Systems Architecture for Quantum Random Access Memory

Shifan Xu
Yale University
Yale Quantum Institute
New Haven, CT, USA
shifan.xu@yale.edu

Connor T. Hann
AWS Center for Quantum Computing
and
California Institute of Technology
Pasadena, CA, USA

Ben Foxman
Yale University
Yale Quantum Institute
New Haven, CT, USA

Steven M. Girvin
Yale University
Yale Quantum Institute
New Haven, CT, USA

Yongshan Ding
Yale University
Yale Quantum Institute
New Haven, CT, USA
yongshan.ding@yale.edu

ABSTRACT

Operating on the principles of quantum mechanics, quantum algorithms hold the promise for solving problems that are beyond the reach of the best-available classical algorithms. An integral part of realizing such speedup is the implementation of quantum queries, which read data into forms that quantum computers can process. Quantum random access memory (QRAM) is a promising architecture for realizing quantum queries. However, implementing QRAM in practice poses significant challenges, including query latency, memory capacity and fault-tolerance.

In this paper, we propose the first end-to-end system architecture for QRAM. First, we introduce a novel QRAM that hybridizes two existing implementations and achieves asymptotically superior scaling in space (qubit number) and time (circuit depth). Like in classical virtual memory, our construction enables queries to a virtual address space larger than what is actually available in hardware. Second, we present a compilation framework to synthesize, map, and schedule QRAM circuits on realistic hardware. For the first time, we demonstrate how to embed large-scale QRAM on a 2D Euclidean space, such as a 2D square grid layout, with minimal routing overhead. Third, we show how to leverage the intrinsic biased-noise resilience of the proposed QRAM for implementation on either Noisy Intermediate-Scale Quantum (NISQ) or Fault-Tolerant Quantum Computing (FTQC) hardware. Finally, we validate these results numerically via both classical simulation and quantum hardware experimentation. Our novel Feynman-path-based simulator allows for efficient simulation of noisy QRAM circuits at a larger scale than previously possible. Collectively, our results outline the set of software and hardware controls needed to implement practical QRAM.

CCS CONCEPTS

• **Computer systems organization** → **Quantum computing**; • **Hardware** → **Quantum technologies**.

MICRO '23, October 28–November 01, 2023, Toronto, ON, Canada
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0329-4/23/10.
https://doi.org/10.1145/3613424.3614270

KEYWORDS

Quantum Computing, Quantum Random Access Memory

ACM Reference Format:
Shifan Xu, Connor T. Hann, Ben Foxman, Steven M. Girvin, and Yongshan Ding. 2023. Systems Architecture for Quantum Random Access Memory. In *56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '23), October 28–November 01, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3613424.3614270

1 INTRODUCTION

Quantum computers hold the potential to solve problems that are beyond the reach of conventional digital computers. Such quantum speedup, as understood theoretically, arises from the utilization of quantum mechanical properties such as superposition and entanglement to process information more efficiently and rapidly [45]. Some of the most promising quantum computing applications include quantum searching [26], optimization problems [55], molecular simulation [21, 39], data processing for machine learning [4, 30], and cryptography [52]. For example, the quantum algorithm by Grover [26] for searching an unordered database of size N makes only order of $\sqrt{N}$ queries to the database. This is a $\sqrt{N}$-speedup over the best classical algorithms, which require order of N queries when given access to the same database.

Over the past three decades, technology for building quantum computing hardware has advanced steadily – prototypes of universal quantum processing units (QPU) housing 100+ individually programmable qubits are becoming available for the first time, and there is great interest in practically realizing these quantum applications. The development of scalable quantum computers is still in its early stages. Current Noisy Intermediate-Scale Quantum (NISQ) hardware [50] is limited by its system size (number of qubits) and fidelity (coherent lifetime of qubits and error rates of quantum gates). Remarkable progress [8] has been made in improving the performance of QPUs, through better quantum control, error correction architectures, as well as compiling and noise mitigation software.

One critical, yet largely missing, ingredient for realizing quantum speedup in practice is the implementation of *quantum queries* [4, 45], which allow data to be loaded into quantum states that the QPU can process. While QPU architectures are designed to process data rapidly, they often cannot encode classical data efficiently

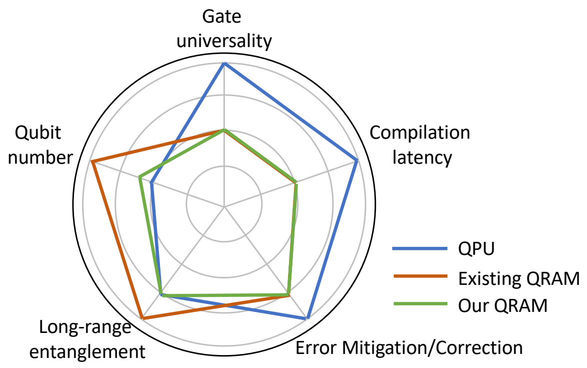


Figure 1: Requirement map by quantum processor unit (QPU) and quantum random access memory (QRAM). There is a need for rethinking the systems architecture for QRAM because QRAM presents a distinct set of architectural constraints. A larger radial coordinate indicates a relatively more stringent requirement – our QRAM design alleviates requirements in multiple dimensions.

or robustly. This inhibits the practical deployment of many quantum algorithms. This issue is known as the *data input and output (I/O) bottleneck* [4, 27]. In terms of gate and qubit overhead, the costs for encoding a large set of data into quantum states can be prohibitively high and dominate the costs of quantum algorithms. Traditional classical random access memory (RAM) [34] allows data stored in memory cells to be loaded rapidly into a central processor unit (CPU). Similarly, a QRAM [23] has been proposed to enable quantum-mechanical loading of data into memory cells. This involves querying the QRAM simultaneously in a large superposition of different addresses.

More precisely, like in classical RAM, a memory cell can be accessed by specifying its address. That is, the data value x_i is stored at address $i \in N$, where N is the memory size.

$$\text{Input: } i \xrightarrow{\text{Classical RAM}} \text{Output: } x_i \tag{1}$$

$$\sum_{i=0}^{N-1} \alpha_i \left|i\right\rangle_{\text{A}} \left|0\right\rangle_{\text{B}} \xrightarrow{\text{Quantum RAM}} \sum_{i=0}^{N-1} \alpha_i \left|i\right\rangle_{\text{A}} \left|x_i\right\rangle_{\text{B}} \tag{2}$$

where α_i is the amplitude of each address in the superposition, and $|\cdot\rangle_{\text{A}}$ ($|\cdot\rangle_{\text{B}}$) is the address (bus) qubit register storing the input (output).

While the design principles of a QRAM are similar to those of a classical RAM, there are unique challenges in scaling up QRAM in practice [22, 27]: *(i) Query Latency.* Like a RAM device, a QRAM allows any data to be accessed in almost the same amount of time regardless of the specified address. This includes a superposition of all addresses at once. Naively, entangling data from one memory cell at a time incurs latency, scaling with N in the worst case. This latency can translate to a slowdown in the application, impeding its practical deployment. *(ii) Memory Capacity.* Quantum algorithms, in principle, offer better quantum speedup when given access to large memory. However, existing QRAM architectures require a rapidly growing number of gates or ancillary qubits when scaling up the memory capacity. *(iii) Fault Tolerance.* Errors in the QRAM (e.g., due to various types of noise in the circuit) could seriously impact its utility, and in many cases eliminate the quantum advantage of an algorithm altogether [51]. As such, it is critical to guarantee the error robustness of QRAM, through either intrinsic noise resilience [28] or error correction.

We propose a general-purpose QRAM architecture to address these challenges, drawing insights from classical RAM, quantum compiling, and quantum error correction. Specifically, we make the following novel contributions:

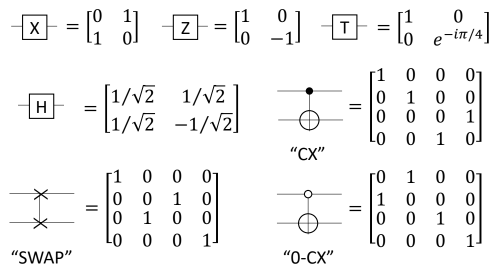
(1) **Goal:** Small QRAM; large virtual memory.
Solution: We propose a new practical architecture that provides a virtual address space that can exceed the capacity of the physical QRAM. By hybridizing two previous query architectures [2, 4], we achieve asymptotic savings in space and time. As a result, our architecture enables queries to large memory that cannot be accomplished by either architecture alone.
(2) **Goal:** Mapping qubits with minimal routing overhead.
Solution: QRAM requires strongly entangling m address qubits with data from $O(2^m)$ memory cells.
Whether we can efficiently embed QRAM on practical (sparsely connected) two-dimensional hardware is highly non-trivial. For the first time, we provide a positive answer to this question. We present a constructive mapping of QRAM on 2D lattice architectures with minimal routing/communication overhead.
(3) **Goal:** Noise-robustness of QRAM.
Solution: We show that our small-scale QRAM can be implemented on current hardware or near-term hardware with moderately improved error rates We also demonstrate that small error correction codes allow us to substantially scale up QRAM with low overhead.

The rest of the article is organized as follows. In Sec. 2, we review the background on quantum compiling and architecture designs for QRAM. Sec. 3 introduces a new QRAM architecture designed explicitly for hybrid QRAM. In Sec. 4, we present the algorithm for mapping QRAM on realistic hardware. In Sec. 5, we analyze the biased-noise resilience property of our circuit and leverage it to reduce error correction overhead. Finally, we validate the results via classical simulation and quantum hardware experiments in Sec. 6 and Sec. 7. In Sec. 7, we also compare the resource usage of different QRAMs and show an asymptotic scaling advantage.

2 BACKGROUND

2.1 Principles of Quantum Computing

In quantum computing, a quantum bit (qubit for short) is the fundamental computing unit. Unlike its classical counterpart, a qubit can be in a superposition state, that is a linear combination of 0 and 1. In the Dirac notation, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α, β are complex coefficients satisfying $|\alpha|^2 + |\beta|^2 = 1$, and $|0\rangle = [1 \;\; 0]^T$ and $|1\rangle = [0 \;\; 1]^T$ are the computational basis vectors. In quantum algorithms, quantum logic gates are used to manipulate the state of the qubits. Some common quantum logic gates are shown below:



2.2 Quantum Compiling

A quantum compiler transforms a high-level quantum program or mathematical algorithm into a sequence of native instructions that the hardware backend recognizes [11, 13]. The transformed quantum circuit must be logically equivalent, resource-efficient, and robust to hardware noise. Due to strict architectural constraints, compiler optimization will have to break traditional abstractions across the software stack and be adapted to algorithmic and device characteristics. We now highlight several important transformation passes in a compiler software.

2.2.1 Gate Synthesis. One of the first steps in quantum compiling is to decompose high-level unitary into a sequence of native gates from an instruction set [29]. A common instruction set is the Clifford+T gates, such as {H, S, CX, T}. When implementing quantum queries using these gates in practice, they can introduce latency into quantum algorithms if the circuit depth is too high.

As such, it is preferable to implement QRAM with a tailored gate set natively, such as the classical reversible gates, including X, CX (controlled-X), Toffoli (double-controlled-X), MCX (multi-controlled-X), and CSWAP (controlled-SWAP) gates. Otherwise, the multi-qubit gates can be decomposed into Clifford+T gates. For example, we can decompose a CSWAP gate to a circuit of depth 12, T depth 3, with no ancillae required [1, 18].

2.2.2 Qubit Mapping and Routing. To execute a quantum circuit, another critical step is to map all logical qubits onto the physical hardware. Current NISQ hardware has a limited number of qubits and restrictive qubit connectivity. Only adjacent qubits can interact with each other, while interactions between distant qubits are resolved via routing qubits closer to each other. Common routing strategies include physically moving qubits (e.g., for trapped ions) [57] or logically swapping qubits (e.g., for superconducting circuits) [44]. Different routing strategies would incur different routing overhead, in terms of the number of additional operations. Future Fault-Tolerant Quantum Computing (FTQC) hardware will have similar constraints. For example, in surface codes, logical qubits can be laid out in a 2D grid topology. Logical gates and qubit routing can be accomplished via lattice surgery [33].

Although the general qubit mapping problem has been proven to be NP-hard [41], it is often useful to leverage information about the circuit and the hardware to improve the quality of a mapping strategy—a well-structured circuit can be easier to map and route [32, 43, 59]. Moreover, hardware noise-aware mapping strategies can enhance the circuit performance significantly [3, 14, 37, 44, 54].

2.3 Quantum Query Architectures

QRAM is an integral part of quantum computer architecture, as it enables quantum computers to efficiently encode classical data into a quantum state for the QPU to process. As shown in Equation 2, a QRAM read operation therefore involves properly entangling data x_i with the corresponding address i in the superposition input.

General-Purpose versus Domain-Specific – Query architectures can be categorized into two classes: (1) General-purpose (GP) architectures that load any possible data values x_i for an arbitrary address $\sum_i \alpha_i |i\rangle$. (2) Domain-specific (DS) architectures that implement only particular data function(s) $f(i) = x_i$. DS architectures are useful when running applications in a target domain, as they are highly tailored to maximize efficiency or fault tolerance.

We now describe the leading designs and implementations of QRAM, namely gate-based and router-based architectures.

2.3.1 Gate-Based Architecture. Conventional wisdom is to use a partition of a universal QPU to implement the functionality of QRAM. As such, many proposals involve synthesizing the QRAM operation using a sequence of quantum logic gates and optionally using ancillary qubits. We provide two examples of these gate-based architectures below: Sequential Query Circuits and Reversible Logic Circuits.

Sequential Query Circuit (SQC) – A quantum query can be implemented by a quantum circuit consisting of sequential MCX (multi-controlled-X) gates with no ancillary qubits. In the literature, the SQC is also known as a basic query circuit (BQC) or quantum read-only memory (QROM) [2]. Figure 2c provides a simple example of such a circuit. In this circuit, a sequence of K MCX gates is applied to query a memory of size N, where each gate has $\log N$ controls on all the address qubits and one target on the so-called *bus qubit* that holds the queried data. Each MCX gate is responsible for loading data stored at one corresponding address, and the full query is realized by iterating sequentially over all possible addresses. The SQC is a general-purpose architecture, capable of querying any function $f(i) = x_i$ for the memory cell data. It uses $O(\log N)$ qubits and has $O(N)$ query latency.

Reversible Logic Circuit (RLC) – An alternative implementation of quantum queries directly on a QPU using quantum gates is through an RLC. When the function that computes the data value is known, that is $x_i = f(i)$, one can synthesize this function directly with classical reversible gates (such as X, CX and Toffoli) and ancillary qubits. Because different circuits are required to implement different functions, RLC implements a domain-specific query. Classical reversible circuits are shown to be easier to optimize and verify than generic quantum circuits [15, 49]. However, as with any DS architecture, the circuits must be synthesized and optimized for each domain application [19]. This is useful if we want to implement a quantum computer to support a particular application. For example, the modular exponentiation step in Shor's algorithm [52] can be implemented by either an RLC or a hand-optimized quantum circuit [20].

2.3.2 Router-Based Architectures. To minimize query latency, several router-based architectures are proposed. The defining feature of these general-purpose architectures is some form of *quantum routing*, wherein quantum data is routed to multiple different locations

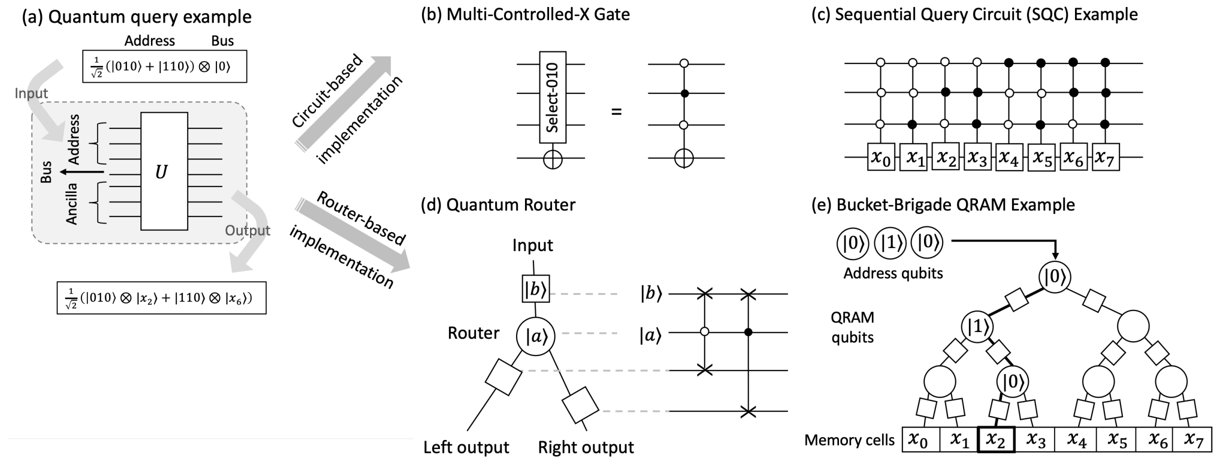

(a) Quantum query example
Address
Bus
$\frac{1}{\sqrt{2}}(|010\rangle + |110\rangle) \otimes |0\rangle$
Input
Bus
Address
Ancilla
U
Output
$\frac{1}{\sqrt{2}}(|010\rangle \otimes |x_2\rangle + |110\rangle \otimes |x_6\rangle)$
Circuit-based implementation
Router-based implementation
(b) Multi-Controlled-X Gate
Select-010
=
(c) Sequential Query Circuit (SQC) Example
x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7
(d) Quantum Router
Input
$|b\rangle$
Router
$|a\rangle$
Left output
Right output
(e) Bucket-Brigade QRAM Example
$|0\rangle$ $|1\rangle$ $|0\rangle$
Address qubits
QRAM qubits
$|0\rangle$
$|1\rangle$
$|0\rangle$
Memory cells
x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7

Figure 2: Different quantum query architectures and corresponding elementary units. (a) Quantum query example with address width 3. (b) Multiple-Controlled-X gate (MCX) as a unit for circuit-based query architecture. (c) Quantum router as a unit for router-based query architecture. (d) Sequential query circuit consisting of sequential MCX gates. (e) Parallel Bucket-Brigade QRAM consisting of recursive quantum routers.

in coherent superposition. The gadget in Figure 2d is a prototypical example; the gadget routes quantum data from an input to one of two different outputs conditioned on the state of a control quantum bit (router qubit). By leveraging such gadgets, router-based QRAMs can trade off space for time to implement $O(\log N)$-latency queries for a memory of capacity N, albeit at the cost of $O(N)$ qubits. This exponential latency reduction relative to SQC is a significant step towards the practical implementation of QRAM. We provide two examples of these router-based architectures below: Fanout and Bucket-Brigade.

Fanout QRAM[45] – Fanout QRAM is the first architecture to achieve an $O(\log N)$-latency query. It arranges quantum routers in a binary tree, recursively using the outputs of the parent router as the inputs of the children routers. A query is implemented in two stages, address loading and data retrieval. During address loading, all routers are first initialized in $|0\rangle$, then a series of CX gates entangle the address and routers. In particular, all 2^k routers at level k of the tree are flipped from $|0\rangle$ to $|1\rangle$ conditioned on the k^{th} address, resulting in the preparation of Greenberger–Horne–Zeilinger-like (GHZ) states [25] across each level of the tree. During data retrieval, a bus qubit is routed down from the root of the tree, following the path indicated by the routers' states. Once it reaches the bottom of the tree, classically-controlled gates copy the data $x_i = f(i)$ into the state of the bus. Subsequently, the bus qubit is back routed out of the tree, and the routers are returned to the all-$|0\rangle$ state by uncomputation. The fanout QRAM's $O(\log N)$ latency results from the fact that both the address loading and data retrieval can be implemented with only $O(\log N)$-depth circuits. However, Fanout QRAM is shown to suffer from decoherence problems due to the high entanglement of GHZ states [28].

Bucket-Brigade QRAM[22, 23] – Bucket-Brigade QRAM improves Fanout QRAM by modifying the address loading stage to reduce the entanglement among the routers, as shown in Figure 2e. Instead of using CX gates to entangle the address and routers, in Bucket-Brigade QRAM the address qubits are themselves routed into the tree, with the states of earlier address qubits controlling the routing of later ones. The resultant state of the routers is more akin to a *W state* [9] than a GHZ state. The former has less entanglement entropy, which has been shown to greatly reduce the sensitivity of Bucket-Bridage QRAM to noise and errors [28]. Importantly, $O(\log N)$ query latency is still achievable with this improvement, making it a competitive query architecture for the NISQ era.

2.3.3 *Other Architectures.*

Other constructions of quantum queries have been proposed. For example, **Select-Swap QRAM** [40] can be viewed as a combination of the gate-based and router-based architectures. Select-Swap QRAM employs a two-stage approach. During the first stage, one sequentially iterates over all possible states of a *subset* of the address qubits, loading corresponding blocks of data for each. During the second stage, the remaining address qubits are used to route this data through a network of CSWAP gates, routing the queried data to a definite location. The sequential iteration in the first stage is analogous to the gate-based SQC, while the coherent routing in the second stage is reminiscent of the router-based architectures.

As another example, arbitrary state preparation algorithms can sometimes be used as a subroutine in QRAM. Some existing work includes the general unitary synthesis method [53, 60] and parameterized circuit method [24, 48]. The general unitary synthesis is more complex than router-based QRAM, with specific gates for the different classical data sets. This increases the difficulty of changing the classical dataset. The parameterized circuit is a popular NISQ-era application that has been proposed to have the ability to realize approximate quantum query with $O(1)$ depth and $O(\log N)$ number of qubits with the price of long training time and approximate queries.

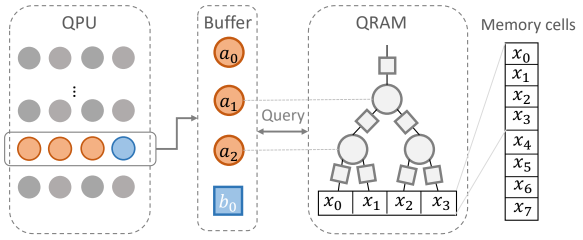


Figure 3: Overview of the proposed virtual QRAM architecture ($k = 1, m = 2$) and its interaction with QPU. The QPU qubits are swapped to the buffer for a quantum query and returned to QPU once the query is done by QRAM.

3 PROPOSED QRAM ARCHITECTURE

This work demonstrates an end-to-end architecture design for QRAM. The prior works, including Bucket-Brigade QRAM and FANOUT, have unaffordable overhead in terms of resource consumption when querying a large database. Our proposed system architecture is built upon a novel router-based construct that allows us to query a virtual address space larger than that is physically available, as shown in Section 3.1. We also provide a series of optimizations in Section 3.2 to significantly cut down both space and time costs. Our new design decouples the address-loading stage from the data-retrieving stage in a quantum query, and deeply optimizes each stage. As a result, we can achieve an asymptotic advantage over prior work (which is shown in Section 7).

3.1 Two-Stage Query Overview

We introduce a novel router-based QRAM using $O(M)$ qubits, where $M = 2^m$. We define the *address width m* as the number of bits to specify an address and the capacity M the size of the address space. For illustration purposes, we include an example architecture shown in Figure 3, and a step-by-step query procedure outlined in Figure 4, with the detailed circuits shown in Figure 5. Like in the router-based QRAM architectures, our construction consists of two stages: address loading and data retrieving. We explain these two stages in turn below.

3.1.1 Stage 1: Address Loading. The task in the address loading stage is to route the m address qubits into the small QRAM. For this purpose, we follow the approach of other router-based QRAM architectures and arrange a collection of quantum routers in a binary tree structure. We affix a layer of *M data qubits* to the outputs of routers at the lowest level of the tree, with each data qubit corresponding to one of the M classical memory cells. These data qubits will be used to facilitate operations during the data retrieval stage.

The address-loading stage follows the conventional Bucket-Brigade procedure: the address qubits are sequentially routed into the tree, with the states of earlier address qubits controlling the routing of later ones. The only difference is that, in our scheme, the data qubits are subsequently prepared in a special state conditioned on the states of the routers (Figure 4a). Specifically, when address i is queried, the i^{th} data qubit is flipped from $|0\rangle$ to $|1\rangle$. This flipping is implemented via a collection of `CX` gates, with the bottom layer of routers as controls and the data qubits as targets, see Fig 5a as query state preparation.

3.1.2 Stage 2: Data Retrieval. We propose a *novel* data-retrieval stage, which, at a high level, performs data compression from the bottom data qubits to the root data qubit in the QRAM. First, M classically-controlled gates act on the bottom of the tree to write $|x_i\rangle$ on the data qubits when address $|i\rangle$ is queried. Specifically, each data qubit is paired with an ancillary qubit initialized in $|0\rangle$. We refer to the two-qubit system of a data qubit and its ancilla as a data node. Then, conditioned on x_i, a `SWAP` gate is applied between the two physical qubits in a data node. This has the effect of encoding the classical data in a dual-rail encoding, i.e. $x_i = 0$ (resp. $x_i = 1$) is encoded as $|10\rangle$ ($|01\rangle$). Fig. 4(b) shows the resultant state.

Then, an array of `CX` gates (Figu 5b) is used to propagate this data up to the root node of the tree, as shown schematically in Fig. 4c. Next, the data at the root node is copied to the bus qubit, conditioned on the k remaining address qubits, using a `MCX` gate. After that, the `CX` array is applied again to uncompute and disentangle QRAM qubits, returning the QRAM to the state for the next data retrieval stage (Fig. 4d). We can then repeat the data retrieval stage for the next segment/page of the memory, after swapping the new segment into the bottom of the tree. Crucially, in this new data-retrieval step, the only non-Clifford gate involved is the `MCX` gate.

3.1.3 Putting It Together: Virtual QRAM. The goal is to implement quantum query access to a memory of capacity $N = 2^n$, where $N > M$. That is, we consider a practical scenario where $O(2^n)$ qubits are not physically available. Can we still implement the query, and if so, at what cost? This scenario resembles classical memory architecture design where "virtual memory" allows a small physical RAM can access a large address space by swapping segments (also known as pages) of memory from disk storage [36]. Our "virtual QRAM" design follows precisely this intuition: we implement a virtual QRAM that allows a small QRAM with $O(M)$ qubits to access a large address space $O(N)$ by swapping segments of classical memory, as shown in the right panel of Fig. 3. To accomplish this, we need to design a QRAM architecture that allows us to query segments of memory coherently.

We first partition the full size-N classical memory into $K = 2^k$ continuous segments (pages), each of which has $M = 2^m$ memory cells. It is equivalent to viewing the original n-bit address as two parts: the most significant k-bits (which we call SQC width) and least-significant m-bits (which we call QRAM width), where $k + m = n$. Our design hybridizes gate-based SQC and router-based QRAM, where a quantum query consists of 6 basic steps: (a) loading m address bits into QRAM, (b) preparing leaf data qubits for data retrieval, (c) retrieving data to root qubit, (d) uncomputing data retrieval, (e) repeat (b-d) for each segment of classical memory, (f) uncomputing address loading. More details are illustrated in Figure 4. A distinct feature of our design is the "load-once" property. Our method only loads the m address qubits into QRAM once at the beginning (and at the end for uncomputation) as shown in Figure 5c, whereas in a previous design from [28] the m address qubits need to be loaded 2^k times. This is one of the major sources of gate savings in our method. We present a pseudocode algorithm to describe

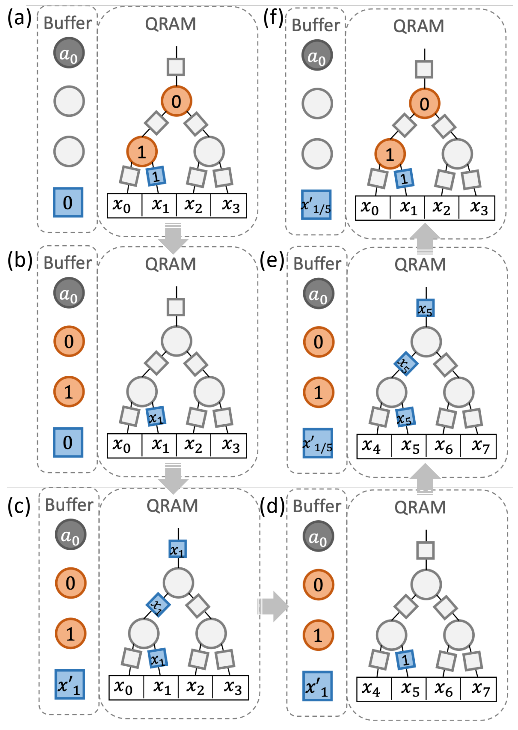


Figure 4: The step-by-step procedure of a quantum query to virtual QRAM ($k = 1, m = 2$). (a) Load the m address qubits (orange) in QRAM and prepare the specific data qubit (blue) for data retrieval. (b) Coherently write classical data x_i in data qubits to state $|x_i\rangle$. (c) Retrieve data by copying to root qubit via CX gates. Data is copied to the bus qubit conditioned on the state of k address qubits (dark gray). (d) Uncompute data retrieval and swap in next segment memory. (e) Repeat data retrieval in (c). (f) Uncompute data retrieval.

the entire query procedure in Algorithm 1. Note that, for illustration purposes, this algorithm does not include the optimization techniques introduced in Section 3.2.

3.2 Implementation and Optimizations

3.2.1 Key Optimization 1: Address Qubit Recycling. As shown in Figure 4, the internal router qubits (orange nodes) are not being used during steps (b)-(e). By recycling/reusing these qubits in replacement of the data qubits, we do not need any data qubits (blue nodes) for the quantum routers internal to the tree. Figure 5 illustrates that the data retrieval stage reuses router qubits for copying data via the CX array. As such, in diagrams such as Figure 6c and Figure 7, we do not draw blue data qubits for internal quantum routers.

3.2.2 Key Optimization 2: Lazy Data Swapping. As shown in the data-retrieval stage in Figure 5b, classical data is loaded and unloaded sequentially for each segment of classical memory. We observed that if the subsequent classical data corresponding to the memory address on the next page is equal to the previous one, unloading and reloading data qubits are redundant and unnecessary.

Algorithm 1 Quantum Query with Virtual QRAM.

```
Require: $|\psi_A\rangle = \sum \alpha_i |i\rangle, n \geq 1, 0 \leq k \leq n-1, m = n-k$
Ensure: $|\psi_{AB}\rangle = \sum_{i=0}^{N-1} \alpha_i |i\rangle_A |x_i\rangle_B$
  $q_t^{(a)}, q^{(b)} \leftarrow t_{th}$ address qubit, Bus qubit
  $q_t^{(c)}, q_t^{(d)} \leftarrow t_{th}$ layer router qubit, data qubit
  $q_{-1}^{(d)} \leftarrow$ input qubit to the root router
  SWAP$[q_0^{(a)}, q_{-1}^{(d)}]$, SWAP$[q_{-1}^{(d)}, q_0^{(c)}]$ {Address Loading}
  for $u \leftarrow 1 \dots m-1$ do
    SWAP$[q_u^{(a)}, q_{-1}^{(d)}]$
    for $v \leftarrow 0 \dots u-1$ do
      CSWAP$[q_v^{(c)}, q_{v-1}^{(d)}, q_v^{(d)}]$
    end for
    SWAP$[q_{u-1}^{(d)}, q_u^{(c)}]$
  end for
  CX$[q_{m-1}^{(c)}, q_{m-1}^{(d)}]$
  for $p \leftarrow 0 \dots 2^k - 1$ do
    Classical-CX$[x_i, q_{m-1}^{(d)}]$ {Data Retrieval}
    for $u \leftarrow 0, m-1$ do
      CX$[q_{m-1-u}^{(d)}, q_{m-2-u}^{(d)}]$
    end for
    MCX$[q_{m,\dots,n-1}, q_{-1}^{(d)}, q^{(b)}]$
    for $u \leftarrow 0 \dots m-1$ do
      CX$[q_{m-1-u}^{(d)}, q_{m-2-u}^{(d)}]$ {Uncompute: Data Retrieval}
      Classical-CX$[x_i, q_{m-1}^{(d)}]$
    end for
  end for
  CX$[q_{m-1}^{(c)}, q_{m-1}^{(d)}]$ {Uncompute: Address Loading}
  for $u \leftarrow 1 \dots m-1$ do
    SWAP$[q_u^{(a)}, q_{-1}^{(d)}]$
    for $v \leftarrow 0 \dots u-1$ do
      CSWAP$[q_v^{(c)}, q_{v-1}^{(d)}, q_v^{(d)}]$
    end for
    SWAP$[q_{u-1}^{(d)}, q_u^{(c)}]$
  end for
  SWAP$[q_{-1}^{(d)}, q_0^{(c)}]$, SWAP$[q_0^{(a)}, q_{-1}^{(d)}]$
```

Instead, by computing $x'_i = x_i \bigoplus x_{i+2^m}$, it is necessary to load the next classical data x_i only when $x'_i = 1$. At the final data-retrieving stage, an alternative classical data unloading is accepted, with a classical value as $c_i = \bigoplus_{i \in \{0,1,\dots,k\}} x_i$. Adopting this technique, named lazy data swapping, provides SWAP gate savings of $O(2^{n-1})$ in average cases, since the subsequent classical data can be the same as the original data with a probability of $p = 0.5$, assuming a uniform distribution for the classical data x_i.

3.2.3 Key Optimization 3: Address Pipelining. With pipelining, we can reduce the depth of address-loading from $O(m^2)$ to $O(m)$. In the naive approach to address loading, address qubits are routed into the tree sequentially, with the $(\ell+1)^{\text{th}}$ address qubit *waiting* to be routed until the ℓ^{th} address qubit has reached its destination at level ℓ of the tree. The total routing time is thus $\sim \sum_{\ell=1}^{m} \ell = O(m^2)$. Instead, the addresses can be routed in a pipelined manner: the $(\ell+1)^{\text{th}}$ address qubit is routed into the tree immediately after the ℓ^{th} qubit has been routed one layer down, i.e. without waiting. Removing the

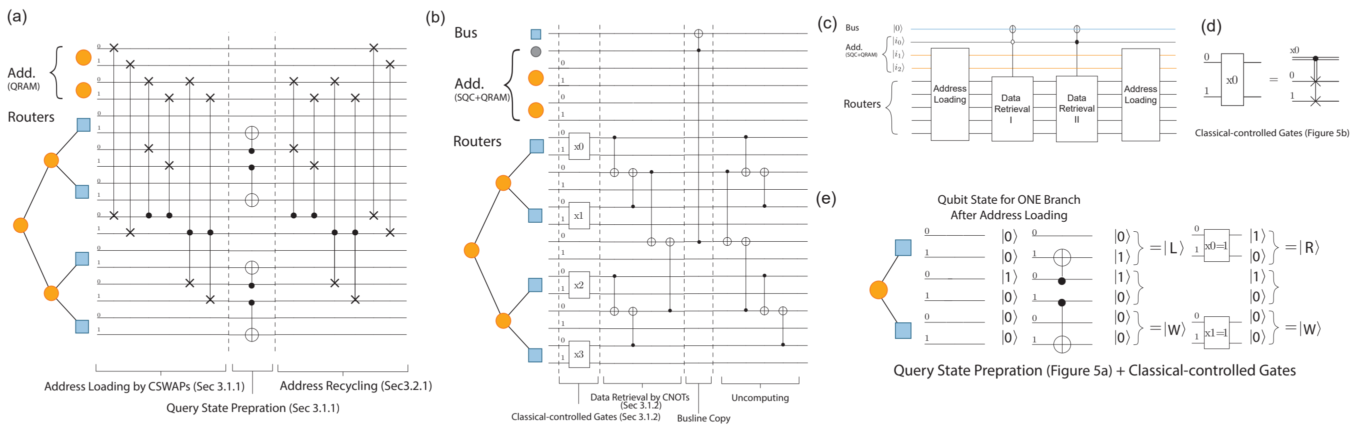

(a)
Add. (QRAM)
Routers
Address Loading by CSWAPs (Sec 3.1.1)
Address Recycling (Sec3.2.1)
Query State Prepration (Sec 3.1.1)
(b)
Bus
Add. (SQC+QRAM)
Routers
Data Retrieval by CNOTs (Sec 3.1.2)
Classical-controlled Gates (Sec 3.1.2)
Busline Copy
Uncomputing
(c)
Address Loading
Data Retrieval I
Data Retrieval II
Address Loading
(d)
Classical-controlled Gates (Figure 5b)
(e)
Qubit State for ONE Branch After Address Loading
Query State Prepration (Figure 5a) + Classical-controlled Gates

Figure 5: Circuit for the architecture design of virtual QRAM, with SQC width $k = 1$ and QRAM width $m = 2$. (a) Circuit for Address Loading Stage. (b) Circuit for Data Retrieval Stage. (c) Outline of a full virtual QRAM circuit. (d) Classical-controlled SWAP gates with dual-rail encoding (e) Qubit state in query state preparation and data retrieval.

waiting reduces the depth to $O(m)$. While the resulting circuit is equivalent to the parallel schedules introduced in [28] and [12], we identify the origin of such parallelism as coming from pipelining.

4 MAPPING QRAM IN 2 DIMENSIONS

4.1 Mapping and Routing: Challenges

Qubit mapping and routing are important steps in a quantum compiler to implement a quantum algorithm on hardware. Routing overhead refers to the number of operations needed to execute a gate operation on two (possibly physically distant) qubits. This overhead can cause a significant increase in the query latency of the quantum algorithm.

Mapping QRAM (of capacity M) is particularly challenging because it is required to map and entangle $O(M)$ qubits. The tree-like structure in router-based QRAM means that the i^{th} layer involves $O(2^i)$ qubits and multi-qubit gates. Naively, keeping a router qubit in layer i equal-distance with its parent router in layer $i-1$ and its children routers in layer $i+1$ is only possible in hyperbolic geometry. To embed a tree in 2D Euclidean space, the root ($i = 0$) of the tree can be far apart from the next layer down ($i = 1$) due to the large size of the two subtrees.

Our research shows that QRAM can be embedded in a 2D nearest-neighbor grid without incurring asymptotic routing overhead. In other words, we can map the QRAM circuit on a 2D grid and route the qubits without increasing the $O(\log M)$ depth of the original circuit. This is achieved by combining a mapping strategy via topological minor graph embedding (Sec 4.2) and a routing method based on teleportation (Sec 4.3).

4.2 Mapping QRAM via H-Tree Recursion

To map QRAM onto a 2D architecture, we need to find an embedding of a binary tree in the connectivity graph of the hardware. In addition, we require the embedding to be a *topological minor graph* embedding. This allows us to implement the teleportation-based routing method by ensuring all routing qubits do not carry any logical information. Given a simple, undirected graph G, another graph H is a topological minor of G if H can be obtained from a subgraph of G by deleting edges, vertices, and contracting edges.

We reduce the problem of mapping QRAM to embedding a complete binary tree in a 2D grid. The problem of embedding complete binary trees into grids has been extensively studied in classical VLSI design [16, 31, 38, 46]. The H-tree recursion is the first efficient mapping strategy introduced by [7]. In Figure 6a, we present the optimal embedding of T_2 into Grid$(3, 3)$ by H-tree recursion, which is also the base case for the recursion. The embedding involves three QRAM router qubits, one unused qubit, one routing qubit, and four data qubits. Note the distinction between a router qubit (in QRAM) and a routing qubit (for teleportation). This design ensures the root QRAM qubit can route to the border of the grid. Recursively, we can construct an embedding of T_{m+2} into Grid$(2n + 1, 2n + 1)$ as shown in Figure 6. As for even-addressed width QRAMs, we can cut half the grid and make it a rectangular one with $M(2n + 1, n)$ to embed T_{m+1} QRAM into the 2D grid.

4.3 Routing via Teleportation

Our teleportation routing method is based on a technique called entanglement swapping [47], commonly used in the context of quantum repeater networks [6, 56]. If the intermediate qubits between two logical qubits are unused, they can be used as ancillae (routing qubits) to perform teleportation, as shown in Figure 6d and 6e. Local Einstein-Podolsky-Rosen (EPR) pair preparation and Bell State Measurement (BSM) are performed in parallel. As such, we can teleport a qubit over a long routing distance with a constant depth circuit. Each QRAM operation (e.g., remote CSWAP) can thus be implemented in $O(1)$ step, regardless of the routing distance.

As a result, our embedding from Sec 4.2 is optimal in terms of routing latency. Since teleportation only introduces an $O(1)$ depth to each gate, the overall QRAM circuit depth remains in $O(\log M)$. Though H-tree is efficient enough to provide the optimal query latency, there are further optimizations are provided by [16, 38, 46], where improved versions of the H-tree recursion are found. As such, we can embed QRAM into a (constant-factor) denser grid.

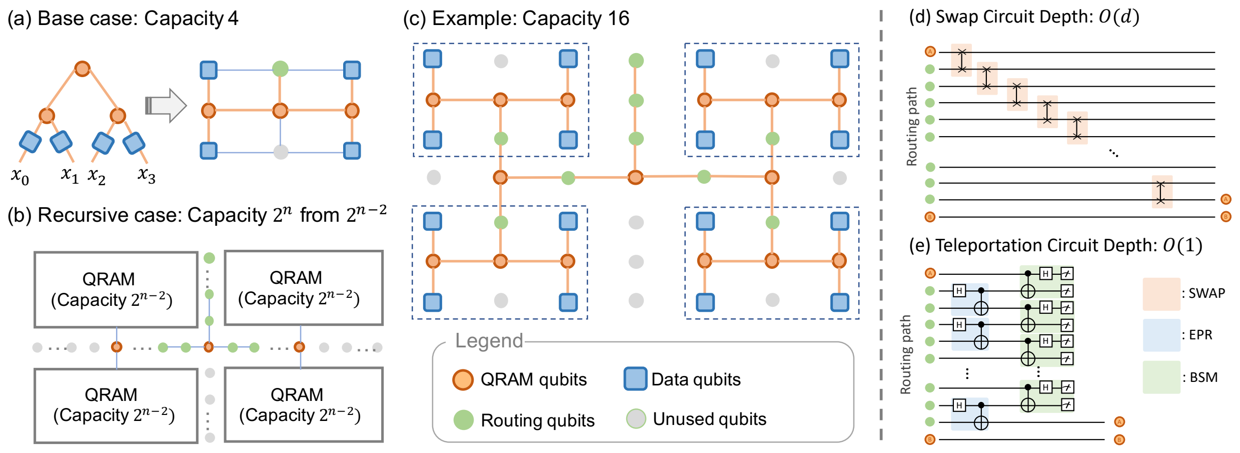

(a) Base case: Capacity 4
x_0
x_1 x_2
x_3
(b) Recursive case: Capacity 2^n from 2^{n-2}
QRAM (Capacity 2^{n-2})
QRAM (Capacity 2^{n-2})
QRAM (Capacity 2^{n-2})
QRAM (Capacity 2^{n-2})
(c) Example: Capacity 16
Legend
QRAM qubits
Data qubits
Routing qubits
Unused qubits
(d) Swap Circuit Depth: $O(d)$
Routing path
(e) Teleportation Circuit Depth: $O(1)$
Routing path
: SWAP
: EPR
: BSM

Figure 6: (a) H-Tree embedding of a capacity-4 QRAM on a 2D grid. (b) Recursive H-Tree embedding for a capacity-n QRAM (Sec. 4.2). (c) Example embedding of a capacity-16 QRAM. (d) Routing via swapping. (e) Routing via teleportation (Sec. 4.3).

5 NOISE-ROBUST IMPLEMENTATIONS

QRAM is highly susceptible to noise and errors, which can cause information loss and reduce the fidelity of the stored states. Thus, QEM or QEC is essential for improving the reliability and accuracy of QRAM operations. This is critical for the success of many quantum algorithms and the development of fault-tolerant quantum computation.

Prior work by [28] revealed intrinsic noise resilience in Bucket-Brigade QRAM by carefully analyzing the error propagation in QRAM circuits. We observed that virtual QRAM shares a similar biased-intrinsic noise resilience property, meaning that Z error in virtual QRAM is constrained to local qubits and will not propagate to the entire circuit even without any active quantum error correction or mitigation. To quantify the noise-resilience of virtual QRAM, we define *query fidelity* for a single query $|\psi_{in}\rangle$ as $F = |\langle\psi_{out}|\psi'_{out}\rangle|^2$, where ψ_{out} is the true output, and ψ'_{out} is the expected output.

With respect to this definition of query fidelity, we will show that in virtual QRAM, the query fidelity is lower bounded for an arbitrary $|\psi_{in}\rangle$. The infidelity is polynomially in terms of the address width m, rather than the overall tree size 2^m.

5.1 Biased-Noise Resilience Analysis

To construct our Z-biased noise model, we assume that each qubit is subjected to the following phase-flip noise quantum channel, $\rho \to \rho' = (1-\epsilon)\rho + \epsilon Z\rho Z$. Equivalently, a Z error is applied to each qubit with probability ϵ. We show that in the presence of this qubit-based error channel, the QRAM part (beside SQC) in virtual QRAM has a lower bound in the query fidelity as

$$F \geq 1 - 4\epsilon \log^2 M = 1 - 4\epsilon m^2 \tag{3}$$

where $m = \log(M)$ is the address width of QRAM. We first present an outline of our methodology. Similar to the approach used in [28], the locality behavior of the noise prevents the error from propagating throughout the entire QRAM, protecting the overall query fidelity. An example of this effect is illustrated in Fig.7 — a Z error in the control qubit of a subsequent CX gate never propagates to the target qubit by commutator relationship of quantum gates. This property also holds in quantum routers using CSWAP gates, ensuring the error locality for the address loading stage.

However, our virtual QRAM cannot prevent other Pauli errors, such as X and Y errors, from propagation. We will show that the fidelity under Z and X error channels has an exponential difference with respect to the QRAM width m.

Consider the computational basis states $|0\rangle, \dots, |2^{m+k}-1\rangle$ corresponding to different memory addresses. For a query Q, let $Q_{good} \subseteq \{0, \dots, 2^{m+k}-1\}$ be the subset of i such that Q behaves ideally on $|i\rangle$: there are no Z errors on any routers on the path of the branch corresponding to i. On the initial state $|\psi_{in}\rangle = \sum_{i=0}^{2^{m+k}-1} \alpha_i |i\rangle_A |0\rangle_B$, a query yields the state $|\psi_{out}\rangle = \left(\sum_{i\in Q_{good}}^{2^{m+k}-1} \alpha_i |i\rangle_A |x_i\rangle_B\right) + |\psi_{bad}\rangle$ where $|\psi_{bad}\rangle$ is a super-position of all non-ideal branches (those with Z errors).

$$\begin{aligned} F &= \left|\left\langle \psi_{out} \middle| \sum_{i\in Q_{good}}^{2^{m+k}-1} \alpha_i |i\rangle_A |x_i\rangle_B \right\rangle + \left\langle \psi_{out} \middle| \psi_{bad} \right\rangle\right|^2 \\ &\geq \left(\sum_{i\in Q_{good}} |\alpha_i|^2 - \left(1 - \sum_{i\in Q_{good}} |\alpha_i|^2\right)\right)^2 \\ &= \left(2\left(\sum_{i\in Q_{good}} |\alpha_i|^2\right) - 1\right)^2 \end{aligned}$$

where the inequality holds since $|\psi_{out}\rangle$ is normalized. Thus, it suffices to show that Q_{good} is sufficiently large in expectation, i.e. most branches perform ideally. If S and S' are sets of indices of the same size, they share equal possibility to coincide with Q_{good}, as our error model on the router is independent of the router itself.

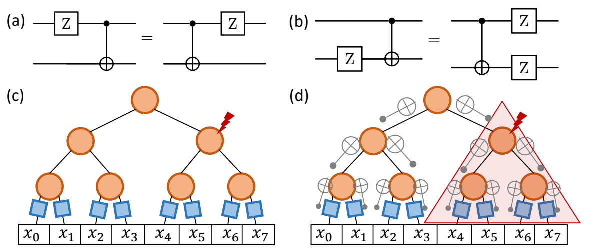


Figure 7: (a) and (b): Commutator relationship for Z and CNOT gates. (c) and (d): Z-error behavior in QRAM. A Z error only propagates to the subtree highlighted in red, due to the direction of the CX gates.

Letting $|Q_{good}| = c$,

$$\mathbb{E}\left[\sum_{i\in Q_{good}} |\alpha_i|^2\right] = \frac{\mathbb{E}[c]}{2^m} \implies \mathbb{E}[F] \geq \left(\frac{2\cdot\mathbb{E}[c]}{2^m} - 1\right)^2 \quad (4)$$

where the expectation is taken over all possible errors. By the fact that the Z error will not propagate up across the tree to destroy other branches, the fulfillment condition for an ideal branch is that all the routers in the path are correct. Since each branch contains m routers, the probability a branch behaves ideally is $(1-\epsilon)^{m^2}$, with $\mathbb{E}[c] = 2^m(1-\epsilon)^{m^2}$. Combining (3) and (4), we conclude that

$$1 - F \leq 1 - \mathbb{E}[F] \leq 4\epsilon m^2 = 4\epsilon \log^2 M$$

where the final inequality is valid for $m \geq 1$, $\epsilon \geq 0$. This proves the fidelity bound in Equation (3).

Precisely, if dual-rail encoding introduced in Section 3 is adopted, the router qubits and data qubits are duplicated, which doubles the errors of each router in the circuit. The above derivation, however, is still valid because the locality of the Z error behavior is not relevant to the choice of encoding. Using the same methodology, we arrive at a bound with only a constant factor difference for dual-rail-encoding virtual QRAM: $F_{\text{dual-rail}} \geq 1 - 8\epsilon m^2$.

On the contrary, the circuit has no noise-resilience property for X errors. Any single X error will propagate to the root qubit of the QRAM, leading to a complete destruction of the query state. Thus to achieve an ideal state, it suffices to show that all the qubits in QRAM are correct. As such, for X error channel with error prob ϵ, the lower bound of infidelity is $1-8\epsilon m\cdot 2^m$, exponential in the total number of qubits. Similar to the X error behavior in the QRAM part, any single Pauli error in SQC is fatal for the query fidelity. Consequently, for an SQC width k, the query fidelity under arbitrary Pauli errors with error rate ϵ is lower bounded by $1-(\epsilon k\cdot 2^k)$. Combining these bounds, We conclude the virtual QRAM with QRAM address width m and SQC width k will lower bound the overall query fidelity as

$$F_{\text{virtual},Z} \geq 1 - 8\epsilon(m+1)\cdot 2^k(k+m) \quad (5)$$

$$F_{\text{virtual},X} \geq 1 - 8\epsilon(m+1)\cdot 2^k(k+2^m) \quad (6)$$

for Z errors and X errors, respectively. Additionally, our biased-noise analysis can be easily extended to a gate-based error channel with errors randomly applied using Monte Carlo sampling to quantum gates, up to a constant factor difference. By the observation that each branch of the QRAM intersects with at most $O(m)$ gates, the lower bound of the fidelity has the same asymptotic scaling as under the qubit-based noise model.

5.2 Asymmetric Error Correction

In this section, we explore the fault-tolerant implementation of the virtual QRAM in future quantum hardware, using rectangular surface code to combine error correction design with intrinsic QRAM noise resilience. We assume that the error rate of physical qubits is unbiased with respect to both X and Z, whereas a logical qubit generated by the rectangular surface code exhibits a biased error rate. Adopting this biased-error surface code as qubits in virtual QRAM, balanced fidelity for X and Z errors can be achieved.

Based on the different lower bounds of query fidelity under X and Z error channels, we need a careful choice of the surface code for the different parts in virtual QRAM. First, the logical error rate ratio of X and Z is related to the physical error rate, surface code threshold, and the code distances dx and dz [5]: $\frac{p_x l}{p_z l} \approx \left(\frac{p}{p_{th}}\right)^{dx-dz}$. To balance the logical error rates of X and Z error channels, we adopt the bound from equation 5 and 6 and let $F_x = F_z$, then $\frac{\epsilon_x}{\epsilon_z} = \frac{p_x l}{p_z l} = \frac{(m+k)\cdot 2^k}{2^k(k+2^m)} = \frac{k+m}{k+2^m}$. We obtain the strategy of designing rectangular surface code for each physical qubit in QRAM is choosing lengths of the surface code dx and dz as:

$$dx - dz \approx \frac{\log(\frac{k+m}{k+2^m})}{\log(\frac{p}{p_{th}})} \quad (7)$$

Since the SQC does not have biased-noise resilience, we can encode k address qubits using regular square surface code to achieve full protection for the entire virtual QRAM.

6 EVALUATION METHODOLOGY

6.1 Baseline Architectures

We theoretically analyze the performance of the new circuit and perform simulation in comparison to two baseline architectures, BB (Bucket-Brigade QRAM) and SS (Select-Swap QRAM), described in Sec 2. These two QRAM architectures present state-of-the-art QRAM architecture designs in both consumption of quantum resources and performance of quantum circuits including circuit depth and noise-resilience properties.

6.2 Feynman Path Simulation

We utilized a classical simulation technique called Feynman-Path Simulation (FPS) to efficiently compute the result of QRAM queries. In FPS, each memory address corresponds to one path. Despite path number and running time scaling exponentially in the address space, FPS can still efficiently simulate and analyze larger QRAMs, on account of the following desirable property. QRAM circuits are constructed from a small, fixed set of classical-reversible gates, meaning that all these gates do not map a single computational basis state to a superposition over basis states. As a consequence, the storage overhead does not increase exponentially in the depth of the circuit, and instead remains constant. Notably, Pauli gates have the same property to ensure we execute noisy QRAM simulations with negligible overhead. Compared to the special-purpose simulator in [28], our Feynman Path implementation is the first general-purpose

QRAM simulation that is capable of handling arbitrary input (e.g., memory capacity, address state, and noise models).

6.3 Experimental Setup

Noise-free or Pauli-noise circuit simulations are implemented with a single core on a single node, with the largest simulations using 1.5 MB of RAM. For each value of QRAM address width m, we execute 1024 shots to achieve the average fidelity, using a gate-based error model applied via Monte Carlo sampling. We assume devices with 2D square grid connectivity, commonly used in NISQ or FTQC architectures.

7 RESULTS

7.1 QRAM Resource Estimation

The improvements breakdown by different optimization techniques in Section 3 are listed in Table 1. Table 2 presents a comprehensive comparison of multiple current quantum architectures, including concrete parameters such as circuit depth, number of qubits, and T gate depth, among others. Notably, the asymptotic scaling in the table indicates that all three architectures have the same scaling with respect to qubit count. SQC+BB (Baseline B) is a load-multiple-times architecture that suffers from deficiencies in exponential $O(2^K)$ overhead in T depth and T counting complexity. SQC+SS (Baseline S) is a load-once architecture; however, its swap-network is not as efficient as the router-based QRAM, since it lacks a pipelining strategy to load the address discussed in Section 3. Consequently, the circuit depth and Clifford depth of SQC+SS will be quadratically larger than our new QRAM architecture with factor $O(m^2)$. Therefore, our virtual QRAM *outperforms* or at least *matches* any resource counting compared with the state-of-the-art QRAM architectures.

7.2 Mapping and Routing Overhead

Figure 8 presents the results of our constructive mapping strategy, counting the extra operation depth induced from connectivity constraints in NISQ hardware. The conventional swap-based routing, when applied to QRAM, leads to exponential extra SWAP depth overhead and loss of the logarithmic scaling of query depth in terms of QRAM width m. In contrast, the teleportation-based routing consistently outperforms the other, exhibiting an exponential advantage in extra operation depth. Figure 8 also indicates that QRAM under embedding only introduces a linear overhead for circuit depth, which protects the query latency to be *unchanged* under qubit mapping and routing. Meanwhile, our H-tree embedding and the teleportation scheme reduce the circuit depth exponentially but with only constant qubit resource overhead. With $m = 2n$ address qubits, unused qubits occupy 25% of the total qubits: $(2^n - 1)^2$ vs $(2^{(n+1)} - 1)^2$. The proportion of unused qubits can be further reduced by improved tree embedding discussed in Sec 4.3.

7.3 Biased-noise simulation

As illustrated in Section 5, the QRAM architecture is robust to Z-biased error. We further validate this property by calculating the virtual QRAM fidelity and comparing it to Baseline B, which is robust to arbitrary error channels but has the same scaling as the new virtual QRAM for Z error. As expected, Baseline S exhibits no

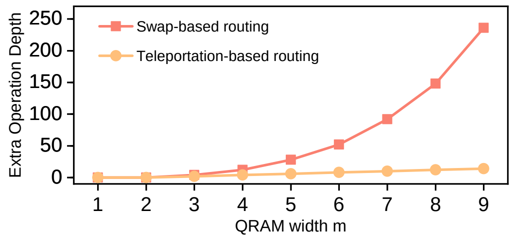


Figure 8: Additional operations after mapping in 2D nearest-neighbor architectures. Swap-based communication scales exponentially worse than teleportation-based communication.

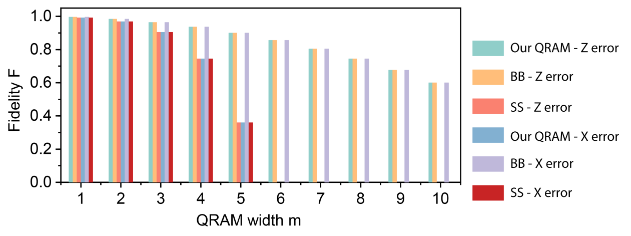


Figure 9: Fidelity comparison for different QRAM architectures. We observe fidelity decays polynomially for Z errors in virtual QRAM and BB QRAM, but for X errors only in BB QRAM.

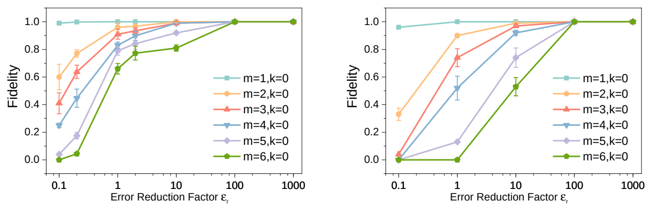


Figure 10: (Left) Fidelity with phase flip error model by numerical simulation. (Right) Fidelity with bit flip error model by numerical simulation.

noise resilience. Figure 9 shows the scaling of the fidelity under Pauli X and Z errors for different architectures with error rate $\epsilon = 10^{-3}$.

Moreover, our simulations in Figure 10 demonstrate the fidelity gap between Z-biased noise and X-biased noise, with far better performance in the former. We also provide Figure 11 to illustrate the trade-off between the QRAM width m and SQC width k under the single qubit Z and X error model. Our plot indicates that the fidelity decays *exponentially* faster when increasing the SQC width parameter k than the QRAM width m.

8 RELATED WORKS

Recently a similar architecture design for QRAM by Chen et al. designed a "load-once" QRAM architecture capable of querying memory with more than 1 bit in each memory cell, i.e., a data width of $k \geq 1$ [10]. The difference between theirs and our work is that we

	RAW	OPT: 1	OPT: 2	OPT: 3	OPT: ALL
Qubits (bit encoding)	$6 \cdot 2^m + k$	$\mathbf{4 \cdot 2^m + k}$	$6 \cdot 2^m + k$	$6 \cdot 2^m + k$	$\mathbf{4 \cdot 2^m + k}$
Circuit depth	$m^2 + (m+1) \cdot 2^k$	$m^2 + (m+1) \cdot 2^k$	$m^2 + (m+1) \cdot 2^k$	$\mathbf{m + (m+1) \cdot 2^k}$	$\mathbf{m + (m+1) \cdot 2^k}$
Classical controlled gates	2^{m+k-1}	2^{m+k-1}	$\mathbf{2^{m+k-2}}$	2^{m+k-1}	$\mathbf{2^{m+k-2}}$

Table 1: Resource overhead improvements from three key optimization methods in Section 3.

	SQC+BB	SQC+SS	Our QRAM
Qubits	$2^m + k$	$2^m + k$	$2^m + k$
Circuit depth	$m \cdot 2^k$	$m^2 \cdot 2^k$	$m \cdot 2^k$
T count	$(2^m + k) \cdot 2^k$	$2^m + k \cdot 2^k$	$2^m + k \cdot 2^k$
T depth	$(m + k) \cdot 2^k$	$m + k \cdot 2^k$	$m + k \cdot 2^k$
Clifford depth	$(m + k) \cdot 2^k$	$(m^2 + k) \cdot 2^k$	$(m + k) \cdot 2^k$

Table 2: Resource overhead comparison between different implementations of virtual QRAM. All costs are in Big-O.

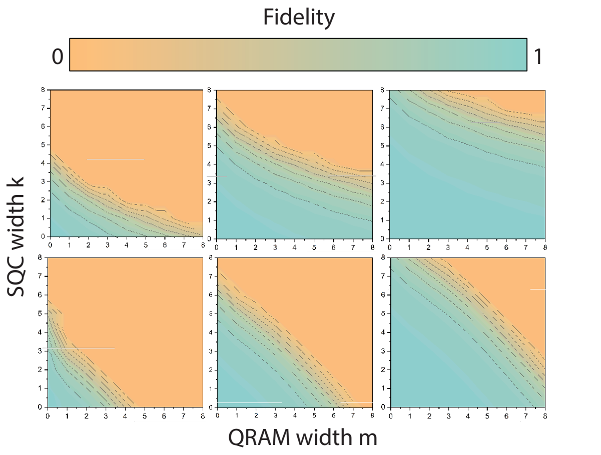


Figure 11: Virtual QRAM fidelity scaling under X and Z error. Top/Bottom: Z/X error with (Left) $\epsilon_r = 1$, (Middle) $\epsilon_r = 10$, (Right) $\epsilon_r = 100$.

target creating larger address widths rather than simply repeating the data retrieved multiple times to query the same address for more bus qubits. Notably, some previous works, e.g., Connor et al. [28], mentioned similar strategies to generalize to high data width. Our virtual QRAM is compatible with a data width larger than 1 by repeatedly querying memory cells one bit at a time, by taking advantage of the parallel retrieval from [10] in our virtual QRAM.

During the preparation of this work, Jaques and Rattew made similar claims on qubit mapping in QRAM in a recent paper [35]. One major limitation of QRAM highlighted in both our work and [35] is the signal latency for communicating qubits within the QRAM. [35] assumes a quantum bus line for communication, which has latency linear to its distance, while our work proposes a novel teleportation-based routing scheme to overcome this bottleneck. Ultimately, via teleportation, we can propagate information across QRAM qubits faster (only limited by classical communication or the speed of light), with negligible quantum delay. This is an important step towards achieving quantum advantage for QRAM with relatively small query delay.

There are significant advances in hardware development towards QRAM using superconducting devices and cold atom arrays. For example, the key element of deterministic `CSWAP` operations between superconducting cavities have been demonstrated in previous work [17, 58]. `CSWAP` operations using superconducting transmon devices have also been demonstrated [42]. `Toffoli` gate and `CSWAP` for QRAM have also been investigated using the Rydberg blockade.

9 CONCLUSION

The use of QRAM is ubiquitous in quantum algorithms. A successful implementation of practical QRAMs could unlock the full potential of quantum computing and bring us closer to realizing practical applications such as optimization, machine learning, and cryptography. Our proposed QRAM architecture addresses the challenges of memory capacity, query latency, and fault-tolerance through innovations in virtualizing QRAM, latency-free mapping to 2D grid architecture, and leveraging intrinsic biased-noise resilience in the circuits. We have shown an end-to-end systems architecture for performing high-fidelity queries of large memories, and identified key technology advances (such as gate error rate reduction or error correction code distance) needed to scale up QRAM and quantum computing platforms as a whole.

ACKNOWLEDGMENTS

YD acknowledges support from NSF (under award CCF-2312754) and Yale University. SMG was supported by the Air Force Office of Scientific Research under award number FA9550-21-1-0209. We thank Liang Jiang, Shruti Puri, Junyu Liu, Nathan Wiebe, and Daniel Weiss for fruitful discussions.

REFERENCES

[1] Matthew Amy, Dmitri Maslov, and Michele Mosca. 2014. Polynomial-time T-depth optimization of Clifford+ T circuits via matroid partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33, 10 (2014), 1476–1489.

[2] Ryan Babbush, Craig Gidney, Dominic W Berry, Nathan Wiebe, Jarrod McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven. 2018. Encoding electronic spectra in quantum circuits with linear T complexity. *Physical Review X* 8, 4 (2018), 041015.

[3] Debjyoti Bhattacharjee, Abdullah Ash Saki, Mahabubul Alam, Anupam Chattopadhyay, and Swaroop Ghosh. 2019. MUQUT: Multi-constraint quantum circuit mapping on NISQ computers. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–7.

[4] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. 2017. Quantum machine learning. *Nature* 549, 7671 (2017), 195–202.

[5] J Pablo Bonilla Ataides, David K Tuckett, Stephen D Bartlett, Steven T Flammia, and Benjamin J Brown. 2021. The XZZX surface code. *Nature communications* 12, 1 (2021), 2172.

[6] H-J Briegel, Wolfgang Dür, Juan I Cirac, and Peter Zoller. 1998. Quantum repeaters: the role of imperfect local operations in quantum communication. *Physical Review Letters* 81, 26 (1998), 5932.

[7] Sally Anne Browning. 1980. *The tree machine: A highly concurrent computing environment.* California Institute of Technology.

[8] Gregory T Byrd and Yongshan Ding. 2023. Quantum Computing: Progress and Innovation. *Computer* 56, 1 (2023), 20–29.

[9] Adán Cabello. 2002. Bell's theorem with and without inequalities for the three-qubit Greenberger-Horne-Zeilinger and W states. *Physical Review A* 65, 3 (2002), 032108.

[10] Zhao-Yun Chen, Cheng Xue, Tai-Ping Sun, Huan-Yu Liu, Xi-Ning Zhuang, Meng-Han Dou, Tian-Rui Zou, Yuan Fang, Yu-Chun Wu, and Guo-Ping Guo. 2023. An Efficient and Error-Resilient Protocol for Quantum Random Access Memory with Generalized Data Size. *arXiv preprint arXiv:2303.05207* (2023).

[11] Frederic T Chong, Diana Franklin, and Margaret Martonosi. 2017. Programming languages and compiler design for realistic quantum hardware. *Nature* 549, 7671 (2017), 180–187.

[12] Olivia Di Matteo, Vlad Gheorghiu, and Michele Mosca. 2020. Fault-tolerant resource estimation of quantum random-access memories. *IEEE Transactions on Quantum Engineering* 1 (2020), 1–13.

[13] Yongshan Ding and Frederic T Chong. 2020. Quantum computer systems: Research for noisy intermediate-scale quantum computers. *Synthesis lectures on computer architecture* 15, 2 (2020), 1–227.

[14] Yongshan Ding, Pranav Gokhale, Sophia Fuhui Lin, Richard Rines, Thomas Propson, and Frederic T Chong. 2020. Systematic crosstalk mitigation for superconducting qubits via frequency-aware compilation. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO).* IEEE, 201–214.

[15] Yongshan Ding, Xin-Chuan Wu, Adam Holmes, Ash Wiseth, Diana Franklin, Margaret Martonosi, and Frederic T Chong. 2020. Square: Strategic quantum ancilla reuse for modular quantum programs via cost-effective uncomputation. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA).* IEEE, 570–583.

[16] Bertrand Ducourthial, G Constantinescu, and Alain Mérigot. 1998. *Implementing image analysis with a graph-based parallel computing model.* Springer.

[17] Yvonne Y Gao, Brian J Lester, Kevin S Chou, Luigi Frunzio, Michel H Devoret, Liang Jiang, SM Girvin, and Robert J Schoelkopf. 2019. Entanglement of bosonic modes through an engineered exchange interaction. *Nature* 566, 7745 (2019), 509–512.

[18] Vlad Gheorghiu, Michele Mosca, and Priyanka Mukhopadhyay. 2022. T-count and T-depth of any multi-qubit unitary. *npj Quantum Information* 8, 1 (2022), 141.

[19] Craig Gidney. 2018. Halving the cost of quantum addition. *Quantum* 2 (2018), 74.

[20] Craig Gidney and Martin Ekerå. 2021. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum* 5 (2021), 433.

[21] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. 2019. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing.* 193–204.

[22] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. 2008. Architectures for a quantum random access memory. *Physical Review A* 78, 5 (2008), 052310.

[23] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. 2008. Quantum random access memory. *Physical review letters* 100, 16 (2008), 160501.

[24] Pranav Gokhale, Olivia Angiuli, Yongshan Ding, Kaiwen Gui, Teague Tomesh, Martin Suchara, Margaret Martonosi, and Frederic T Chong. 2019. Minimizing state preparations in variational quantum eigensolver by partitioning into commuting families. *arXiv preprint arXiv:1907.13623* (2019).

[25] Daniel M Greenberger, Michael A Horne, and Anton Zeilinger. 1989. Going beyond Bell's theorem. *Bell's theorem, quantum theory and conceptions of the universe* (1989), 69–72.

[26] Lov K Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing.* 212–219.

[27] Connor T Hann. 2021. *Practicality of Quantum Random Access Memory.* Ph. D. Dissertation. Yale University.

[28] Connor T Hann, Gideon Lee, SM Girvin, and Liang Jiang. 2021. Resilience of quantum random access memory to generic noise. *PRX Quantum* 2, 2 (2021), 020311.

[29] Aram Harrow. 2001. *Quantum compiling.* Ph. D. Dissertation. Massachusetts Institute of Technology, Department of Physics.

[30] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. 2009. Quantum algorithm for linear systems of equations. *Physical review letters* 103, 15 (2009), 150502.

[31] Ralf Heckmann, Ralf Klasing, Burkhard Monien, and Walter Unger. 1992. Optimal embedding of complete binary trees into lines and grids. In *Proc. 17th Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG'91), Lecture Notes in Computer Science.* 25–35.

[32] Stefan Hillmich, Alwin Zulehner, and Robert Wille. 2021. Exploiting quantum teleportation in quantum circuit mapping. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference.* 792–797.

[33] Clare Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter. 2012. Surface code quantum computing by lattice surgery. *New Journal of Physics* 14, 12 (2012), 123011.

[34] Richard C Jaeger, Travis N Blalock, and Benjamin Joseph Blalock. 1997. *Microelectronic circuit design.* McGraw-Hill New York.

[35] Samuel Jaques and Arthur G Rattew. 2023. QRAM: A Survey and Critique. *arXiv preprint arXiv:2305.10310* (2023).

[36] Tom Kilburn, David BG Edwards, Michael J Lanigan, and Frank H Sumner. 1962. One-level storage system. *IRE Transactions on Electronic Computers* 2 (1962), 223–235.

[37] Gushu Li, Yufei Ding, and Yuan Xie. 2019. Tackling the qubit mapping problem for NISQ-era quantum devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems.* 1001–1014.

[38] Y-B Lin, Zevi Miller, Manley Perkel, Dan Pritikin, and Ivan Hal Sudborough. 2003. Expansion of layouts of complete binary trees into grids. *Discrete applied mathematics* 131, 3 (2003), 611–642.

[39] Guang Hao Low and Isaac L Chuang. 2017. Optimal Hamiltonian simulation by quantum signal processing. *Physical review letters* 118, 1 (2017), 010501.

[40] Guang Hao Low, Vadym Kliuchnikov, and Luke Schaeffer. 2018. Trading T-gates for dirty qubits in state preparation and unitary synthesis. *arXiv preprint arXiv:1812.00954* (2018).

[41] Dmitri Maslov, Sean M Falconer, and Michele Mosca. 2008. Quantum circuit placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 4 (2008), 752–763.

[42] Connie Miao, Gideon Lee, Liang Jiang, and David Schuster. 2023. Implementation of a Quantum Switch with Superconducting Circuits. *Bulletin of the American Physical Society* (2023).

[43] Abtin Molavi, Amanda Xu, Martin Diges, Lauren Pick, Swamit Tannu, and Aws Albarghouthi. 2022. Qubit Mapping and Routing via MaxSAT. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO).* IEEE, 1078–1091.

[44] Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. 2019. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems.* 1015–1029.

[45] Michael A Nielsen and Isaac Chuang. 2002. Quantum computation and quantum information.

[46] Jaroslav Opatrny and Dominique Sotteau. 2000. Embeddings of complete binary trees into grids and extended grids with total vertex-congestion 1. *Discrete Applied Mathematics* 98, 3 (2000), 237–254.

[47] Jian-Wei Pan, Dik Bouwmeester, Harald Weinfurter, and Anton Zeilinger. 1998. Experimental entanglement swapping: entangling photons that never interacted. *Physical review letters* 80, 18 (1998), 3891.

[48] Koustubh Phalak, Junde Li, and Swaroop Ghosh. 2022. Approximate Quantum Random Access Memory Architectures. *arXiv preprint arXiv:2210.14804* (2022).

[49] Aditya K Prasad, Vivek V Shende, Igor L Markov, John P Hayes, and Ketan N Patel. 2006. Data structures and algorithms for simplifying reversible circuits. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 2, 4 (2006), 277–293.

[50] John Preskill. 2018. Quantum computing in the NISQ era and beyond. *Quantum* 2 (2018), 79.

[51] Oded Regev and Liron Schiff. 2008. Impossibility of a quantum speed-up with a faulty oracle. In *Automata, Languages and Programming: 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I 35.* Springer, 773–781.

[52] Peter W Shor. 1994. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science.* Ieee, 124–134.

[53] Xiaoming Sun, Guojing Tian, Shuai Yang, Pei Yuan, and Shengyu Zhang. 2023. Asymptotically optimal circuit depth for quantum state preparation and general unitary synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2023).

[54] Swamit S Tannu and Moinuddin K Qureshi. 2019. Not all qubits are created equal: A case for variability-aware policies for NISQ-era quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems.* 987–999.

[55] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. 2020. Convex optimization using quantum oracles. *Quantum* 4 (2020), 220.

[56] Rodney Van Meter. 2014. *Quantum networking.* John Wiley & Sons.

[57] Xin-Chuan Wu, Dripto M Debroy, Yongshan Ding, Jonathan M Baker, Yuri Alexeev, Kenneth R Brown, and Frederic T Chong. 2021. Tilt: Achieving higher fidelity on a trapped-ion linear-tape quantum computing architecture. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA).* IEEE, 153–166.

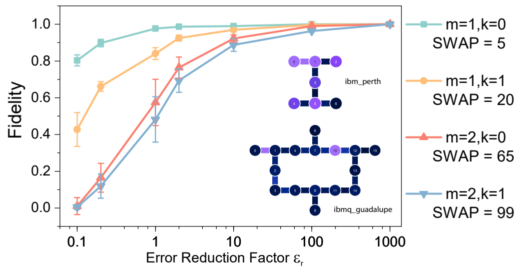


Figure 12: Fidelity with reduced error model from IBMQ. Extra SWAP gates are reported under the legend.

[58] Sophia Xue, Stijn de Graaf, Benjamin Chapman, Yaxing Zhang, James Teoh, Jacob Curtis, Takahiro Tsunoda, Alec Eickbusch, Alexander Read, Akshay Koottandavida, et al. 2023. A hybrid controlled-SWAP gate between two bosonic modes. *Bulletin of the American Physical Society* (2023).

[59] Chi Zhang, Ari B Hayes, Longfei Qiu, Yuwei Jin, Yanhao Chen, and Eddy Z Zhang. 2021. Time-optimal qubit mapping. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 360–374.

[60] Jun Zhang, Jiri Vala, Shankar Sastry, and K Birgitta Whaley. 2004. Optimal quantum circuit synthesis from controlled-unitary gates. *Physical Review A* 69, 4 (2004), 042309.

A IS QRAM VIABLE ON CURRENT QPUS?

Current QPUs have limited qubit count, device connectivity, and fidelity. In this section, we investigate the main technological improvements needed to implement QRAM. Our real hardware noise simulation experiments are implemented in Python 3.9, interfacing with IBM's Qiskit software library. All the IBMQ noisy simulation experiments are executed with 200 shots, using 48 cores and 30GB of RAM with error models from IBM's *ibm_perth* for $m = 1$ cases and from IBM's *ibmq_guadalupe* for $m = 2$ cases. The topology of the two machines is listed in Figure 12. Our analysis in Section 5 reveals a lower bound of the fidelity that suggests high fidelity can be achieved for small-scale QRAM by enhancing the error rate of the quantum hardware. To this end, we introduce an *Error Reduction Factor* (ϵ_r) that describes the error rate ratio between expected future hardware and current hardware. $\epsilon_r = \frac{\text{current error rate}}{\text{future error rate}}$, with the current hardware error rate assumed to be 10^{-3}.

In Figure 12, we assess the fidelity of the small-scale virtual QRAM under a realistic noise model obtained from IBM quantum hardware and simulate the virtual QRAM with error rates ϵ_r times lower than the current model to predict its performance in the future machine. Since IBM hardware has sparser connectivity than 2D square grid analyzed in Sec 4, extra SWAP gates by Qiskit default transpiler Sabre [37] are needed to implement QRAM operations, with counting in Figure 10. Despite the extra SWAP operations, encouragingly, our results indicate that with a future noise model *10 times* better than the current technology, we can achieve significantly improved query fidelity. Furthermore, when error rates are reduced to 10^{-5} possibly via near-term small-scale error correction, then the query fidelity can reach higher than 0.98.