UNDERSTANDING GROVER’S SEARCH ALGORITHM THROUGH A SIMPLE CASE OF STUDY

F. Orts¹, G. Ortega², N.C. Cruz¹, E.M. Garzón¹

¹Dpt. of Informatics, Univ. of Almería, ceiA³, Almería (SPAIN)
²Computer Architecture Department, Campus Teatinos, Univ. of Málaga (SPAIN)

Abstract

Quantum computers are emerging as one of the most novels technologies nowadays. These computers work through quantum circuits which, unlike classic computer circuits, follow the set of rules determined by quantum mechanics. However, those rules are counterintuitive, and it is necessary to define appropriate techniques to explain them to students. This work aims to improve the existing approaches to explain Grover's search algorithm, which is known for demonstrating that quantum computers can outperform the traditional ones. We propose a new methodology to teach it that avoids complicated descriptions and uses a simple application example, namely, the classical division. In this context, the division is seen as the search of a number X that is equal to the dividend when multiplied by the divisor. We develop a complete explanation of the algorithm for this example. It covers the beginning, evolution, and finalization of the method conceptually and physically while also exposing its pros and cons. With this example, it is not necessary knowledge about quantum mechanics, nor neither geometry, only basic knowledge about quantum computing is necessary (which is also introduced in this work). Although there are better ways to solve divisions, we believe it is a perfect example to explain Grover’s algorithm. The objective of this paper is that Computer Engineering students understand the behavior and the bases of Quantum Computation and Grover's algorithm in particular. Therefore, the proposed methodology has been designed for being included in current degrees in Computer Engineering, which do not generally cover this topic despite its future potential.

Keywords: Education, Computer science, quantum computation.

1 INTRODUCTION

Quantum computation is based on the principles of quantum mechanics, a mechanical framework for the construction of physical theories. Its appearance comes from the idea that an algorithmic process can be simulated efficiently using a Turing machine [1]. However, it is not the case for randomized algorithms, which cannot be efficiently solved on a deterministic Turing machine [2]. Following such idea, David Deutch defined a class of computing machines which was capable of efficiently simulating an arbitrary physical system [3]. These machines are the predecessors of the current quantum machines. The best examples of the highest power of the quantum computers are Shor and Grover's algorithms [4], [5]. Although there are several problems which are better solved with quantum computers, the kind of problems that are better solved through quantum computation is now defined. Even though quantum computers are labelled as being counterintuitive, classic algorithms have their quantum counterparts, which are, at least, as good as the classic ones in terms of performance [6]. However, the way of solving a problem is usually dependent on the kind of computer -classic and quantum-, because each kind has its own properties and paradigms [7]. It can be observed even in the simplest elements of a quantum circuit: the logic gates. Quantum computers use several basic gates whose rules do not exist in the classic computation, so it is necessary to find different approaches to solve known problems in an efficient way, as it is explained below.

As it has been mentioned, quantum computers work using a special kind of circuit called a quantum circuit. The bases of such circuits are the quantum gates. A quantum gate is like a gate from a classical logic circuit, but they have some differences since quantum gates must follow the concepts of quantum mechanics. Several classical ideas like loops or joining wires are impossible in the framework of quantum mechanics. Moreover, quantum mechanics have their own rules that are not present in classical circuits. For instance, all quantum gates (all quantum circuits) must be reversible. Quantum circuits never lose information. This usually means that more resources are needed in order to keep the information and make it possible to recover all the states a qubit has had during the circuit.
Quantum parallelism is one of the advantages that quantum computers offer. A quantum computer can save lots of different states and can compute operations with all these states at the same time using superposition, a property of quantum mechanics. This kind of parallelism is ideal when using brute force. For instance, in a cryptographic problem, every possible key can be loaded and tested at the same time. Nevertheless, there is a problem: the wave-particle duality. Only one random state will be recovered if the recovery of them is carried out. And, what is more, all the states have the same probabilities to be the final result. Luckily, Lov Grover found an algorithm to manipulate the probabilities of the states in superposition [5]. It allows the probabilities of a certain state to increase while those of others decrease. This process can be repeated until the probability of the desired state is almost guaranteed. This process involves $N$ steps using several basic exponential computations, which divides by two the size of a password in a brute force attack [8]. Moreover, Grover’s algorithm can be used to simplify several problems.

The objective of this paper is that Computer Engineering students understand how Grover’s algorithm works. To achieve this purpose, the algorithm is explained using a divisor circuit instead of following a classical approach. Grover’s algorithm is normally used in hard problems in the literature, being such problems more difficult to understand than the algorithm itself. By simplifying the case study, it is easier to identify the steps, the main ideas and the benefits of the algorithm. On the other hand, there are better methods than Grover’s algorithm to solve divisions due to several facts (which are later described in this paper). However, the division is a clear example to explain Grover’s algorithm thanks to its simplicity and the possibility of identifying the advantages and disadvantages of using Grover’s algorithm.

The remainder of this paper is structured as follow. Section 2 contains the necessary knowledge about quantum circuits to make this paper self-contained. Section 3 describes the methodology used in the paper and Grover’s search algorithm. Section 4 presents the final divisor circuit as a combination of Grover’s search algorithm and current multiplier circuits. Finally, Section 5 summarizes the conclusions.

2 QUANTUM CIRCUITS

A quantum circuit consists of a set of quantum gates. A quantum gate is like a classical logic gate, but there is an important difference: quantum gates must follow the principles of quantum mechanics [6]. These principles can be summarized as follows:

- They operate on quantum qubits. So, they must deal with quantum properties such as entanglement and superposition.
- They must be reversible. That is, applying a gate twice must result in the original input. This also means that quantum circuits never lose information. In other words, it is always possible to recover the previous states of a qubit.

For instance, the classical XOR gate is irreversible because it is not possible to recover the inputs from a given output. It could not be a quantum gate as it does not fulfill the second rule. In a higher level, a set of quantum gates (that is, a quantum circuit) has the following constraints:

- Loops are not allowed. This means that feedback is not possible.
- Wires cannot be joined. Joining wires is usual in classical circuits, but it implies irreversibility.
- Qubits cannot be duplicated, so copies of qubits are not an option.

Returning to quantum gates, they can be expressed as matrices. A qubit can be expressed as a vector of norm 1 inside a sphere of radius 1 (Fig. 1). They only rotate the vector, but the norm does not change. There are many quantum gates, but all of them can be expressed as a combination of the gates explained in the following subsections. It is fundamental to know how they work in order to understand the final divisor circuit.
2.1 Pauli gates

There are three quantum gates which are used to rotate the value of the qubit among the X, Y and Z axes. They are called Pauli-X, Pauli-Y and Pauli-Z gates, respectively, and they perform a rotation on a single qubit, as it can see in Figs. 2, 3 and 4. As they act on a single qubit, Pauli gates consist of 2 x 2 matrices. The most used one in this work is the Pauli-X gate, which is described as a classical NOT gate because its output is $|0\rangle$ if the input is $|1\rangle$, and $|1\rangle$ if the input is $|0\rangle$. However, if the input is $|a\rangle$, provided that $a \neq 0$ and $a \neq 1$, the gate does not negate that quantum state. Instead of doing so, the gate swaps the probabilities of $|0\rangle$ and $|1\rangle$. For instance, if $|a\rangle = x|0\rangle + y|0\rangle$, then applying Pauli-X gate will result in $|a\rangle = y|0\rangle + x|0\rangle$.

2.2 Controlled-NOT gate

The Controlled-NOT gate, also called CNOT gate, is like the Pauli-X gate. The difference is that this gate acts on two qubits: a control qubit and a target qubit. The idea behind the CNOT gate is to perform the Pauli-X operation on the target qubit only if the control qubit is set to 1. For instance, if the control qubit is set to $|1\rangle$, the gate will always perform the operation. The idea of the controlled gates in quantum circuits (not only the CNOT gate) is to attach an identity matrix (a) to the top-left corner of another matrix (b). Matrix a performs the control operation, that is, it will allow the operation only if the control qubit takes the value 1. On the contrary, b is the matrix (operation) to be applied on the target qubit, for instance, the Pauli-X in the CNOT gate. This is shown in Fig. 5.
2.3 Hadamard gate

The Hadamard gate acts on a single qubit, like the Pauli gates. This gate is responsible for setting a qubit into superposition. It performs a rotation of $\Phi$ radians around the X axis and then another rotation of $\Phi/2$ radians around the Y axis, as shown in Fig. 6. For instance, a qubit in the state $|0\rangle$ will be transformed into $|+\rangle$, that is, $(1/\sqrt{2})|0\rangle + (1/\sqrt{2})|1\rangle$, and a qubit in $|1\rangle$ will be transformed into the state $|-\rangle$, that is, $(1/\sqrt{2})|0\rangle - (1/\sqrt{2})|1\rangle$. In such examples, several parallel states are being considered.

The power of superposition is that with $N$ qubits there are $2^N$ parallel states. There is not anything similar in classical computation [9]. Moreover, this gate is also necessary to get the Bell states, which have interesting applications like quantum teleportation.

2.4 Toffoli gate

The Toffoli gate is like the CNOT gate. However, it has two control qubits. So, the gate only operates on the target qubit if the other 2 qubits are set to 1. If only the standard (orthonormal) basis $|0\rangle$ and $|1\rangle$ are considered, the Toffoli gate can be used to simulate the classical NAND gate and to do FANOUT. Taking into account that this work is about the integer division, this consideration is always true. However, in any other situation in which $|a\rangle = x|0\rangle + y|0\rangle$ with $x \neq 0$ and $y \neq 0$ that is not true because negating a quantum state is not as simple as negating a bit.

Building a Multiple-controlled Toffoli (MCT) gate, i.e., a Toffoli gate with more than 2 control qubits, is not trivial. In [10], an MCT gate which is smaller than the others and has fewer ancillary qubits is presented. In this work, this gate is used when an MCT operation is necessary.

In summary, Table 1 shows the circuit symbols and matrix representations of all the gates described in this section.

3 METHODOLOGY

A division of two positive integer numbers, being $A$ the dividend and $B$ the divisor, can be expressed as $A = BQ + R$, where $Q$ is the quotient and $R$ the remainder. In this work, the remainder is not considered for the sake of clarity, so the division can be simplified as $A = BQ$. This operation could be solved by finding a value of $Q$ which satisfies the expression. By using quantum parallelism, we could try all the possibilities of $Q$ to compute the product $BQ$. This could be done through superposition and a multiplier circuit, which is smaller in terms of qubits and gates than a divisor circuit. However, it is only necessary to find the value of $Q$ which satisfies $BQ = A$, and the mentioned operation would return $BQ$ for any random $Q$ value. But this can be solved with Grover’s algorithm, which allows
manipulating the probability of the desired solution, making it possible to recover it among the other results.

As stated, the objective is to find a $Q$ such that $BQ = A$. In a search space of $2^N$ elements, where $N$ is the number of digits of the solution, a classical search would need $2^N$ iterations in the worst case. Theoretically, in a quantum search, it can be solved with only $\sqrt{N}$ iterations with Grover’s search algorithm [5]. However, that approximation does not take into account that those iterations can only be carried out consecutively or partially in parallel. So, the quantum search requires $N^{1/2}$ operations [11]. Although it is still better than a classical search, the runtime is exponentially worse than a classical division. Nevertheless, as previously highlighted, the purpose of the circuit is only didactic.

Table 1. Circuit symbol and matrix form of the principal gates.

<table>
<thead>
<tr>
<th>GATE</th>
<th>CIRCUIT SYMBOL</th>
<th>MATRIX</th>
<th>GATE</th>
<th>CIRCUIT SYMBOL</th>
<th>MATRIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pauli-X</td>
<td><img src="image1.png" alt="Image" /></td>
<td>$\begin{bmatrix} 0 &amp; 1 \ 1 &amp; 0 \end{bmatrix}$</td>
<td>Pauli-Y</td>
<td><img src="image2.png" alt="Image" /></td>
<td>$\begin{bmatrix} 0 &amp; 1 \ 1 &amp; 0 \end{bmatrix}$</td>
</tr>
<tr>
<td>Pauli-Z</td>
<td><img src="image3.png" alt="Image" /></td>
<td>$\begin{bmatrix} 0 &amp; 1 \ 1 &amp; 0 \end{bmatrix}$</td>
<td>Hadamard</td>
<td><img src="image4.png" alt="Image" /></td>
<td>$\begin{bmatrix} 1 &amp; 1 \ 1 &amp; -1 \end{bmatrix}$</td>
</tr>
<tr>
<td>CNOT</td>
<td><img src="image5.png" alt="Image" /></td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>Toffoli</td>
<td><img src="image6.png" alt="Image" /></td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 0 &amp; 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 1 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
</tr>
</tbody>
</table>

3.1 Grover’s search algorithm

Grover’s algorithm is the fastest way to search a vector starting from an initial state [12]. The desired state can be obtained by repeating iterations. An iteration of Grover’s algorithm consists of a rotation in the two-dimensional space covered by the initial vector $\Psi$ and a state which is the superposition of all the solutions to the search problem [13], as shown in Fig. 7. In this figure, $\beta$ represents the solution state. $\theta/2$ is the angle between $\Psi$ and $\alpha$, which is the perpendicular state to the solution state. $\Psi$ is set to a superposition state of all the computational basis states $\alpha$ and $\beta$. In a search space of $2^N$, $\Psi$ will be a superposition of $N$ states. The solution is usually a superposition. However, this is not the case because only one solution is possible. To get close to $\beta$, $O(\sqrt{N})$ iterations are needed [12].

The inputs are $N$ qubits and several auxiliary qubits which can be needed for the Oracle. The $N$ qubits are put in superposition with Hadamard gates. Fig. 8 shows the subroutine which computes an iteration. This iteration is repeated until the solution is obtained. Following the ideas described in the last paragraph, the procedure is as follows [6]:

![Image](image7.png)

Figure 7. Representation of a single iteration of the Grover’s search algorithm. $\beta$ represents the solution state, and $\alpha$ is the perpendicular initial state to $\beta$. 
1. Apply the Oracle.
2. Set the $N$ qubits into a superposition state.
3. Perform a conditional phase shift, which consists in reverting all the states (apart from those in $|0\rangle$) with a Pauli-Z gate.
4. Apply superposition again.

### 3.2 The Oracle

The purpose of the Oracle is to find an integer $Q$ such that $QB = A$. Since the search is focused on $Q$, and $A$ and $B$ are already known, only the $N$ qubits of $Q$ must be in superposition. Thus, the Oracle must compute $QB$ and then to compare the result with $A$. That is, the output must be 1 if $A = BQ$ and 0 otherwise. The Oracle, as a part of Grover's search algorithm, is shown in the circuit of Fig. 9 (Oracle box), where the first $N$ qubits are the digits of $Q$ and the auxiliary ones include $A$ and $B$, as well as any other necessary qubit. Since the computation made by the Oracle flips only the phase of the solution state, the circuit is able to find the specific inputs ($Q$) which are a solution by performing a full search [14]. Some oracles are shown in the next section.

![Figure 8. This circuits performs the idea presented in Fig. 7.](image)

### 4 DIVISOR CIRCUIT

There are several works about quantum divisor circuits [15], [16], [17], [18], [19], [20]. The quantum divisor circuit presented in [17] reduces the number of quantum gates and ancilla qubits of [15], [18], [19], [20]. The circuit in [16] is the most recent one, and it improves in efficiency. This circuit employs subtractor and conditional addition circuits to compute the division as shown in Fig. 9. The proposed circuit is focused on teaching how to use Grover’s algorithm, so it computes the division with this algorithm. However, it is only a didactic example: the circuits mentioned above are better than the proposed one. Those circuits optimize resources and time, and they should be used for computing divisions in a real case. In general, Grover's algorithm should be used in search problems for which there is not an optimal resolution approach.

Since the conditional phase shift is given by the algorithm itself, it is only necessary to build the Oracle. Its objective should be to find an integer $Q$ such that $QB = A$. Therefore, any circuit which computes $QB$ and compares the result to $A$ can be used. We can find circuits in the state-of-the-art which compute such operations.

#### 4.1 Quantum divisor circuit for the general case

To compute $A / B$, a full multiplier integer circuit is needed. Any available circuit is valid, but that presented in [21] (Fig. 9) is currently the most optimized in terms of garbage and ancilla qubits. It is not necessary to understand that circuit: the only important thing is that it computes the product, so we can use it as our Oracle (It is only necessary to add a comparator to check if $A = BQ$). However, it is briefly explained for the sake of clarity. It uses a novel addition and rotation methodology which allows reducing the amount of resources. This methodology involves the following steps:

1. For $m = 0$ to $N * 2$ repeat steps 2 and 3.
2. Compute the Addition block. After that, $Q[m]$ (the $m$-digit of $Q$), $B$ and $Zcin$ (a qubit used to propagate the carry generated from the previous qubit position) are restored, and the auxiliary qubits, $P$, are modified according to several equations described in [5]. This part of the circuit performs the addition on $P$ and $B$ only if the corresponding multiplier qubit $Q[m]$ is high.
3. Compute the Rotate block, which performs a rotate right operation with a delay of 6.
4. $m = m + 1$. 

1735
4.2 Quantum divisor circuit for particular cases

A circuit to compute a product between an integer \( A \) and \( B = 2^K+1 \) is presented in [22]. This circuit is more optimized than the previous one for this particular case, so it can be used as an Oracle if \( B = 2^K+1 \) [23]. It could be built with shifts and adds, but the mentioned circuit reduces the number of unnecessary ancillary qubits. It is only necessary to add a comparator to check if \( A = (2^K + 1)Q \). This is shown in Fig. 10. Any circuit which computes the product could be used as an Oracle, but it is important to get an optimized circuit in terms of quantum cost and delay [24].

Figure 9. Reversible multiplier circuit proposed in [21]. P is a set of auxiliary qubits. Zcin is used to propagate the carry generated from the previous qubit position.

Figure 10. Oracle circuit used to compute the division based on the circuit proposed in [22]. Several CNOT gates and a Pauli-X gates can be observed.

5 CONCLUSIONS

In this work, Grover’s search algorithm has been described step by step for Computer Engineering students and using the division as a simple case study. We have explained that the division should not be solved with Grover’s algorithm. However, this example is appropriate for educational purposes since it allows explaining the main advantages and disadvantages of the algorithm. The use of such a simple example let students be abstracted of the difficulties of quantum mechanics and quantum computation, focusing only on the particularities of Grover’s algorithm.

ACKNOWLEDGEMENTS

This work has been supported by the Spanish Science and Technology Commission (CICYT) under contract TIN2015-66680; Junta de Andalucía under contracts P11-TIC-7176 and P12-TIC-301 in part financed by the European Regional Development Fund (ERDF). G. Ortega is supported by a ‘Juan de la Cierva Incorporación’ Fellowship from the Spanish Ministry of Economy, Industry and Competitiveness. We want to thank IBM for the possibility of using their resources.

REFERENCES


