QOSF: Task 1

Arif Mohd

(Dated: September 18, 2021)

We address the design task of a quantum circuit that considers as input the vector of integers $[1, 5, 7, 10]$ and returns a quantum state which is a superposition of the *indices* of the inputs where two adjacent bits have different values, i.e., $\frac{1}{\sqrt{2}}(|01\rangle + |11\rangle)$ for the input above. We first use a quantum RAM (qRAM) to store the data given in the input vector that can be located by the index register of the qRAM. Next we use this qRAM in designing an Oracle that marks the input integers of interest (namely those with strings made of alternating bits) and correspondingly marks the index of the memory location of the integers of interest. Finally the Grover amplitude amplification is applied to the index register to obtain an output that is close to the desired output with a high probability. We note a subtlety that the direct application of Grover's algorithm requires that the number of search solutions be less than half of the total number of search entries. Since we have essentially four search entries (four data points saved in four memory locations) and two search solutions (two data points occupying two memory locations) we can't apply the Grover algorithm as it is. Instead, we first add one more qubit to the index register of the qRAM and pad the extra four memory locations with zeros. This effectively modifies the search problem to find two entries out of eight and now Grover's algorithm applies. Simple counting tells that one query to the Oracle, and thus one Grover iteration, is optimum. Although the math works out as expected and our solution in principle achieves the goal, we note that an unsatisfactory aspect of our proposed solution to the design task is that we are not able to simulate it on the simulator because of the use of too many qubits. This we believe is due to our treatment of the input data as quantum. We expect that treating the input data as classical and modifying the Oracle accordingly will reduce this overhead and then a simulation would be possible, but we leave it for future work.

I. INTRODUCTION

Consider the following problem. We are given a vector containing four integers $[1, 5, 7, 10]$. Suppose we use two qubits to indicate the location, i.e., the index of the integers. $|00\rangle$ points to the first entry which is '1', $|01\rangle$ points to the second entry which is '5', $|10\rangle$ points to the third entry which is '7', and $|11\rangle$ points to the fourth entry which is '10'. Now each entry in the vector can be presented in the binary form: '1' is 0001, '5' is 0101, '7' is 0111, and '10' is 1010. The problem is to design a quantum circuit that takes the vector as input and gives as output the superposition of indices of the integers whose binary representation has alternating bits (in this case 5 and 10). Therefore, the output of the quantum circuit for the above input vector should be $\frac{1}{\sqrt{2}}(|01\rangle + |11\rangle)$. The goal of this article is to explain the design of a quantum circuit that accomplishes this.

It is clear that this problems falls into the category of problems known as *unstructured database search*, see for e.g., ref. [1]. The design of such a circuit will need a memory component that can store the data and has unique indices pointing to the location of each data point. One should be able to randomly access from the memory any data point from the memory-index of the location of that data point. In other words we need a quantum random access memory, henceforth called qRAM. It is quantum because the indices of the memory locations will be qubits and we could access the superposition of data points by creating a superposition of these memory index qubits. The data stored in qRAM could itself be classical or quantum. In our work, we will think of the data as quantum although we are not going to make any use of this quantum nature. We discuss more about this in the sec. VI. Once we have the qRAM we could in principle use the Grover search algorithm applied to the memory indices in order to retrieve the indices of our data of interest (solution of the search). All it needs is an Oracle to be used in the Grover's search. With the qRAM and the Oracle in hand we could use the Grover search algorithm to obtain the solution to our design problem discussed above. However, for Grover's search to be useful the number of search solutions must be less than half of the total search entries. In our case however, we have four search locations and two search solutions. So a straightforward application of Grover's search wouldn't be helpful. The trick, as explained in ref. [1], is to add a qubit to the memory indices. This doubles the number of memory locations.We could pad in the extra memory locations with irrelevant data which is not a solution of the search problem. This trick thus doubles the number of search entries while the number of search solutions remain the same. Now we can use Grover's algorithm to find the data of interest.

The plan of this article is as follows. In sec. II we discuss the quantum circuit and the working of the qRAM following ref. [2]; in sec. III we discuss the design of the Oracle that gives a negative phase to solution of the search problem. Here we also emphasize the role of *uncomputation*; in sec. IV, we append the amplitude amplification protocol of Grover's algorithm and give a complete working example of Grover's search using the qRAM Oracle followed by the amplitude amplification; in sec. V we propose our solution of the search problem that we described above; finally, in sec. VI we discuss the unsatisfactory aspect of our solution and some ideas on improvement.

II. QRAM

A particular model of quantum random access memory, called the bucket-brigade model, was proposed in ref. [3], see also ref. [4]. A quantum circuit for the bucket-brigade model was given in ref. [5] and is nicely explained in ref. [2]. It's the circuit given in the latter that we use in our work. The bucket-brigade model shown in fig. 1 consists of two-qubit *address* register, four-qubit *fanout* register that maps the address register values to the corresponding memory locations, a 16-qubit memory register with each 4-qubit memory sub-register that stores the binary form of the integers. For example, the memory location $m00$ has four-qubits $m00_0, m00_1, m00_2, m00_3$. In fig. 1, $m00$ stores the string 0001 corresponding to the integer '1', similarly, $m11$ has four qubits $m11_0, m11_1, m11_2, m11_3$ that are storing the string 1010 which is the binary representation of the integer '10'. Finally, the four-qubit *readout* register in fig. 1 reads out the content of the memory location that is indexed by the index register.

Too see how the qRAM works, in fig. 2 we have set the address register to $|11\rangle$. This is fanned out to the memory location $m11$ which contains the state $|1010\rangle$. We then measure the readout register and the result is shown is fig. 3. We see that our readout register correctly retrieves the state stored in the four qubits of the memory location $m11$.

We remind the reader that we are treating the data stored in the memory as quantum although we will not use this feature. We comment more on this in sec.VI. As described in the introduction in sec. I, for solving our state-preparation, or equivalently our search problem, we will be using three-qubit memory register. The fanout register will then be eight-qubit and there will be eight accessible memory locations.

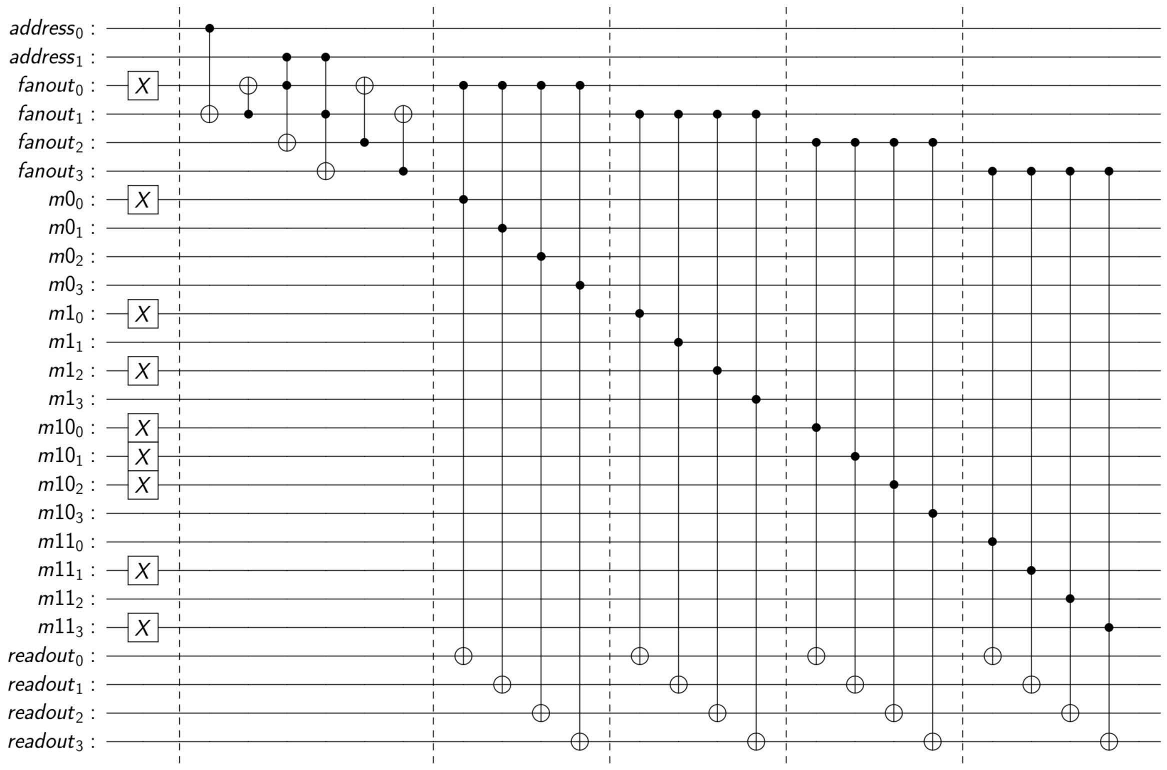

FIG. 1. Quantum circuit of a quantum RAM. Data [1, 5, 7, 10] is fed into the memory register and and is read out by the readout register. The data is accessed by the address register which is fanned out to different memory locations based upon the state of the address register.

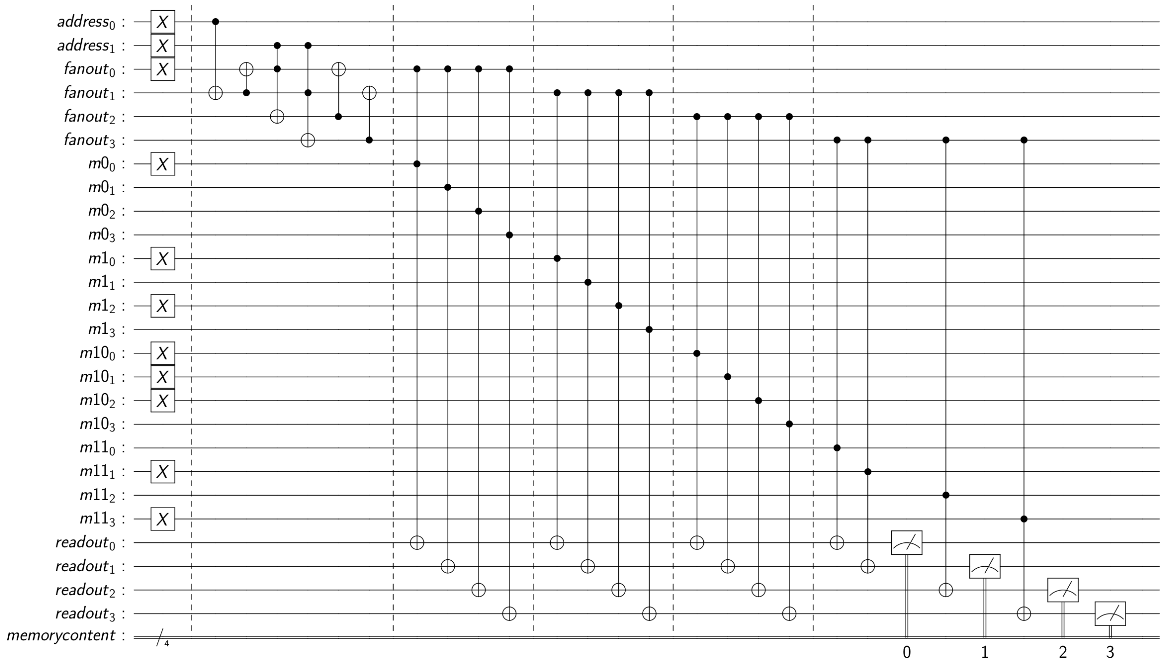

FIG. 2. Quantum Circuit to check the qRAM. The state of the address register is $|11\rangle$ and it accesses the memory location m11 through the fanout register. The memory location m11 has four qubits in the state $|1010\rangle$ which are read by the readout register.

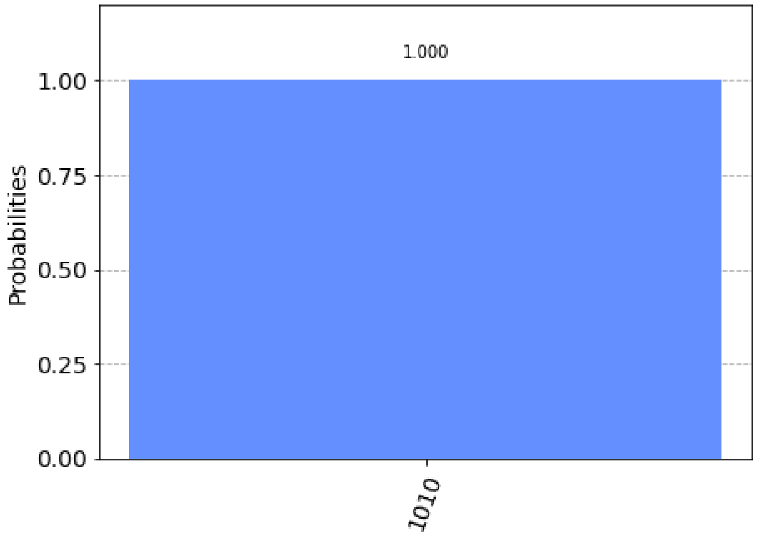

FIG. 3. Measurement result of the readout register for the circuit shown in fig. 2. We see that the result is peaked at $|1010\rangle$.

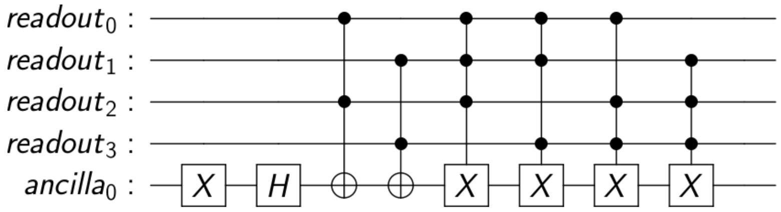

FIG. 4. Quantum Circuit to give negative phases to $|1010\rangle$ and $|0101\rangle$ while keeping the other states as is. It is achieved by first initializing the ancilla in the state $|-\rangle$.

III. THE ORACLE

Before looking at the workings of the Oracle, let's first see how to give a negative phase to the states read out by the readout register which correspond to 1010 and 0101. For this purpose we add an ancilla qubit that is initialized to $|-\rangle$. The quantum circuit for this task is shown in fig. 4. When the readout register contains either $|1010\rangle$ or $|0101\rangle$, the ancilla gets a negative sign. For any other state in the readout register the ancilla remains as it is. This means that if the readout contains a superposition of all the states (this can be obtained by inserting Hadamard gates in the input lines in the circuit in fig. 4), i.e., $\frac{1}{2}\sum_{x=0}^{2^2-1}|x\rangle$ then the output will be $\frac{1}{2}\sum_{x=0}^{2^2-1}(-1)^{f(x)}|x\rangle$. Here, $f(x)$ is the decision function: $f(x)=1$ for $x=0101$ or 1010 and is zero otherwise, where $|x\rangle$ is a state of the readout register.

It is now clear that to get the Oracle we need to add an ancilla to the input and append the readout register with the circuit in fig. 4. However, we need to form a superposition of the *indices* of the memory locations of 10 and 5 (1010 and 0101) and not the data itself. This can be obtained by *uncomputing* the readout register and the fanout register. So, if we give as input to the address register of our qRAM a state that is the linear superposition of all the memory locations, i.e, the state $\frac{1}{2}\sum_{x=0}^{2^2-1}$, where x is now the state of the address register , then the readout will read the superposition of all the data points. Then using the circuit in fig. 4 we give a negative phase to the data of interest. After uncomputing the readout and the fanout register the only change is that the *indices* of the memory register corresponding to the location of data of interest get a negative phase and the state in the memory address register has evolved to $\frac{1}{2}\sum_{x=0}^{2^2-1}(-1)^{f(x)}|x\rangle$. Here $f(x)$ is now the decision function: $f(x)=1$ if x points to the location of 0101 or 1010 and is zero otherwise. This quantum circuit is shown in fig.5.

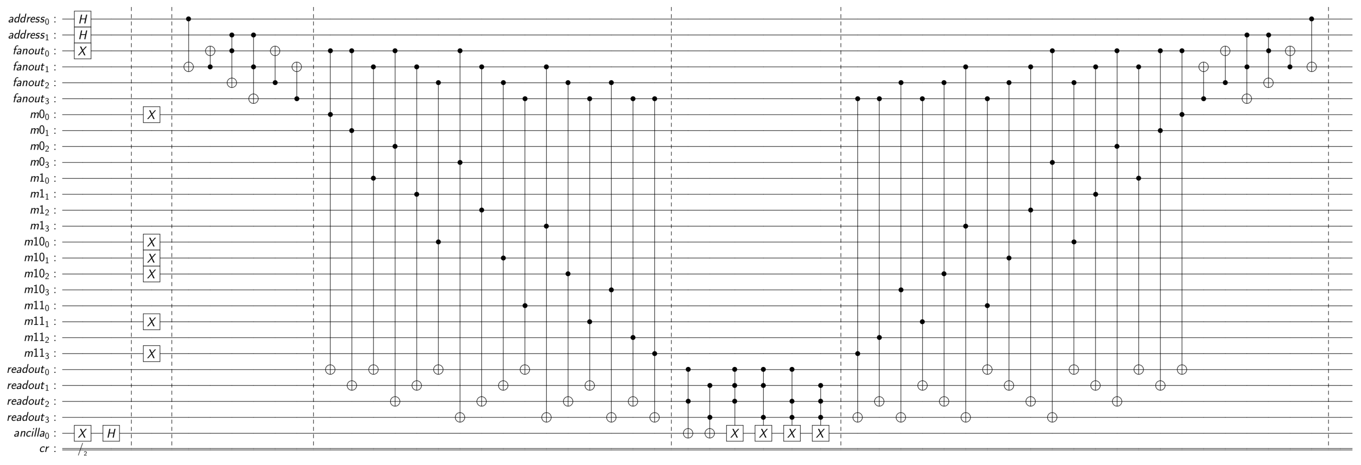

FIG. 5. A complete Oracle to mark the $m11$ state $|1010\rangle$ that leads to marking its location after readout and fanout registers are *uncomputed*. Finally, the address register state $|11\rangle$ receives a negative phase while the other address registers state remain unchanged.

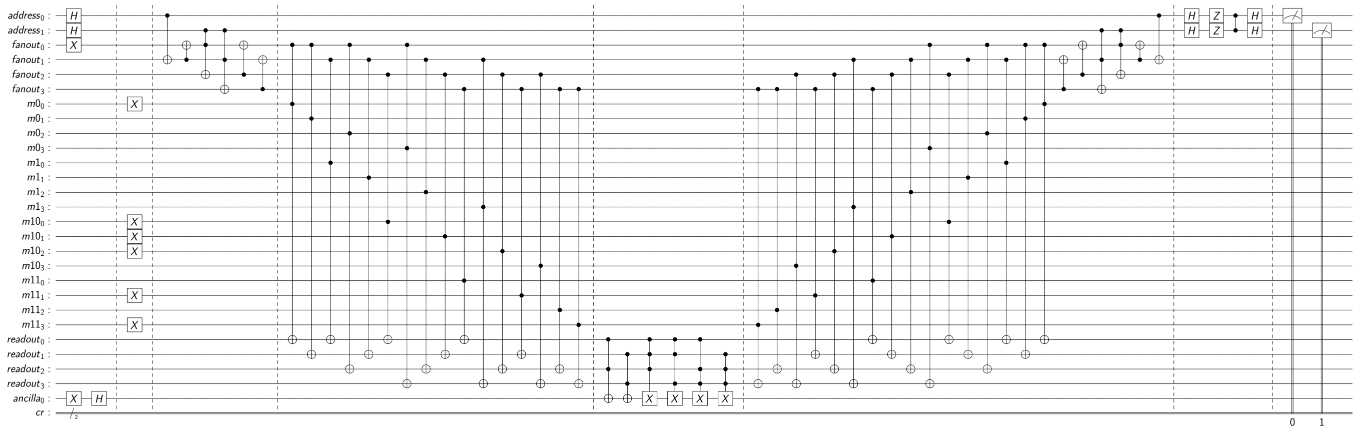

FIG. 6. Complete circuit implementing the Grover search on two-qubit address register. It picks out the state of the address register that points to the memory location holding the state $|1010\rangle$.

IV. SEARCH USING THE QRAM ORACLE: AN EXAMPLE

If our idea of adding a qubit to the address register and then using Grover's algorithm for solving the task at hand is going to be successful we should better convince ourselves first that for a two-qubit register and only one search solution, the case that we know that Grover's algorithm works, our quantum circuit does work. But does it? In this section we show that the answer is in affirmative.

Let's suppose that we have only one marked entry of the input vector, say 10. We want to show that in this case, with two qubit address register of the qRAM, and one application of Grover's iteration we can retrieve the state of the address register that points to the memory location holding the state $|1010\rangle$. A Quantum circuit for this task is shown in fig. 6. The only new ingredient is that we have appended the Oracle with the amplitude-amplification circuit also called the diffuser. The job of the diffuser shown in fig. 7 is, forgetting about all the Hadamards, to leave the state $|00>$ as it is while giving a phase = -1 to all the other states.

The output of the circuit is seen after measuring the address register qubits. It is shown in fig. 8. We see that our circuit correctly identifies the location of the $|1010\rangle$ that corresponds to the entry of input vector '10'.

V. SOLUTION OF THE ASSIGNED DESIGN TASK

Finally, we are ready to give the solution of the problem described in sec. I. As already mentioned there, the only change between what we did in the complete example in sec. IV and now is that now we use a three qubit address

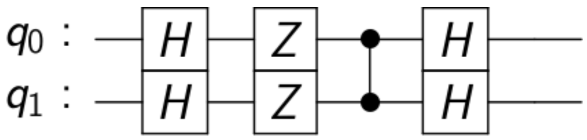

FIG. 7. Amplitude amplification circuit used in fig. 6 .

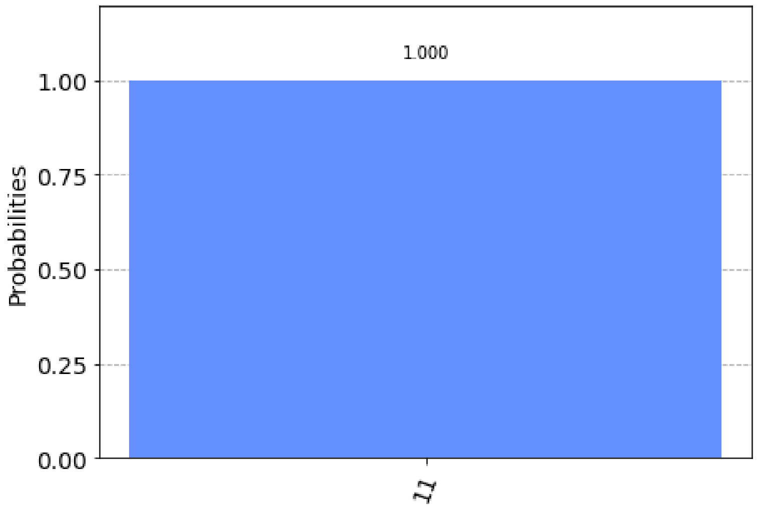

FIG. 8. Result of the search performed in fig. 6. Since the memory m11 holds the state $|1010\rangle$, and it is accessed by the address register's state $|11\rangle$ we see the probability is trivially peaked about $|11\rangle$.

register. This increases the fanout register and the memory register to eight. Also, the amplitude amplification circuit also needs to be modified. The circuit for the amplitude amplification is shown in fig. 9. We pad the extra memory locations by zeros. Now the Grover's search algorithm can be used and only one Grover iteration is optimum to bring the output state of the address register to be close to the linear combination (of almost with the same amplitudes) of the superposition of states of the address register that each point to the location containing the data of interest, namely, 0101 and 1010. Since the memory register is three qubits now, and we know that the search-solutions, i.e., data of interest are contained within the first four memory locations, i.e., the first two qubits, once we have the output of the Grover's iteration we can trace out the third qubit of the address register (it won't be entangled with the first two qubits of the memory register, so the final state would still be pure) . We would thus obtain with a high probability the desired output to be close to that asked in the problem. Since the total number of qubits now are about 48, we unfortunately are not able to run it on a simulator. However, the math is clear and we do expect our quantum circuit to do the job. The circuit diagram in this case is also too big to be put here.

VI. OUTLOOK AND DISCUSSION

We have proposed a solution of the task mentioned in the abstract, and also in the introduction sec. I. While we have a reasonable confidence that our quantum circuit solves the problem there is a disagreeing aspect to it. We are using 48 qubits and it's a large number for a simulator to be able to keep track of. This is the reason why we could

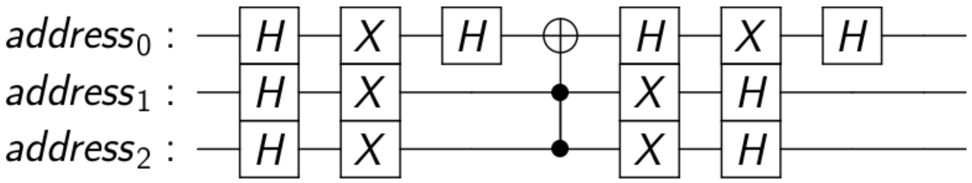

FIG. 9. Amplitude amplification circuit used in the actual solution using 3qubit address register. This is taken from [1].

not check our solution with a simulator. We believe that this large overhead could be ameliorated to a large extent by using the classical data in the quantum memory. In our program we have represented the input data as quantum, however we think that it's not really needed.

We have also not discussed the case of the general sized database here. It's clear that we would need to be first able to manipulate the classical data in the qRAM to mark the data of interest. We leave this for future.

Finally, we have not given any analysis of the breadth/depth/complexity of our quantum circuit and how it would fare against other circuits to solve the problem. We hope to come back to tihs point after we have seen some other solutions to the problem being addressed here.

ACKNOWLEDGMENTS

I am thankful to Alberto Maldonado of QOSF's channel # mentorship-applicants for patiently addressing my confusions and for many very helpful comments.

[1] Michael A. Nielsen and Isaac L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).
[2] Olivia Di Matteo, "A primer on quantum ram," (2020).
[3] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone, "Quantum random access memory," Physical Review Letters **100** (2008), 10.1103/physrevlett.100.160501.
[4] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone, "Architectures for a quantum random access memory," Physical Review A **78** (2008), 10.1103/physreva.78.052310.
[5] Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, and Priyaa Varshinee Srinivasan, "On the robustness of bucket brigade quantum ram," New Journal of Physics **17**, 123010 (2015).