Introduction to Quantum Computing through Shor's Factorization Algorithm and Grover's Search Algorithm

Andrés Sicard
asicard@sigma.eafit.edu.co

Mario Vélez
mvelez@sigma.eafit.edu.co

Fredy Gallego
fgallego@delta.eafit.edu.co
student assistant

Cathalina Gutiérrez
cgutierr@delta.eafit.edu.co
student assistant

Luis Ospina
lospinad@delta.eafit.edu.co
student assistant

Carlos Pérez
cperezmo@delta.eafit.edu.co
student assistant

Universidad EAFIT; Medellín, Colombia

12th December 1999

Abstract

We give a brief introduction about quantum computing, we introduce some basic concepts and then we present Shor's algorithm and Grover's Algorithm and their simulation in $MATHEMATICA^{TM}$ and QCL respectively.

1 Introduction

Quantum Computing is a growing field of investigation that involves primarily physic and theoretical computation as long as this computation takes advantage of the properties of quantum mechanics, where the parallelism implicit in the quantum operations is the point in which the computation acquires its maximal power.

According to some authors "If currents trends continue, by the year 2020 the basic memory of a computer will be the size of individual atoms"[9, p.1]. This affirmation together with other more like the solution to the heat dissipation bring us to see the Quantum Computing like an inevitable step in the same advance of the technology.

Through quantum parallelism, the computation offers us the attractive possibility of breaking complexities. A good example of this is the Shor's algorithm [7], which reduces the complexity of the fastest classical algorithm for factoring an integer n in exponential time to polynomial time [7]. This complexity breakage in Shor's algorithm is the reason because the Quantum Computing can break unbreakable codes like RSA cryptosystem. Another example is Grover's Algorithm which finds a key in an unsorted data base and which also reduces the complexity, in this case, the reduction is from a polinomial complexity to a minor polinomial complexity.

We are going introduce to Quantum Computing through the Shor's Algorithm and Grover's Algorithm; in section 2 we present some fundamental concepts for them. This concepts are at level of people of computer science that have basic concepts about statistics and linear algebra. We explain generally steps of the algorithms and finally we will show the simulation of Shor's Algorithm in *MATHEMATICA*^{TM} and Grover's Algorithm in QCL [5].

We hope the reader, after reading this article, understands the main ideas presented on it, feels able to read other papers about Quantum Computing, and also be stimulated to investigate on this field.

2 Preliminar Concepts

These concepts are only the basic concepts for this article. For more elements about the preliminar concepts of Quantum Computing refer to [1, 6, 8, 9].

2.1 Quantum Registers and Normalization

The minimum information unit in quantum computing is the qubit which is analogous to the classical bit and is represented by:

Classical Bit	Quantum Bit (Qubit)
0	$\|0\rangle$
1	$\|1\rangle$

$|0\rangle$ is called the ket zero and $|1\rangle$ is called the ket one. This notation is known as Dirac bra-ket notation which is the most usual and natural for quantum computing.

But there is a difference between bits and qubits. A bit can only express one of two states (0 or 1), while a qubit is a vector and as a vector it can be expressed as linear combinations of the basis vectors $\{|0\rangle, |1\rangle\}$. The vector representation of qubits is:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{1}$$

Let $|\Phi\rangle$ be a qubit, then its state is represented by:

$$\begin{aligned} |\Phi\rangle &= a|0\rangle + b|1\rangle \qquad |a|^2 + |b|^2 = 1 \qquad a, b \in \mathbb{C} \\ &= a\begin{pmatrix} 1 \\ 0 \end{pmatrix} + b\begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} a \\ b \end{pmatrix}. \end{aligned} \tag{2}$$

The restriction $|a|^2 + |b|^2 = 1$ means that the state must be normalized (magnitude=1).

A simple qubit is not very useful. For representing multiple qubits we use tensor product ($\otimes$). The tensor product between $|0\rangle$ and $|1\rangle$ is:

$$\begin{aligned} |0\rangle \otimes |1\rangle &= |0,1\rangle \\ &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}. \end{aligned} \tag{3}$$

Let $|\Phi\rangle = a|0\rangle + b|1\rangle$ and $|\Psi\rangle = c|0\rangle + d|1\rangle$ be two qubits. General tensor product $|\Phi\rangle \otimes |\Psi\rangle$ is:

$$|\Phi\rangle \otimes |\Psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} \tag{4}$$

$$= \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix} = ac \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + ad \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + bc \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + bd \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{5}$$

$$= ac|0,0\rangle + ad|0,1\rangle + bc|1,0\rangle + bd|1,1\rangle. \tag{6}$$

For 2 qubits we have 4 basis vectors $\{|0,0\rangle, |0,1\rangle, |1,0\rangle, |1,1\rangle\}$, and in a general way, for n qubits we have 2^n basis vectors.

A quantum register is a sequence of qubits such as $|0,0,1,0\rangle$. For simplicity we will use decimal notation; for example, the vector $|1,0,1\rangle$ in binary is equal to vector $|5\rangle$ in decimal.

A register can be divided into 2 or more registers. For example, the 8-qubit register $|a_1, a_2, \ldots, a_8\rangle$ where $a_i \in \{0,1\}$, could be divided into a register of, let's say 6 qubits, and a register of 2 qubits: $|register1, register2\rangle$. It can also be expressed as $|a_1, a_2, a_3, a_4, a_5, a_6\rangle |a_7, a_8\rangle$.

2.2 Superposition of states: Quantum Parallelism

In order to do computations, quantum computing uses linear operators. A linear operator U is a transformation where for the vectors x, y and the number a we have:

$$U(ax) = aU(x), \tag{7}$$

$$U(x+y) = U(x) + U(y). \tag{8}$$

And because of this, the appliance of a computation on the system will simultaneously apply the computation above each basis vector. For example, the computation represented by the linear operator U applied on the state given by equation (2):

$$U \,|\, \Phi\rangle = U\Big[a \,|\, 0\rangle + b \,|\, 1\rangle\Big] = aU \,|\, 0\rangle + bU \,|\, 1\rangle \,. \tag{9}$$

This shows the effect of quatum parallelism.

Linear operators are represented by matrices. Operation between operators and vectors is matrix multiplication. An example of linear operator is the operator NOT which works as follows:

$$NOT \,|\, 1\rangle = |\, 0\rangle \,, \qquad\qquad NOT \,|\, 0\rangle = |\, 1\rangle \,. \tag{10}$$

And for the equivalence between kets and vectors given by equation (1):

$$NOT \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \qquad\qquad NOT \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \tag{11}$$

So as the operator is a matrix, and as the operation between operators and vectors is not other than matrix multiplication:

$$NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{12}$$

There is another restriction about operators: they must be unitary. An operator U is unitary if $U^\dagger = U^{-1}$. $U^\dagger$ is the conjugate transpose of the matrix U. If we define U as:

$$U = \begin{pmatrix} a+ib & c+id \\ e+if & g+ih \end{pmatrix}, \tag{13}$$

then, its conjugate transpose is:

$$U^\dagger = \begin{pmatrix} a-ib & e-if \\ c-id & g-ih \end{pmatrix}. \tag{14}$$

Because U is unitary computation is reversible. So if we want to apply a function on the system, we must make it reversible. The non-reversible function $f(x)$ made reversible is:

$$f(x, y) = (x, y \oplus f(x)), \tag{15}$$

where $\oplus$ is the exclusive-or operation. And specially for $y = 0$,

$$f(x, 0) = (x, 0 \oplus f(x)) = (x, f(x)), \tag{16}$$

because $(0 \oplus a)$ always depends on a. We can see this in the exclusive-or table:

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Now let F be the operator that implements the function $f(x)$ besides we have the double register $|\, x, y\rangle$. After applying the operator over the state:

$$|\, \Psi\rangle = \frac{1}{2}[|\, 0, 0\rangle + |\, 1, 0\rangle + |\, 2, 0\rangle + |\, 3, 0\rangle] = \frac{1}{2}\sum_{i=0}^{3} |\, i, 0\rangle\,, \tag{17}$$

we have:

$$\begin{aligned} \frac{1}{2}F\Big[\sum_{i=0}^{3} |\, i, 0\rangle\Big] &= \frac{1}{2}\sum_{i=0}^{3} |\, i, f(i)\rangle \\ &= \frac{1}{2}\Big[\, |\, 0, f(0)\rangle + |\, 1, f(1)\rangle + |\, 2, f(2)\rangle + |\, 3, f(3)\rangle\,\Big]. \end{aligned} \tag{18}$$

2.3 The Hadamard Transformation

This transformation assigns at the both possible qubit states the same probability [6, 8]. It is used to initialize the system although it has other uses as we will see ahead. It only operates on one qubit and it is defined as follows:

$$\mathrm{H}\,|\,0\rangle \rightarrow \frac{1}{\sqrt{2}}(|\,0\rangle + |\,1\rangle), \tag{19}$$

$$\mathrm{H}\,|\,1\rangle \rightarrow \frac{1}{\sqrt{2}}(|\,0\rangle - |\,1\rangle). \tag{20}$$

The matrix for this transformation is given by:

$$\mathrm{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{21}$$

When it is applied to $|\,0\rangle$, H creates a superposed state $\frac{1}{\sqrt{2}}(|\,0\rangle + |\,1\rangle)$.

2.3.1 The *Walsh - Hadamard* Transformation

The Hadamard transformation applied to n qubits is known as the *Walsh - Hadamard* transformation and produces a superposition of all 2^n possible states. This transformation is defined as follows:

$$\mathrm{W} = \mathrm{H} \otimes \mathrm{H} \otimes \ldots \otimes \mathrm{H} \tag{22}$$

$$\begin{aligned} \mathrm{W}\,|\,0,0,0,\ldots,0\rangle &= \frac{1}{\sqrt{2^n}}[(|\,0\rangle + |\,1\rangle) \otimes (|\,0\rangle + |\,1\rangle) \otimes \ldots \otimes (|\,0\rangle + |\,1\rangle)] \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |\,x\rangle\,. \end{aligned} \tag{23}$$

As an example, the *Walsh - Hadamard* transformation matrix for 2 qubits is defined as follows:

$$\begin{aligned} \mathrm{W} &= \mathrm{H} \otimes \mathrm{H} \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ &= \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}. \end{aligned} \tag{24}$$

2.4 Measurement and Probability

Quantum system is a black box, we can not measure the system without disturbing it [1]. If the system is in a superposition of states, after measurement, the system is forced to be in only one basis state. In what state will be the system after measurement? We can not know that, but we can know the probability of being in any of the states.

The probability of being in a state after measurement is given by the coefficients of the basis vectors. So, if we measure the state given by equation (2), the probability of getting the value 0 and being in state $|\,0\rangle$ is $|a|^2$ and the probability of getting 1 and being in state $|\,1\rangle$ is $|b|^2$, it means:

$$P(|\,0\rangle) = |a|^2 \qquad \text{and} \qquad P(|\,1\rangle) = |b|^2. \tag{25}$$

But we do not need to measure all the system. If we have a double register, we can measure either the first or the second one. For example, if we have a double register of one qubit each register, and we have the state:

$$|\,\Phi\rangle = a\,|\,0,0\rangle + b\,|\,0,1\rangle + c\,|\,1,0\rangle + d\,|\,1,1\rangle\,, \tag{26}$$

the probability of getting 0 after measuring the second register is $|a|^2 + |c|^2$ and the probability of getting 1 is $|b|^2 + |d|^2$. Suppose we get 0, so after measurement the system collapses to [6, 8, 9]:

$$|\,\Phi'\rangle = \frac{1}{\sqrt{|a|^2 + |b|^2}}\Big[a\,|\,0,0\rangle + c\,|\,1,0\rangle\Big], \tag{27}$$

where the term $\frac{1}{\sqrt{|a|^2+|b|^2}}$ is for state normalization.

2.5 Quantum Fourier Transform: QFT

Fourier Transform (FT) in a few words, projects a function from the time domain to the frequency domain; so, functions of period T are projected into functions that have zeros in all values except at multiples of the frequency $f = \frac{1}{T}$ [6]. Essence of success of Shor's algorithm and most of the known quantum algorithms rests in this feature of Fourier Transform.

The Quantum Fourier Transform (QFT) is a variant of the Fast Fourier Transform (FFT), and only gives approximated values for functions with periods that are not powers of 2.

The linear operator for the QFT is [1, 6, 7, 9]:

$$QFT\,|\,x\rangle \rightarrow \frac{1}{\sqrt{2^m}} \sum_{c=0}^{2^m-1} \exp\{\frac{2\pi i c x}{2^m}\}\,|\,c\rangle\,, \tag{28}$$

where m is the number of qubits of the register $|\,x\rangle$.

[1] This is a consequence of the quantum mechanics properties.

3 Shor's Algorithm for Factoring Integers on a Quantum Computer

Shor's algorithm factors an integer in polynomial time, reducing the number of steps from $exp(c(\log n)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}})$ (for some constant c) to $O((\log n)^2(\log\log n)(\log\log\log n))$ [7]. In order to do that it uses quantum parallelism. But we know that we cannot measure the quantum state without changing it and losing information. For that reason we must use something different: to use a periodic function and get its period with help of the Fourier Transform.

The periodic function we are going to use is [7]:

$$f(x) = a^x \mod N, \tag{29}$$

where N is the number we attempt to factor and a is a random number between $1 < a < N$ such that $gcd(N, a) = 1$. [2]

Let r be the period of the function (29), so we have that [3]:

$$a^r \equiv 1 \mod N, \tag{30}$$

$$(a^r - 1) \equiv 0 \mod N, \tag{31}$$

$$(a^{r/2} + 1)(a^{r/2} - 1) \equiv 0 \mod N. \tag{32}$$

This means that dividing $(a^{r/2} - 1)(a^{r/2} + 1)$ by N gives a remainder of 0. So, if r is even and $a^{r/2} \not\equiv \pm 1 \mod N$ at least $(a^{r/2} - 1)$ or $(a^{r/2} + 1)$ must have a non-trivial common factor with N and we can find it by calculating $gcd(a^{r/2} + 1, N)$ and $gcd(a^{r/2} - 1, N)$.

The algorithm proceeds as follows [1, 6, 7, 9]:

1. Find a random number a for the function (29) and create a superposition of the integers that will become the arguments of the function. So, first we have the state $|\,0, 0\rangle$ [4], then we create a superposition on the first register and finally we apply the function on the state:

$$|\,0, 0\rangle \rightarrow \frac{1}{\sqrt{M}} \sum_{i=0}^{M-1} |\,i, 0\rangle \rightarrow \frac{1}{\sqrt{M}} \sum_{i=0}^{M-1} |\,i, f(i)\rangle\,, \tag{33}$$

   where $M = 2^m$ and m is the number of qubits on the first register. M must be of order $O(N^2)$ (N is the number we attempt to factor) so the approximation given by the QFT will be good enough for the algorithm to work. So, we will choose an M such that $N^2 \leq M < 2N^2$.

[2] gcd(N,a) is the greatest common divisor of N and a.

[3] $a \equiv b \mod N$ is called a congruence an it means that $(a - b)|N$ or $(a - b) \mod N = 0$. Dividing $(a - b)$ by N gives a remainder of 0.

[4] $|\,0, 0\rangle$ is a decimal representation of the registers. First and second register will have more than one qubit, depending on the number we attempt to factor.

2. Measure the second register and obtain the value u. The new state is [1, 9]:

$$\frac{1}{\sqrt{\|A\|}} \sum_{i=0}^{\|A\|-1} |\, ir + s, u\rangle\,, \tag{34}$$

where $A = \{a/f(a) = u\}$, $\|A\|$ is the number of elements in the set A and s is the minimum number such that $f(s) = u$.

3. Apply Quantum Fourier Transform on first register. This give us a function with peaks of probability on integers closed to multiples of $\frac{1}{r}$. In our case more exactly multiples of $\frac{M}{r}$.

4. Measure first register. We will obtain with high probability a value c closed to a multiple of $\frac{M}{r}$, let's say $\lambda\frac{M}{r}$. This way we can get the period of the function and solve the problem.

5. Repeat the procedure if necessary. There are some reasons to repeat the algorithm [6]:

   (a) The measured value c was not closed enough to a multiple of $\frac{M}{r}$.
   (b) $gcd(r, \lambda) \neq 1$.
   (c) The algorithm yields N as N's factor.
   (d) The period r is odd.

3.1 An Example: Factoring 15

1. First we choose a random number, let's say 2 and construct our function: $f(x) = 2^x \mod 15$. We create a superposition of the numbers $i = 1, 2, \ldots, 15$ ($M = 16 = 2^4$). In this case we choose $M = 16$ for simplicity on the example.

$$|\, 0, 0\rangle \rightarrow \frac{1}{\sqrt{16}} \sum_{i=0}^{16-1} |\, i, 0\rangle = \frac{1}{4}\Big[|\, 0, 0\rangle + |\, 1, 0\rangle + |\, 2, 0\rangle + \cdots + |\, 15, 0\rangle\Big]. \tag{35}$$

Then we apply the function (see figure (1)):

$$\begin{aligned} \rightarrow \frac{1}{4}\sum_{i=0}^{15} |\, i, f(i)\rangle &= \frac{1}{4}\sum_{i=0}^{15} \big|\, i, 2^i \mod 15\big\rangle \\ &= \frac{1}{4}\Big[|\, 0, 1\rangle + |\, 1, 2\rangle + |\, 2, 4\rangle + |\, 3, 8\rangle + |\, 4, 1\rangle + \cdots + |\, 15, 8\rangle\Big]. \end{aligned} \tag{36}$$

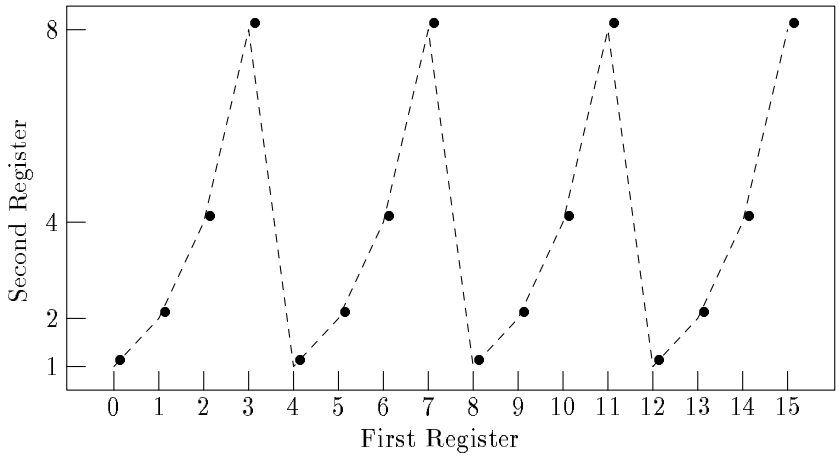

Figure 1: After applying function $f(x)$

2. We can see that the period r of the function is 4. We measure the second register and obtain a value, let's say 4 (this does not mean that we have obtained the period). Now we have the set $A = \{a/f(a) = 4\} = \{2, 6, 10, 14\}$. Quantum state collapses to (see figure (2)):

$$\begin{aligned} \rightarrow \frac{1}{\sqrt{4}} \sum_{i=0}^{4-1} |\, 4i+2, 4\rangle &= \frac{1}{2} \sum_{i=0}^{3} |\, 4i+2, 4\rangle \\ &= \frac{1}{2}\Big[|\, 2, 4\rangle + |\, 6, 4\rangle + |\, 10, 4\rangle + |\, 14, 4\rangle\Big]. \end{aligned} \tag{37}$$

3. Then we apply the Quantum Fourier Transform, QFT, on first register (see figure(3)). We will not care more about second register.

$$\begin{aligned} \rightarrow & \frac{1}{2} QFT\Big[|\, 2\rangle + |\, 6\rangle + |\, 10\rangle + |\, 14\rangle\Big] \\ & = \frac{1}{2}\Big[|\, 0\rangle - |\, 4\rangle + |\, 8\rangle - |\, 12\rangle\Big] \end{aligned} \tag{38}$$

$$P(|\, 0\rangle) = P(|\, 4\rangle) = P(|\, 8\rangle) = P(|\, 12\rangle) = \frac{1}{4}. \tag{39}$$

4. Finally we measure first register an obtain, let's say $c = 12$. So,

$$\lambda \frac{M}{r} = c, \qquad \frac{\lambda}{r} = \frac{c}{M}, \qquad \frac{\lambda}{r} = \frac{12}{16}, \qquad \frac{\lambda}{r} = \frac{3}{4}, \qquad \lambda = 3, \qquad r = 4. \tag{40}$$

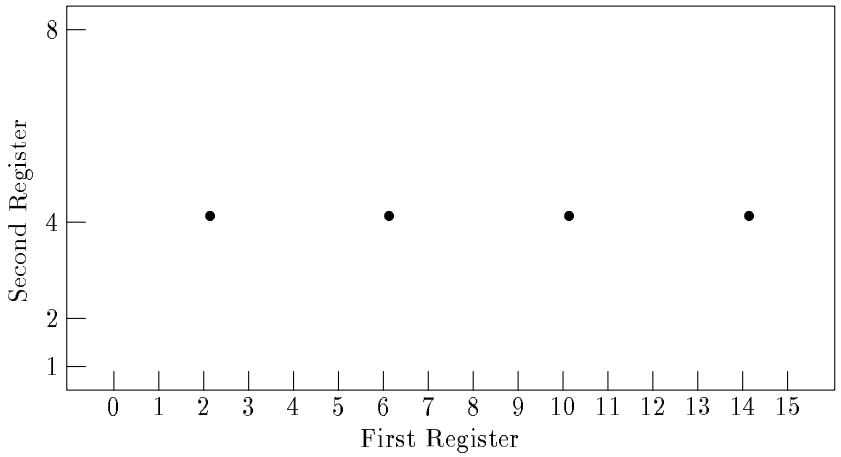

Figure 2: After first measure. Value gotten $c = 4$

Our posible factors are:

$$gcd(2^4 - 1, 15) = gcd(3, 15) = 3, \tag{41}$$

$$gcd(2^4 + 1, 15) = gcd(5, 15) = 5, \tag{42}$$

$$3 \times 5 = 15. \tag{43}$$

We have found the factors of 15 in our first try, but if the algorithm had failed by any of the reasons mentioned above, with a few repetitions of the algorithm we would surely obtain the factors with a high probability.

3.2 Simulation on *MATHEMATICA*^{TM}

The simulation was written on *MATHEMATICA*^{TM}, because it provides many of the operators required by the simulation and it makes easier to manipulate numerical and symbolic expressions.

The simulation we are going to use was implemented by the Colin P. Williams and Scott H. Clearwater [9], this simulates the kets and the operators through vectors and matrices. Other features as the measurement and parallelism are simulated by pseudo-random algorithms and pseudo-parallel algorithms what causes simulation to be slower than made by classical algorithms.

Now, we will see the factorization of 21 by the simulator.
We just need to type the line:

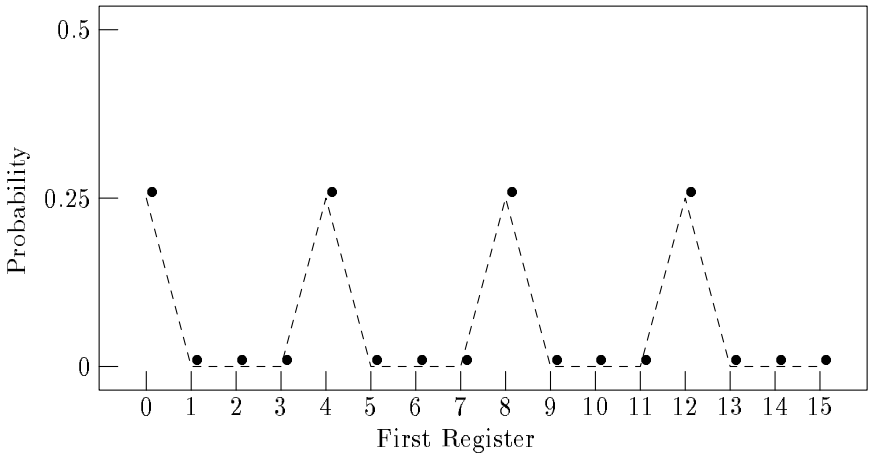

Figure 3: After applying Fourier Transform on first register

```
1  RunShorsAlgorithm[21]
```

And let the algorithm works. The first step is to get a random number for the function (lines 2-4) and then get a number of order $0(N^2)$ which will be our M (lines 5-8). At this point simulation differs from our presentation of Shor's Algorithm because our M is always a power of 2, while the M of the simulation is not.

```
2  Step 1: Picking a random integer a, 1 < a < 21, that is co-prime
3  to 21.
4  >> Picked a == 8.
5  Step 2: Picking a "smooth" M, i.e. an integer of order O(N^2) ~
6  441 that has small prime factors.
7  >> Picked M == 450.
```

Now it follows to compute the period of the function. But it differs again from our presentation. The simulation repeats this step $O(\log M)$ times, while we just repeat the algorithm if necessary.

```
8                                    x
9  Step 3: Computing the period of 8  mod 21. Repeat steps 3(a)-3(g)
10 O(log(M)) times.
```

Lines 12 through 36 comprend the first point of our presentation of Shor's algorithm.

```
Step 3(a): Initially, place Reg1 and Reg2 in the state |0,0>.

Step 3(b): Loading Reg1 with a superposition of all integers in
the range 0 to M-1, i.e. creating the superposition 1/Sqrt[M]
Sum[ket[x], {x,0,M-1}

This puts Reg1 in the state: Reg1 = 0.0471405 ket[0] + 0.0471405
ket[1] + 0.0471405 ket[2] +
   0.0471405 ket[3] + 0.0471405 ket[4] + 0.0471405 ket[5] +
   0.0471405 ket[6] + 0.0471405 ket[7] + 0.0471405 ket[8] +
   0.0471405 ket[9] + 0.0471405 ket[10] + 0.0471405 ket[11] +
   <<432>> + 0.0471405 ket[444] + 0.0471405 ket[445] +
   0.0471405 ket[446] + 0.0471405 ket[447] + 0.0471405 ket[448] +
   0.0471405 ket[449]

Step 3(c):
                    x
Loading Reg2 with 8  mod 21 This puts Reg2 in a state representing
a superposition of the integers: {1, 8, 1, 8, 1, 8, 1, 8, 1, 8, 1
                  ...  8, 1, 8, 1, 8, 1, 8, 1, 8, 1, 8}
```

Then, as in second point of the presentation, it follows the measurement of the second register.

```
Step 3(d): Measuring the state of Reg2 measureReg2 = 8 The
measurement of Reg2 has a side effect on the state of Reg1. In
fact the state of Reg1 is projected into: projectReg1 = 0.0666667
ket[1] + 0.0666667 ket[3] +
   0.0666667 ket[5] + 0.0666667 ket[7] + 0.0666667 ket[9] +
   0.0666667 ket[11] + 0.0666667 ket[13] + 0.0666667 ket[15] +
   0.0666667 ket[17] + 0.0666667 ket[19] + 0.0666667 ket[21] +
   0.0666667 ket[23] + <<207>> + 0.0666667 ket[439] +
   0.0666667 ket[441] + 0.0666667 ket[443] + 0.0666667 ket[445] +
   0.0666667 ket[447] + 0.0666667 ket[449]
```

Simulation continues applying Fourier Transform on first register (Point 3 of the explanation).

```
Step 3(e): Computing discrete Fourier transform of the contents
 of Reg1.
This may take several minutes! (Remember you are watching a
classical simulation of a quantum
  algorithm).
The Fourier transform puts Reg1 in the state:
```

```
47
48  Fourier = 0.707107 ket[0] - 0.707107 ket[225]
```

It follows the measurement of first register. As we mention above the algorithm is repeat several times, each time getting a value near to a multiple of $\frac{M}{r}$.

```
49  Steps 3(f)&3(g): Sample from Fourier spectrum created in Reg1 This
50  entails repeating the previous steps & measuring Reg1 each time.
```

The values gotten from the repetition of the algorithm.

```
51  Found samples: {ket[225], ket[0], ket[0], ket[0], ket[225],
52  ket[0],
53     ket[0], ket[0], ket[225], ket[0], ket[0], ket[225]}
```

Next step is to get the period of the function.

```
54  Step 4: Extract the period, r, from the given samples. Each sample
55  provides a value c Finding the closest rational to c/M, whose
56  denominator is less than N. Each such rational has the form
57  lambda/r, where r is the period. Period was r=2
```

And finally, we have achieved our goal: to get the factors of 21.

```
58  Step 5: Obtain the factors of N from the period, r. Computing the
59  factors of n from GCD[x^(r/2) - 1, N] and GCD[x^(r/2) + 1, N] In
60  this case, a=8, r=2, N=21 GCD[8 - 1, 21] = 7 GCD[8 + 1, 21] = 3 So
61  the factors of N=21 are 7 and 3. {7, 3}
```

4 Searching in an unsorted data base

We can imagine an unsorted data base, something like a phone book, unsorted because we know the telephone number and we are looking for the name. Searching for this name in a classical way would suppose a minimum amount of $\frac{N}{2}$ queries[5], N is the number of elements in the data base.

When we have a data base of 4 elements, the amount of queries to find an element is 2, but if the number of elements is 1.000.000, the amount of queries is 500.000, this is a very large number when we are talking

[5] Generally $\frac{N}{2}$ is the amount of queries we need in the average, but Grover refer it as the minimal [2].

about data base transactions; quantum computation use some properties that let us to reduce this large number from $\frac{N}{2}$ to $\sqrt{N}$, for example: in the data base of 4 elements, the amount of queries is the same: 2, but in the data base of 1.000.000 elements, the amount of queries now is only 1.000, this means a substantial improvement in searching time.

4.1 Grover's Algorithm

The problem that seeks to solve this algorithm is the following one:

We have a data base that has $N = 2^n$ elements: S_1, S_2,...,S_N. This elements are represented as n bits chains. If one of the elements, called S_v, complies with the condition $C(S_v) = 1$, i.e, is the element that we are seeking and for the other states this same condition is equal to 0, i.e. $C(S_{i \neq v}) = 0$, the problem is to identify the state S_v.

In the implementations that actually exists of Grover's algorithm, is not really implemented the complete data base, as if we only have the list of telephones, nameless, and to the end of the algorithm, what we obtain is the position in which is located the element.

We suppose a list of 4 elements (by simplicity). We will call *marked element* to the telephone number that we are looking for. This list is implemented with 2 qubits, being the 4 possible combinations of 0 and 1 each one an element.

The steps of the algorithm are:

1. Initiate the system to the following distribution: $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |\, x\rangle$ each one of the N states remains with the same probability amplitude; this operation is carried out with the *Walsh - Hadamard* transformation.
2. The following operations are repeated $\mathbf{Round}[\frac{\pi}{4}\sqrt{N}]$ times[6]:
   (a) Apply the diffusion transformation D: Diffusion transformation was implemented thus: D = WRW, where W is the *Walsh - Hadamard* transformation matrix and R is the rotation matrix, that is defined as follows [2]:
   $$\begin{aligned} R_{ij} &= 0 \text{ if } i \neq j, \\ R_{ii} &= 1 \text{ if } i = 0, \\ R_{ii} &= -1 \text{ if } i \neq 0. \end{aligned} \tag{44}$$
   (b) Apply a transformation on each element of the system:
   $$\mathrm{I}_{x_a} = \begin{cases} -|\, x\rangle & \text{if } x = x_a, \\ |\, x\rangle & \text{if } x \neq x_a, \end{cases} \tag{45}$$
   where $a \in \{0, ..., N-1\}$, i.e., x_a is the marked element.
   Then, we are going to invert the probability amplitude of the marked element in each iteration. What is achieved with this step is that at the end of iterations, the probability amplitude of the marked element is the biggest.

[6] In all references the amount of iterations appears as $\sqrt{N}$, but Grover says that the number of iterations is $\frac{\pi}{4}\sqrt{N}$ [3].

3. The resultant state is measured. This final state will be the element S_v, the only one that will have a probability of at least $\frac{1}{2}$.

The number of iterations (**Round**$[\frac{\pi}{4}\sqrt{N}]$) of the operations should be exact, because this is what guarantees the probability of the marked element to be greater than $\frac{1}{2}$ after the iterations. When the operations are not applied the exact number of times, the marked element remains with a probability less than the remainder of the possible states, whether that the number of iterations to be greater or smaller than $\frac{\pi}{4}\sqrt{N}$.

The Grover's iteration, that consists of the Diffusion matrix and the I_{x_0} operation defined previously, we are going to apply it as an operator $\mathrm{Q} = -I_{x_0}\mathrm{WI}_0\mathrm{W}$ defined in [4], that consists of the diffusion matrix, where R is replaced with I_0, an operation that we will define later.

4.2 Development of Grover's Algorithm: An example with 2 qubits

We are going to develop the algorithm with $|\,0,1\rangle$ as the marked element, this means that to the end, the result of measure will be 1, for our example have operator W given by equation (24), operator I_0 given by equation (45) where $x_a = 0$ and operator I_{x_0} given by equation (45) where $x_a = 1$.

1. The system is initiated in 0:

$$|\,\Psi\rangle = |\,0,0\rangle\,. \tag{46}$$

2. Apply the *Walsh - Hadamard* transformation to normalize the system:

$$\begin{aligned}\mathrm{W}\,|\,0,0\rangle &= \frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{pmatrix} \times \begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \frac{1}{2}\begin{pmatrix}1\\1\\1\\1\end{pmatrix}\\ &= \frac{1}{2}\,|\,0,0\rangle + \frac{1}{2}\,|\,0,1\rangle + \frac{1}{2}\,|\,1,0\rangle + \frac{1}{2}\,|\,1,1\rangle\,.\end{aligned} \tag{47}$$

3. Iterate **Round**$[\frac{\pi}{4}\sqrt{N}] = 2$ times:

   (a) Apply W upon the system:

$$\begin{aligned}\mathrm{W}(\frac{1}{2}\,|\,0,0\rangle + \frac{1}{2}\,|\,0,1\rangle + \frac{1}{2}\,|\,1,0\rangle + \frac{1}{2}\,|\,1,1\rangle) &= \frac{1}{4}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{pmatrix} \times \begin{pmatrix}1\\1\\1\\1\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}\\ &= |\,0,0\rangle\,.\end{aligned} \tag{48}$$

   (b) Apply $\mathrm{I}_0\,|\,x\rangle$ that equals to the rotation matrix and changes the phase (sign) of the state $|\,0,0\rangle$:

$$\mathrm{I}_0\,|\,0,0\rangle = -\,|\,0,0\rangle\,. \tag{49}$$

(c) Apply again W to the system:

$$\mathrm{W}(-\,|\,0,0\rangle) = -\frac{1}{2}\,|\,0,0\rangle - \frac{1}{2}\,|\,0,1\rangle - \frac{1}{2}\,|\,1,0\rangle - \frac{1}{2}\,|\,1,1\rangle\,. \tag{50}$$

(d) Apply I_{x_1} upon the system, and takes charge of changing the phase (sign) to the element marked ($|\,0,1\rangle$).

$$\mathrm{I}_{x_1}(-\frac{1}{2}\,|\,0,0\rangle - \frac{1}{2}\,|\,0,1\rangle - \frac{1}{2}\,|\,1,0\rangle - \frac{1}{2}\,|\,1,1\rangle) = -\frac{1}{2}\,|\,0,0\rangle + \frac{1}{2}\,|\,0,1\rangle - \frac{1}{2}\,|\,1,0\rangle - \frac{1}{2}\,|\,1,1\rangle\,. \tag{51}$$

Here we begin to repeat the steps of the iteration:

(a) Apply W:

$$\mathrm{W}(-\frac{1}{2}\,|\,0,0\rangle + \frac{1}{2}\,|\,0,1\rangle - \frac{1}{2}\,|\,1,0\rangle - \frac{1}{2}\,|\,1,1\rangle) = -\frac{1}{2}\,|\,0,0\rangle - \frac{1}{2}\,|\,0,1\rangle + \frac{1}{2}\,|\,1,0\rangle - \frac{1}{2}\,|\,1,1\rangle\,. \tag{52}$$

(b) Apply $\mathrm{I}_0\,|\,x\rangle$:

$$\mathrm{I}_0 - \frac{1}{2}\,|\,0,0\rangle - \frac{1}{2}\,|\,0,1\rangle + \frac{1}{2}\,|\,1,0\rangle - \frac{1}{2}\,|\,1,1\rangle = \frac{1}{2}\,|\,0,0\rangle - \frac{1}{2}\,|\,0,1\rangle + \frac{1}{2}\,|\,1,0\rangle - \frac{1}{2}\,|\,1,1\rangle\,. \tag{53}$$

(c) Apply again W:

$$\mathrm{W}(\frac{1}{2}\,|\,0,0\rangle - \frac{1}{2}\,|\,0,1\rangle + \frac{1}{2}\,|\,1,0\rangle - \frac{1}{2}\,|\,1,1\rangle) = |\,0,1\rangle\,. \tag{54}$$

(d) Apply I_{x_1} upon the system:

$$\mathrm{I}_{x_1}(|\,0,1\rangle) = -\,|\,0,1\rangle\,. \tag{55}$$

4. Measure:

When the system is measured we obtain 1, that is the position of: $|\,0,1\rangle$, the marked element[7].

[7] The operator Q has a sign "$-$" that is the one that changes the sign to the probability that obtain finally.

4.3 Grover's Algorithm Implementation

The quantum computation has been slowly adopted for anyone who work in computation due to the existence of a great variety of formalism (notations, matrices, operators, etc), and do not have a representation that has a similarity with the classical programming languages.

This reason is the one that has motivated the QCL's author to develop it and thus to be able to offer the opportunity of an approach in a nice way to the quantum programming and to the concepts related to it.

QCL is a quantum programming language that contains all the elements for the implementation of quantum algorithms and let us to simulate the execution of the algorithms. It is an application for Linux that can be obtained in the internet at: http://tph.tuwien.ac.at/~oemer. With the application comes as an example the implementation of the Shor's Algorithm (section 3).

Using QCL we develop an implementation of Grover's algorithm, where we can define a N elements data base delivering the number of qubits that one want to use to represent the data base and the element that one wants to mark.

For this development the following functions were implemented:

- Function Inv: This function inverts from 0 to 1 and from 1 to 0, all the qubits of the register q, when i and x are equals; in order to be able to apply the operator $CPhase$ [5], which only operates upon the element $1111\ldots1$.

```
qufunct inv(int x, qureg q) {
  int i;
  for i = 0 to #q-1 {
    if not bit(x,i) { Not(q[i]); }
  }
}
```

- Operator Ix: Is the transformation described in the numeral 45 of the section 4.1, which changes the phase (sign) of the element that receives.

```
operator Ix(int x, qureg q) {

  inv(x,q);
  CPhase(pi,q);
  !inv(x,q);
}
```

- Operator Q: This is the operator Q defined in section 4.1, where Mix(q), represents the *Walsh - Hadamard* transformation.

```
operator Q(int x, qureg q) {
  Mix(q);
  Ix(0,q);
  Mix(q);
  Ix(x,q);
}
```

- Procedure Grover: This procedure is the one that executes the algorithm and its job is initialize the registers, the variables that are going to be used, determine the number of iterations; to execute the iterations, measure the system and give the result.

```
//Receive the number of qubits and Marked element
procedure Grover(int b,int x) {
  int N=2^b; //Number of elements in the data base
  int C=ceil((Pi/4)*(sqrt(N))); //Number of iterations
  int m; //Measure
  int i; //Index
  qureg q[b]; //Quantum register

  reset; //Initialize the system to 0
  Mix(q); //Normalize
  for i = 1 to C { //begin the iteration
    Q(x,q);        //of the operator Q
  }
  measure q,m; //Measure the system
  print "Measure: ",m; //Print the result
}
```

For the execution of the algorithm we invoke the application in the following way:

```
qcl -bN -i groverN.qcl
```

Where `b` indicates that we are going to use N qubits, `i` indicates that QCL will be used in an interactive way and `groverN.qcl` is the name of the file with the source code of the implementation.

An example of this implementation will show the execution for 2 qubits, with 1 as the marked element.

1. We execute the algorithm:

```
1  qcl -b2 -i groverN.qcl
```

2. At this time we show that we are going to use 2 qubits that let us have 4 elements, from 0 to 3. In the execution a message appears indicating us how to use the algorithm (lines 2-4).

```
2  Use: Grover(n,q)
3  n -> Number of qubits.
4  x -> Marked element.
```

3. Now we initiate the procedure indicating the number of qubits and the marked element:

```
5  Grover(2,1)
```

4. Apply the first operations, reset that carries the initial state to zeros (lines 6- 7) and apply the first *Walsh - Hadamard* transformation (W) (lines 8-9), that we use for normalize the system[8]:

```
6  @RESET
7  %|00>
8  @Mix(qureg q=|10>)
9  %0.5 |00> + 0.5 |10> + 0.5 |01> + 0.5 |11>
```

5. Then the cycle begins, it will repeat 2 times the operator Q on the system (lines 10-61)

   (a) Apply the first operation in Q, W (lines 10-11); then apply I_0 (lines 12 - 22) the rotation matrix and changes the phase (sign) of the state $|0,0\rangle$ (line 22):

```
10 @ Mix(qureg q=|10>)
11 % 1 |00>
12 : I_0
13 @ Not(qureg q=|.0>)
14 % 1 |01>
15 @ Not(qureg q=|0.>)
16 % 1 |11>
17 @ CPhase(real phi=3.141593,qureg q=|10>)
18 % -1 |11>
19 @ !Not(qureg q=|0.>)
20 % -1 |01>
21 @ !Not(qureg q=|.0>)
22 % -1 |00>
```

   (b) Apply again W to the system (lines 23 -24) and then apply I_{x_1}, that changes the phase (sign) to the marked element, in this case the element 1 (lines 25-31):

[8] The symbol @ indicates the operation executed and % indicates how the system remains after the execution of the operation.

```
23 @ Mix(qureg q=|10>)
24 % -0.5 |00> + -0.5 |10> + -0.5 |01> + -0.5 |11>
25 : I_x_1
26 @ Not(qureg q=|0.>)
27 % -0.5 |10> + -0.5 |00> + -0.5 |11> + -0.5 |01>
28 @ CPhase(real phi=3.141593,qureg q=|10>)
29 % -0.5 |10> + -0.5 |00> + 0.5 |11> + -0.5 |01>
30 @ !Not(qureg q=|0.>)
31 % -0.5 |00> + -0.5 |10> + 0.5 |01> + -0.5 |11>
```

(c) From here on steps of the iteration are repeated. Apply W (lines 32-33), I_0 (lines 34-44), W again (lines 45-46) and finally I_{x_0} (lines 47-53) :

```
32 @ Mix(qureg q=|10>)
33 % -0.5 |00> + 0.5 |10> + -0.5 |01> + -0.5 |11>
34 : I_0
35 @ Not(qureg q=|.0>)
36 % -0.5 |01> + 0.5 |11> + -0.5 |00> + -0.5 |10>
37 @ Not(qureg q=|0.>)
38 % -0.5 |11> + 0.5 |01> + -0.5 |10> + -0.5 |00>
39 @ CPhase(real phi=3.141593,qureg q=|10>)
40 % 0.5 |11> + 0.5 |01> + -0.5 |10> + -0.5 |00>
41 @ !Not(qureg q=|0.>)
42 % 0.5 |01> + 0.5 |11> + -0.5 |00> + -0.5 |10>
43 @ !Not(qureg q=|.0>)
44 % 0.5 |00> + 0.5 |10> + -0.5 |01> + -0.5 |11>
45 @ Mix(qureg q=|10>)
46 % 1 |01>
47 : I_x_1
48 @ Not(qureg q=|0.>)
49 % 1 |11>
50 @ CPhase(real phi=3.141593,qureg q=|10>)
51 % -1 |11>
52 @ !Not(qureg q=|0.>)
53 % -1 |01>
```

6. Now we measure (lines 54-55) and we obtain the result (line 56) :

```
54 @ MEASURE
55 % 1 |01>
56 :Measure:  1
```

5 Conclusion

Quantum Computing is a relative new computation model that implies a new way of thinking algorithms for getting a maximal advantage of its properties, and it demands to us to understand its characteristics, because as technology trends show, Quantum Computer is possibly a feature for computers's next generation.

6 Acknowledgements

The paper was financed by EAFIT University, under the research project number 817407.

References

[1] Aharonov, D. Quantum Computation. Eprint: quant-ph/9812037, 1998.

[2] Grover, L. K. A fast quantum mechanical algorithm for database search. Eprint: quant-ph/9605043, 1996.

[3] Grover, L. K. Rapid sampling through quantum computing. Eprint: quant-ph/9912001, 1999.

[4] Jozsa, R. Searching in Grover's algorithm. Eprint: quant-ph/9901021, 1999.

[5] Ömer, B. A Procedural Formalism for Quantum Computing. Master's thesis, Department of Theorical Physics, Technical University of Viene, 1998.

[6] Rieffel, E., and Polak, W. An Introduction to Quantum Computing for Non-Physicists. Eprint: quant-ph/9809016, 1998.

[7] Shor, P. W. Polinomial-time algorithms for prime factorizacion and discrete logarithms on a quantum computer. *Siam Journal on Computing 26*, 5 (Octuber 1997), 1484–15090.

[8] Sicard, A., and Vélez, M. Algunos elementos introductorios acerca de la computación cuántica. In *Memorias VII Encuentro de la ERM, Realizado en la Universidad de Antioquia* (Agosto 1999), Escuela Regional de Matemáticas. En: Informe de Investigación: ¿Máquina de Turing Cuántica Autorrefencial: Una posibilidad?, Universidad EAFIT, 1999.

[9] Williams, C. P., and Clearwater, S. H. *Explorations in Quantum Computing.* Springer-Telos, 1997.