An Overview of Quantum Random Access Memory

ROHITH K

21226

Indian Institute of Science Education and Research, Bhopal

Submitted to:

Dr. Kuntal Roy

Dept. of Electrical Engineering and Computer Science, IISER Bhopal

Project Report submitted towards the completion of Summer Internship 2023, at IISER Bhopal under the topic of Quantum Machine Learning

ABSTRACT

Innovation in memory storage and memory access were crucial in the development of classical computers. This was one of the main reasons for the drastic improvement in the efficiency of classical computation. As we near quantum supremacy, the quantum analogue of classical RAM, would without a doubt play a huge role in near term quantum computers. Considering the already theorized speedup in machine learning techniques using quantum computations, research into the implementation of Quantum Random Access Memory (QRAM) would be of paramount importance in reducing the error rates and reducing the computational complexity. This project attempts to understand the proposed architectures of QRAM and compare it with the architecture of classical RAM. We also look at the physical implementation of QRAM using bucket brigade architecture.

Keywords: Quantum Random Access Memory, Quantum Machine Learning

INTRODUCTION

There are multiple proved advantages of QRAM that would make quantum machine learning and allied areas practical as well as efficient. For machine learning purposes, we typically employ quantum protocols on classical information. This would require encoding of classical information on qubits. The two methods for this are basis encoding and amplitude encoding. Basis encoding associates the basis state of a qubit with each input, while amplitude encoding normalizes the inputs and encodes them in the amplitudes of a quantum superposition state.

The operation of QRAM can be developed from the implementation of classical RAM, which has three basic components. A memory array which contains some N number of memory cells which contain information, an input or address register to which the address location of the memory cell to be accessed is input, and an output register to which the data stored in the specific memory cell is output. Conventionally, memory access is done in the form of a bifurcation graph or a binary tree with memory cells at the end leaves.

While there is notable similarity in the working of QRAM and classical RAM, QRAM has added advantages on many levels. QRAM allows queries in the form of superposition of multiple memory addresses, thanks to the usage of qubits. But direct translation of implementation designed for classical RAM would not be efficient for QRAM, since as number of interactions increase, decoherence in the system

RESULTS

We begin with an overview of the implementation of classical RAM. There are two implementations:

3.1 Classical Fanout Architecture

This implementation of classical RAM is the most popular. A binary tree is constructed such that the nodes at each level are controlled by a bit of the index register. These nodes act as switches and directs incoming channel to the target memory cell using the index register bit as control. For the jth bit (j starting from 0) of the index register, all the switches under its control (2^j) direct either to the left branch or to the right branch. Fig. 1.2 depicts the electronic implementation of such a model, using transistors as switch nodes.

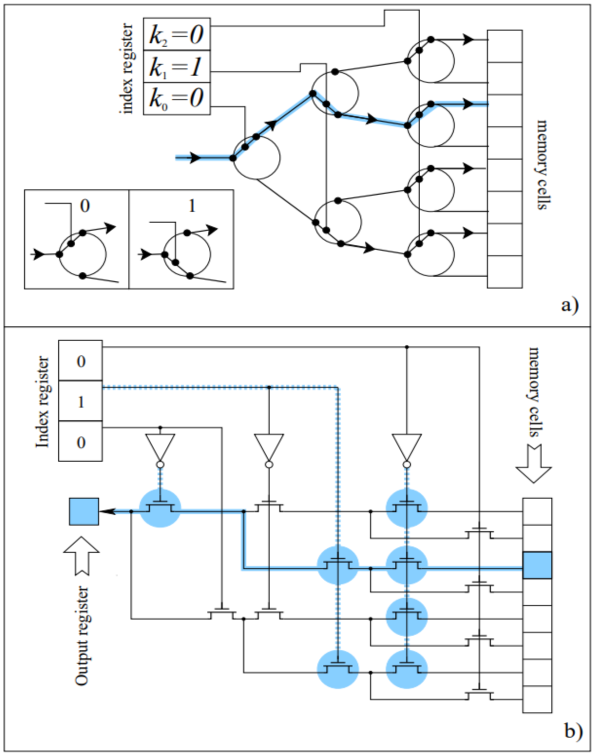

Fig. 3.1(a) Functioning of fanout architecture in classical RAM. Fig. 3.1 (b) Electronic implementation of fanout architecture in classical RAM

Here, since all the switched are controlled by the index register bits regardless of whether the switches are used or not, such an implementation can be impractical to implement on a quantum system. This is due to the interactions of qubits which induce decoherence in the system. Hence, fanout architecture is generally inefficient for QRAM. However, on conventional RAM, since there is no issue of decoherence, and the energy is negligible, fanout is highly practical.

3.2 Classical Bucket Brigade Architecture

Bucket brigade architecture is an improvement over fanout architecture, that does not require as many interactions as fanout architecture does. Here, we use 'trits', that is, units of information with three basis states. The three states can be thought of as '0', '1', and '•' as the switch nodes. When '•' state encounters a bit, the value of the bit is copied to the trit and when '0' or '1' state encounters a bit, it transmits the bit to the next level along one of the two branches. In the binary tree, all the switch trits on the nodes are initialized to the • state. The index register bits themselves are

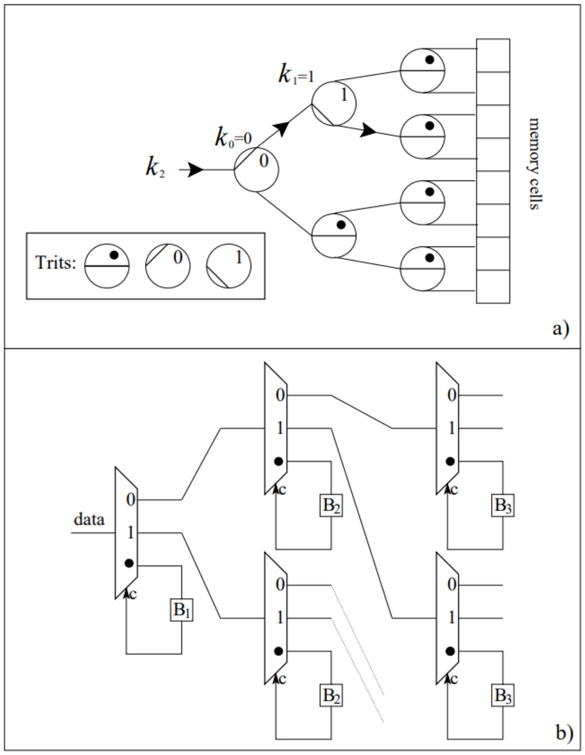

Fig. 3.2(a) Working of bucket brigade architecture in classical RAM.
Fig. 3.2(b) Electronic implementation of bucket brigade architecture in classical RAM

input into the binary tree and through the behavior of the switch trits, the specific memory cell is accessed. Bucket brigade architecture on classical RAM can be implemented using a series of multiplexers and some memory elements as the selection lines. Such an implementation is given in Fig. 1.2. All the memory elements initially store the • state. The first bit of the index register would pass through the multiplexor to the memory element and would then act as the select line. Thus, the next bit of the index register is sent to the left or right branch of the binary tree according to the value of the first bit. If the first bit is 1, then the next bit of the index register is routed to the left branch and if the first bit is 0, it would route the next bit to the right branch of the binary tree.

3.3 Quantum Bucket Brigade Architecture

Quantum bucket brigade architecture involves qutrits, having three computational basis states. The states are $|0\rangle$, $|1\rangle$, and $|W\rangle$. Since these qutrits control the path towards the memory cell, they are called switch qutrits. The state transformation is done through a unitary operation U as follows:

$$\mathrm{U}|\mathbf{0}\rangle|\boldsymbol{W}\rangle = |\boldsymbol{f}\rangle|\boldsymbol{left}\rangle$$

and

$$\mathrm{U}|\mathbf{1}\rangle|\boldsymbol{W}\rangle = |\boldsymbol{f}\rangle|\boldsymbol{right}\rangle$$

Here, the state of j^{th} bit of the index register is stored exactly in one qutrit, at the j^{th} level of the binary tree and routes subsequent qubits as per its state. The first qubit of the index register transforms the head qutrit from $|W\rangle$ state to $|0\rangle$ or $|1\rangle$ state, and directs the second qubit of the index register into either of the two branches depending on the state of the head switch qutrit. After all n qubits of the index register has passed through the binary tree, a distinct path towards the desired memory cell is allowed. For a single memory access form an array of 2^n memory cells, bucket brigade would activate only n qutrits and the rest of the nodes would be in state $|W\rangle$. This would considerably reduce the error rate when compared to fanout architecture. We use a quantum BUS to access the content of the memory cell and store the information. Once the quantum bus is retrieved, we have to reset the active switch qutrits into the $|W\rangle$ state. For this, we use the inverse of the unitary operator U, U†.

$$\mathrm{U}^{\dagger}|\boldsymbol{f}\rangle|\boldsymbol{left}\rangle = |\mathbf{0}\rangle|\boldsymbol{W}\rangle$$

and

$$\mathrm{U}^{\dagger}|\boldsymbol{f}\rangle|\boldsymbol{right}\rangle = |\mathbf{1}\rangle|\boldsymbol{W}\rangle$$

After this, the states of the switch qutrits are set to the $|W\rangle$ state and the system is ready for another memory access. This inverse operator is activated by a series of counters at each node, which activate the operator after some specific numbers of signals have passed through a node.

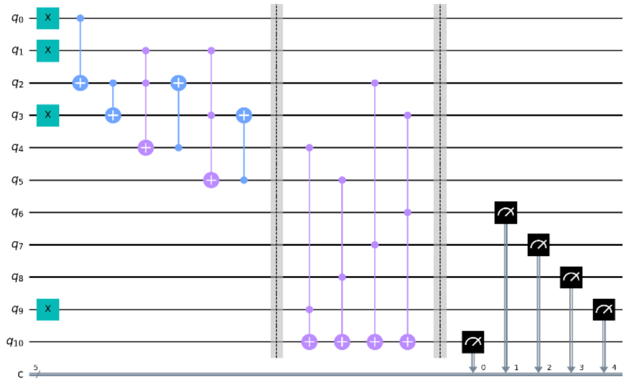

Fig.3.3 Simulated bucket brigade architecture on qiskit. q0 and q1 are address qubits, q2 to q5 are route qubits and q6 to q9 are memory qubits. q10 is the output register.

The logic of the quantum bucket brigade was simulated on Qiskit using a sample quantum circuit with 4 memory cells. The address qubits q0, and q1 are in state $|1\rangle$ and thus, the memory cell q9 which is accessed, and input to the classical register c0. By measuring all the memory cells, we confirm that the value of c0 and c4, which is the measured value of q9, are equal. As predicted, the value of c0 was 1, and the value of c5 was also 1.

3.4 Physical Implementation of Quantum Bucket Brigade Architecture

For the actual implementation of quantum bucket brigade architecture, photons and trapped atoms are used. Trapped atoms in a network of coupled cavities act as qutrit switches while state of the index register qubit is encoded in the polarization state of a photon. The switching action of the qutrit is emulated through Raman transition of the incoming photon induced by a strong laser field. We first send a photon, polarized to encode the state of the first index register qubit to the cavity, and through Raman transition technique, the photon is absorbed into the trapped atom. If the photon is polarized in the $|0\rangle$ state, it is absorbed into the $|zero\rangle$ state of the trapped atom, and the next photon, which encode the second qubit of the index register is Raman transitioned into the $|up\rangle$ level and re-emitted towards the dashed branch of the binary tree which is coupled with the $|up\rangle$ level. Now, if the first photon is polarized in $|1\rangle$ state, it is absorbed into the $|one\rangle$ state and the subsequent photon is bsorbed and re-emitted by the $|down\rangle$ state to the coupled dotted branch in the binary tree. This is repeated by the following qubits and qutrits, resulting in a path to the target memory cell. This would also store all the n address qubits in the trapped atoms, one at each level of the binary tree. Now, another photon is used as the quantum bus and data is retrieved. To reset the switch qutrits, all atoms starting from the last level are made to emit their stored qubits, with sequenced Raman transition.

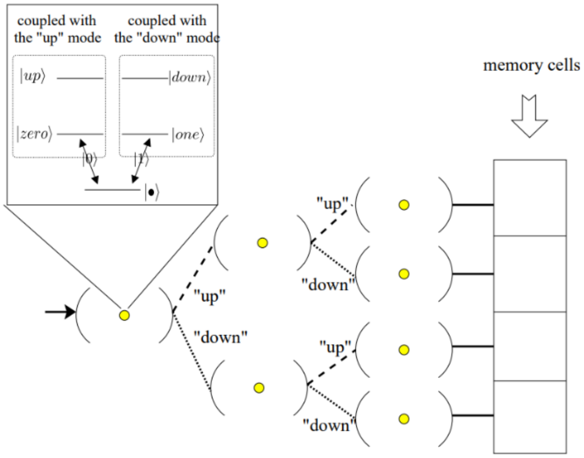

Fig. 3.4 Physical implementation of Quantum Bucket Brigade Architecture

SUMMARY

An efficient and fault-tolerant implementation of QRAM is necessary for the theorized speed up of machine learning as well as other protocols using quantum computers. While it is used in many quantum machine learning techniques, it is an absolute requirement in protocols like Grover's search. Inspired from the implementation of classical RAM, we attempted to understand the proposed implementations of QRAM. Memory access is performed classically, with a bifurcation graph, with route bits at the nodes which directs the bus towards the targeted memory location. In fanout architecture, which is widely used in classical RAM, these route bits are controlled by the bits of the index register. Electronically implemented with a series of transistors, this method would have 2^{n-1} active transistors for n index register bits. Bucket brigade architecture uses trits, instead of bits, and the bits if the index register are themselves copied into the switch trits. This is implemented using a series of multiplexers with some memory element as the selection line. It is possible to implement a quantum analogue of the bucket brigade architecture using qutrits and a quantum bus. Physically, this is implemented using photons and trapped atoms in a network of coupled cavities.

REFERENCES

Vittorio Giovannetti, Seth Lloyd, Lorenzo Maccone (2008), Architectures for a quantum random access memory

Vittorio Giovannetti, Seth Lloyd, Lorenzo Maccone (2008), Quantum Random Access Memory

Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, Priyaa Varshinee Srinivasan (2015), On the robustness of bucket brigade quantum RAM