Classical Realization of Grover's Quantum Search Algorithm using Toffoli gates

Manuel-Iván Casillas-del-Llano[1] and Álvaro-Lorenzo Salas-Brito[2]

[1]Universidad Autónoma Metropolitana. Unidad Azcapotzalco. D.F., México
al210180113@alumnos.azc.uam.mx
[2]Universidad Autónoma Metropolitana. Unidad Azcapotzalco. D.F., México
asb@correo.azc.uam.mx

Abstract. Grover's algorithm is used to search for quantum data. However, this algorithm procedure is described by means of concepts and operators from quantum theory; concepts hardly known by computer scientists. In this work we propose an alternative classical computing model of Grover's algorithm, using Toffoli gates connected with elementary gates. Our model has been programmed on a high-level programming language and tested using arbitrary elements on a data set. Our results are concordant with those presented on the reference section.

Keywords: Grover algorithm, Toffoli gates, Quantum computing.

1 Introduction

A computer is a physical device that aids us to process information while running some algorithms. An algorithm is well defined procedure, with finite description, that executes some information processing task. A task of this kind can be done by means of physical processes.

At the design level of complex algorithms, it is useful and essential to work with some idealized computational model. However, while analyzing the true limitations of a computer device, especially for practical reasons, it is important not to forget the link between computing and physics Idealized models can not fully represent all the details of these computational devices.

Classical computing has several limitations. There are problems that cannot be deal with actual computing, such as the impossibility to run on polynomial time the travelling agent problem algorithm or integer factorization.

However, it has been shown that those kinds of problems can be handled by **quantum computing**. Quantum computing uses the phenomena described by quantum theory in order to process information and execute tasks faster than classical computing. Devices that process quantum information are named **quantum computers**.

Advances in Computer Science and Engineering
Research in Computing Science 58, 2012 pp. 375–390
Paper Recived 30-09-2012 and Acepted 07-11-2012

2 Grover's Algorithm

Suppose there is a non sorted data base of size N consisting (without loss of generality) of numbers from 0 to N – 1. Using traditional algorithms, we must look up for every element on the data base in order to find the desired item. The average number of steps needed is N/2, and N on the worst case scenario; therefore, searching for an element has order of $O(N)$ time complexity. However, using quantum mechanics procedures, Grover's algorithm only requires $O\left(\sqrt{N}\right)$ steps. [1]

Initially, an *n* qubit system, with $N = 2^n$ elements, is set in an equal superposition of all basis states, expressed as

$$\left|\Psi_0\right\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} \left|i\right\rangle \tag{1}$$

It is posible to search for a specific element inside this system. This particular element is defined as the **marked state**, while the remaining elements of the set are defined as the **collective state** [1]. In order to perform the searching of the marked state, two special operators C and D are used, defined as inversion and diffusion operator, respectively. Operator C has the effect to invert or change the sign of the amplitude in the marked state, and ignores the rest of the elements belonging to the collective state. When operator D is applied to the superposition of states, it increases the amplitude of the marked state, decresing the amplitude of the collective state. If we define the compound operator as $U \equiv DC$, then each operation of *U* is called an iteration. It has been shown that after *U* is repeated $O\left(\sqrt{N}\right)$ times, the probability of getting the marked state when a mesaurement is made approeaches 1 [1].

2.1 Representation of inversion and diffusion operators.

Before reading the following sections, we encourage you to read [4], where the most important operations on quantum computing are explained in great detail. Also, for a wide explanation of Grover's Algorithm insights, we recommend reading [1].

Given the superposition of states $\left|\Psi_0\right\rangle = \left(1/\sqrt{N}\right)\sum_{i=0}^{N-1}\left|i\right\rangle$, we will denote the marked state as $\left|M\right\rangle$.

Inversion operator C is defined as $C \equiv I - 2|M\rangle\langle M|$, where *I* is the identity matrix[5]. Similary, diffusion operator D is defined as $D \equiv 2|\Psi\rangle\langle\Psi| - I$.

To be able to represent these operators by means of Toffoli gates and elementary operations, it is necessary to know the effect produced by them on the superposition of states. Let's split $|\Psi_0\rangle$ into two parts: the marked state and the collective state, that is, a linear combination of $|\Psi_0\rangle$ defined as:

$$|\Psi_0\rangle = \alpha|\Psi\rangle + \beta|M\rangle \tag{2}$$

where α and β are their respective amplitudes of the collective state and the marked state.

Next, we will apply the inversion operator to this linear combination of states:

$$C(\alpha|\Psi\rangle + \beta|M\rangle) = (I - 2|M\rangle\langle M|)(\alpha|\Psi\rangle + \beta|M\rangle)$$

$$C(\alpha|\Psi\rangle + \beta|M\rangle) = \alpha|\Psi\rangle - \left(\frac{2\alpha}{\sqrt{N}} + \beta\right)|M\rangle \tag{3}$$

At this moment, C operator has changed the sign of the marked state. Next step is to apply the diffusion operator on equation (2). That is:

$$D\left[\alpha|\Psi\rangle - \left(\frac{2\alpha}{\sqrt{N}} + \beta\right)|M\rangle\right] = (2|\Psi\rangle\langle\Psi| - I)\left[\alpha|\Psi\rangle - \left(\frac{2\alpha}{\sqrt{N}} + \beta\right)|M\rangle\right]$$

$$D\left[\alpha|\Psi\rangle - \left(\frac{2\alpha}{\sqrt{N}} + \beta\right)|M\rangle\right] = \left(\alpha - \frac{4\alpha}{N} - \frac{2\beta}{\sqrt{N}}\right)|\Psi\rangle + \left(\frac{2\alpha}{\sqrt{N}} + \beta\right)|M\rangle \tag{4}$$

We have applied once the compound operator $U \equiv DC$, and so, an iteration of Grover's algorithm have been made. According to [7], the number of iterations needed to approach the amplitude of the marked state to 1 is $\lfloor \pi\sqrt{N}/4 \rfloor$.

2.2 Alternative representation of Grover's algorithm using different approaches.

Grover algorithm can be also represented by circuits using interconnected quantum gates and the Toffoli gate [8]. A circuit for 2 qubit system is shown in Figure 1.

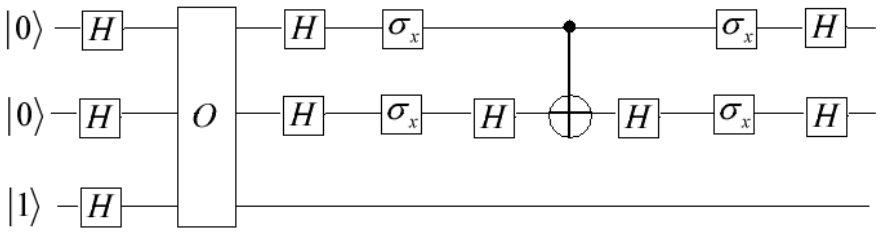


Fig. 1. A quantum circuit that implements Grover's algorithm for N = 4 elements. Pauli matrix σ_x behaves as a NOT gate. Block with letter O denotes a query to the oracle.

Another representation of Grover's algorithm can be implemented using optical approaches, as described in [16] for a system of 2 qubit elements. An implementation using two trapped atomic ion qubits for a system of 2 qubit elements is also proposed [17]. Other representations such as nuclear magnetic resonance are described in [5].

As it will be shown on the next sections, we will build a model that will also represent Grover's algorithm, but using only classical (non-quantum) elementary gates (with its limitations, see Conclusions section).

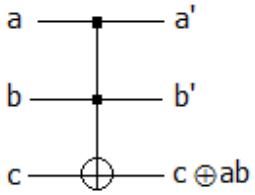
3 Toffoli gates

Toffoli gates were invented by Tommaso Toffoli [15]. Its main characteristic is that it is a universal reversible logic gate. It is a universal gate because any logic gate can be constructed by means of several Toffoli gates interconnected. It is a reversible logic gate because, given a certain output, we can obtain its corresponding input. Toffoli gates can be modeled using the billiard ball model [2]. This gate is also known as a controlled-controlled-NOT gate, because it flips the third bit on a 3-bit gate if and only if the first two bits are 1. Fig. 2a) shows Toffoli gate truth table when applied to three bits, and Fig. 2b) shows its circuit representation.

Toffoli gates are crucial for our proposed model, since it will aid us in the construction of the inversion operator C because it can detect if the marked state was found or not (see Section 4.2). The output of the inversion operator C will be zero if the element is not the marked state, and it will be 1 if the marked state was found. These outputs will be used then for further calculations.

Input			Output		
a	**b**	**c**	**a'**	**b'**	**c'**
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

a)



b)

Fig. 2. a) Toffoli gate truth table when applied to three bits. b) Circuit implementation.

4 Realization of Grover's Algorithm using Toffoli gates

Now that we have obtained the neccesary equations from the preceeding section, we now are able to model Grover's algorithm. If we look closely to Eq. 3 and 4, we can see that the operations involved are elementary additions, substractions, multiplications and divisions, such as $\alpha - 4\alpha/N - 2\beta/\sqrt{N}$. Thus, we need basic gates that perform these operations, such that this model will be constructed interconnecting classical logic gates. The elements involved in these basic operations are needed for the model, so we must supply them at the beginning of the execution. We call these elements as the **control data** of the model. Also, the **input data** will consist of the initial element list, which contains the superposition of states. Fig. 3 shows a general diagram for the proposed model.

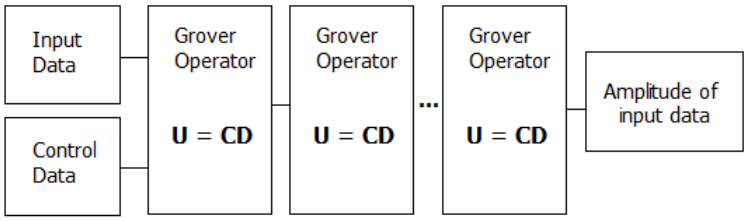


Fig. 3. General shematic of Grover's algorithm procedure. Grover operator is applied the optimum number of iterations in order to increase the amplitude of the marked state, and thus increasing its probability.

Elements from Figure 3 are described below.

Input data: It consists of the element list, which stores the marked state and the colective state.

Control data: These are fixed data that must be supplied to the algorithm before its execution. Control data consists of α, which represents the amplitude of the marked state; β, which stores the amplitude of all the collective set; and finally $1/N$ and $1/\sqrt{N}$, where N represents the total number of states on the database.

Grover operator: This operator was defined in section 2 as $U \equiv DC$.

Input data amplitudes. This is the set of the final amplitudes of the marked state and the collective state. After applying Grover operator a total of $\lfloor \pi\sqrt{N}/4 \rfloor$ iterations to the superposition of states, we expect that the amplitude of the marked state is almost 1.

4.1 Elementary gates

In order to implement Grover's algorithm using non-quantum operations through classical gates, we need to define them first. Such gates constitute the set of basic operators of the model.

Π GATE

This gate requires two input elements. It returns the product of both elements. (See Fig. 4)

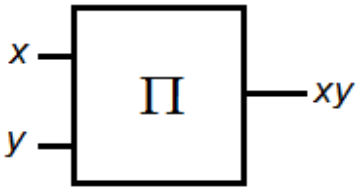


Fig. 4. π-gate: it returns the product of x times *y*

$\sum$ GATE

This gate also requires two input elements. It returns the sum of both elements. (See Fig. 5)

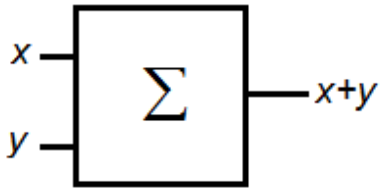


Fig. 5: ∑-gate: it returns the sum of x and *y*.

σ GATE

For this gate, two input data are needed. If the second element is set to 1, the first element will suffer a change of its algebraic sign, otherwise, it will remain unaltered. (See Fig. 6)

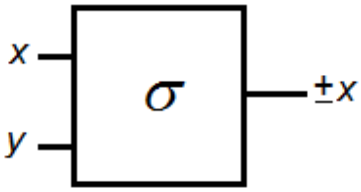


Fig. 6: σ-gate: if *y* is set to 1, this gate will change x's sign. Oterwhise, x will keep unchanged.

4.2 Modeling of Grover operator using Toffoli and elementary gates.

As stated before, Grover operator consists of the application of C operator, followed by D operator, that is, $U \equiv DC$.

Inversion operator **C** is constructed according to the element we are searching for, that is, the marked state. We constructed inversion operator using the binary representation of the element as follows: if the marked state contains zeroes, two NOT gates are put sequentially, otherwise, no gate is needed. These gates are connected by means of a **Toffoli gate**; this gate acts as follows: if every bit is set to 1, it means that the marked state was found, and it will return a 1 as an output, otherwise, it will return a zero, (that is, the marked state was not found).Suppose we want to search for the element 10 on the superposition of states (whose binary representation is 1010 for a 4-bit system). Construction for the inversion operator C using elementary and Toffoli gates is shown in Fig 7.

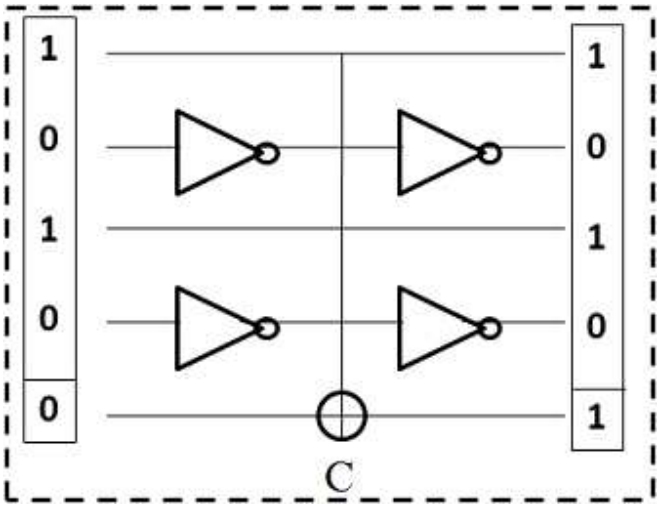


Fig. 7: Construction of inversion operator C for element 10 (which binary representation is 1010). Fifth bit is set to 0, so if every remaining bit at the end was set to 1, this bit will be also switched to 1, meaning that the marked element was found.

Diffusion operator D is needed to increase the amplitude of the marked state. According to Equation 3, this operator can be constructed using elementary gates as shown in Fig 8.

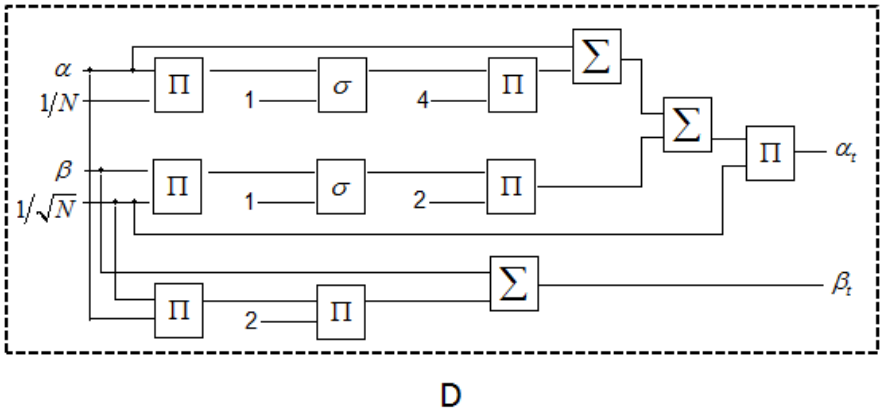


Fig. 8: Diffusion operator. Interconnection of classical gates allows to increase the amplitude of the marked state. This operator along with the inversion operator succesfully simulates Grover operator U = CD. α_t and β_t are variables that will be used on further calculations.

For illustrative purposes, let's take a closer look at the first part of Fig. 8. Eq. 4 shows that the partial chain of operations $\alpha - 4\alpha/N$ is performed. This is done by the section described in Fig. 9, taken from the diagram in Fig. 8

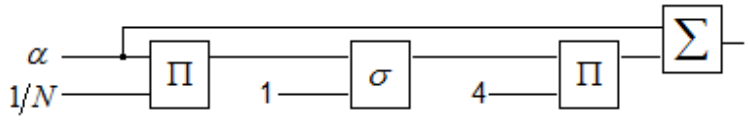


Fig. 9: Carefully following each element on the diagram, we see that the operation $\alpha - 4\alpha/N$ is successfully made.

4.3 Searching for a marked state using the model.

In order to show how the model works, let's look for the element labeled "4" stored in a list of 3 q-bit elements. First, we construct the Inversion operator C for element 4 using a Toffoli gate and NOT gates (see Fig. 10)

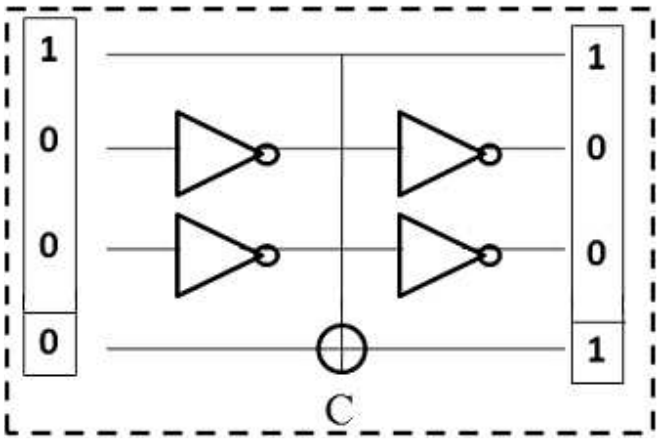


Fig. 10: Construction of inversion operator C for element 4 (which binary representation is 100). Fourth bit is set to 0, so if every remaining bit at the end was set to 1, this bit will be also switched to 1, meaning that the marked element was found.

Toffoli gates are useful on the task of finding the marked state, since it detects wherever the marked state is there or not. Since the entire list consists of 8 elements (from 0 to 7), we must apply the inversion operator for every single element belonging to the list. Fig. 11 shows the results of applying the inversion operator on element 3 and 4.

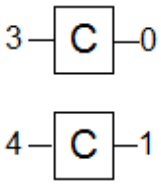


Fig. 11: When inversion operator is applied to element "3", it returns a 0 since it is not the marked state. On the other hand, C operator returns a 1 when applied to the element 4, that is, the marked state was found. This outputs are used on further calculations.

Next step is to increase the amplitude of the marked state using the diffusion operator. Fig. 12 shows how while combining the outputs from both in andversion diffusion operators we increase the amplitude of the marked state, while the collective state remains without change. Notice that only the element 3 and 4 are shown for simplicity, but this has to be done for every element on the list of elements.

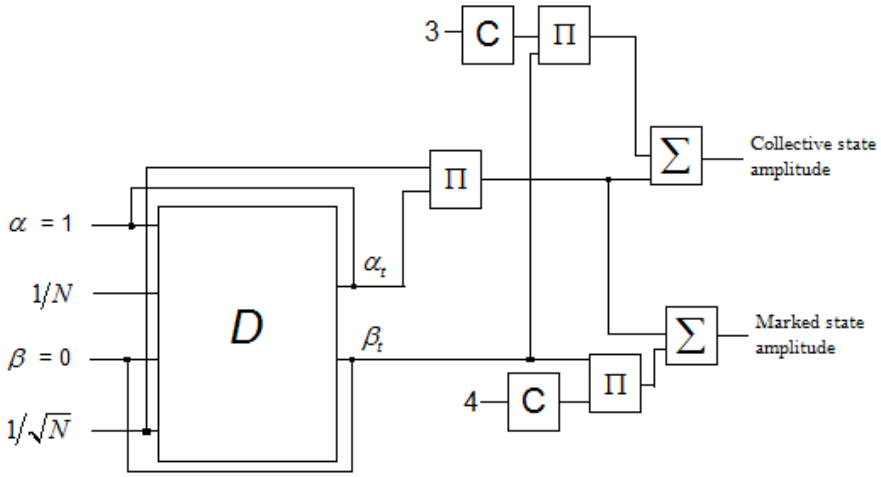


Fig. 12: With the ouput of inversion operator, it is now possible to increment the amplitude of the marked state. To accomplish this, we apply diffusion operator D and combine it with the output of inversion operator C, just like Eqs. 3 and 4 describe. Notice how the outputs from the inversion operator are multiplied by the auxiliary variable β_t and then added to the final result. Since the ouput from the inversion operator is always zero for elements of the collective state, the product is also zero and nothing is added, except from the marked state.

5 Simulation of the model using a high-level programming language.

At this stage, we can program an algorithm that simulates the processes of Grover's algorithm for quantum search. This algorithm will be programmed on a high-level programming language to run some tests, in order to validate our model.

5.1 General Algorithm

We present the general steps needed to simulate correctly Grover's Algorithm. This algorithm is based on Equations 3 and 4. Its advantage consists of the few steps needed to simulate Grover's algorithm.

```
Classic Grover Algorithm

Input
    N: the total number of elements on the system.
    B⃗ = (β_1, β_2, ..., β_{N-1}): the amplitude quoeficient vector
    of the collective state.
    α_M : the amplitude of the marked state.

Output
    B⃗ = (β'_1, β'_2, ..., β'_{N-1}): the changed amplitude quoefi-
    cient vector of collective state.
    α'_M : the new amplitude of the marked state.
Variables
    β:the value of the amplitude of the collective
    state. We can store it on a single variable since
    all the collective state will have the same ampli-
    tude through the entire algorithm.
    coef: an auxiliary variable used to store interme-
    diate values.

Begin
    //Initialize every element of the vector β_i ∈ B⃗ and
    //the amplitude of the marked state to α_M an
    //equally superposition of states
    β ← 1/√N
    α_M ← 1/√N
```

```
Repeat from i = 1 to ⌊π√N/4⌋
  //Apply the operations from Equation 3 to the
  //amplitude of the marked state and the
  //collective state.
  α'_M ← α_M − 4α_M/N − 2β/√N
  β ← 2α/√N + β

  //Store the new normalized marked state to the
  auxiliary variable.
  coef ← α'_M/√N

  //Assign each element β_i ∈ B⃗ its new amplitude.
  β_i ← coef
  //Assign the new amplitude of the marked state.
  α'_M ← coef + β

End Repeat
```

In order to valídate our proposed model, we implemented it using the programming language C++. Data input consists of the number of qubits *n* and the number of desired iterations. It let you choose the optimum number of iterations $\lfloor \pi\sqrt{N}/4 \rfloor$, or any other number of iterations.

5.2. Tests using the optimum number of iterations

Table 1 shows the results for databases built from 1 to 10 qubits, using the optimum number of iterations. The **probability of the marked state** is the probability of obtaining the marked state when a measure of the superposition of states is made. The **probability of the collective state** (or probability of failure) is the probability of obtaining one of the elements of the collective state. Since the probability of the marked state is always same, despite the element searched for, it is unnecessary to list the marked state. This means that if we are looking for the element numbered as 3 on a list of 16 elements, the probability of obtaining it after applying Grover's algorithm is 96.1319%, and that probability will not change if we are looking for the element numbered as 5 on that same list.

Table 1. Probabilities of obtaining the marked state and the collective state embedded into an *n*-qubit system, using the optimum number of iterations.

No. of qubits *n*	**Number of elements N**	**Number of iterations** $\lfloor \pi\sqrt{N}/4 \rfloor$	**Probability of the marked state**	**Probability of the collective sate**
1	2	1	50%	50%
2	4	1	100%	0%
3	8	2	94.5313%	5.4691%
4	16	3	96.1319%	3.8685%
5	32	4	99.9182%	0.0806%
6	64	6	99.6586%	0.3402%
7	128	8	99.5620%	0.4318%
8	256	12	99.9947%	0.0052%
9	512	17	99.9448%	0.0511%
10	1024	25	99.9461%	0.1023%

5.3 Tests using an arbitrary number of iterations

We made tests using an arbitrary number of iterations, instead of the optimum number. Table 2 shows the results for states from 1 to 10 qubit.

Table 2. Probabilities of obtaining the marked state and the collective state embedded into an *n*-qubit system, using an arbitrary number of iterations.

No. of qubits *n*	**Number of elements N**	**Number of iterations**	**Probability of the marked state**	**Probability of the collective sate**
1	2	3	50%	50%
2	4	3	25%	75%
3	8	6	99.9786%	0.0217%
4	16	7	36.4913%	63.5085%
5	32	7	20.9918%	79.0097%
6	64	12	0.0071%	99.9936%
7	128	10	91.9442%	8.0518%
8	256	18	54.1236%	45.7980%
9	512	20	94.2684%	5.7232%
10	1024	27	97.8187%	2.1483%

6. Analysis of results.

Table 1 shows that if we use the optimum number of iterations, probability of obtaining the marked state is almost 100% (except for the 2-qubit case, which is an expected outcome [14]). Particulalry, Table 1 shows that for a 3-qubit system, a probability of 94.5313%, which is also a result obtained at [1]. On Table 2 it is shown that unexpected results are obtained if a different number of iterations than the optimal is used. There is no way to know if there is an improvement of getting a better probability for the marked state. For example, we again can see on Table 1 that for a 3-qubit system, we get a probability of 94.5313% using the optimum number of iterations (in this case, 2); however, Table 2 shows that we get a better probability (about 99.9786%) if we use 6 iterations. This does not necessarily mean that if we use a bigger number of iterations, the probability of obtaining the marked state will be better. For example, calculating the probability of a 6-qubit system yields a 99.6586% for the marked state, using the optimum number of iterations (6). But if we use twice that number of iterations, Table 2 shows that we get a quite bad result (0.0071%), meaning that the probability of failure (that is, the probability of obtaining an element of the collective state when making a measure on the system) is about 99.9936%. Similar results can be seen when we are measuring probabilities on a database with 16 and 32 elements, or 4 and 5-qubits systems, respectively.

7. Conclusions

As powerful as classical computing is, it has several limitations. Research on unconventional computing, such as biological inspired computing and quantum computing is made in order to deal with these limitations. Despite that a lot of research work has been made on quantum computing, there is still so much work to do. There are a lot of obstacles that must be solved before a complete physical implementation of a quantum computer can be made. To be able to analyze and foresight the inner concepts and advantages of quantum computing, it is necessary to have a good understanding about theoretical quantum physics. Such knowledge is often lacked by classical computer scientists. For the aforementioned reasons, we need an alternative model capable of explaining the quantum processes (in this case, quantum data search) on terms that can be understood by classical computing developers and researches. In this work, we have presented an alternative theoretical implementation of Grover's quantum search algorithm. With this model, we are able to analyze the behavior of Grover's algorithm on each iteration, without knowing quantum physics or owning an actual quantum computer. With the analysis of the main quantum operators of the algorithm, an equivalent model using only elementary operators by means of interconnecting Toffoli gates and elementary gates was built. This model was implemented on a high level programming language using equations 2 and 3. Many simulations were made using different databases with several elements. Results obtained coincide with those obtained on [1] and [13].

It is important to mention that our proposed model shows the behavior of the Grover's algorithm for quantum data. However, since we are implementing a quan-

tum algorithm on a classical computer, advantages acquired from quantum theory are lost, such as superposition of states, that is, the process that affects the elements on the database is made one by one, and not at the same time like Grover's algorithm quantum implementation. Our model only shows how the probability of obtaining the marked states increases with the algorithm, but it behaves just as a classical algorithm.

References

1. C. Lavor, L.R.U. Mansur, R. Portugal. "Grover's Algorithm: Quantum Database Search", arXiv:quant-ph/0301079v1.
2. E. Fredkin and T. Toffoli. "Conservative logic", International Journal of Theoretical Physics, 21, pp. 219–253, 1982.
3. A. Steane and E. Rieffel. "Beyond bits: the future of Quantum Information Processing", IEEE Computer Society, Vol. 33, pp.38-45, January 2000.
4. N. Mermin. "Lecture Notes on Quantum Computation", Cornell University, Physics, pp-481-681, 2006.
5. M. Nielsen and I. Chuang. "Quantum Computation and Quantum Information", Cambridge University Press, 2000.
6. A. Childs and W. Dam. "Quantum Algorithms for Algebraic Problems", Reviews of Modern Physics, Vol. 82, No. 1, pp. 1-52, January-March 2010.
7. P. Kaye, R. Laflamme, M. Mosca. "An Introduction to Quantum Computing", Oxford University Press, 2007.
8. G. Benenti, G. Casati, G. Strini. "Principles of Quantum Computation and Information", World Scientific, 2004.
9. S. Lomanco. "A lecture on Shor's Quantum Factoring Algorithm", arXiv:quant-ph/0010034v1.
10. I. L. Chuang, N. Gershenfeld, and M. Kubinec, ''Experimental implementation of fast quantum searching'', Physical Review Letters, 80, pp. 1050–1053, 1999.
11. Z. Diao, "Exactness of the Original Grover Search Algorithm", arXiv:quant-ph/1010.3652v1.
12. L. Grover, "A fast quantum mechanical algorithm for database search", arXiv:quant-ph/9605043
13. S. Lomanco. "A lecture on Grover's Quantum Search Algorithm", arXiv:quant-ph/0010034v1
14. K.A. Brickman, P.C. Haljan, P.J. Lee, M. Acton, L. Deslauriers, C. Monroe, "Implementation of Grover's Quantum Search Algorithm in a Scalable System", arXiv:quant-ph/0510066v2
15. T. Toffoli, "Reversible computing," in Proceedings of the 7th Colloquium on Automata, Languages and Programming. Springer-Verlag London, UK, 1980, pp. 632–644.
16. P.G. Kwiat, J.R. Mitchell, P. D. D. Schwindt, A. G. White, "Grover's search algorithm: An optical approach", arXiv:quant-ph/9905086v1
17. K.A. Brickman, P.C. Haljan, P.J. Lee, M. Acton, L. Deslauriers, C. Monroe, "Implementation of Grover's Quantum Search Algorithm in a Scalable System", arXiv:quant-ph/0510066v2