Quantum Fourier Transform Circuit Simulator

Víctor H. Tellez*, Antonio Campero**, Cristina Iuga**, Gonzalo I. Duchen***

*Electrical Engineering Departament, **Chemical Departament, DCBI, Universidad Autonoma Metropolitana Iztapalapa, Av. San Rafael Atlixco 186, Col. Vicentina, Iztapalapa 09340 D.F. Mexico, email: vict@xanum.uam.mx
***SEPI, ESIME Culhuacan, Av. Santa Ana 1000, Col. San Francisco Culhuacan, C.P. 04430, D.F.

ABSTRACT

Quantum Fourier transform is of primary importance in many quantum algorithms. This paper presents the development of a Quantum Fourier Transform Circuit Simulator system that processes classical analog signals and presents the results of the processing data. The data is acquired by an analog to digital classical converter, on a classical computer. The data stored is processed by computer using an algorithm that executes a Quantum Fast Fourier Transform (QFT).

1 INTRODUCTION

In quantum mechanics, quantum information is physical information that is held in the "state" of a quantum system. The most popular unit of quantum information is the qubit, a two-state quantum system. However, unlike classical digital states (which are discrete), a two-state quantum system can actually be in a superposition of the two states at any given time. A quantum bit, or qubit is described by a state vector in a two-level quantum mechanical system which is formally equivalent to a two-dimensional vector space over the complex numbers.

The Quantum Fourier transforms (QFT) plays essential roles in various quantum algorithms such as Shor's algorithms [1, 2, 3] and hidden subgroup problems [2, 4, 5]. Inspired by the exponential speed-up of Shor's polynomial algorithm for factorization [1], many people investigated the problem of efficient realization of QFT in a quantum computer [3, 6, 7, 8, and 9]. Up to now, many improvements have been made. In [6], Moore and Nilsson showed that QFT can be parallelized to linear depth in a quantum network, and upper bound of the circuit depth was obtained by Cleve and Watrous [7] for computing QFT with a fixed error. In reference [8] the actual time-cost for performing QFT in the quantum network was examined. Further, Blais [9] designed an optimized quantum network with respect to time-cost for QFT.

The present work shows a sound processing system based QFT circuit simulator of such sounds. A minimum system based on QFT circuit simulator and acquisition was developed, using a classical computer and the QFT was made on Python compiler. The paper is organized as follows. After the introduction the acquisition system is described, followed by the methodology to processing the sound. Results and conclusions are presented.

2 SYSTEM DESCRIPTION

The analog to digital converter part is based on a 68HC11 microcontroller minimum system; this processor that can perform the required memory, A/D conversion and transfer to Personal Computer. The data received from the Analog to Digital Converter is processing by a personal computer, using the Python compiler, which execute the QFT program and presents the results of the processing.

3 METHODOLOGY

The system requests a subject to emit a sound and it is recorded through the microphone; the program seeks for the suitable tone in order to be recognized as a valid signal. The capture algorithm is executed from the analog to digital converter, and then the processing algorithms are executed from the personal computer.

3.1 The QFT modeling circuit

The processing algorithm, is based on the QFT as an unitary operation on n qubits defined as follows

$$F : \left| x \right\rangle \to \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{2\pi i xy/2^n} \left| y \right\rangle \tag{1}$$

where xy is a normal ``decimal" multiplication of numbers x and y, which are represented by the quantum registers

$$\left| y \right\rangle = \left| y_{n-1} \right\rangle \otimes \left| y_{n-2} \right\rangle \otimes \cdots \otimes \left| y_0 \right\rangle \tag{2}$$

where $\left| x_k \right\rangle$ and $\left| y_k \right\rangle$ are individual qubits.

Comparing this with the notation in the section about Simon Oracle, where $x \bullet y$ meant

$$x \bullet y = x_0 \bullet y_0 + 2x_1 \bullet y_1 + 2x_2 \bullet y_2 + \cdots + 2x_{n-1} \bullet y_{n-1}$$

There we treated **x** and **y** as arrays of bits rather than integer numbers. Of course in computing a single integer number is implemented as an array of bits, but the point is how you interpret this array, and so *xy* in the Fourier Transform formula is not the same as $x \bullet y$ in the Simon Oracle formula. The former is an integer operation on two scalar numbers and the latter is a binary operation on two binary vectors. The former can be expressed in terms of a binary operation too, but it will not be $x \bullet y$.

Observe that once you know what **F** does to the basis vectors $|x\rangle$, you can figure out what **F** does to any other vector. This other vector can be $\sum_{\infty} f(x)|x\rangle$, which yields the following formula for Quantum Fourier Transform of function f:

$$F\left(\sum_{x=0}^{N-1} f(x)|x\rangle\right) = \sum_{x=0}^{N-1} f(x) F(|x\rangle)$$

$$= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x) \sum_{y=0}^{N-1} e^{2\pi i xy/N} |y\rangle \qquad (3)$$

$$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} f(x) e^{2\pi i xy/N} |y\rangle \qquad (4)$$

From this formula the y^{th} component of **F** is

$$F_y(f) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x) e^{2\pi i xy/N} \qquad (5)$$

which is beginning to look quite like a normal Discrete Fourier Transform.

3.2 The QFT Circuit Simulator

The QFT circuit simulator is development on Python compiler, on a personal computer running over Linux operating system. In order to implement the circuit that calculates the QFT:

$$F : |x\rangle \to \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{2\pi i xy/2^n} |y\rangle \quad (6)$$

we shall deploy the trickery of the Fast Fourier Transform. Let us have a look at:

$$e^{2\pi i xy/2^n}$$

This expression is periodic in *xy* and the period is 2^n. The trick about the Fast Fourier Transform is that it only uses the terms of $e^{2\pi i xy/2^n}$ that correspond to the ``first circle'', i.e., the terms for which $x\,y / 2^n < 1$. Let us evaluate then $xy/2^n$ while truncating very thing that would go onto the second and third circle:

$$\frac{xy}{2^n} \equiv \frac{1}{2^n}\left(x_0 + x_1 2 + x_2 2^2 + x_3 2^3 + \cdots + x_{n-1} 2^{n-1}\right) \times$$
$$\left(y_0 + y_1 2 + y_2 2^2 + y_3 2^3 + \cdots + y_{n-1} 2^{n-1}\right) = \ldots$$

Here we have decomposed x and y into their binary components, so that each of the x_k and y_k terms is either 0 or 1.

$$= \frac{1}{2^n}\left(y_0\left(x_0 + x_1 2 + x_2 2^2 + \cdots + x_{n-1} 2^{n-1}\right) + y_1 2^1\left(x_0 + x_1 2 + x_2 2^2 + \cdots + x_{n-1} 2_{n-1}\right) + \cdots\right.$$
$$\left.\cdots + y_1 2^{n-1}\left(x_0 + x_1 2 + x_2 2^2 + \cdots + x_{n-1} 2^{n-1}\right)\right)$$

$$\frac{1}{2^n} = (y_0\left(x_0 + x_1 2 + x_2 2^2 + \cdots + x_{n-1} 2^{n-1}\right) +$$
$$y_1\left(x_0 2 + x_1 2^2 + x_2 2^3 + \cdots + x_{n-2} 2^{n-1}\right) +$$
$$+ y_2\left(x_0 2^2 + x_1 2^3 + x_2 2^4 + \cdots + x_{n-3} 2^{n-1}\right) + \cdots + y_{n-1} x_0 2^{n-1})$$

$$= y_0\left(\frac{x_0}{2^n} + \frac{x_1}{2^{n-1}} + \frac{x_2}{2^{n-2}} + \cdots + \frac{x_{n-1}}{2}\right) + y_1\left(\frac{x_0}{2^{n-1}} + \frac{x_1}{2^{n-2}} + \frac{x_2}{2^{n-3}} + \cdots + \frac{x_{n-2}}{2}\right) +$$

$$+ y_2\left(\frac{x_0}{2^{n-2}} + \frac{x_1}{2^{n-3}} + \frac{x_2}{2^{n-4}} + \cdots + \frac{x_{n-3}}{2}\right) + \cdots + y_{n-1}\frac{x_0}{2} = ..$$

There is a special notation, which covers the sums in the brackets:

$$\frac{x_0}{2} \to (x_0), \qquad \frac{x_0}{2^2} + \frac{x_1}{2} \to (x_0 x_1),$$
$$\frac{x_0}{2^3} + \frac{x_1}{2^2} + \frac{x_2}{2} \to (x_0 x_1 x_2) \ldots$$

Using this notation:

$$\frac{xy}{2^n} \equiv y_0(x_0 x_1 \ldots x_{n-1}) + y_1(x_0 x_1 \ldots x_{n-2}) +$$
$$y_2(x_0 x_1 \ldots x_{n-3}) + \cdots + y_{n-1}(x_0) \qquad (7)$$

So now we can write our Quantum Fourier Transform thusly:

$$F|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{2\pi i xy/2^n} |y\rangle$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{2\pi i\left(y_0(x_0 x_1 \ldots x_{n-1}) + .. + y_{n-1}(x_0)\right)} |y\rangle$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^{n-1}} e^{2\pi i y_0(x_0 x_1 \ldots x_{n-1})} |y_0\rangle \otimes e \otimes \cdots \otimes e^{2\pi y_{n-1}(x_0)} |y_{n-1}\rangle$$

Now observe that y_k is either 0 or 1. If it is 0 then the corresponding term is, for example,

$$e^{2\pi i 0(x_0 x_1 \cdots x_{n-3})}\left|0\right\rangle = \left|0\right\rangle$$

If it is 1 then the corresponding term is: $e^{2\pi i 1(x_0 x_1 \cdots x_{n-3})}\left|1\right\rangle = \left|1\right\rangle$

The sum over all possible values of **y** will eventually assign both 0 and 1 to every y_k, therefore the following superposition is equivalent to the above:

$$F\left|x\right\rangle = \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(x_0 x_1 \ldots x_{n-1})}\left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(x_0 x_1 \ldots x2)}\left|1\right\rangle\right) \otimes \cdots$$
$$\otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(x_0)}\left|1\right\rangle\right)$$

And this already points to the way we can implement a QFT circuit. Consider the following circuit:

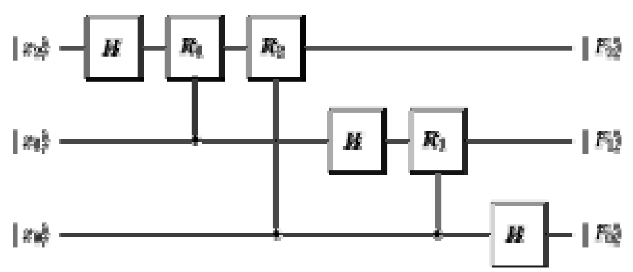

Figure 1 QFT circuit.

Here, as before, **H** is the Hadamard operator and $\mathbf{R}_d$ is a controlled gate defined by:

$R_d = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2^d} \end{pmatrix}$ Where d is the *distance* between the lines. Let us analyze this circuit step by step: after the first Hadamard gate the top line becomes

$$\frac{1}{\sqrt{2}}\sum_{y=0}^{1}(-1)^{x_2 y}\left|y\right\rangle = \frac{1}{\sqrt{2}}\sum_{y=0}^{1} e^{2\pi i x_2 y/2} \quad (8)$$

The second step applies $\mathbf{R}_1$ to the top line under the control of the middle line. Observe that $\mathbf{R}_1$ does nothing to $\left|0\right\rangle$ and phase shifts $\left|1\right\rangle$. The phase shift factor is $e^{i\pi/2}$ if the control line $\left|x_1\right\rangle$ is $\left|1\right\rangle$ and there is no phase shifts if $\left|x_1\right\rangle = \left|0\right\rangle$. We can therefore write that the phase shift inflicted by $\mathbf{R}_1$ on $\left|1\right\rangle$ is *always* $e^{i\pi x/2}$, where x is the control signal.

Applying this to states on the top and on the middle line yields

$$R_1\left|x_1\right\rangle \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(x_2)}\left|1\right\rangle\right) = \left|x_1\right\rangle \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(x_2)}e^{i\pi x_1/2}\left|1\right\rangle\right) \quad (9)$$
$$= \left|x_1\right\rangle \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(x_2/2 + x_1/4)}\left|1\right\rangle\right) = \left|x_1\right\rangle \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(x_1 x_2)}\left|1\right\rangle\right)$$

The third gate applies $\mathbf{R}_2$ to the top line, but this time under the control of the bottom line. This operator, again, will do nothing to $\left|0\right\rangle$, but will phase shift $\left|1\right\rangle$ by additional $e^{i\pi x_0/4}$ so the state of the whole system now becomes:

$$\begin{aligned} & R_2\left|x_0\right\rangle \otimes \left|x_1\right\rangle \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(x_1 x_2)}\left|1\right\rangle\right) = \\ & = \left|x_0\right\rangle \otimes \left|x_1\right\rangle \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(x_1 x_2)}e^{i\pi x_1/4}\left|1\right\rangle\right) \\ & = \left|x_0\right\rangle \otimes \left|x_1\right\rangle \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(x_0/8 + x_1/4 + x_2/2)}\left|1\right\rangle\right) = \\ & = \left|x_0\right\rangle \otimes \left|x_1\right\rangle \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(x_0 x_1 x_2)}\left|1\right\rangle\right) \end{aligned} \quad (10)$$

Reasoning as above we can see immediately that the next two gates applied to $\left|1\right\rangle$ will convert it into $\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(x_0 x_1)}\left|1\right\rangle\right)$. So that now the state of the computer is:

$$\begin{aligned} & \left|x_0\right\rangle \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(x_0 x_1)}\left|1\right\rangle\right) \\ & \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{e\pi i(x_0 x_1 x_2)}\left|1\right\rangle\right) \end{aligned} \quad (11)$$

And finally the single Hadamard transform on the bottom line converts $\left|x_0\right\rangle$ to $\left|0\right\rangle + e^{2\pi i(x_0)}\left|1\right\rangle / \sqrt{2}$, so that in effect the final state of the computer is:

$$\begin{aligned} & \frac{1}{\sqrt{2}}\left(\left|0\right\rangle\right) + e^{2\pi i(x_0)}\left|1\right\rangle \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(x_0 x_1)}\left|1\right\rangle\right) \\ & \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi(x_0 x_1 x_2)}\left|1\right\rangle\right) \end{aligned} \quad (12)$$

But this is a 3-point Quantum Fourier Transform, so the circuit shown above is a QFT circuit.

4 RESULTS

In the table 1 and 2, we can see the results comparing QFT circuit with FFT development in commercial software (MATLAB). These bounds allow simulation for many choices of N and ϙ. However the choices for M and L given in odd integer can usually be improved, and were merely given to show such values can be found. For example, the following table shows, for different N and ϙ combinations, a triple (g,m, l) of integers, with the choice from line 86 being M = 2g; yet in each case M = 2m and L = 2l is the pair with minimal m satisfying the hypotheses for odd integer Thus choosing M and L carefully may allow lower qubit counts, such as the N = 13, ϙ = 0.10 case.

Table 1, Values for QFT circuit simulator

ϱ	N=13	N=25	N=51	N=101	N=251	N=501
.001	45,45,28	47,47,28	48,48,29	50,50,29	52,52,30	53,53,30
.01	36,35,21	37,37,22	38,38,23	40,40,23	42,42,23	43,43,24
.05	29,28,17	30,30,17	31,31,18	33,33,18	35,35,19	36,36,19
.10	26,25,15	27,27,15	28,28,16	30,30,16	32,32,17	33,33,17
.20	23,22,13	24,24,13	25,25,14	27,27,14	29,29,15	30,30,15
.30	21,20,12	22,22,12	24,24,12	25,25,13	27,27,13	29,28,14
.40	20,19,11	21,21,11	22,22,12	24,24,12	26,26,13	27,27,13

Table 2, Values for classical FFT

ϱ	N=13	N=25	N=51	N=101	N=251	N=501
.001	43,43,27	46,46,27	48,49,30	54,51,28	51,532,31	52,52,29
.01	35,34,22	36,36,23	39,39,22	41,41,22	41,41,22	42,42,23
.05	30,29,19	31,31,16	32,32,19	34,34,19	34,33,19	38,39,16
.10	25,25,16	26,26,15	27,28,15	31,32,15	31,31,15	32,32,16
.20	23,2113	25,24,14	26,24,15	26,26,14	30,30,16	31,32,16
.30	20,20,11	22,21,10	26,25,13	27,26,14	28,26,12	28,27,16
.40	21,14,16	23,27,12	23,24,14	25,23,13	27,28,14	26,28,12

5 CONCLUSIONS

We processes the same acquiring sound on classical and commercial software using the FFT (Matlab), and the results are presented on table 1 and 2. And from the statistical results, we can see on comparison that there is a similar result using the QFT circuit simulator as the similar FFT. The problems that we resolved is the time for processing using the QFT, is because this algorithm have very cost on memory and resources on hardware, and the time for processing is almost 3 and four hours. So, we recommend using a good processor with enough memory for then.

The age of Quantum Information Processing has arrived, and the solution for different applications is high. Right know we can say that the using for QFT for processing gives us the next advantage:

- Massive parallelism, this for the superposition theory
- Reversible logic
- When the QFT can be done, the processing time will be faster

6 REFERENCES

[1]. *P.W.Shor*, SIAM J.Comput 26, 1484 (1997).

[2]. M.A.Nielesn, I.L.Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, England, 2000).

[3]. *S.Beauregard*, Circuit for shor's algorithm using 2n+3 qubits, quant-ph/0205095.

[4] *M.Mosca and A.Ekert*, The hidden subgroup problem and eigenvalue estimation on a quantum computer, quantph/9903071.

[5]. *M.Ettinger and P.Hoyer*, On quantum algorithms for non-commutative hidden subgroups, quant-ph/9807029.

[6]. *C.Moore and M.Nilsson*, SIAMJ. Comput 31, 799 (2001).

[7]. *R.Cleve and J.Watrous*, Fast parallel circuits for the quantum fourier transform, quant-ph/0001113.

[8]. *A.Saito, K.Kioi, Y.Akagi*, N.Hashizume, and K.Ohta, Actual computational time-cost of the quantum fourier transform in a quantum computer using spins, quantph/0001113.

[9]. *Ethan Bernstein and Umesh Vazirani*, Quantum complexity theory, SIAM Journal on Computing, 26 (1997), no. 5, 1411–1473.

[10].*Thomas Beth, Markus P¨uschel*, and Martin R¨otteler, Fast quantum Fourier transforms for a class of non-abelian groups, Proc. of Applied Algebra Algebraic Algorithms, and Error Correction Codes (AAECC-13, Springer-Verlag, 1999, volume 1719 in Lecture Notes in Computer Science, pp. 148–159.

[11]*I. L. Chuang and M. A. Nielsen*, Quantum computation and quantum information, Cambridge University Press, Cambridge, 2000.

[12].*R. Cleve, E. Ekert, C. Macchiavello, and M. Mosca*, Quantum algorithms revisited, Proc. Roy. Soc. Lond. A 454 (1998), 339–354.