for further processing as a unified register [1]. The quantum address and bus registers become entangled but (of course) remain unentangled with the classical memory, which we can treat as read-only.

To understand downstreaming, RAM databases can be modeled as a complete binary tree in which the address bus consists of bits that direct a request down the tree ("route" the request), taking the left or right path on the i^{th} level of the tree based on the value of the i^{th} address bit. To address N total data elements, we need an address that is $n = \log_2 N$ bits. A Fanout qRAM can be built using an n-level binary tree arrangement of nodes, providing $\Theta(\log(N))$ latency.

This approach, however, suffers from decoherence problems because each of the $O(N)$ active nodes each temporarily holds

This work is supported by JST Moonshot R&D Grant (JPMJMS2061).

[1]For example, a polarization photonic qubit can be split to run through two paths via a polarizing beamsplitter, allowing the different states to be operated on independently. The amplitudes will be split accordingly, retaining any entanglement or relationship to other qubits.

979-8-3315-4137-8/24/$31.00 ©2024 IEEE
DOI 10.1109/QCE60285.2024.00153

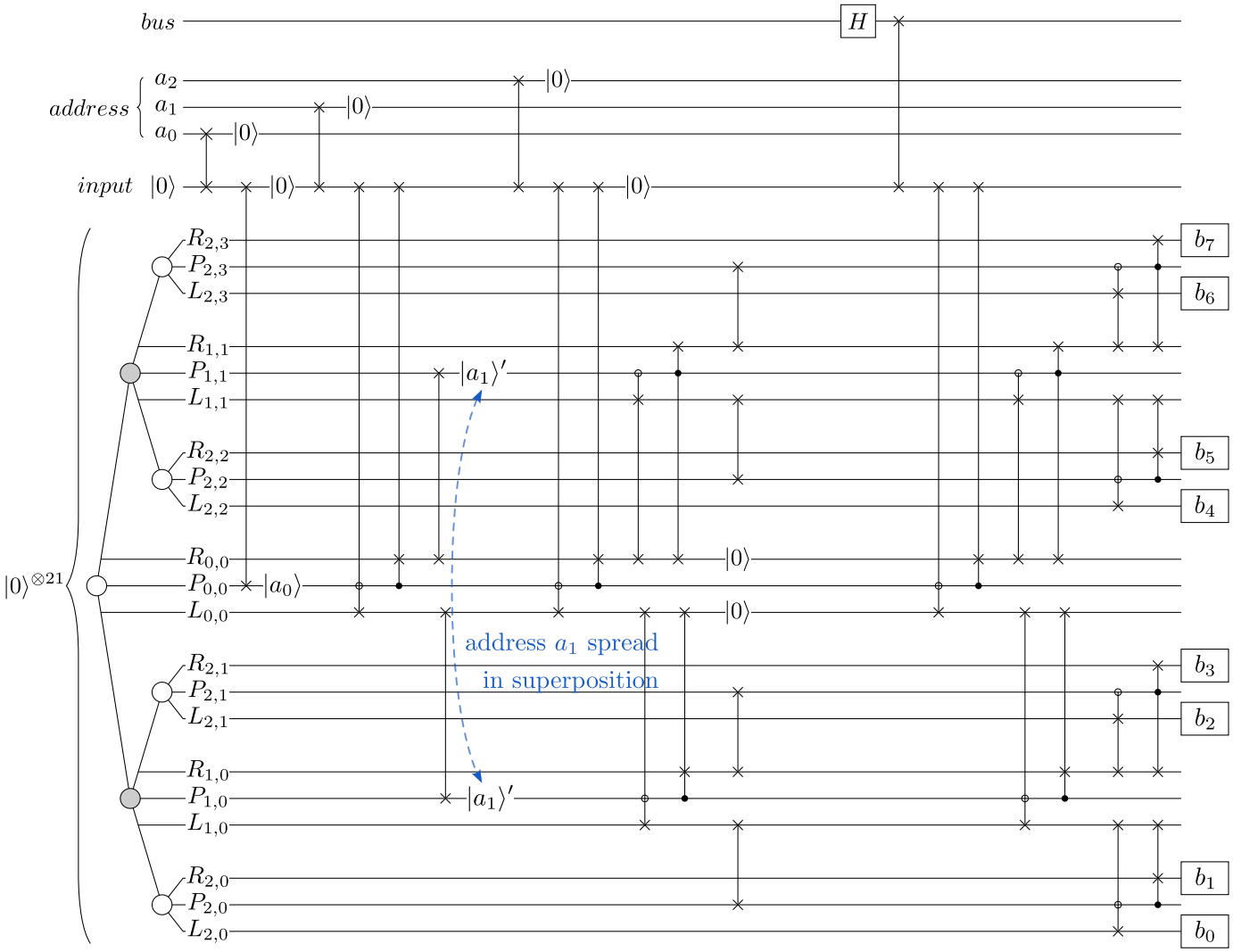

Fig. 1. Hann's bucket brigade qRAM circuit for the case of $n = 3$ [1]. The address register is first loaded into the binary tree (illustrated on the left), with three qubits per node. First, address qubit $|a_0\rangle$ is transferred to the root of the tree. Lower-order address qubits such as $|a_1\rangle \rightarrow |a_1\rangle'$ are spread to nodes lower in the tree to route the bus qubit to the classical memory cells (b_i). Many of the qubits in the circuit are used only briefly, and spend most of their lifetime in a $|0\rangle$ state separate from our qubits of interest; a few of these are marked on the figure.

a qubit during the read operation. The importance of errors in qRAM, and therefore whether addresses, data elements, or both need to be protected via QEC, depends on the intended application.

A significant improvement over Fanout qRAM, known as *bucket brigade*, reduces the number of entangled qubits scattered throughout the device and reduces the number of gates that must directly interact with each address qubit [9]. Two significant changes are made. Address qubits and a bus qubit are literally routed into a binary tree to reach the classical memory cells, reducing the demand on each address qubit as it comes into play by "copying" it in binary tree fashion. The implementation of the tree is achieved through the use of three states $|0\rangle, |1\rangle,$ and $|W\rangle$ (wait) instead of the two in Fanout qRAM. All the nodes are initially initialized in the (inactive) state $|W\rangle$. Despite using $\sim 3N$ qubits, only N are entangled at one time.

Paler *et al.* [13] have proposed a parallel version of the bucket brigade that reduces the circuit depth from $\Theta(N)$ to $\Theta(\log(N))$ in the worst case scenario (when the query is executed on the $N = 2^n$ memory cells, where n is the number of address bits) without ancillary qubits, as opposed to the construction of Di Matteo *et al.* [11], which proposed an $O(N)$ number of ancillary qubits in the parallel version.

However, the bucket brigade approach is still sensitive to noise and errors; the error scales polylogarithmically with the size of the qRAM [15]. Unfortunately, it has been seen that conventional QEC techniques for the bucket-brigade architecture have large resource costs [14]. The main issue lies in the fact that the bucket-brigade circuit uses the Toffoli gate, which cannot be implemented transversally using a CSS code [14] or surface code [11]. Instead, the Toffoli gate is generally achieved via magic state distillation for the T gate, which entails a significant investment of time

and qubits, despite recent improvements [16]. Moreover, the bucket brigade decomposed in terms of Clifford+T gates has significant resource consumption [17].

This paper demonstrates that the non-transversality problem in bucket-brigade qRAM can be addressed by breaking the overall circuit into pieces that can individually be implemented fault-tolerantly, resulting in better logical error rates with fewer resources, unlike existing approaches. Hence, we present a model of bucket-brigade qRAM that can be executed fault tolerantly using pieceable techniques by breaking down logical circuits into more straightforward, more manageable components without the need for ancillary magic states or other gates. We show formally how bucket-brigade qRAM can be achieved in circuits whose depth is polynomial in the number of address qubits (i.e., $O(\log^k N)$ for some constant k) for an N-memory element qRAM, although they require $O(N)$ gates, all of which are either Toffoli or CNOTs. Then, the pieceable fault-tolerance principle is implemented to maintain error resistance for these two gate types.

Subsequently, we evaluate the resource requirements in comparison to alternative fault-tolerant implementations employing magic state distillation like surface code. As a result, we demonstrate that our proposal is more resource-efficient than the previous approach. Compared to our approach, fault-tolerant bucket-brigade qRAM implementations that employ magic state distillation have a constant multiplicative gain (approximately 5, 12, and 1.2 for time, physical, and total costs, respectively). In addition, we expect that these results will lead to a low rate of logical errors.

We begin with a full introduction to bucket brigade qRAM, including pseudocode for a basic version and an optimized version in Sec. II [2]. We present the concept of Pieceable Fault-Tolerance in Section III. We introduce the proposed Pieceable Fault-Tolerance Bucket-Brigade in section IV. We analyze our proposals in section V. We conclude this paper with a discussion in Section VI.

II. Circuits for $O(N)$ Bucket-Brigade qRAM with $O(\text{polylog}(N))$ Depth

Our circuit representation of qRAM builds on the detailed bucket brigade circuit of Hann [1]. The circuit consists of five phases: address distribution, bus distribution, data read, bus un-distribution, and address un-distribution. The un-distribution phases are the same gates as the distribution, in reverse order. The first three phases of an $N = 8$ instance are shown in Fig. 1. Algorithm 1 is a pseudocode for generating the complete circuit (*not* the execution-time logic) for the first three phases of a scalable circuit. In the pseudocode, CSWAP$_\bullet$ represents the ordinary CSWAP that swaps its second and third arguments if the control bit is 1, while CSWAP$_\circ$ swaps them if its control bit is 0.

[2]Although quite a number of papers present basic diagrams for fixed-size versions and describe larger extensions, to the best of our knowledge this is the first complete pseudocode in the literature.

The address register (typically initialized in $|+\rangle^{\otimes n}$ for the first call) is swapped one qubit at a time into the binary tree via the input register, converting the binary representation into a unary one. Each "node" of the tree consists of three qubits, called $|P\rangle$, $|L\rangle$, and $|R\rangle$ (for path select, left, and right). The latter two are only briefly used as address qubit $|a_{i+1}\rangle$ is distributed left or right based on $|a_i\rangle$, except for the last level of the tree, as described below. When subscripted $|P_{i,j}\rangle$, etc., i is the level in the binary tree (starting with $i = 0$ at the root) and $0 \le j < 2^i$ indicates which node within that level.

After the address distribution, the $|P\rangle$ qubits hold the address qubits, and the bus qubit is routed down the tree, also temporarily using the $|L\rangle$ and $|R\rangle$ qubits, until reaching the bottom of the tree, where $|R\rangle$ and $|L\rangle$ actually read all N physical classical memories simultaneously, represented by b_j in Fig. 1. The read operation is achieved by a classically controlled Z gate applied to the $|R_{n-1}\rangle$ and $|L_{n-1}\rangle$ states. Following this Z operation with a Hadamard will give the readout as defined in (1).

The limiting factor in performance will be the number of CSWAP gates and their performance. To work with pieceable fault tolerance (below), we use a CCZ gate and the decomposition shown in Fig. 3. When the lower qubit is known to begin in $|0\rangle$ (as happens in our circuits), the first CNOT can be eliminated.

Descriptions of bucket brigade qRAM often begin with the $|P\rangle$ qubits in a tertiary state $|W\rangle$, but to work with error correction, we assume all of the qubits in the tree begin in $|0\rangle$. This assumption also allows us to somewhat optimize the circuit, since the common three-CNOT implementation of SWAP can be reduced to two CNOTs if one of the qubits is known to be $|0\rangle$. The optimized circuit is shown in Algorithm 2. As with the CSWAP$_\circ$, Toffoli$_{\circ,\bullet}$ indicates that the NOT on the target qubit is executed if the first control qubit is 0 and the second control qubit is 1. (Information and gates not included are the same as Algorithm 1.)

For an $N = 2^n$-element qRAM, Hann uses $3(N-1)+n+2$ qubits, but only N non-zero amplitudes are present at any time. Depending on actual physical implementation, including how qubits are physically transferred within the device, some of these qubits could potentially be eliminated during further optimization. For example, the circuits as presented include unconditional SWAP gates where one of the qubits is known to be $|0\rangle$; physically routing the qubit instead will reduce the number of qubits in the system.

Although the device requires $O(N)$ quantum devices and $O(N)$ quantum gates, the lowest level of the tree also executes $O(N)$ operations simultaneously. Overall, the circuit depth of the address distribution, the bottleneck in the process, is $O(\log^2 N)$. More details on gate counts and performance are presented in Sec. V.

Algorithm 1 Hann Bucket Brigade Downstreaming and Copying

Input: An n-qubit address $(a_0, \ldots, a_{n-1})$ with a_0 the high-order bit, and a bus qubit

Output: Memory data for the given address

Resources: N classical memories, $O(N)$ q. ancillae

Preconditions: $|P\rangle_{i,j} = |0\rangle$

Runtime (circuit depth): $\Theta(n^2)$

Gates: (Before we distribute the bus qubit) $\Theta(n^2 \times 2^n)$ SWAP, and $\Theta(n^2 \times 2^n)$ CSWAP gates

{binary→unary address conversion}
for $i = 0$ to $n-1$ **do** {iterate over levels}
  SWAP($|a_i\rangle$, $Input$)
  if i == 0 **then**
    SWAP($Input$,$P_{0,0}$)
  else
    CSWAP$_\bullet$($P_{0,0}$, $Input$, $R_{0,0}$)
    CSWAP$_\circ$ ($P_{0,0}$, $Input$, $L_{0,0}$)
    if i == 1 **then**
      SWAP($L_{0,0}$, $P_{1,0}$)
      SWAP($R_{0,0}$, $P_{1,1}$)
    else
      for $k = 1$ to $i-1$ **do**
        for $j = 0$ to $2^k - 1$ **do**
          if j % 2 == 0 **then**
            CSWAP$_\circ$($P_{k,j}$, $L_{k,j}$, $L_{k-1,j/2}$)
            CSWAP$_\bullet$($P_{k,j}$, $R_{k,j}$, $L_{k-1,j/2}$)
          else
            CSWAP$_\circ$($P_{k,j}$, $L_{k,j}$, $R_{k-1,(j-1)/2}$)
            CSWAP$_\bullet$($P_{k,j}$, $R_{k,j}$, $R_{k-1,(j-1)/2}$)
          end if
          SWAP($L_{i-1,j}$, $P_{i,2*j}$)
          SWAP($R_{i-1,j}$, $P_{i,2*j+1}$)
        end for
      end for
    end if
  end if
end for
{next, distribute the bus qubit ($|+\rangle$)}
H($|bus\rangle$)
SWAP($Input$,$|bus\rangle$)
CSWAP$_\bullet$($P_{0,0}$, input, $R_{0,0}$)
CSWAP$_\circ$ ($P_{0,0}$, input, $L_{0,0}$)
for $i = 1$ to $n-1$ **do**
  for $j = 0$ to $2^i - 1$ **do**
    if j % 2 == 0 **then**
      CSWAP$_\circ$($P_{i,j}$, $L_{i,j}$, $L_{i-1,j/2}$)
      CSWAP$_\bullet$($P_{i,j}$, $R_{i,j}$, $L_{i-1,j/2}$)
    else
      CSWAP$_\circ$($P_{i,j}$, $L_{i,j}$, $R_{i-1,(j-1)/2}$)
      CSWAP$_\bullet$($P_{i,j}$, $R_{i,j}$, $R_{i-1,(j-1)/2}$)
    end if
  end for
end for
Copy data from the selected memory to the bus qubit.
Following this, the above must be uncomputed.
return memory data

Algorithm 2 An optimized downstreaming and copying process for Hann Bucket Brigade

Gates: (Before we distribute the bus qubit) $\Theta(n^2 \times 2^n)$ CNOT, and $\Theta(n^2 \times 2^n)$ Toffoli

{binary→unary address conversion}
for $i = 0$ to $n-1$ **do** {iterate over levels (n is excluded)}
  CNOT($Input$, $|a_i\rangle$)
  CNOT($|a_i\rangle$, $Input$)
  if i == 0 **then**
    CNOT($Input$, $P_{0,0}$)
    CNOT($P_{0,0}$, $Input$)
  else
    Toffoli$_{\bullet,\bullet}$($P_{0,0}$, $Input$, $R_{0,0}$)
    CNOT($R_{0,0}$, $Input$,)
    Toffoli$_{\circ,\bullet}$($P_{0,0}$, $Input$, $L_{0,0}$)
    CNOT($L_{0,0}$, $Input$)
    if i == 1 **then**
      CNOT($L_{0,0}$, $P_{1,0}$)
      CNOT($P_{1,0}$, $L_{0,0}$)
      CNOT($R_{0,0}$, $P_{1,1}$)
      CNOT($P_{1,1}$, $R_{0,0}$)
    else
      for $k = 1$ to $i-1$ **do**
        for $j = 0$ to $2^k - 1$ **do**
          if j % 2 == 0 **then**
            Toffoli$_{\circ,\bullet}$($P_{k,j}$, $L_{k-1,j/2}$, $L_{k,j}$)
            CNOT($L_{k,j}$, $L_{k-1,j/2}$)
            Toffoli$_{\bullet,\bullet}$($P_{k,j}$, $L_{k-1,j/2}$, $R_{k,j}$)
            CNOT($R_{k,j}$, $L_{k-1,j/2}$)
          else
            Toffoli$_{\circ,\bullet}$($P_{k,j}$, $R_{k-1,(j-1)/2}$, $L_{k,j}$)
            CNOT($L_{k,j}$, $R_{k-1,(j-1)/2}$)
            Toffoli$_{\bullet,\bullet}$($P_{k,j}$, $R_{k-1,(j-1)/2}$, $R_{k,j}$)
            CNOT($R_{k,j}$, $R_{k-1,(j-1)/2}$)
          end if
          CNOT($L_{i-1,j}$, $P_{i,2*j}$)
          CNOT($P_{i,2*j}$, $L_{i-1,j}$)
          CNOT($R_{i-1,j}$, $P_{i,2*j+1}$)
          CNOT($P_{i,2*j+1}$, $R_{i-1,j}$)
        end for
      end for
    end if
  end if
end for
{bus qubit ($|+\rangle$ distribution is unchanged)}
Copy data from the selected memory to the bus qubit.
Uncompute the bus distribution and address encoding above.
return memory data

TABLE I
THE [[7, 1, 3]] QUANTUM CODE STABILIZERS

K4 = XIXIXIX	K1= ZIZIZIZ
k5 = XXIIXXI	K2 =ZZIIZZI
K6 = XXXXIII	K3 = ZZZZIII

III. PIECEABLE FAULT-TOLERANCE

Based on the realization that circuits that implement logical gates can be broken into pieces that *minimize contagious errors (or error propagation)*, Yoder, Takagi and Chuang proposed the concept of *pieceable fault tolerance* (PFT) to perform error correction without needing ancillary magic states, decomposition, or other gates [18]. PFT works because non-contagious errors (those that commute with the circuit pieces) can safely be corrected at the end of the circuit. This saves on stabilizer measurements, which otherwise would introduce more ancillae and potential errors. For qRAM, the elimination of magic state distillation will provide an important saving in resources, as detailed over the coming sections.

Error detection and correction involves extracting information from logically encoded data qubits without destroying them [19], [20]. Here, we focus on *stabilizer codes*, where a logical state $|\psi\rangle$ has a stabilizer S such that $|\psi\rangle = S|\psi\rangle$, i.e., $|\psi\rangle$ is a $+1$ eigenstate of S. The stabilizer is designed to use ancilla qubits to determine the existence of errors through syndrome measurement without compromising the data qubits. Indeed, the quantum error correction circuits attempt to use syndrome measurement to extract eigenvalues without collapsing the superposition or introducing errors.

Before constructing our PFT-based qRAM, in this section we briefly introduce the key concepts of PFT. We consider the logical circuit $\mathcal{L} = L_m \dots L_2 L_1$, the Pauli group on n qubits $\mathcal{P}_n$ and the stabilizer set S, where $\mathcal{L}$ is a logical gate and L_i are circuit pieces of that gate. The commutativity of two operators A and B is symbolized by the expression $[A, B] = 0$, where $[A, B] = AB - BA$ is the commutator.

Definition III.1 (7-qubit Steane code)**.** The Steane code is a $[[7, 1, 3]]$ CSS code, where 7, 1, and 3 indicate the number of physical qubits, the number of logical qubits, and the distance between valid code words, respectively. Valid code words are in a $+1$ eigenstate of all of the stabilizers in Tab. I, and the code is capable of correcting a single error [21]. Although the Steane code is fully capable of concatenated (recursive) implementation, in this paper we focus on only a single level, allowing us to refer to "physical" and "logical" qubits without confusion.

Definition III.2 (Contagious errors)**.** Errors that *can* propagate to other qubits when executing circuit pieces L_i are described as Pauli error operators

$$\mathcal{E}_C = \{E \in \mathcal{P}_n \ \exists i \mid [E, L_i] \neq 0\}. \tag{2}$$

Contagious errors must be dealt with before further L_i operations are performed.

Definition III.3 (Non-contagious errors)**.** Errors that cannot propagate to other qubits are described as Pauli error operators,

$$\mathcal{E}_C = \{E \in \mathcal{P}_n \ \exists i \mid [E, L_i] = 0\}. \tag{3}$$

Because they do not spread from qubit to qubit, correction of these errors can be deferred without fear of error propagation.

Definition III.4 (Constant Stabilizer (S_C))**.** Constant stabilizers commute with the L_i operations to be performed,

$$\mathcal{S}_C = \{s \in S \ \forall i \mid [s, L_i] = 0\}. \tag{4}$$

Yoder et al. state that constant stabilizer measurements should be performed *first*.

Definition III.5 (Round-robin circuit)**.** A round-robin circuit applies gates to the full combinatoric set across multiple sets of qubits, used in fault-tolerant execution of logical gates. For instance, a round-robin circuit of CZ gates on two sets of three qubits A_i and B_i is formed by the nine gates $\prod_{j,k\in\{1,2,3\}} CZ(A_j, B_k)$ [18], [22]. Our logical circuit pieces L_i will be round robin on the physical qubits.

Overall, the authors consider a stabilizer group and a logical circuit (for instance, a logical CZ circuit). The latter is subdivided into parts for the simple reason that although the entire logic circuit is not fault-tolerant, certain sections nevertheless can be. However, it is important to note that some elements of the stabilizer group will be transformed into non-Pauli operators when conjugated by non-Clifford gates. Thus, it is necessary to be cautious in managing errors that may cross between parts (contagious error in (2)). To this end, it is necessary to inquire about the nature of the errors to be controlled. The authors propose completing error correction halfway through the circuit to prevent errors from propagating until they become impossible to handle. For this reason, the intermediate level is defined based on the concept of constant stabilizers of (4) and aims to correct contagious errors. Other types of errors, like non-contagious and single-qubit errors, may be allowed to remain uncorrected when entering the following circuit pieces.

IV. PIECEABLE FAULT-TOLERANCE BUCKET-BRIGADE

As shown in Sec. II, the bucket-brigade circuit can be decomposed into CNOT and CSWAP gates as shown Algorithm 1 or into Toffoli and CNOT as in Algorithm 2. As a result, we only have to consider fault tolerance of these two gates. As noted above, we did not consider using the tertiary $|W\rangle$ state since it is not required in the full circuit model for bucket brigade, slightly improving the error scaling [23].

To design our fault tolerance, we choose to use the Steane code [21] due to its rich set of transversal Clifford gates and connection to classical coding theory. Specifically, the gates H (Hadamard), S (Phase gate), and CNOT (Controlled-NOT gate) are all transversal in their simplest form. Therefore, we only have to deal with fault tolerance for CSWAP. Unfortunately, any QECC that corrects arbitrary Pauli errors cannot perform CSWAP transversally, seen easily from its

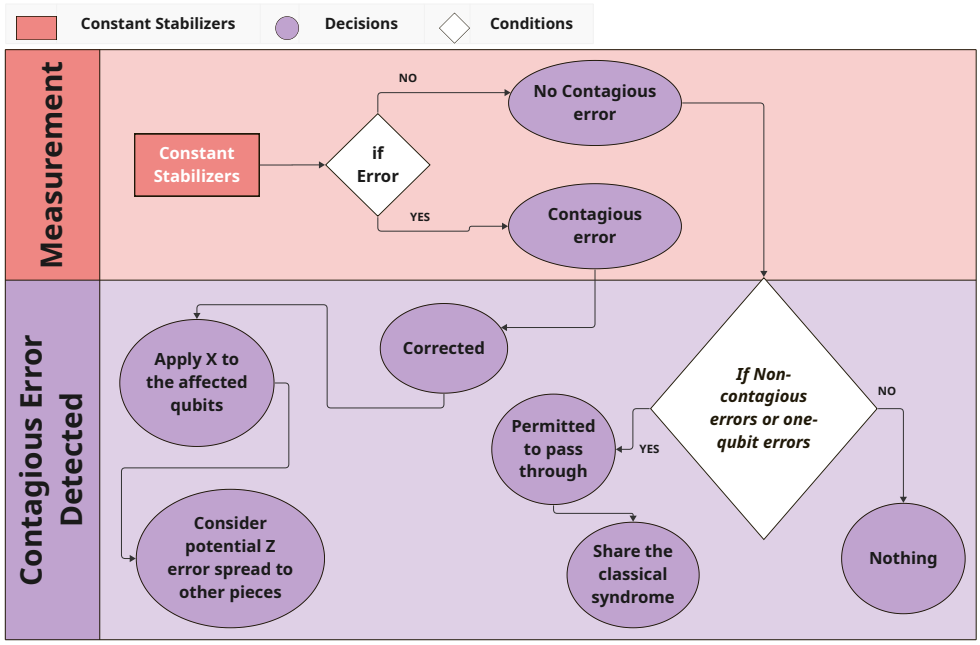

Fig. 2. The error correction strategy on an encoding block (with a maximum of one X or Y error [18]) depends on measuring constant stabilizers and taking action based on whether errors are present or not. A focus is placed on how contagious errors should be handled and how the reliability of non-constant stabilizer measurements should be maintained after an error is identified.

decomposition into a Toffoli gate in Fig. 3 [24]. This constraint will be overcome by using the pieceable technique on logical CSWAP circuits.

We can observe that each of the logical CSWAP circuits ($\mathcal{L}$) represents a set of simple actions (gates, state preparation, measurements, etc.) that take place on physical qubits. Alternatively, each of them can be described as an ordered list of these operators or actions as $\mathcal{L} = \{L_n, L_{n-1}, \ldots, L_1\}$ where the operators L_i are multiplied, with L_{i-1} acting before L_i [18].

It is important to note that each of these L_i operators will likely cause errors on the qubits on which it acts, thus introducing the faulty component problem. To meet this challenge, we will apply the Steane quantum code $\mathcal{S}$ on the circuit $\mathcal{L}$, leading to the circuit $\mathbf{L} = S_C^n L_n \ldots S_C^2 L_2 S_C^1 L_1$. After each portion, we apply an error correction (with the constant S_C^i) without altering $\mathcal{L}$ (because the portions L_i commute with stabilizers S_C^i, see (4)).

The strategy adopted consists of dividing the group of stabilizers S of the Steane code $\mathcal{S}$ into two subgroups: the first group, called constant stabilizers (S_C), includes all the stabilizers $s \in S$ which commute with all the parts L_i of the circuit (4), while the second group, called non-constant stabilizers, includes stabilizers which do not commute with all the parts here. Thus, the constant stabilizers, responsible for maintaining the Pauli circuit, will handle contagious errors, introducing no more than one single-qubit error to the qubits in the code. The elements in the constant stabilizer can be measured using Shor's fault-tolerant approach [25] with repeats and majority voting.

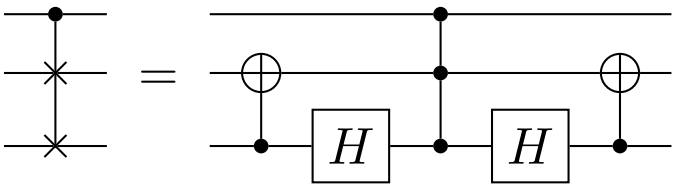

Fig. 3. Equivalent circuit representations of CSWAP. When constructing fault-tolerant CSWAP based on pieceable techniques, the construction composed of two CNOTs, two Hadamard gates, and one CCZ is essential.

The outlined strategies are shown in Fig. 2. As a first step, the eigenvalues of constant stabilizers must be measured to detect contagious errors. If a constant stabilizer detects an error, it signals a fault and allows corrections to be made as a result. If the constant stabilizers do not detect contagious errors, existing non-contagious errors can remain uncorrected. On the one hand, because the non-contagious errors do not propagate (by definition), they will be considered later in the final error correction process. Alternatively, the intermediate error correction (the correction that uses the constant stabilizer) shares information such as the position of the contagious errors regarding the syndrome measurement. As a result, the final error correction will be aware of where to look for non-contagious errors. These elements justify the need for a final error correction step to fill the gaps in the intermediate corrections.

We use the round-robin approach as in Def. III.5 to build a fault-tolerant logical circuit for a CSWAP gate by using the decomposition of CSWAP into CCZ gate in Fig. 3. Yoder *et al.*

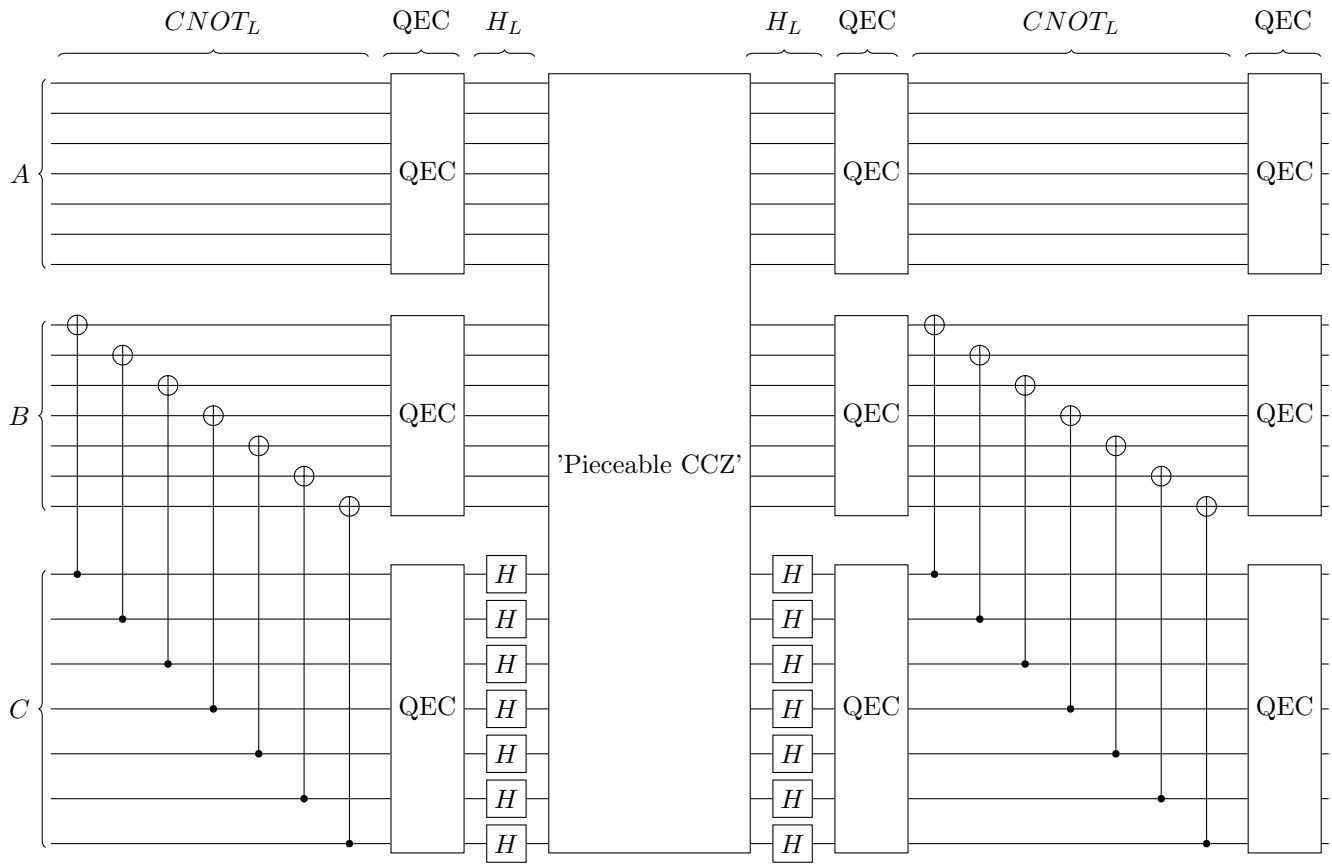

Fig. 4. Constructing the FT CSWAP using the pieceable CCZ on the Steane [[7,1,3]] code, following Fig. 3. The piecable CCZ (Fig. 5) checks constant stabilizers. QEC blocks are full checks of all six stabilizer eigenvalues and any necessary correction. The seven transversal physical CNOTs that constitute a logical CNOT can be executed in parallel.

proved that a logical CCZ gate can be formed in round-robin fashion [18]. As illustrated in Fig. 4, the logical CSWAP gate is formed between qubits of the three code blocks, denoted $\prod_{\alpha,\beta,\gamma\in\{5,6,7\}} CSWAP(A_\alpha, B_\beta, C_\gamma)$, where A, B, and C are blocks of qubits. The piecable CCZ in that figure is detailed in Fig. 5. Each CCZ interacts with a qubit at most twice as illustrated, therefore, our round-robin CSWAP can also be described as a fault-tolerant logical circuit.

V. Evaluation

Using the circuit defined in Algorithm 2, we determine the resources of the bucket brigade in terms of Toffoli gates (Table III).

As part of this section, we analyze the resource requirements for implementing fault-tolerant bucket-brigade circuits when dealing with many queries. For this purpose, we use the cost metrics in [26], defined as follows,

$$Cost = \log_2(\text{Logical qubits} \times \text{Quantum code Cycles}). \quad (5)$$

Various considerations can be applied to the cost definition, but as in [26], we consider the execution of the query's quantum circuit as the primary factor. Hence, time and space are traded off at this cost. In the following, we refer to a bucket brigade employing a fault-tolerant pieceable technique as BB-pieceable, while BB-magic state refers to a bucket brigade employing a magic state distillation technique.

A. Resource estimation settings

We use the Python-based code described in [11] to calculate the resources required [3]. This code provides a flexible environment for testing and improving the qRAM. Our pieceable Steane bucket brigade is compared to alternative fault-tolerance implementations employing magic state distillation like surface code. We use factors such as the number of physical qubits, time, and cost (5) as a means of comparing competing implementations.

Our simulation assumes a depolarizing error model, with the error probability of a gate with three qubits, two qubits, and one qubit being the same (pg=$1e-5$) [11]. However, other physical operations, such as state preparations and qubit measurements, are considered perfect [27]. Moreover, using the same approach as in [27], we observe that the state preparation has less than a 1% resource impact, so we decide it can be disregarded.

[3]https://github.com/glassnotes/FT_qRAM_Circuits

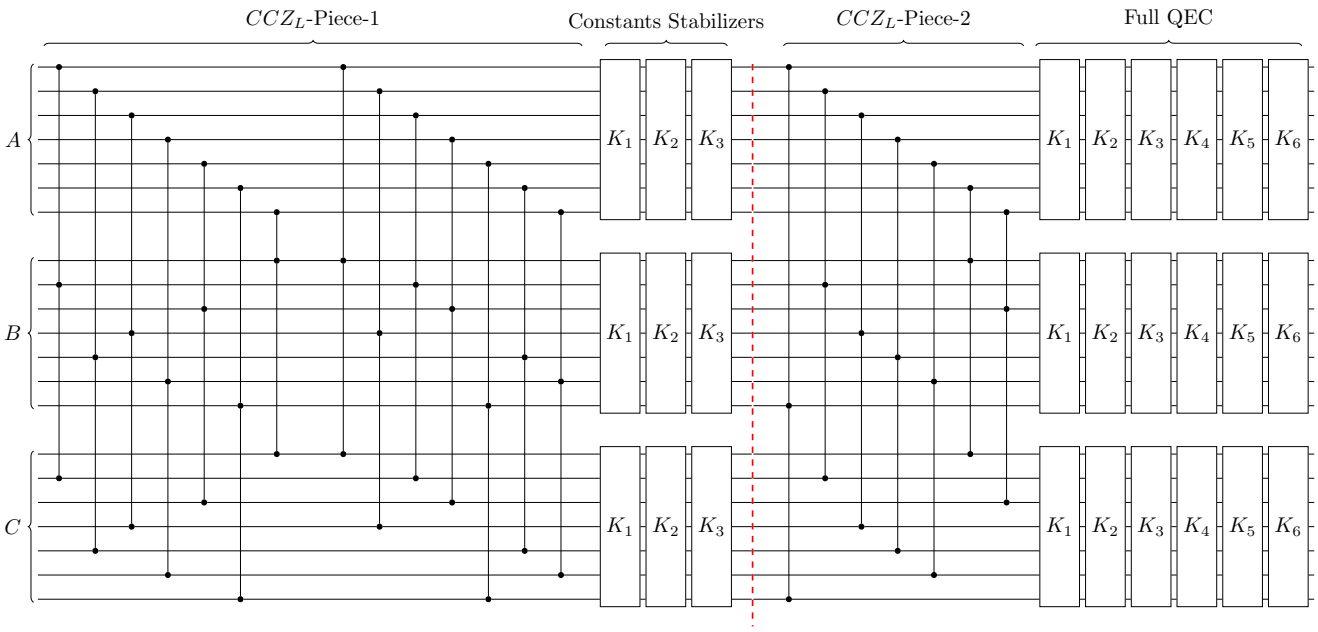

Fig. 5. The pieceable logical $\mathrm{CCZ}_L(|A\rangle|B\rangle|C\rangle)$, using the Steane code. After the first piece, the X stabilizers K_i (Tab. I) are applied to each block to deal with a maximum of one contagious X or Y error. In contrast, non-contagious Z errors are delayed for later correction (during the final correction, showing all six stabilizers) since they do not propagate. Blocks also share information about syndrome information to limit error propagation. In the final step, we measure all stabilizers, considering the crucial information at the intermediate level regarding the position of X and/or Y errors. This way, the final correction can pinpoint precisely where to look for Z errors

.

We use the algorithm in [26] to evaluate the number of rounds during the magic state distillation. As long as the magic state injection error rate is higher than the distillation output error rate, the number of cycles will increase. When considering the Clifford gates, we seek the smallest code distance that satisfies the inequality defined in [26].

Table III contains the parameters we use in our resource estimation and the configuration parameters. We follow the workflow illustrated in Fig. 6 for our resource estimation.

B. Performance evaluation

During our resource estimation, we varied n from 15 to 36 and evaluated the resource requirements for the pieceable fault tolerance technique and magic state distillation. The values 15 and 36 were selected due to their importance as $n = 15$ corresponds to 4 KB and $n = 36$ corresponds to 8 GB of classical data [11]. Tables IV&V summarize the numerical results for $n = 16$, $n = 26$, and $n = 36$.

In Fig. 7, we evaluate respectively the time (Fig. 7(a)), the number of physical qubits (Fig. 7(b)) and the cost (Fig. 7(c)) according to address size n under bucket brigades using surface code or pieceable techniques.

In Fig. 7(a), we observe that as n increases, the time required for BB-pieceable and BB-magic increases. The time varies linearly as a function of n, with deviations occurring at $n = 24$. This trend can be attributed to an increase in the number of cycles per state when distilling magical states. In fact, the number of cycles increases from 80 for $n = 23$ to 130 for $n = 24$, whereas for the other values of n, the increase was only 10 per cycle. The BB-pieceable technique on the 4 KB qRAM takes 0.832 ms compared to the BB-magic state distillation that takes 4.752 ms. Similarly, the 8 GB qRAM takes 5.153 ms compared to the BB-magic state distillation, which takes 78.528 ms. A constant multiplicative factor exists between BB-magic and BB-pieceable (approximately five to fifteen times). In other words, BB-magic is approximately five to fifteen times more time-consuming than BB-pieceable.

Figure 7(b) represents the variation of the number of physical qubits as a function of n during the implementation of the bucket brigade using pieceable technique and surface code. We observe a linear variation in the number of physical qubits with n. On the one hand, we note that a fault-tolerant bucket brigade (4KB classical memory) can be implemented using surface code and pieceable techniques with approximately 277 million and 21 million physical qubits, respectively. On the other hand, when considering 8 GB of classical data, BB-magic requires 2.17e+15 physical qubits, while BB-pieceable requires 167.62 e+12 physical qubits. Hence, we conclude that BB-magic has a constant multiplicative factor of approximately ≈ 12 compared to BB-pieceable concerning physical qubit overhead.

In Fig. 7(c), we note a linear variation of the cost (5) with n. The BB-magic has a constant multiplicative factor of approximately 1.2 compared to the BB-pieceable regarding cost overhead.

VI. Discussions and Limitations

A quantum random access memory (qRAM) has the potential to enable large-scale quantum computation systems, which is why they are of considerable interest in quantum

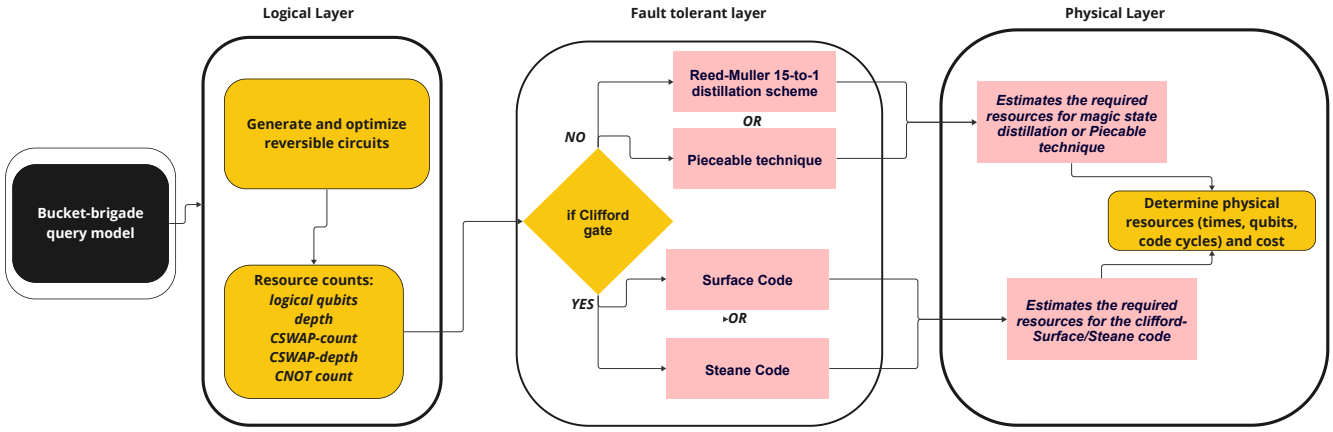

Fig. 6. An Analysis Framework for Fault-Tolerant Resource Requirements. Upon optimizing the bucket-brigade circuit and decomposing it into CNOTs and CSWAPs gates, a test is conducted on the nature of the gates. When the gate is not Clifford, it goes through a process of distillation of magical states or a pieceable procedure. In contrast, if the gate is of the Clifford type, we use the Clifford surface/Steane code to perform logical Clifford gates. A lower-level assessment of physical resources concludes this process. [26]

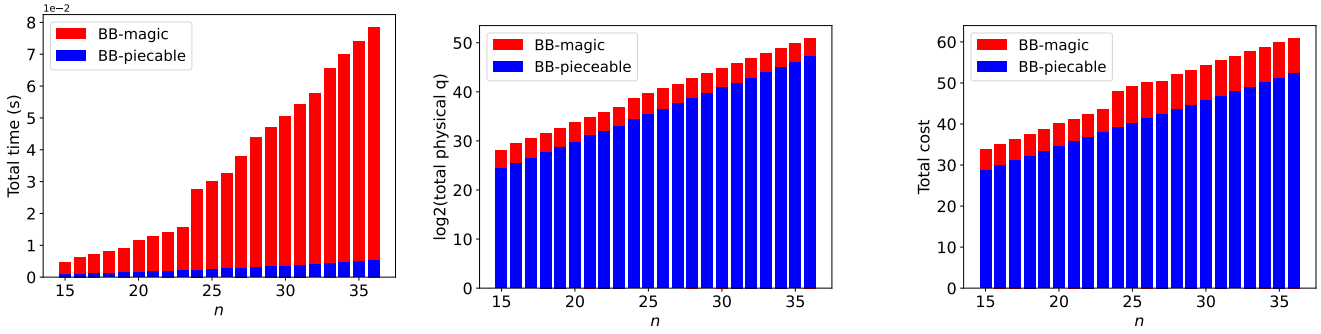

Fig. 7. Variation of the physical qubits, the time, and the cost according to address size n under bucket brigades using surface code or pieceable techniques.

TABLE II
BUCKET-BRIGADE LOGICAL RESOURCE COUNTS

Designation	Counts
Logical qubits	$3 \times 2^n + n - 1$
depth	$8 \times n^2 - 24 \times n + 36$
Toffoli	$2 \times 2^n(n^2 - 3 \times n + 2) + 4 \times (n - 1) + 2$
Toffoli-depth	$2 \times n^2 - 4 \times n + 6$
CNOT	$2^n \times (6 \times n^2 - 4 \times n + 10) - 4 \times n + 9$

TABLE III
SURFACE CODE PHYSICAL PARAMETERS [11], [26]

Designation	Counts
Magic state input failure probability	1e-4
Gate failure probability	1e-5
surface code cycle	200e-9 seconds

computing. In qRAM, the error probability associated with each query is $\approx \log^2(N) \times 10^{-5}$, corresponding to a net error rate in the range $10^{-3} \sim 10^{-2}$ for N in the range $10^3 \sim 10^9$ when no error correction is considered. Such an error per query might be sufficient for some qPCA-related applications, but might not be good enough for Grover Search (which needs $\sqrt{N}$ queries). In light of this fact, error-corrected QRAM becomes a necessity.

However, fault-tolerant designs require techniques such as magic state distillation, which is both time and qubit-intensive [24], [28].

Hence, this paper presents a fault-tolerant resource-efficient implementation of qRAM based on Yoder et al.'s "pieceable fault tolerance". First, we demonstrate that qRAMs with bucket-brigade gates can be implemented using polynomial depth computing through CSWAPs and CNOTs or Toffoli and CNOTs. Because we only need to maintain fault tolerance for two gates, CSWAP and CNOT, we employ pieceable techniques on their logical circuits through their division into pieces and applying error correction to each part of the circuit. Through this approach, we show that compared with BB-pieceable, BB-magic has a constant multiplicative factor (approximately 5, 12, and 1.2 for time, physical, and total cost, respectively). Additionally, we expect low logical errors

TABLE IV
ANALYSIS OF SURFACE CODES FOR FAULT-TOLERANCE BUCKET BRIGADES FOR $n = 16$, $n = 26$, $n = 36$,

Distillation parameters							
n	Dist_Factories	Dist_time	Cycles/State	Dist_logical_q /Factory	Dist_logical_q	Dist_physical_q	Dist_Total Cycles
16	60629	0.006356	70	16	970064	594649232.0.0	31780
26	32109505	0.032604	130	240	7706281200	1.541256e+12	163020
36	22215750426	0.078528	160	240	5331780102240	1.668847e+15	392640
Resource counts							
n	Toffoli_Depth	Toffoli_Count	Depth	Cliffords_Count			
16	454	27525182	1700	97124298			
26	1254	80530636902	4820	265885319074			
36	2454	163552354631822	9540	525154241216378			
Totals							
n	Total_time (s)	Total_Physical_q	Total_Cost				
16	0.006356	715179131.0	35.153983				
26	0.032604	1.792915e+12	50.218472				
36	0.078528	2.173935e+15	60.933933				

TABLE V
ANALYSIS PIECEABLE FOR FAULT-TOLERANCE BUCKET BRIGADES FOR $n = 16$, $n = 26$, $n = 36$

Pieceable parameters					
n	physical_Toffoli_count	physical_cx_count	pieceable_time	pieceable_logical_q	pieceable_phys_q
16	578028822	4844432032	0.000953	779515.714286	5456610
26	1691143374942	14173392094752	0.002633	8.256730e+08	5779710810
36	3434599447268262	28785214415200672	0.005153	8.568932e+11	59982526149301
Resource counts					
n	Toffoli_Depth	Toffoli_Count	Depth	Cliffords_Count	
16	454	27525182	1700	97124298	
26	1254	80530636902	4820	265885319074	
36	2454	163552354631822	9540	525154241216378	
Totals					
n	Total_time (s)	Total_Physical_q	Total_Cost		
16	0.000953	43994718	29.844984		
26	0.002633	86310357610	41.30284		
36	0.005153	167626461925442	52.266122		

in our proposal as a result of these results. However, we considered using the Steane code, which has low correction and detection capacity. As a result, better quantum codes with high correction capabilities may be adopted for more robust solutions in the future.

ACKNOWLEDGMENT

This work is supported by JST Moonshot R&D Grant (JPMJMS2061). L.J. acknowledges support from the AFOSR MURI (FA9550-21-1-0209), NSF (CCF-2312755), and the Packard Foundation (2020-71479).

REFERENCES

[1] C. T. Hann, *Practicality of quantum random access memory*. PhD thesis, Yale University, 2021.

[2] G. Schaller and R. Schützhold, "Quantum algorithm for optical-template recognition with noise filtering," *Phys. Rev. A*, vol. 74, p. 012303, Jul 2006.

[3] K. Jebari, "Selection methods for genetic algorithms," *International Journal of Emerging Sciences*, vol. 3, pp. 333–344, 12 2013.

[4] J. Adcock, E. Allen, M. Day, S. Frick, J. Hinchliff, M. Johnson, S. Morley-Short, S. Pallister, A. Price, and S. Stanisic, "Advances in quantum machine learning," 2015.

[5] J. D. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, pp. 195–202, 2016.

[6] P. Wittek, *Quantum machine learning: what quantum computing means to data mining*. Academic Press, 2014.

[7] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. Sawaya, *et al.*, "Quantum chemistry in the age of quantum computing," *Chemical reviews*, vol. 119, no. 19, pp. 10856–10915, 2019.

[8] I. Kerenidis and A. Prakash, "Quantum recommendation systems," *arXiv: Quantum Physics*, 2016.

[9] V. Giovannetti, S. Lloyd, and L. Maccone, "Architectures for a quantum random access memory," *Phys. Rev. A*, vol. 78, p. 052310, Nov 2008.
[10] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum random access memory," *Phys. Rev. Lett.*, vol. 100, p. 160501, Apr 2008.
[11] O. Di Matteo, V. Gheorghiu, and M. Mosca, "Fault-tolerant resource estimation of quantum random-access memories," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–13, 2020.
[12] F.-Y. Hong, Y. Xiang, Z.-Y. Zhu, L.-z. Jiang, and L.-n. Wu, "Robust quantum random access memory," *Phys. Rev. A*, vol. 86, p. 010306, Jul 2012.
[13] A. Paler, O. Oumarou, and R. Basmadjian, "Parallelizing the queries in a bucket-brigade quantum random access memory," *Phys. Rev. A*, vol. 102, p. 032608, Sep 2020.
[14] 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*, vol. 17, p. 123010, dec 2015.
[15] C. T. Hann, G. Lee, S. Girvin, and L. Jiang, "Resilience of quantum random access memory to generic noise," *PRX Quantum*, vol. 2, p. 020311, Apr 2021.
[16] D. Litinski, "Magic state distillation: Not as costly as you think," *Quantum*, vol. 3, p. 205, Dec. 2019.
[17] P. A. M. Casares, "Circuit implementation of bucket brigade qram for quantum state preparation," *arXiv: Quantum Physics*, 2020.
[18] T. J. Yoder, R. Takagi, and I. L. Chuang, "Universal fault-tolerant gates on concatenated stabilizer codes," *Phys. Rev. X*, vol. 6, p. 031039, Sep 2016.
[19] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Reports on Progress in Physics*, vol. 76, no. 7, p. 076001, 2013.
[20] D. Gottesman, "An introduction to quantum error correction and fault-tolerant quantum computation," *arXiv: Quantum Physics*, 2009.
[21] A. M. Steane, "Error correcting codes in quantum theory," *Phys. Rev. Lett.*, vol. 77, pp. 793–797, Jul 1996.
[22] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, "Mixed-state entanglement and quantum error correction," *Phys. Rev. A*, vol. 54, pp. 3824–3851, Nov 1996.
[23] C. T. Hann, G. Lee, S. Girvin, and L. Jiang, "Resilience of quantum random access memory to generic noise," *PRX Quantum*, vol. 2, p. 020311, Apr 2021.
[24] M. Newman and Y. Shi, "Limitations on transversal computation through quantum homomorphic encryption," *Quantum Info. Comput.*, vol. 18, p. 927–948, sep 2018.
[25] P. W. Shor, "Fault-tolerant quantum computation," *Proceedings of 37th Conference on Foundations of Computer Science*, pp. 56–65, 1996.
[26] M. Amy, O. Di Matteo, V. Gheorghiu, M. Mosca, A. Parent, and J. Schanck, "Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3," in *Selected Areas in Cryptography – SAC 2016* (R. Avanzi and H. Heys, eds.), (Cham), pp. 317–337, Springer International Publishing, 2017.
[27] H. Goto, "Minimizing resource overheads for fault-tolerant preparation of encoded states of the steane code," *Scientific reports*, vol. 6, no. 1, p. 19578, 2016.
[28] B. Eastin and E. Knill, "Restrictions on transversal encoded quantum gate sets," *Phys. Rev. Lett.*, vol. 102, p. 110502, Mar 2009.