On the Robustness of Bucket Brigade Quantum RAM

Srinivasan Arunachalam[1,2], Vlad Gheorghiu[2,3], Tomas Jochym-O'Connor[2,4], Michele Mosca[2,3,5,6], and Priyaa Varshinee Srinivasan[7,8]

1 **Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands**
2 **Institute for Quantum Computing, University of Waterloo, Waterloo, ON, N2L 3G1, Canada**
3 **Department of Combinatorics & Optimization, University of Waterloo, Waterloo, ON, N2L 3G1, Canada**
4 **Department of Physics & Astronomy, University of Waterloo, Waterloo, ON, N2L 3G1, Canada**
5 **Perimeter Institute for Theoretical Physics, Waterloo, ON, N2L 6B9, Canada**
6 **Canadian Institute for Advanced Research, Toronto, ON, M5G 1Z8, Canada**
7 **David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, N2L 3G1, Canada**
8 **Institute for Quantum Science and Technology, University of Calgary, Calgary, AB, T2N 1N4, Canada**

Abstract

We study the robustness of the bucket brigade quantum random access memory model introduced by Giovannetti, Lloyd, and Maccone [Phys. Rev. Lett. **100**, 160501 (2008)]. Due to a result of Regev and Schiff [ICALP'08 pp. 773], we show that for a class of error models the error rate per gate in the bucket brigade quantum memory has to be of order $o(2^{-n/2})$ (where $N = 2^n$ is the size of the memory) whenever the memory is used as an oracle for the quantum searching problem. We conjecture that this is the case for any realistic error model that will be encountered in practice, and that for algorithms with super-polynomially many oracle queries the error rate must be super-polynomially small, which further motivates the need for quantum error correction. We introduce a circuit model for the quantum bucket brigade architecture and argue that quantum error correction for the circuit causes the quantum bucket brigade architecture to lose its primary advantage of a small number of "active" gates, since all components have to be actively error corrected.

1998 ACM Subject Classification J.2 Physical Sciences and Engineering

Keywords and phrases Quantum Mechanics, Quantum Memories, Quantum Error Correction

Digital Object Identifier 10.4230/LIPIcs.TQC.2015.226

1 Introduction

A random access memory (RAM) is a device that stores information in an array of memory cells in the form of bits. In contrast to other types of information storage devices, the access latency to any memory cell is constant and does not depend on the location of the information in the RAM. Information stored in the RAM is retrieved by inputting the address of the desired memory cell in a routing circuit. Any address in a RAM with $N = 2^n$ memory cells can by addressed via a unique n bit input query string. The corresponding output register contains the contents of the addressed memory location.

© Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, and Priyaa Varshinee Srinivasan;
licensed under Creative Commons License CC-BY

The typical physical implementation of the addressing mechanism uses the fanout architecture [17, 26], in which the routing scheme corresponds to a binary tree. Each node consists of a pair of transistors which routes the electronic signal down one of the two paths to the subsequent level. In the fanout architecture, a given level has all nodes sharing the same routing direction (left or right), set by the corresponding address bit. An n bit query string determines a unique path in the binary tree, corresponding to the desired memory location. In the process, $\mathcal{O}(2^n)$ transistors are activated.

Alternative routing schemes with $\mathcal{O}(\text{poly}(n))$ activated transistors have been proposed, corresponding to exponentially lower energy consumption. One such example is the "bucket brigade" scheme [13, 12]. However, most of the classical implementations follow the simpler fanout architecture, as the power consumption of RAM is negligible in comparison with the power consumption of other components in the architecture.

The classical RAM addressing scheme can be generalized to a quantum RAM (which we simply call qRAM from here on) scheme, where the input is a quantum state, the routing components are inherently quantum, and the information stored can be either classical, i.e. $|0\rangle$ or $|1\rangle$ but not a superposition of both, or quantum, i.e. any arbitrary superposition of $|0\rangle$ and $|1\rangle$. In the present paper we consider qRAM that stores only classical information. Such memory allows querying superposition of addresses

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

where $\sum_j \alpha_j|j\rangle$ is a superposition of queried addresses and $|m_j\rangle$ represents the content of the j-th memory location. A memory that stores classical information but allows queries in superposition is required for quantum algorithms such as Grover's search on a classical database [23], collision finding [6], element distinctness [1], dihedral hidden subgroup problem [20] and various practical applications mentioned in [3]. In fact, such a quantum memory plays the role of the oracle and is ideal in implementing any oracle-based quantum algorithm, in which the oracle is used to query classical data in superposition.

A conceptually simple physical implementation of a qRAM corresponds to a direct generalization of the fanout architecture used in classical RAMs. However, the number of faulty components that can be tolerated by the quantum architecture is of prime importance due to the difficulty in maintaining quantum coherence. This motivates searching for schemes with fewer faulty components. A fundamental assumption of the qRAM architecture is that "active" gates[1] are the only ones with significant errors.

In this paper we investigate the bucket brigade qRAM proposal introduced in [13, 12]. Assuming one requires a constant error probability for the oracle query, then with the bucket brigade error model it suffices to have an error rate that is on the order of $\mathcal{O}(1/n^2)$. In the bucket brigade model, one assumes that each computational path only contains $\mathcal{O}(n)$

[1] The concept of "active" gates introduced in [13, 12] is somewhat unnatural when extended to quantum gates. At the physical level, a gate is considered active if it physically acts on its input. Since the qRAM may be in a superposition of querying many (or all) possible bit values in the memory, every gate may be in a superposition of being active or not. Implicitly, there is a physical process that is checking whether each gate is active, and then acting in that case, and such a process will not be perfect in practice. Translated into the circuit model, such gates may be modelled as controlled-gates, i.e. gates that act on its input provided that the control qubit is set to $|1\rangle$. Therefore, such a gate is considered "active" if its control is set to $|1\rangle$ and "non-active" otherwise.
In practice, even non-active gates will be prone to errors. The implicit assumption is that these errors are much smaller than the errors in active gates, and the focus of the bucket brigade models is to reduce the impact of the higher order errors found in the active part of a gate.

components that are faulty, and that a total of $\mathcal{O}(n^2)$ faulty operations are performed. One can argue that it is optimistic to assume that the so-called "non-active" components will be completely error-free. And, one could counter-argue that the error rates will be much lower, and thus ignored for problem instances of appropriate size. For the purposes of this article, we set aside these concerns and accept the premise of there only being $\mathcal{O}(n)$ faulty components.

In contrast to such a qRAM, if one just used a regular fanout circuit for the lookup, with no error correction, one would need to maintain quantum coherence over an exponential number of components [12]. In order to achieve a constant error rate for the query in this case, one would need to implement a fault-tolerant version of the look-up circuit, which would normally incur an overhead that is polynomial in n. One advantage of bucket brigade qRAM is thus to bypass the poly-log overhead of fault tolerant quantum error correction needed to achieve a constant error rate for a look-up. Such an error rate would be sufficient if the qRAM is used in an algorithm making a constant number of queries, for example, for certain state generation algorithms [22, 14]. In general, for an algorithm with inverse polynomially many queries, it would suffice to reduce the query error rate to be inverse polynomial in n, e.g. [21, 11].

In this article, we firstly shed doubt on the usefulness of a qRAM that provides queries with constant probability of error, when used with algorithms that make super-polynomially many oracle queries. As an aside, we note that if the imperfect query operation is assumed to be unitary, and if one can apply the inverse of this imperfect query, then one can apply simple amplification methods to achieve queries with arbitrarily small error δ using a number of repetitions that is proportional to $\log(1/\delta)$. It was shown that this logarithmic overhead is not necessary for quantum searching [16] and other problems [9]. However, there is no reason to expect the errors in a realistic qRAM to behave this way, and in this article we consider incoherent errors.

We first show that a very simple model of incoherent physical errors induces an overall query error similar to the one described by Regev and Schiff [25]. Consequently, a qRAM that produces queries with constant error will not permit the quadratic speed-up in Grover's search algorithm or any other quantum search algorithm one might design. We show that one cannot escape achieving an error rate that is super-polynomially small. We conjecture that this error model nullifies the asymptotic speed-ups of other quantum query algorithms as well, and leave as open questions the extension of this result to other important query problems.

This negative result implies the need for some means of error reduction for the qRAM, with a look-up error rate exponential in n. For consistency we assume a physical error rate that is inverse polynomial in n, the logarithm of the size of the database. We thus explore a natural approach, using quantum error correcting codes, to provide this error reduction, and argue that the apparent advantage of qRAM disappears in this case; in principle, one can make the error rate arbitrarily small, however the advantage of a small number of activated gates in the bucket brigade architecture appears to be lost when active error correction has to be performed on each gate. The main motivation for the quantum bucket brigade approach over a straightforward binary-tree approach is that the equivalent of the active gates are the only gates prone to error, and thus an inverse polynomial in n error rate suffices in order to achieve an overall constant error per qRAM look-up.

The remainder of this paper is organized as follows. In Sec. 2 we describe the bucket brigade qRAM architecture and prove that for the Regev and Schiff model [25] the error rate per gate must scale as inverse polynomial in the size of the database. In Sec. 3 we

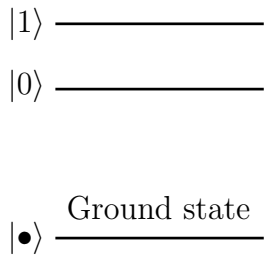

Figure 1 Representation of the energy levels of a qutrit used at the nodes of the routing binary tree. The states $|0\rangle$ and $|1\rangle$ form a metastable subspace since the energy difference between the states is required to be much smaller than the difference between the ground state $|\bullet\rangle$ and $|0\rangle$.

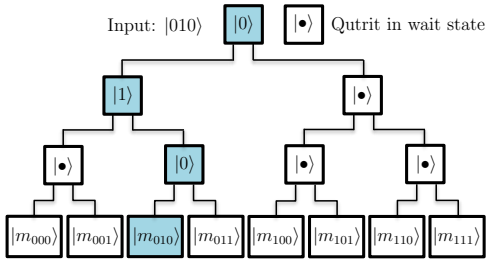

Figure 2 (Color online) Bucket brigade scheme for a qRAM with 8 memory locations. The address register is $|010\rangle$, corresponding to the memory location m_{010}. The path $0 \to 1 \to 0$ is established by sequentially introducing the address qubits $|010\rangle$ into the root of the tree.

develop and analyze a simple error model that provides intuition for the overall behaviour of the memory with realistic noisy environments. In Sec. 4, in order to discuss approaches for introducing quantum error correction inside the qRAM architecture, we introduce a circuit model for the bucket brigade architecture. We then argue in Sec. 5 that a fault-tolerant bucket brigade qRAM loses the advantage of small number of active components. Finally, in Sec. 6 we conclude and present some open problems and directions for future research.

2 Quantum RAM Architectures

In [13, 12], Giovanetti *et al.* proposed a quantum bucket brigade addressing scheme requiring only $\mathcal{O}(n)$ activations per memory call. The nodes of the routing binary tree are three level quantum systems (qutrits), with an energy spectrum schematically depicted in Fig. 1.

The 2^n qutrits at the nodes of the binary tree are initially prepared in the ground state $|\bullet\rangle$, named the "wait" state, and the memory address is specified by the n-qubit state $|a_0 a_1 \dots a_{n-1}\rangle$. At time t_0, the address qubit $|a_0\rangle$ is input at the root of the tree and it interacts with the qutrit at node 0 changing its state from $|\bullet\rangle$ to $|a_0\rangle$. The states $\{|0\rangle, |1\rangle\}$ of the node qutrit are coupled to two spatial directions (paths), right and left respectively. The role of the coupling is to route the following incoming address photon along the correct path of the binary routing tree. At time t_1, the subsequent address qubit $|a_1\rangle$ is input at the root of the tree. The address qubit $|a_1\rangle$ interacts with the qutrit at node 0 and is physically routed down the left or right path of the tree depending upon the state $|a_0\rangle$ of node 0. Consequently it changes the state of the corresponding node at level 1 to $|a_1\rangle$. The process continues until all the remaining address qubits are sent through the tree, with the k-th address qubit changing the state of the node at the k-th level from $|\bullet\rangle$ to $|a_k\rangle$. After $\mathcal{O}(n^2)$ time steps[2], a routing path is assigned from the root of the tree to the desired memory location, with only n nodes in the path (one node per level) having a state different from $|\bullet\rangle$. A bucket brigade routing scheme for an 2^3-address qRAM is schematically depicted in Fig. 2. The proposed physical implementation of bucket brigade in [12] uses atoms in a cavity as routing nodes and polarization photon states as addressing qubits.

[2] The k-th address qubit interacts with the first $k-1$ routing nodes, followed by a single interaction with the corresponding node at the k-th level. Considering each interaction takes a single time step, the k-th address qubit changes the state of the corresponding node at the k-th level after k time steps. Considering there are a total of n address qubits, the overall time required is $\mathcal{O}(n^2)$.

In [12], the authors claim that the bucket brigade scheme is coherent as long as the error per gate, ε scales as $\mathcal{O}(1/n^2)$. For this error scaling, as n increases, the *overall* error rate of the qRAM oracle asymptotically approaches a constant. Yet constant overall error rates are not favourable for some important quantum algorithms. For example, Regev and Schiff [25] showed that the quadratic speed-up in Grover's searching algorithm vanishes when using oracles with a constant error rate. Namely, in order to regain the quadratic speed-up, the error rate per oracle call should scale no worse than $\mathcal{O}(2^{-n/2})$ (therefore the error rate not only needs to be non-constant it must vanish at a fast enough rate with increasing n).

In the next few sections, we construct a simple model of bucket brigade qRAM with errors and show in Appendix A that Regev and Schiff error model [25] resembles the model we construct. Based on this resemblance and assuming $\mathcal{O}(n^2)$ faulty operations per memory call, we conjecture that in order to implement the qRAM for quantum searching, the overall error rate per memory call has to be in $\mathcal{O}(2^{-n/2})$. In fact, for this to hold, the error rate per gate ε should decrease faster than $1/f(n)$, where $f(n) \in \omega(2^{n/2})$. Thus ε has to be in $o(2^{-n/2})$ and hence much smaller than $\mathcal{O}(1/n^2)$, since the overall error rate per memory call must scale as

$$1 - \left(1 - \frac{1}{f(n)}\right)^{\mathcal{O}(n^2)} \in \Omega\left(\frac{n^2}{f(n)}\right), \tag{2}$$

and in order to satisfy

$$1 - \left(1 - \frac{1}{f(n)}\right)^{\mathcal{O}(n^2)} \in \mathcal{O}(2^{-n/2}), \tag{3}$$

it is required that

$$f(n) \in \Omega\left(n^2 2^{n/2}\right) \implies \frac{1}{f(n)} \in \mathcal{O}\left(\frac{1}{n^2 2^{n/2}}\right) \implies \varepsilon \in o(2^{-n/2}). \tag{4}$$

Recently Hong et al. [15] proposed a bucket brigade qRAM scheme in which the number of time steps required per memory call is reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. While this reduction decreases the overall error rate, the error rate per gate ε must still be in $o(2^{-n/2})$.

The need for super-polynomially small (in n) error rate per gate for real world applications motivates a more thorough analysis of the bucket brigade qRAM scheme and the need for quantum error correction, these topics being the subject of the following sections.

3 Errors Analysis

In this section we introduce a simple toy error model for the physical implementation proposed in [12], in which the qutrits are implemented by trapped atoms in cavities. The address qubits are implemented by photons that propagate along the network of cavities, and excite the corresponding qutrit to either of the states $|0\rangle$ or $|1\rangle$, depending on their polarization. In this way, the incoming address photons create a "path" through the binary tree of cavities, leading to the desired memory location. The readout is performed by injecting a "bus" qubit (photon) at the root of the tree that interacts with the desired memory location, copies its value (the states stored by the memory are $|0\rangle$ or $|1\rangle$, and not any superposition), and finally is sent back along the routing tree exiting through the root with the corresponding memory location content. For more details about the physical model an interested reader is referred to [12].

3.1 Toy Error Model

In the following we assume that the only source of errors in the above model is due to random flips between the states $|0\rangle$ and $|1\rangle$ of the qutrit. We assume a typical symmetric bit-flip error, in which at each time step the state $|j\rangle$ can either flip to $|j \oplus 1\rangle$ with probability ε or remain unchanged with probability $1 - \varepsilon$. The motivation for considering this error model is that, since the states $\{|0\rangle, |1\rangle\}$ are close together in the energy spectrum, significantly less energy is required to cause a flip between them, hence such flips are more likely to occur. In reality, there may be other sources of errors such as coupling errors, decaying of excited qutrit states to the ground state, loss of photons during the routing process and so on. However, our toy model illustrates the effects of an error that would naturally occur in a realistic physical realization of a qRAM. There is no reason to expect these other sources of errors would help matters (otherwise, one could seek to deliberately introduce or simulate such errors).

It is not hard to observe that any error in the routing process can propagate through the tree resulting in various possibilities. Considering all possible errors in such a model, the possible paths that the bus photon could take in the final step termed as *right path*, *wrong path* and *no-path*, respectively. For convenience, we further assume the operations used to un-compute the path information encoded in the qRAM are error-free.

1) Right path. This scenario occurs when no flips (errors) arise during the routing process. In this ideal scenario, the bus reaches the correct location in the qRAM as specified by the input address. Fig. 2 depicts an example of a *right path* given an input address $|010\rangle$.

To compute the probability p_{rp} of such an event, we require that no bit flip occurs at each of the j levels. Taking the intersection of such events for all $n - 1$ levels of the binary tree gives the probability of the *right path*

$$p_{rp} = \prod_{j=0}^{n-1} (1-\varepsilon)^{n-j} = (1-\varepsilon)^{\sum_{j=0}^{n-1}(n-j)} = (1-\varepsilon)^{n(n+1)/2}. \tag{5}$$

2) Wrong path. This error refers to the cases wherein the the bus reaches *any other* location in the qRAM other than the location corresponding to the input address. A *wrong path* error occurs at level i if the state $|j\rangle$ of the active routing qutrit at level i flips to $|j \oplus 1\rangle$ and no other errors occur subsequently (at later time steps). The scenario where another error occurs at a later time step in the levels preceding to the j-th level leads to a *no-path* error which we discuss later. The following two figures illustrate two possible *wrong path*s for the input address $|010\rangle$. In Fig. 3, the error is assumed to occur in the third time step, due to which the bus accesses the wrong location corresponding to $|011\rangle$. In Fig. 4, the error is assumed to occur in the second time step, with the bus wrongly accessing the location corresponding to $|000\rangle$.

In order to calculate the probability of a *wrong path* occurring, we consider the probability of any path occurring, regardless of whether it is the *right* or *wrong* path, we denote this probability by p_{path}. Suppose the state $|\psi_j\rangle$ is being routed down the qRAM circuit to the j-th level. If any of the $(j-2)$ first routing nodes have flipped then the state will be routed down an unexpected branch and will not excite the j-th level of the tree, resulting in a *no-path*. The probability of success at this given time step is therefore $(1-\varepsilon)^{j-1}$, where ε is the probability of a node flipping, recall we must include the level-0 root node here. This can only for levels 2 and above. The overall probability of success is therefore the product of each of the individual probabilities of success at each time step (including the time step

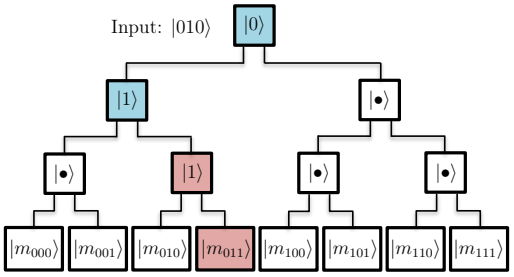

Figure 3 (Color online) Example of a *wrong path* produced by an error at the third time step, given the address $|010\rangle$.

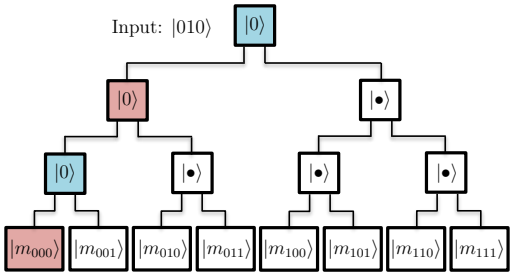

Figure 4 (Color online) Example of a *wrong path* produced by an error at the second time step, given the address $|010\rangle$.

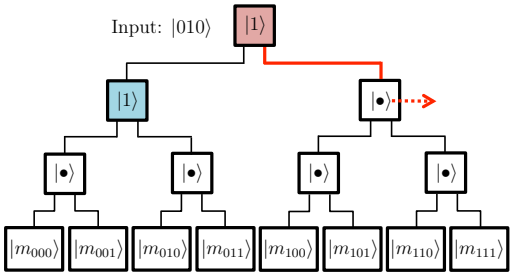

Figure 5 (Color online) Example of a *no-path* given the address $|010\rangle$.

to send the bus qubit down the tree to recover the information stored in the RAM). This probability is given by:

$$p_{path} = p_{wp} + p_{rp} = \prod_{j=2}^{n}(1-\varepsilon)^{j-1} = (1-\varepsilon)^{\sum_{j=2}^{n}(j-1)} = (1-\varepsilon)^{n(n-1)/2}. \tag{6}$$

As we computed before the probability of a *right path* p_{rp} in Eq. (5), the probability of a *wrong path* is then

$$p_{wp} = p_{path} - p_{rp} = (1-\varepsilon)^{n(n-1)/2} - (1-\varepsilon)^{n(n+1)/2}. \tag{7}$$

3) No-path. This error refers to the scenario where the bus never reaches *any* location of the qRAM. Such an error arises when a bit flip error occurs in levels 0 to $n-3$. The smallest such tree where this error can occur is therefore a three-level tree (corresponding to a qRAM with 2^3 memory cells), as shown in Fig. 5. The difference between a *wrong path* and a *no-path* is that, in the latter, the bus photon does not reach the memory address, hence does not read any information, whereas in the former scenario the bus reaches the wrong address in the qRAM and after the un-computing stage, the bus contains the information of *some* particular address in the qRAM.

We present an example of a *no-path* error in Fig. 5, for an input address 010. At the first time instant, the first address photon (i.e. $|0\rangle$) activates the switch (qutrit) in the first layer of the tree. At the second time instant, the address photon $|1\rangle$ interacts with the switch in the first layer, now in state $|0\rangle$, to decide the direction in which it has to be routed. Assuming no error during the second time step, the second address photon is correctly routed to the left path. Assume now that at the third time instant, a flip error occurs on the root qutrit,

which flips its state from $|0\rangle$ to $|1\rangle$. The third address photon would then be incorrectly routed to the path on the right. As it can be seen from Fig. 5, at the third time instant there are two activated switches in the second level. The readout bus photon can no longer reach any of the memory locations, and will be lost in the second level of routing tree.

The probability of a *no-path* event is simply

$$p_{np} = 1 - p_{wp} - p_{rp} = 1 - (1-\varepsilon)^{n(n-1)/2}. \tag{8}$$

If the qRAM is used to implement a quantum oracle O, then O will be faulty, with an error model described by

$$\rho \xrightarrow{O} p_{rp}\hat{O}\rho\hat{O}^\dagger + p_{wp}\mathcal{E}_{wp}(\rho) + p_{np}\mathcal{E}_{np}(\rho), \tag{9}$$

with $\hat{O}$ denoting a perfect oracle. Here $\mathcal{E}_{wp}(\cdot)$ and $\mathcal{E}_{np}(\cdot)$ are error channels that corresponds to the *wrong path* and *no-path* errors, respectively.

Our error model Eq. (9) is less optimistic than the one of Regev and Schiff [25] of the form $\rho \xrightarrow{O} (1-p)\hat{O}\rho\hat{O}^\dagger + p\rho$. The main difference is that the latter does not mix the amplitudes of the initial starting superposition state in Grover's search algorithm, whereas our model decoheres the system much faster due to the non-trivial errors $\mathcal{E}_{wp}$ and $\mathcal{E}_{np}$. Although we do not have a proof that the quantum query complexity of our model cannot be less than the one considered in [25] (i.e. linear in N), we argue (based on a formal proof for a similar decoherence model, see Appendix A) that this is indeed the case.

3.2 Asymptotic Behaviour

In Figs. 6, 7 and 8 we analyze the probabilities of the three types of errors discussed in the previous subsection. The parameters of interest are the error probability per gate, denoted by ε, the overall fidelity of the addressing circuit (i.e. the probability of a *right-path*), denoted by p_{rp}, and the number of levels in the qRAM addressing binary tree denoted by n (corresponding to 2^n memory locations).

For a fixed ε, we see that the *no-path* behaviour becomes the dominating term in the error model, asymptotically with n, as depicted in Fig. 6.

For a fixed n, again the *no-path* term dominates when the error per gate ε becomes large, see Fig. 7.

Finally, for a fixed desired overall fidelity p_{rp}, the maximum allowed error probability per gate ε to achieve the overall fidelity p_{rp} decays exponentially as a function of n, as plotted in Fig. 8.

From Fig. 8 it can be seen that, the error rate per gate of $\mathcal{O}(1/n^2)$ (blue line in Fig. 8) as considered in Giovannetti et al. [12] is more optimistic than our error rate $\varepsilon(n)$ (red line in Fig. 8)

For larger output fidelity p_{rp}, $\varepsilon(n)$ will always be bounded above by $1/n^2$, with the gap between the two increasing as p_{rp} approaches towards 1. Asymptotically in n, the two graphs converge towards zero.

Simply, the difference between our error $\varepsilon(n)$ and the one in [12] can best be understood by investigating the series expansion

$$p_{rp} = (1-\varepsilon)^{n^2} = 1 + 2\log(1-\varepsilon)\frac{1}{n(n+1)} + \mathcal{O}(\frac{1}{n^4}). \tag{10}$$

In [12] the authors considered only the first order $1/n^2$ as a desirable error rate per gate. However, when the output fidelity p_{rp} approaches 1, this approximation is no longer accurate,

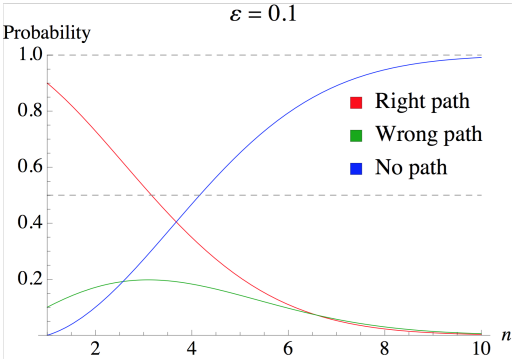

Figure 6 (Color online) Comparison of errors for fixed ε as a function of n.

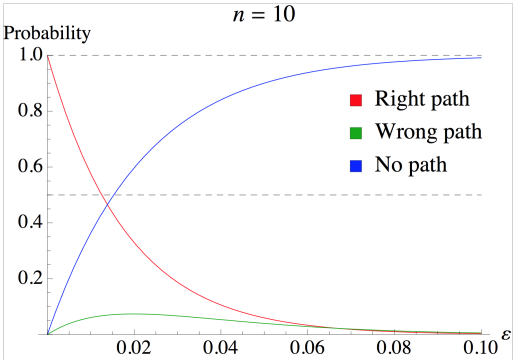

Figure 7 (Color online) Comparison of errors for fixed n as a function of ε.

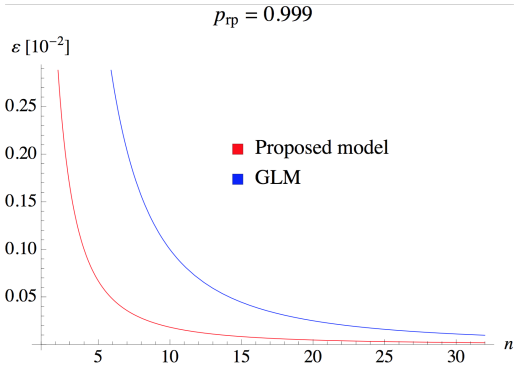

Figure 8 (Color online) Required ε (in dimensionless units of 10^{-2}) as a function of n, for a fixed circuit fidelity. GLM denotes the model proposed in [12].

and higher order terms are important. As mentioned at the end of Sec. 2, inverse polynomial error rates are not good enough in implementing Grover's search with a qRAM-based oracle. In fact, overall error rates of at most $\mathcal{O}(2^{-n/2})$ are essential.

The dominant *no-path* error term poses a fundamental implementation problem, due to lack of oracle information, similar (see Appendix A) to the noise model investigated by Regev and Schiff [25]. If in the future, qRAM designs could be constructed without the presence of such a *no-path* term (i.e. with only *wrong-path* noise), one can attempt error correction to efficiently reduce the error rate. We demonstrate in Appendix B a possible error correction scheme for a simplified *wrong-path* term governed by bit-flip channels, then show however that the scheme is not applicable to our errormodel or to the Regev and Schiff error model [25].

4 Circuit Model

In Fig. 9 we present a circuit description for an $N = 2^3$ qubit bucket brigade qRAM, in which the memory contains only states in the computational basis $\{|0\rangle, |1\rangle\}$. Our circuit is immediately extendable to $N = 2^n$ and closely simulates the physical model proposed in [12].

The circuit description of the bucket brigade addressing scheme accounts for the temporal aspects of the bucket brigade scheme. Namely, since the address qubits are introduced into

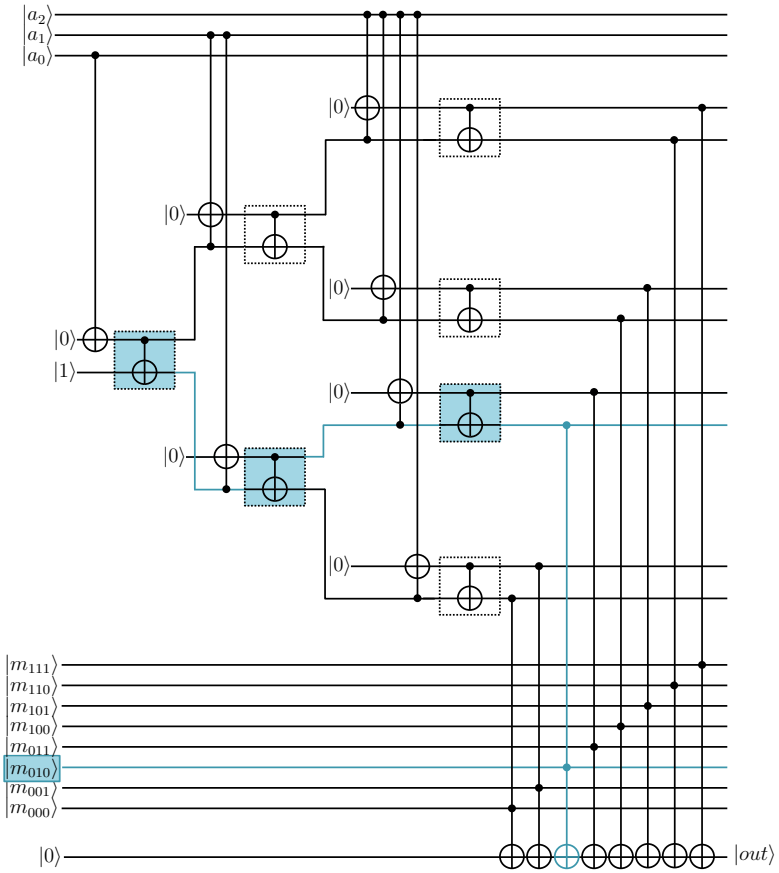

Figure 9 (Color online) Circuit for bucket brigade qRAM. Nodes to the left of the memory cell are *routing nodes*. The dashed squares represents the memory locations. The first layer of nodes immediately to the right of the memory are the *coupling nodes*. Finally, the nodes on the right are the *read out nodes*. A possible input is e.g. $|a_0a_1a_2\rangle = |010\rangle$, for which the circuit reads the memory location m_{010}. The path leading to the location m_{010} is represented in blue colour, and the active routing and readout nodes are highlighted. One could more closely mimic the physical flow of information in the bucket brigade qRAM by adding an additional qubit at each node in the binary tree we see in the diagram. Then, for each $k = 0, \ldots, n-1$, we add an initial controlled-NOT gate to copy a_k to the root node, followed by a series of $\mathcal{O}(2^k)$ controlled-SWAPs that will bring the value of a_k to the unique node in level k defined by the bits $a_0, a_1, \ldots, a_{k-1}$. While this adds exponentially many gates, it does not change the overall gate complexity, and these additional gates only add $\mathcal{O}(k)$ to the depth of the circuit. This also illustrates that the exponential depth implicit in the circuit we describe in the diagram can easily be reduced to polynomial depth by mimicking the ideas presented in the qRAM proposal. We leave the circuit diagram in this simpler form, since it does not affect our arguments in Sections 3 and 5.

the binary tree architecture sequentially, the circuit description should respect this ordering. The input to the circuit are the address qubits $|a_0a_1 \ldots a_n\rangle$. The circuit resembles a binary tree composed of $2^n - 1$ routing nodes, 2^n memory cells and 2^n readout nodes that perform the inverse operations of the routing circuit, used to decouple the qRAM from the address qubits. Additionally, a bus qubit is introduced that interacts with the memory nodes to extract the information stored in the appropriate memory location. It is worth noting that this bus qubit as described may not be physically realistic since it may interact with all the bits in the qRAM. We leave it as such, for simplicity. In practice, if such a non-local qubit is

not feasible, one may either work with a phase shift oracle (as described in Ch. 8 of [19]), or one may use a binary-tree circuit to bring the result of the qRAM look-up to a specific qubit that will be accessed by the quantum algorithm that performs the look-up.

The address qubit $|a_0\rangle$ is used to activate the appropriate branch at the first level of the routing. The address qubit is coupled via a CNOT to an ancillary state prepared in the state $|0\rangle$. This qubit then serves as one of the input qubits along with an additional qubit prepared in the $|1\rangle$ state for the routing node (a CNOT gate with the first qubit as control). Depending on the state of the address qubit, the resulting two-qubit output of the routing node will have a single excited qubit in the $|1\rangle$ state, which we shall call the activated qubit. The activated branch of the tree governs the routing of the subsequent interactions with the address qubits, playing the role of the routing atom in the case of the bucket brigade outlined in [12].

The two qubits at the exit of the level-0 routing node serve as inputs to the second register of the two level-1 routing nodes. These qubits control which of the routing nodes are activated at the next level of the qRAM binary tree architecture. Namely, the qubit that is excited in the $|1\rangle$ state allows for the coupling between the address qubit and an introduced $|0\rangle$ state ancilla via a Toffoli gate. Therefore the input to the active routing node is either $|01\rangle$ or $|11\rangle$ depending on the state of the address qubit $|a_1\rangle$. Effectively, the routing operation given by a CNOT gate activates a branch of the tree. For the node that is non-active, the state at the output of the previous level is $|0\rangle$, meaning the Toffoli is not activated and the resulting input and output state to the routing node remains $|00\rangle$. Therefore, after two routing node levels, the output of the routing qubits is composed of 2^2 qubits, with only a single branch being excited depending on the state of the first two address qubits $|a_0a_1\rangle$. Therefore, this corresponds to an isometry:

$$|00\rangle_{a_0a_1} \to |0001\rangle, \quad |01\rangle_{a_0a_1} \to |0010\rangle, \quad |10\rangle_{a_0a_1} \to |0100\rangle, \quad |11\rangle_{a_0a_1} \to |1000\rangle, \tag{11}$$

where the excited output qubits in the $|1\rangle$ state represent an active physical path for the subsequent qRAM operations. This procedure is repeated for n levels, where at the k-th level there are 2^k Toffoli gates and routing nodes. The 2^k Toffoli gates are required to route of the address qubit $|a_k\rangle$ through the previous k levels and the routing node establish the output states in order to route the subsequent address qubits. Since such a circuit performs the appropriate unitary mapping of the address qubits for all computational basis state inputs, by linearity it will extend to all superpositions of input address qubits. An example of the routing procedure for a three-qubit input address state $|010\rangle$ is presented in Fig. 9, where the blue highlighted nodes correspond to the activated nodes.

After the completion of the n routing node levels, memory readout is performed. The reading is performed by introducing a bus qubit prepared in the $|0\rangle$ state which is the target of 2^n Toffoli gates. Each of the 2^n qubits at the output of the qRAM routing nodes pair with one of the memory cells to serve as control qubits for the Toffoli gates. Since only a single output qubit from the routing scheme is activated, only a single Toffoli gate couples with a memory location to the bus qubit. The bus qubit is represented by the bottom qubit in Fig. 9 while the 2^3 memory qubits are represented between the qRAM routing architecture and the bus qubit.

Having completed the coupling of the address qubit, the state of the qRAM routing qubits must be decoupled from the address and bus qubits. Each of the gates from the routing circuit are performed in reverse order, which corresponds to performing the inverse unitary coupling transformation between the address qubits and the routing qubits. The resulting state couples the address qubits with the corresponding memory qubit and has decouples the routing qubits to their input ancillary states.

5 Error Correction

The results from Sec. 3 motivate the need for quantum error correction to be implemented at each node in order to protect against errors that may cause detrimental faults in path information.

5.1 Imposing a quantum error correcting code

In choosing a quantum error correcting code (QECC) to protect the path information that is stored in each node, it is essential to choose an encoding that can be implemented fault-tolerantly, to allow for the generalization to large computational systems. Moreover, the QECC should be chosen such that it can naturally be incorporated from the quantum computer that is accessing the qRAM. In order to analyze the desired error correction properties of a bucket brigade qRAM architecture we consider the circuit presented in Fig. 9. The key gate components at each site are the CNOT and Toffoli gates.

The most natural construction of a QECC that can implement such operations with minimal overhead would be the 15-qubit Reed-Muller code. The reason for choosing such a code would be that decomposing the gate operations in the routing circuit as a sequence of CNOT and Toffoli gates has the advantage that each of these gates can be implemented in a transversal manner. Transversality is defined as the ability to implement a logical gate by applying physical gates that have support at at most a single location per encoded codeblock: it is the most natural way to guarantee fault-tolerance. However, if the quantum computing device leads more naturally to another form of quantum error correction encoding, methods such as state distillation or other schemes for universal fault-tolerance can be used [8, 24, 4, 18, 2].

The focus of many fault-tolerant implementations are through the CSS code construction [28, 10, 29]. A CSS quantum code is constructed using two classical error correcting codes, each individually used to address X and Z type errors. Given that any quantum error can be decomposed in terms of a linear combination of Pauli operators, developing an error correcting code that can address both types of errors will be sufficient for the construction of a QECC.

Let $\mathcal{C}_X$ be a classical error correcting code of length n that has the associated parity check matrix H_X, where each 0 in the parity check matrix of the classical code is replaced by the two-dimensional identity matrix I and each 1 in the classical parity check matrix is replaced by the Pauli X operator. Similarly, let $\mathcal{C}_Z$ be a second classical error correcting code of length n with an associated parity check matrix H_Z, where each 1 in the classical parity check matrix has been now replaced by the Pauli Z operator. If $\mathcal{C}_X^{\perp} \subseteq \mathcal{C}_Z$ then by combining the stabilizers generated from the parity check matrices of both codes, H_X and H_Z, the resulting stabilizer code forms a QECC. The number of physical qubits in the code is n, the number of logical qubits is given by $k_X + k_Z - n$, where k_i is the number of logical states in the given classical code i and the distance of the code is at least the minimum of the distance of the two classical codes. One of the many appealing features of the CSS code construction is the transversality of the CNOT gate, a feature of the X and Z stabilizers being independent. A particular example of a CSS code is the 15-qubit Reed-Muller code mentioned above.

5.2 Number of activations in a CSS code

In the implementation of Giovanetti *et al.* [12], one of the primary advantages is the number of gate activations that are needed per level of the bucket brigade scheme. More simply, a

CNOT (Toffoli) gate in their scheme is "activated" only when the control qubit(s) is (are) in the state $|1\rangle$. Since only one register is in such a state at a given level, the total number of activations can thus be kept low. In a physical implementation, this is relevant as an activated path may represent the presence of a physical excitation without which no physical process occurs, therefore one can think of these non-activated gates as in fact being the identity operation. However, such an advantage vanishes when imposing a CSS code in order to protect from errors due to the symmetry in the number of $|0\rangle$ and $|1\rangle$ states the logical states encoding the path information.

In the CSS code construction, two classical codes were taken to form a QECC. Therefore, given some codeword of the classical code $c \in \mathcal{C}_Z$, the equivalent quantum state written out in computation basis $|c\rangle$ must be stabilized by the Z generators of the code, by definition of being a codeword of the classical code $\mathcal{C}_Z$. However, in order to be a logical state of the CSS code, it must also be stabilized by the elements of the group generated by the X stabilizers. Therefore, the codestate will be the superposition of the application of all X stabilizers upon $|c\rangle$,

$$|c + \mathcal{C}_Z^{\perp}\rangle = \sum_{x \in \mathcal{C}_Z^{\perp}} |c + x\rangle = \frac{1}{2^{n-k_X}} \prod_i (I^{\otimes n} + S_{X,i})|c\rangle, \tag{12}$$

where $\{S_{X,i}\}$ are the generators of the X stabilizer group, equivalently given by the rows of the parity check matrix H_X.

Consider the form of Eq. (12), given the state $|c\rangle$ written in the computational basis, the action of the operator $(I^{\otimes n} + S_{X,1})$ will be the equally weighted superposition of the state $|c\rangle$ and $S_{X,1}|c\rangle$, which will differ at the location where $S_{X,1}$ has a Pauli X in its description. Therefore, at these locations half of the states in the superposition will have a physical $|0\rangle$ state and half will have a physical $|1\rangle$ state. Then acting upon the state with the operator $(I^{\otimes n} + S_{X,2})$ will have the same effect on all the states in the superposition, with now an even number of physical $|0\rangle$ and $|1\rangle$ states occurring at location with Pauli X in $S_{X,2}$. Repeating this for all X generators, any location with a X operator in one of the stabilizers will necessarily have half of the states in the superposition in each of the physical basis states. In order for the code to protect against any arbitrary single qubit error, each physical qubit must be protected by at least one X stabilizer operator with support the given location, otherwise it would be vulnerable to a single Z error at this location. As such, all relevant CSS codesstate will have an equal number of each of the physical basis states when writing out the expansion of the state in the computational basis.

In a physical implementation, such as that of Giovanetti *et al.* [12], a qubit in the state $|1\rangle$ represents an activated physical process, and as such the advantage of the bucket brigade scheme is that the number of such processes are kept low. However, due to the symmetry in the number of activations that must exist in both the logical ground and excited states, this advantage no longer exists when considering CSS codes. More generally, non-symmetric codes, that is codes where the logical $|0\rangle$ state and logical $|1\rangle$ have a differing number of physical states in the excited state $|1\rangle$, are not desirable for the purposes of error correction as they will be more susceptible to Z errors. The three-qubit repetition code is an extreme example of such a property.

In principle, for the physical error model discussed in Section 3, one can envision using the detection of a photon lost in the routing structure as a means to correct for *no-path* errors (see Fig. 5). However, detecting the exact node at which a photon was lost reveals path information about the state being read by the qRAM (since the previous node in the routing structure would have necessarily been activated by the address qubits) which leads

to a loss of coherence in the system. Therefore, any photon detection has to identify the level at which the photon was lost, while not revealing exactly where. It is hard to envisage a practical means for experimentally realizing a photon detection with this property (for example, by somehow symmetrizing the loss of the photon across the exponentially many nodes at a given level). Furthermore, even if this is achieved, one still faces the problem that the lost photon contained path information. Thus, destroying the photon with this path information is equivalent to a dephasing error leading to a further loss of coherence.

In conclusion, if one encodes each node of the bucket brigade qRAM in an error correcting code, then all nodes of the circuit are activated at a physical level, and essentially the qRAM architecture becomes equivalent to a fanout architecture. Even it the latter case, designing a good quantum error correcting code is highly non-trivial. An important issue is that the syndrome measurement should not reveal any information whatsoever about the physical location of the nodes affected by errors. Otherwise, path information is being revealed, which decoheres the system.

6 Conclusions and open questions

We analyzed the robustness of the bucket brigade qRAM scheme introduced in [13, 12] under an optimistic error model. The primary advantage of the bucket brigade scheme is the need for a polynomial in n (rather than exponential) number of gate activations per memory reading. Yet, we give evidence for the hypothesis that for realistic error models, whenever the qRAM is used as a oracle for quantum searching, its error rate per gate has to scale as $o(2^{-n/2})$. Such an error rate is exponentially smaller than the error $\mathcal{O}(1/n^2)$ proposed in [12] (which is sufficient for algorithms with low query complexity), motivating the need for quantum error correction.

We argued that using traditional error correcting techniques offsets the main advantage of the bucket brigade scheme when used with algorithms that make super-polynomially many oracle queries. Since each component of the routing architecture has to be actively error corrected in order to protect against detrimental faults, the overall scheme requires an exponential number of physical gate activations, even if the number of logical gate activations remains polynomial.

An interesting open question is the existence of a realistic architecture-specific error correction technique that could recover the polynomial number of physical gate activations of the routing scheme while still guaranteeing fault-tolerance. For example, if one tries to use an error correction mechanism whereby one only uses multi-qubit code states along the active path, then one has the problem of extracting syndromes and applying corrections in a way that does not identify which path has the non-trivial syndromes (since such information would lead to decoherence). If in this case, for example, one attempts to extract the syndrome without leaving a trace of which node in a given level it came from, then the problem seems at least as challenging as implementing a reliable qRAM.

Moreover, it would be interesting to investigate whether the requirement for a super-polynomial suppression of the error rate is a characteristic of quantum searching algorithms or a more general feature of query complexity with faulty oracles.

Acknowledgements. We thank Daniel Gottesman, Stacey Jeffery, Seth Lloyd, Lorenzo Maccone and Matteo Mariantoni for numerous insightful discussions and suggestions. We thank Seth Lloyd for pointing out to us nice examples of important algorithms that only use polynomially many queries. The authors were supported in part by NSERC, CIFAR,

FQRNT, OGS, CryptoWorks21 and Industry Canada. IQC and PI are supported in part by the Government of Canada and the Province of Ontario. S.A. is grateful to Ronald de Wolf's ERC Consolidator Grant QPROGRESS and Michele Mosca for support from NSERC.

References

1. Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM J. Comput.*, 37(1):210–239, April 2007.
2. Jonas T. Anderson, Guillaume Duclos-Cianci, and David Poulin. Fault-tolerant conversion between the steane and reed-muller quantum codes. *Phys. Rev. Lett.*, 113:080501, Aug 2014.
3. Srinivasan Arunachalam. Quantum speed-ups for boolean satisfiability and derivative-free optimization. Master's Thesis, University of Waterloo, 2014.
4. H. Bombin. Optimal transversal gates under geometric constraints. e-print arXiv:1311.0879 [quant-ph].
5. H. Bombin and M. A. Martin-Delgado. Topological quantum distillation. *Phys. Rev. Lett.*, 97:180501, Oct 2006.
6. Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. *SIGACT News*, 28(2):14–19, June 1997.
7. Sergey Bravyi and Jeongwan Haah. Magic-state distillation with low overhead. *Phys. Rev. A*, 86:052329, Nov 2012.
8. Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Phys. Rev. A*, 71:022316, Feb 2005.
9. Harry Buhrman, Ilan Newman, Hein Roehrig, and Ronald de Wolf. Robust polynomials and quantum algorithms. e-print arXiv:quant-ph/0309220.
10. A. R. Calderbank and Peter W. Shor. Good quantum error-correcting codes exist. *Phys. Rev. A*, 54:1098–1105, Aug 1996.
11. Andrew M. Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A. Spielman. Exponential algorithmic speedup by a quantum walk. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC'03, pages 59–68, New York, NY, USA, 2003. ACM.
12. Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Architectures for a quantum random access memory. *Phys. Rev. A*, 78:052310, Nov 2008.
13. Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Phys. Rev. Lett.*, 100:160501, Apr 2008.
14. Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. e-print arXiv:quant-ph/0208112.
15. Fang-Yu Hong, Yang Xiang, Zhi-Yan Zhu, Li-zhen Jiang, and Liang-neng Wu. Robust quantum random access memory. *Phys. Rev. A*, 86:010306, Jul 2012.
16. Peter Høyer, Michele Mosca, and Ronald de Wolf. Quantum search on bounded-error inputs. In JosC.M. Baeten, JanKarel Lenstra, Joachim Parrow, and GerhardJ. Woeginger, editors, *Automata, Languages and Programming*, volume 2719 of *Lecture Notes in Computer Science*, pages 291–299. Springer Berlin Heidelberg, 2003.
17. R. C. Jaeger and T. N. Blalock. *Microelectronic Circuit Design*. McGraw-Hill, Dubuque, 2003.
18. Tomas Jochym-O'Connor and Raymond Laflamme. Using concatenated quantum codes for universal fault-tolerant quantum gates. *Phys. Rev. Lett.*, 112:010505, Jan 2014.
19. Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An Introduction to Quantum Computing*. Oxford University Press, 2007.
20. Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput.*, 35(1):170–188, July 2005.

21 Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. e-print arXiv:1307.0411 [quant-ph].

22 Michele Mosca and Phillip Kaye. Quantum networks for generating arbitrary quantum states. In *Optical Fiber Communication Conference and International Conference on Quantum Information*, page PB28. Optical Society of America, 2001.

23 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information (Cambridge Series on Information and the Natural Sciences)*. Cambridge University Press, 1 edition, January 2004.

24 Adam Paetznick and Ben W. Reichardt. Universal fault-tolerant quantum computation with only transversal gates and error correction. *Phys. Rev. Lett.*, 111:090505, Aug 2013.

25 Oded Regev and Liron Schiff. Impossibility of a quantum speed-up with a faulty oracle. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Part I*, ICALP'08, pages 773–781, Berlin, Heidelberg, 2008. Springer-Verlag.

26 A. S. Sedra and K. C. Smith. *Microelectronic Circuits*. Oxford Press, New York, 1998.

27 Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:R2493–R2496, Oct 1995.

28 A. M. Steane. Error correcting codes in quantum theory. *Phys. Rev. Lett.*, 77(5):793–797, Jul 1996.

29 Andrew M. Steane. Multiple-particle interference and quantum error correction. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 452(1954):2551–2577, 1996.

A A simple decoherence model

Let us consider the error model considered in [25],

$$\mathcal{R}_p(\rho) := (1-p)\hat{O}\rho\hat{O}^\dagger + p\rho, \tag{13}$$

with $\hat{O}$ denoting the perfect oracle for quantum searching and let us define

$$\mathcal{D}_q(\rho) := (1-q)\rho + q\vec{X}\rho\vec{X}^\dagger \tag{14}$$

as the multi-qubit bit-flip channel where $\vec{X}$ is a shorthand notation for a tensor product of σ_X bit-flip operators acting on some fixed subset of the oracle qubits. The proof technique presented below for $\mathcal{D}_q$ also applies to the case of multi-qubit dephasing channels).

The error model proposed in this paper (see Eq. (9)) is

$$O(\rho) := p_{rp}\hat{O}\rho\hat{O}^\dagger + p_{wp}\mathcal{E}_{wp}(\rho) + p_{np}\mathcal{E}_{np}(\rho). \tag{15}$$

We show that the composition $\mathcal{R}_p \circ \mathcal{D}_q$ resembles (although it is not exactly the same) our error model Eq. (9), for suitable chosen p and q. It follows immediately that $\Omega(N)$ lower bound for the searching algorithm considered in [25] is also a lower bound for the composition $\mathcal{R}_p \circ \mathcal{D}_q$, since channel composition cannot decrease the query complexity (one can simply incorporate $\mathcal{D}_q$ into an appropriate unitary for the $\mathcal{R}_p$ algorithm).

A simple calculation yields:

$$\begin{aligned}
\mathcal{R}_p \circ \mathcal{D}_q(\rho) &= (1-p)\hat{O}\mathcal{D}_q(\rho)\hat{O}^\dagger + p\mathcal{D}_q(\rho) \\
&= (1-p)(1-q)\hat{O}\rho\hat{O}^\dagger + (1-p)q\hat{O}(\vec{X}\rho\vec{X}^\dagger)\hat{O}^\dagger + p(1-q)\rho + pq\vec{X}\rho\vec{X}^\dagger \\
&= (1-p)(1-q)\hat{O}\rho\hat{O}^\dagger + (1-p)q\hat{O}(\vec{X}\rho\vec{X}^\dagger)\hat{O}^\dagger + p\mathcal{D}_q(\rho).
\end{aligned} \tag{16}$$

We now identify the coefficients in Eq. (15) and Eq. (16)

$$\begin{cases} p_{rp} = (1-p)(1-q) \\ p_{wp} = (1-p)q \\ p_{np} = p, \end{cases} \tag{17}$$

and note that for any given probabilities p_{rp}, p_{wp}, p_{np} satisfying $p_{rp} + p_{wp} + p_{np} = 1$, the system of equations Eq. (17) has the solution

$$\begin{cases} p = p_{np} \\ q = \dfrac{p_{wp}}{p_{wp} + p_{rp}}. \end{cases} \tag{18}$$

We can therefore write

$$\mathcal{R}_p \circ \mathcal{D}_q(\rho) = p_{rp}\hat{O}\rho\hat{O}^\dagger + p_{wp}\hat{O}(\vec{X}\rho\vec{X}^\dagger)\hat{O}^\dagger + p_{np}\mathcal{D}_q(\rho). \tag{19}$$

Comparing Eq. (15) and Eq. (19), we observe that the term $\hat{O}(\vec{X}\rho\vec{X}^\dagger)\hat{O}^\dagger$ is very similar to our wrong path term $\mathcal{E}_{wp}(\rho)$ (the error that corresponds to reading out an incorrect memory location). The last term term $\mathcal{D}_q(\rho)$ in Eq. (19) is not of the form of our no-path error term $\mathcal{E}_{np}(\rho)$, as the latter consists of depolarizing channels of different strengths depending on the position of the address qubit (i.e., the qubits are affected in decreasing order of significance, that is, the first qubit is affected the most, whilst the last one the least). However, $\mathcal{D}_q(\rho)$ is a decohering term, which seems to be a "weaker" form of noise than $\mathcal{E}_{np}(\rho)$. We showed above that even with this weaker decoherence term the quadratic speedup of any searching algorithm is lost. Therefore we have strong reasons to believe that adding a stronger decoherence term will not lower the quantum query complexity for the quantum searching problem. A rigorous proof of this conjecture remains an open problem.

B Error correction schemes

B.1 Correcting simple bit-flip errors

We show below that for a qRAM governed by a toy error model of the form

$$O(\rho) = (1-p)\hat{O}\rho\hat{O}^\dagger + p\hat{O}(\vec{X}\rho\vec{X}^\dagger)\hat{O}^\dagger, \tag{20}$$

the query error rate can be made arbitrarily small by using quantum error correction. Here $\hat{O}$ denotes the perfect oracle and $\vec{X}$ represents a multi-qubit bit-flip channel (a tensor product of individual bit-flip operators acting on an arbitrary subset of qubits). While such error models are not realistic for the architecture presented in this work, it may be that future designs allow for simpler error propagation. Such schemes could benefit from quantum error correction to sufficiently reduce their error rate to enable Grover search.

As Grover's algorithm requires $\mathcal{O}(\sqrt{N})$ steps, one desires a target logical error rate of $\delta = \mathcal{O}(1/\sqrt{N})$. Since the faulty oracle has an error model that consists of a bit flip channel followed by the perfect oracle call, one can use a quantum error correcting code and apply the oracle in parallel along the qubits composing the code. The parallelism of the oracle calls mimics majority counting and allows for error correction to be performed between logical oracle call steps. For simplicity, we provide an example that corrects against bit flip errors only using the repetition code, however such an analysis could be extended to correct for

phase flips using code families such as the color codes [5], higher dimensional Shor codes [27], or triorthogonal codes [7].

For example, consider an oracle of the form $|a\rangle|b\rangle \to |a\rangle|b \oplus f(a)\rangle$, where $a,\ b \in \{0,1\}$. A logical oracle call that uses an n-qubit repetition code behaves as follows for states in the computational basis:

$$|a\rangle|b\rangle \xrightarrow{V} |a\rangle^{\otimes n}|b\rangle^{\otimes n} \xrightarrow{\hat{O}^{\otimes n}} |a\rangle^{\otimes n}|b \oplus f(a)\rangle^{\otimes n}, \tag{21}$$

where V denotes the isometric encoding. Therefore, given a repetition code of length d, the code corrects for all errors up to $d/2-1$ physical bit flips by majority counting, using non-destructive Z-type stabilizer measurements. Therefore, the logical error rate becomes $p_L = p^{d/2}$. Choosing d large enough allows the logical error rate to satisfy $p_L = p^{d/2} < \delta$, where δ is the desired target fidelity. Therefore

$$d > \frac{2\log\delta}{\log p} = \frac{2\log\left(1/\sqrt{N}\right)}{\log p} = \frac{n}{\log\left(1/p\right)}. \tag{22}$$

Each of the n address qubits that serve as input to the oracle call must be encoded into a repetition code of length d. Hence, the total number of oracle calls for the complete Grover search algorithm is $\mathcal{O}(nd\sqrt{N}) = \mathcal{O}(n^2\sqrt{N}) = \mathcal{O}(\sqrt{N}(\log N)^2)$. As such, there is a logarithmic penalty for error correction, yet the scaling is not linear as in the error model of Regev and Schiff [25].

B.2 The failure of repetition codes for Regev and Schiff error model

The above error correction scheme is not applicable to the error model presented in [25], described by $\mathcal{R}_p(\rho) = (1-p)\hat{O}\rho\hat{O}^\dagger + p\rho$, since the failure of an oracle call can lead to an uncorrectable error, as demonstrated below. Consider the following example of the 3-qubit repetition code, where rather than all three oracles calls succeeding, the oracle call on the first set of qubits fails. The computational states evolve as:

$$|000\rangle|000\rangle \xrightarrow{\hat{O}_2\hat{O}_3} |000\rangle|0f(0)f(0)\rangle \tag{23}$$

$$|111\rangle|000\rangle \xrightarrow{\hat{O}_2\hat{O}_3} |111\rangle|0f(1)f(1)\rangle. \tag{24}$$

Consider the action of such a faulty oracle on the encoded state $(|000\rangle + |111\rangle)/\sqrt{2}$, for $f(0) = 0$ and $f(1) = 1$. The resulting mapping is

$$\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) \otimes |000\rangle \xrightarrow{\hat{O}_2\hat{O}_3} \frac{1}{\sqrt{2}}(|000\rangle|000\rangle + |111\rangle|011\rangle). \tag{25}$$

The syndrome check operators for the repetition code are the parity check operators $\{Z_1Z_2, Z_2Z_3\}$. They are used to determine if an oracle call has failed by measuring the ancilla qubits. However, the measurement collapses the state to either $|000\rangle|000\rangle$ or $|111\rangle|011\rangle$. Upon applying the appropriate correction based on the measured syndromes, the resulting state becomes either $|000\rangle|000\rangle$ or $|111\rangle|111\rangle$. Therefore, the logical oracle call has failed, since the correct result must yield the superposition $(|000\rangle|000\rangle + |111\rangle|111\rangle)/\sqrt{2}$.

As expected, the error correction properties of the repetition code are not in violation of the results of Ref. [25], which state that a linear number of noisy black-box oracle calls are required, even with the addition of error correction.

B.3 The failure of repetition codes for our error model

Consider the oracle error model:

$$O(\rho) := p_{rp}\hat{O}\rho\hat{O}^{\dagger} + p_{wp}\mathcal{E}_{wp}(\rho) + p_{np}\mathcal{E}_{np}(\rho), \tag{26}$$

where $\hat{O}$ is the perfect oracle call while $\mathcal{E}_{wp}(\rho)$ and $\mathcal{E}_{np}(\rho)$ are the *wrong path* and *no-path* terms, respectively. We model the *wrong path* term as a convex combinations of bit-flip channels followed by perfect oracle calls. An example of one of those terms is the second term in Equation 20. We model the *no-path* term as taking any input state and mapping it to a fixed state $|g\rangle$, which represents the loss of a qubit to be replaced by any fixed ancillary state. It should be noted that in the *no-path* case, the readout ancilla state does not change. Consider the action of the noisy channel on the five-qubit repetition code. Each instance of the channel has a certain probability of failure given by the associated weights. Focusing on one particular instance where the first address photon is lost and the second is affected by a bit flip, the resulting mapping on the computational basis states is given by:

$$|00000\rangle|00000\rangle \longrightarrow |g1000\rangle|0f(1)f(0)f(0)f(0)\rangle \tag{27}$$

$$|11111\rangle|00000\rangle \longrightarrow |g0111\rangle|0f(0)f(1)f(1)f(1)\rangle. \tag{28}$$

Again choosing $f(0) = 0$ and $f(1) = 1$, a superposition of input states in the computational basis evolves as

$$\frac{1}{2}\mathrm{P}\left[(|00000\rangle + |11111\rangle) \otimes |00000\rangle\right] \longrightarrow \frac{1}{2}\left(\mathrm{P}\left[|g1000\rangle \otimes |01000\rangle\right] + \mathrm{P}\left[|g0111\rangle \otimes |00111\rangle\right]\right), \tag{29}$$

where P[•] denotes the projector onto its argument. The measurement of the stabilizers of the 5-qubit code on the ancillary states results in the collapse of the state into one of two terms depending on the syndrome measured. Note that the *no-path* term is the term that destroys coherence, similarly to the error term in the Regev and Schiff model [25].