Quantum Random Access Memory

Aaron Green and Emily Kaplitz

Fall 2019

Abstract

qRAM is a necessary component for larger quantum computers. In this paper, we discuss the background, usage, and practicality of qRAM. We also show the importance of qRAM in quantum algorithms. Lastly, we will study how qRAM might be implemented in the future.

1 Introduction

Many algorithms throughout different fields of computing require the storage and use of memory. A ubiquitous form of memory storage in classical computing is random access memory (RAM). Similar to classical computing, many quantum computing algorithms also depend on the use of stored states. There are two main ways to achieve this. First, an algorithm that combines classical and quantum computing, only using quantum states for very short periods of time. Second, using quantum random access memory (qRAM) [4]. This paper surveys the background, usage, and practicality of qRAM, including why it is useful and how it can be implemented.

qRAM is necessary for larger quantum computers that may arise in the future, rather than the current ones which are much smaller in size, and will function very similarly to classical RAM. It is composed of the same three components, Memory, an Input Register, and an Output Register. However unlike classical RAM, qRAM will use qubits for its Input and Output Registers, and possibly its memory array as well [10].

In section 2, we will go over the background of qRAM. This includes going over classical RAM, the development of qRAM, and typical architectures and

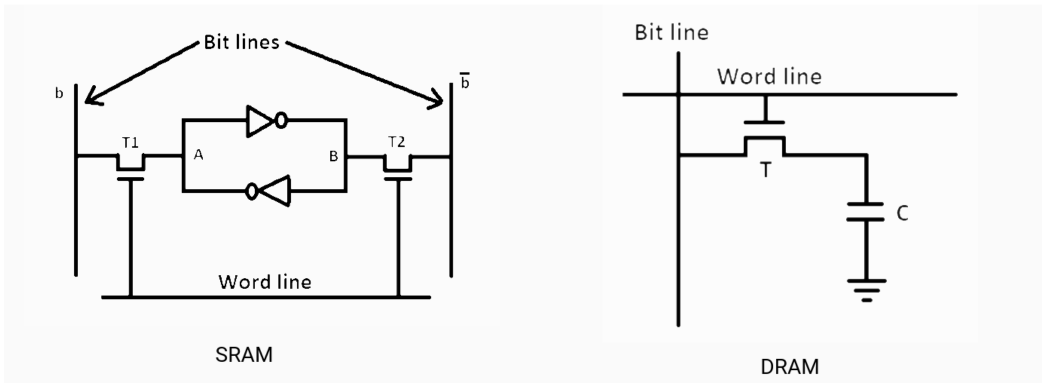

Figure 1: *SRAM and DRAM Circuit designs [1]*

their issues. In section 3, we will review the importance of qRAM by studying three different algorithms. For our last section, we will discuss the candidate systems for implementation of the architecture Bucket Brigade, state of the art advancements, recent experiments, and open questions.

2 Background

2.1 Classical RAM

Memory consists of cells that can store information [3]. Each cell has an address, or wordline, to find the cell location. A memory with n cells has an address of 0 to $n-1$ bits. All cells contain the same amount of bits for the address. Memory that can be read and written is called Random Access Memory, or RAM. RAM is made up of three parts: Memory, Input Register and Output Register. There are two types of RAM: static and dynamic. Static RAM, SRAM, retains data bits in memory as long as power is being supplied to the computer. It is used in cache which is small fast memory. SRAM has faster access then DRAM, but it is more expensive. DRAM stores bits in only a capacitor and transistor making it simple and cheap. However, it needs to be refreshed. Both RAM architectures have a problem with leakage of memory, however, different techniques are used to prevent memory from being lost. Circuits for SRAM and DRAM can be found in figure 1.

2.2 Development of qRAM

In order to access memory in a quantum system there needs to be a quantum equivalent to classical RAM. This quantum equivalent is referred to as qRAM. qRAM uses the same three components as classical RAM: Memory Array, Input Register, and Output Register. In qRAM, the Input and Output Registers are composed of qubits, while the Memory Array can be classical or quantum depending on the application.

An effective implementation of qRAM can create exponential speedup for many algorithms as well as being required for the implementation of other algorithms. qRAM also leads to new quantum computation primitives for use in quatum cryptography and quantum networking.

Although RAM and qRAM are essential for developing efficient, modern algorithms, both memory regimes are exponentially expensive when accessing one of 2^n memory slots where n represents the number of bits in a memory address. This need for so many slots can cause a high decoherance rate for qRAM which has slowed its development. Previously, for a d-dimensional lattice, many architectures required $O(N^{\frac{1}{d}})$ operations to access a memory location.

2.3 Typical RAM Architectures and Issues

qRAM works by using quantum superposition to perform memory access [10]. In order to access a superposition of the memory cells the address register, a, must contain a superpostion of the address. The qRAM returns a superpostition of data in a data register, d, correlated to the address register. Let D_j be the content of the jth memory cell. This is shown in the formula below.

$$\sum_j \psi_j |j\rangle_a \rightarrow \sum_j \psi_j |j\rangle_a |D_j\rangle_j$$

In order to understand qRAM, let us first look at RAM in the classical sense through the Basic RAM Addressing Scheme [10]. Let N be the number of memory cells. These cells are placed at the end of a bifurcation graph with n levels, as shown in figure 3. If we want to find the value of the jth bit in the address register we do it by following a path to the jth level of the graph. We do this by taking the left path if we read in a 0 and the right path if we read in a 1. Each of the N possible values of the address register is a unique path to a memory cell. The classical implementation requires

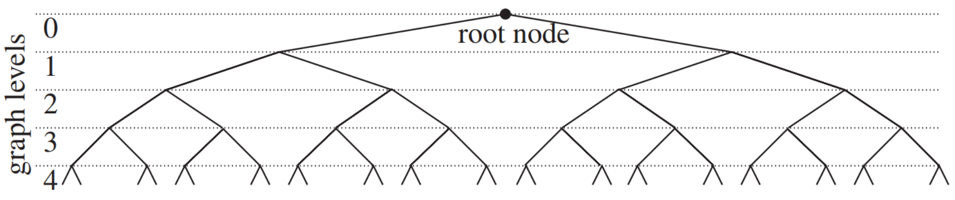

Figure 2: *Bifurcation Graph used in RAM Addressing [10]*

using a transistor at each of the two paths at each node of the graph. Thus, each address bit controls the transistors at a particular level of the graph. This leads to an exponential number of transistors being activated for each memory call.

Now lets consider using this scheme in terms of quantum computing [10]. n qubits of the address register control n quantum control lines. These act on the entire level of the bifurcation graph. At each branch we avoid the signals along the left path if we read in a 0 and avoid the signals along the right path if we read in a 1. Each binary address corresponds to a set of switches that find a unique path through the graph associated with that address. The connected superposition of the address is entangled with a set of switches that find a superposition of paths through the graph. $O(N)$ quantum transistors are entangled with the desired memory. A quantum bus is used in the memory call to follow the super position of paths through the graph. The state of the bus changes according to the quantum information in the memory slot through a Controlled-NOT gate. Disconnection from the bus's position in the address register is done by returning to the root by the path taken.

This is a very impractical way of implementing qRAM as to query a superposition of memory cells, the address qubits are entangled with $O(N)$ switches or quantum gates. This super position is susceptible to decoherence and thus requires quantum error correction whenever the rate is bigger than 2^{-n} which can be very costly. We will discuss better architectures in later sections.

2.3.1 Quantum Bus

A Quantum Bus is a signal that is sent back and forth in a binary tree that represents the RAM [9]. The bus can be routed through every through every possible path of the binary tree. It is characterized by some internal degree of freedom that can be used to store and process information. If each memory cell contains a single bit of information, the quantum bus is a single qubit with a copy operation being a single Controlled-NOT gate. Each memory cell can contain d bits of information. We can transfer information bit by bit with a two dimensional quantum bus that is repeated d times.

2.3.2 Fanout

In the Fanout scheme [9], the index register gives the direction to reach the memory cell needed. The index register is a binary string. Each bit tells which direction to take at a bifurcation of the tree. If a bit in the string is a 0, then all the switches will point up. Fanout schemes are common in classical RAM chips. In a classical implementation, RAM Chips that translate schemes into electronic circuits where the switches of the binary tree are replaced by a pair of transistors. The last few bits of the index register are connected to an exponentially large number of transistors. This does not translate well to the quantum setting because of quantum coherence. If we were to turn this scheme into qRAM, we would do this by turning it into a reversible process and making sure that quantum coherence is preserved. The kth qubit still needs to control 2^k bifurcations of the tree. The gates required to control this scheme is costly in qRAM because of decoherence and noise. If a single gate out of the entire scheme is decohered, the fidelity of the system is reduced by a factor of two [10]. However, this scheme can identify failures as the memory gets large.

In order to use this scheme, a quantum bus is implemented. At a bifurcation in the path of the tree, the direction is chosen using quantum gates. To change this into quantum we would do a unitary operation controlled by a qubit from the Input Register. The amount of transformations along the tree convert a binary value from the index register into a position of the bus. The bus follows more than one path in superposition and interacts with all of the memory cells relative to the path if the index register contains a superposition of multiple addresses. After accessing the memory cell, the quantum coherence correlation needs to be undone. This is done by running the translation that was first done in reverse. However, this means that the bus can not be sent back through the binary tree as each bifurcations decorrelates the value of the control qubit in the bus.

2.3.3 Bucket Brigade

About 10 years ago, Biamonte, et al. developed a new architecture for qRAM called "Bucket Brigade" which lowers that complexity to $O(logN)$ [9], exponentially decreasing access complexity.
This architecture can be applied to both classical and quantum systems, although it is unnecessary for classical systems due to the already existing scaled systems which are not as efficient, but do not need to be. Instead of encountering binary switches of "0" or "1" going down a path to memory locations, trits are used. These trits can take the values of "wait", "left", and "right". Initially, each trit is in the "wait" state. As bits representing location are encountered, each trit in a "wait" state will change to "left" or "right" based on the value of that bit being 0 or 1. Translating this to a quantum system, we can take the values of "wait", "left", and "right" to simply be quantum states $|wait\rangle$, $|left\rangle$, and $|right\rangle$.
At first glance, this architecture is very similar to the Fanout scheme in the way that it involves a tree of paths that determines which memory location is accessed to read from or write to. The core difference in this architecture is the two phase operation of Bucket Brigade. Rather than accessing and retrieving/writing information in one step, Bucket Brigade initializes its path before sending in a Quantum Bus. As mentioned previously, the coherence complexity of accessing one memory cell using Bucket Brigade is $O(logN)$ because only that many qubits are not in the $|wait\rangle$ state. Therefore the complexity of accessing r memory cells is $O(rlogN)$ [10]. In this architecture, if a fraction ϵ of gates are decohered, then the average fidelity of the system is $O(1 - \epsilon logN)$ [10]. This is much better than the fidelity of the Fanout scheme.

3 Uses in Algorithms

3.1 Quantum Searching on a Classical Database [6]

The quantum searching on a classical database problem is as follows: we have an unstructured list or database and we want to find a particular element in that list. Let $N = 2^n$ be the amount of items each with a length of l bits. Our database looks like $\{d_1, ...d_N\}$. We want to find a particular element s in the list. For a quantum computer to solve this problem, we need two units a CPU and memory. The CPU consists of four registers: an n qubit index register initialized to $|0\rangle$, an l qubit register initialized to the state $|s\rangle$ that remains in that state for all of the computation, an l qubit data register initialized to $|0\rangle$, and a 1 qubit register initialized to $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$. The memory unit can be implemented in two different ways. One way is to have a memory containing $N = 2^n$ cells with l qubits each and a database entry $|d_x\rangle$. This is qRAM using only quantum concepts. Another way to do this is to use classical memory with $N = 2^n$ cells of l bits each and containing d_x as the database entry. We can address these cells using a superposition of multiple values which allow us to load the cell values from memory. This is qRAM using classical and quantum concepts.

3.2 Collision Finding [2]

The r-collision problem states that given n and a black box function f is either one-to-one or r-to-one given d inputs, where r divides d. In order for an algorithm to solve this problem, we need to be able to access information from previous queries. The quantum collision problem uses a unitary black box that allows the use of adaptive queries. This problem can be solved in $O\left(\sqrt[3]{\frac{d}{r}} log\left(\frac{d}{r}\right)\right)$.

So, the quantum computer would need to access the outcomes from its memory. This is were qRAM is needed in order to solve the problem. We need to check the query outcomes against eachother to see if all outcomes are distinct when all queries are distinct.

3.3 Element-Distinctness in the Classical and Quantum Setting [12]

The element distinctness problem is described as determining if all elements in a list are distinct. Classically, we can solve this problem by using sorting algorithms. If we sort a list f, we can see if any elements in the list repeat by traversing the list once. A quantum computer solves distinctness faster then sorting because the collision problem is equivalent to the element distinctness problem. There is no need for sorting in a quantum setting. As we discussed above, we need a way to access previous outcomes, thus qRAM is a solution to quickly access the queries that we have made. Without qRAM, we would not be able to solve the collision problem or the element distinctness problem.

4 Possible Implementation and Modern Developments

4.1 Candidate Systems for Implementation of Bucket Brigade

Since the turn of the century, several different physical systems to implement quantum computation have been implemented. Bucket Brigade qRAM is a structure which operates coherently on a small number $O(logN)$ of a large amount $O(N)$ of first-neighbor connected qubits [10]. Two candidate systems that have been proposed for this architecture are optical lattices and Josephson arrays [8] [14].

One way to think about Bucket Brigade in a slightly more, but still not entirely, experimental setting is with photons, with state encoded in polarization, and trapped ions, with state encoded in energy level [10]. Each node in the structure is initially an ion in its ground state. As photons travel through the structure, if they encounter an ion in the $|wait\rangle$ state, then the photon will excite the ion to either the left state or the right state depending on the polarization. If the photon encounters an ion in the $|left\rangle$ or $|right\rangle$ state, it will be scattered by the ion and continue on its path. After the path through the structure has been created, a photon in the $|0\rangle$ traverses through the graph and eventually pick up the state in the memory cell. It then deflects back through the graph. This process allows the ion to excite to a $|left'\rangle$ and $|right'\rangle$ state, which eventually will decay back into the $|wait\rangle$ state. This process is a Raman process [10].

4.2 State of the Art

One way to speed up quantum RAM access has very recently been shown to be a process of quantum forking [7]. This process is similar to classical process forking, when one large process is split into several smaller ones. This process can speed up calculations such as ensemble averaging and inner product calculations. The most basic example of this quantum forking is for inner product calculation in the form of the swap test, with which quantum forking drops the number of qRAM calls by a factor of about $\frac{1}{2}$.

One other very modern aspect of qRAM is the implementation in hardware. It has been shown that acoustic resonators with high quality factors can act as effective quantum memories [11]. This architecture is called circuit quantum electrodynamics (cQED).

4.3 Recent Experiment

Earlier this year, a group from Tsinghua University in China created a 105 qubit random access memory using 210 memory cells using dual-rail representation of qubits [13]. Like some of the ways mentioned earlier in this paper, the group used photonic pulses as bus qubits and atomic spin states as memory qubits. The setup uses temporal multiplexing storing all of its qubits onto a solid-state ensemble. The ensemble was composed of ^{87}Rb atoms that have been trapped and cooled down.
Rather than using a Fanout or Bucket Brigade architecture, this setup uses a pair of crossed acoustic-optical deflectors. After setting up the memory, the group tested the fidelity of each qubit after a storage time of 1.38 [13]. Note that other implementations hope to achieve storage times of up to the millisecond scale using methods such as nitrogen vacancy defects in diamond [5].

In Figure 3 we see a visualization of the average fidelity of each memory cell across several tests for several different stored states. We can see that there are noticeable changes in fidelity across the physical location of qubits. We can gather from this experiment that though we can store qubits and access them after a reasonable amount of time, the fidelity of qubits still has room to be improved. Part of the reason for the lack of fidelity away from the center of the memory qubits may be due to the optical setup used to multiplex the optical beams when accessing each cell.

4.4 Open Questions

The biggest open question is how can a working qRAM be built. It might be possible to create a hybrid quantum system that uses superconducting qubits and spin qubits [5]. However, a working qRAM has not been created.

Another question that needs to be worked on are finding more uses for quantum forking to lower qRAM calls by even a constant amount [7].

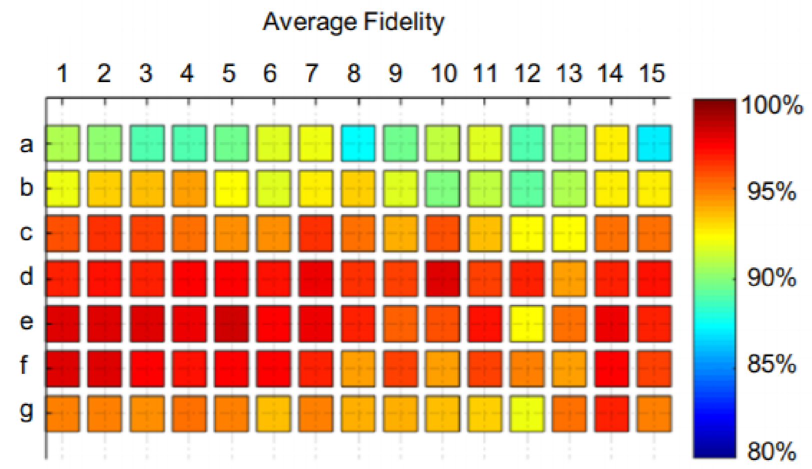

Figure 3: *Fidelity mapping of each memory cell [13]*

There is still more that can be discovered for the reading and writing onto qRAM. Right now, there is a lot of theory and not as many physical implementations. So, a natural question would be how reading and writing in the physical setting would work. Further, the question of how rewriting would work. It is possible to change stored states into the states that we now want to store, however that would require a lot more work.

Lastly, how the physical implementation of Bucket Brigade would work is still unknown. In theory, Bucket Brigade runs faster and more efficient than fanout schemes. It is an interesting question to see how much more efficient Bucket Brigade runs in the physical world.

5 Conclusion

There are still a lot of open questions when it comes to qRAM. We know that it is a necessity as quantum computers grow. However, the theory for architectures like Fanout and Bucket Brigade still needs to be implemented in the real world context. Recent experiments have shown what qRAM might look like, but there is still a lot of work to be done. However, it is clear that qRAM is vital for quantum computers as they begin to solve harder problems.

References

[1] Structured computer organization. https://www.geeksforgeeks.org/different-types-ram-random-access-memory/.

[2] Aram W. Harrow Andrew M. Childs and Pawel Wocjan. Weak fourier-schur sampling, the hidden subgroup problem, and the quantum collision problem. *STACS 2007*, 1:598–609, February 2007.

[3] Todd Austin Andrew S. Tanenbaum. *Structured Computer Organization*. Pearson, 2013.

[4] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549:195 EP –, Sep 2017.

[5] Miles Blencowe. Quantum ram. *Nature*, 468:44–45, Nov 2010.

[6] Michael A. Nielsen Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, UK, 2010.

[7] Francesco Petruccione June-Koo Kevin Rhee Daniel K. Park. Circuit-based quantum random access memory for classical data. *Scientific Reports*, 9:3949, 2019 Feb.

[8] L.-M. Duan, E. Demler, and M. D. Lukin. Controlling spin exchange interactions of ultracold atoms in optical lattices. *Phys. Rev. Lett.*, 91:090402, Aug 2003.

[9] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Architectures for a quantum random access memory. *Phys. Rev. A*, 78:052310, Nov 2008.

[10] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Phys. Rev. Lett.*, 100:160501, Apr 2008.

[11] Connor Hann, Chang-Ling Zou, Yaxing Zhang, Yiwen Chu, Robert Schoelkopf, Steven Girvin, and Liang Jiang. Hardware-efficient quantum random access memory with hybrid quantum acoustic systems. 06 2019.

[12] Mark Heiligman Peter Hoyer Frederic Magniez Miklos Santha Harry Buhrman, Christoph Durr and Ronald de Wolf. Quantum algorithms for element distinctness. *SIAM Journal on Computing*, 34:1324–1330, Sep 2000.

[13] N. Jiang, Y.-F. Pu, W. Chang, C. Li, S. Zhang, and L.-M. Duan. Experimental realization of 105-qubit random access quantum memory. *npj Quantum Information*, 5(1):28, 2019.

[14] Alessandro Romito, Rosario Fazio, and C. Bruder. Solid-state quantum communication with josephson arrays. *Phys. Rev. B*, 71:100501, Mar 2005.