The quantum Random Access Memory

International Iran Conference on Quantum Information (IICQI)
Kish Island
7-10 September 2007

Lorenzo Maccone
Quantum Information theory group.
Dip. “A. Volta”, Univ. of Pavia

Vittorio Giovannetti
Scuola Normale Superiore

Seth Lloyd
MIT

www.qubit.it
Abstract: We present a protocol to address a memory array using a quantum register as index register. This permits the addressing of arbitrary quantum superpositions of memory cells. If the memory array is classical, a small quantum computer suffices to implement the protocol: the array can be exponentially larger than the required quantum resources.

Outline:

- Ram
- quantum RAM
- conventional architecture: noisy!
- “Bucket brigade” architecture
- implementations
Each cell of a memory array can be univocally determined by its numerical address. An \( n \)-bit register can then address \( 2^n \) possible locations:

- \( i \) bit register
- \( m_i \) output register: content of the \( i \)th memory cell.

\[ \text{RAM} \]

Circuit

\[ \begin{array}{cccc}
1 & 2 & 3 & \ldots & i & 2^n \\
\end{array} \]

Memory array

takes an \( n \) bit address as input, provides the content of the memory cell as output.
quantum Random Access Memory (qRAM)

Same as the previous, but the **index** and **output** registers are made of qubits! (Eventually also is the memory array):

- **$|i\rangle$** is the $n$ qubit register.
- The qRAM circuit takes an $n$ qubit address as input, provides the content of the $i$th memory cell as output.
- The output register contains the content of the $i$th memory cell.
...this has important consequences!
Suppose that the input register is in a superposition of querying for $N$ different addresses:

\[
\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i\rangle
\]  

\[
\text{qRAM} 
\]  

\[
\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i\rangle |m_i\rangle
\]

Now the output register is **ENTANGLED** with the input register!
YES!

It is explicitly or implicitly invoked in MANY known quantum algorithms:

- Quantum searching in a classical database
- Collision finding
- Element distinctness (in the classical and quantum settings)
- Pattern recognition
- New algorithms for evaluating general NAND trees

It's useful also for NEW algorithms we are developing:

- Quantum private queries (interrogate a database securely)
- Quantum routing (route signals in a quantum internet)
1. Conventional architectures: modify the circuits for classical RAMs

   not good!  → **Difficult** to implement and **noisy**

2. Bucket brigade architecture

   Solves the main problems of implementing a qRAM
Internal workings of conventional RAMs

E.g. address register = "2" = 0 0 1 0

each register bit deviates the signal in one level of a bifurcation tree:
This ideal structure is almost faithfully copied in circuit diagrams of conventional RAMs.
...and was also proposed for building a qRAM [Nielsen and Chuang]

PROBLEM!
The last few graph levels are BIG Schroedinger cats!!

It's difficult to maintain the necessary quantum coherence

⇒ Impractical
There's another way to implement the binary decision tree.

Bucket brigade protocol

Put trit memory elements in the tree nodes.

1. Initialize all of them in the “waiting” state:
Bucket brigade 2

2. send the address bits in the network one by one

When it encounters a “waiting” trit a bit “0” becomes a “left” trit
a bit “1” becomes a “right” trit
3. when all the address bits have been sent, there's a route carved in the tree:

   e.g. for address register = 010:

4. extract the information through this route.
It's very simple to quantize this RAM architecture:

just use **qutrits** in place of trits

(and little else)

three-level quantum systems

\[ |waiting\rangle \quad |left\rangle \quad |right\rangle \]
The difficulty of maintaining coherence over many gates is traded for quantum memories.

<table>
<thead>
<tr>
<th></th>
<th>Conventional RAM arch.</th>
<th>Bucket brigade</th>
</tr>
</thead>
<tbody>
<tr>
<td>coherent control</td>
<td>$2^n$</td>
<td>$n$</td>
</tr>
<tr>
<td>q. memories</td>
<td>$n$</td>
<td>$2^n$</td>
</tr>
<tr>
<td>index register</td>
<td></td>
<td></td>
</tr>
<tr>
<td>qutrits</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
HOWEVER, note that:

1. $O(2^n)$ memory elements are necessary **ANYHOW**, when the memory array is composed of quantum memories.

2. Most of the qutrits are always in the “waiting” state. If it is chosen appropriately, there is **very little noise** in the bucket brigade!

   just choose the “waiting” state as a ground state.

$\Rightarrow$ BB is more suited for qRAMs!
Implementation

- Qutrits composed of trapped ions

- Address register stored in the polarization of photon states

- “Bus” system composed of a photon, which flies through the network to the selected memory cell, stores its contents and flies back.
Implementation 2

Ions level structure

1. All the photons of the address register are sent in.
   - if they encounter a ion in $|\text{waiting}\rangle$, they are stored either in level $|\text{left}_d\rangle$ or $|\text{right}_d\rangle$, depending on their polarization.
   - if they encounter a ion in $|\text{left}_d\rangle$ or $|\text{right}_d\rangle$, they are re-emitted (using a Raman transition mediated through $|\text{left}_u\rangle$ and $|\text{right}_u\rangle$) and continue along the tree, to the left or to the right, respectively (the $|\text{left}_x\rangle$ and $|\text{right}_x\rangle$ levels are spatially coupled only to the left and right outgoing modes).
2. Now the bus photon is sent in
It follows the path carved by the address register photons (stored in the ions).
It reaches the memory cell, where it copies (or swaps) its content.
It is reflected back and exits at the tree root node with the memory cell content.
3. The address register photons are re-emitted one by one (starting from the last nodes in the tree)
4. The memory has been accessed, the address register reobtained, and the network reset.
Conclusions

We have seen:

- What is a Random Access Memory
- What is a Quantum RAM
- How conventional RAMs work
- Why this isn't good for quantum computers
- The Bucket Brigade protocol
- Why this is good for quantum computers
- A proof-of-principle BB implementation
A new architecture for random access memories, which is particularly suitable for quantum computers.

Comments and questions to Lorenzo Maccone, maccone@qubit.it

quant-ph arXiv:0708.1879