CSCI 398 Team Project Phase 6:

Final Paper

Team # 5

Cody Evans, Min Toe, Ryan McGuire

An Analysis of the Quantum Fourier Transform Circuit

Cody Evans
Computer Science Department
California State University, Chico
Chico, USA

Min Toe
Computer Science Department
California State University, Chico
Chico, USA

Ryan McGuire
Computer Science Department
California State University, Chico
Chico, USA

Abstract— **The Quantum Fourier Transform is one of the most relevant quantum subroutines at present, used in most algorithms such as Shor's Algorithm, that have exponential speed-up compared to other classical ones. It essentially transforms a qubit into its constituent frequencies. The Quantum Fourier Transform is performed with a particular decomposition into a product of unitary matrices. In this research study, we present the detailed process of both the analysis, and the creation of a Quantum Fourier Transform circuit by using IBM Quantum Experience.**

Keywords- QFT; gate; circuit; complexity;

I. Introduction

To understand the Quantum Fourier Transform (QFT), one must first understand Fourier Transforms in general. Fourier Transforms are an incredibly powerful mathematical tool, useful in many areas of engineering and science such as image and signal processing or circuit design. At a high level, a Fourier Transform decomposes a mathematical function, usually functions of time, or a signal, into the frequencies that it is made of. Put simply, the Fourier Transform transforms a function from the time domain to the frequency domain. [3] This is extremely useful for functions which have underlying periodicity.

Functions with an underlying periodicity have the quality that they repeat over certain intervals. By definition, "a function $f(t)$ is periodic of period T if there is a number $T > 0$ such that $f(t + T) = f(t)$ for all t" [2]. A simple example of a periodic function would be $f(x) = sin(x)$, where the function repeats itself every 2π.

It turns out that there are a few popular implementations of the Fourier Transform as an algorithm. First, we have the Discrete Fourier Transform (DFT). While powerful, DFT has a computational complexity of N^2, where *N* is the size of the data. [4] A more popular algorithm for computing the Fourier Transform of a function is the Fast Fourier Transform (FFT), which has an improved computational complexity of $N \log(N)$ [4]. The Fast Fourier Transform takes advantage of the inherent symmetry the Fourier series has -- which is the basis for the Fourier Transform. [2] The next contender for algorithmic implementations of the Fourier Transform is the Quantum Fourier Transform. The Quantum Fourier Transform is based on the Fast Fourier Transform, but also provides a massive speedup over the FFT algorithm, which we will discuss later. [4]

II. Problem

As mentioned in the introduction, we want to be able to decompose a function into its constituent frequencies, and use that to find the period of a function. [3] While this might seem simple given the example of $f(x) = sin(x)$, more complicated functions present a number of challenges. Even if we know as a precondition that a given function is periodic, it is extremely difficult to find it. Imagine we have a function, and we are very zoomed in. We might find a region that appears to be repeating, however if we zoom out, we find that indeed that the zoomed in portion is in fact not periodic. This function is periodic though, so we keep going. We increase the size of the interval we are looking at to try and find any patterns, but none are found. [2] It is very possible that the period of this function is on an interval that is extremely large. As you can see, it is very hard to determine the global periodicity of a function, as there can be many red herrings which may appear to be the periodic interval, but in fact are not. We need a way to do this efficiently, and the classical computing implementations of the Fourier Transform are not nearly efficient enough for complicated functions.

Classically, our best example of doing this has a runtime of an exponential of n, where n is the number of bits needed to describe the period. This makes period finding for complicated functions unfeasible. [2]

An example problem would be noticing the pattern of exponentials in Shor's algorithm. This is difficult on a traditional computer but on a quantum computer with QFT it is much quicker. [3]

III. Goal

Since there are a number of uses for Fourier Transformations, we will focus on the importance of the Quantum Fourier Transform in Shor's algorithm. At a high level, Shor's algorithm can be used to factor integers in polynomial time. This is extremely useful, as integer factorization is the basis for a dominant portion of the field of cryptography, and the foundation of security on the internet. [3]

First, we need to implement the Quantum Fourier Transform as a "gate" in a quantum circuit. One requirement of all quantum gates is that they are reversible. This means that we can undo the operation a gate did, and return the original state prior to the first gate. As it turns out, the Fourier Transform also has an inverse Fourier Transform. [2]

Second, we need our Quantum Fourier Transform to preserve the superposition "norm". This means that regardless of the current state of a qubit, the probabilities of observing either state must always add up to 1. Given $a|0\rangle+\mathrm{b}|1\rangle$, where *a* and *b* are the respective probabilities of observing either state, the condition $a+b=1$ must be true. In general, this means that the Quantum Fourier Transform operation is unitary. [1]

IV. Research Questions

Time metrics are typically used but in this case we will also have processing capability metrics because the quantum version of the Fourier Transform requires a certain number of qubits. [5]

V. Significance

What makes the Quantum Fourier Transform significant? Being a quantum algorithm, instead of operating on binary 1's and 0's, QFT can also operate on states in superposition. Additionally, the Quantum Fourier Transform is far more efficient than the Discrete Fourier Transform or the Fast Fourier Transform. As mentioned earlier, efficiency is usually measured with time. However, there is another benefit: the Quantum Fourier Transform uses exponentially less (qu)bits than classical implementations of the Fourier Transform. [2]

In general, a Quantum Fourier Transform operating on 2^n amplitudes can be implemented in a quantum circuit using $O\left(n^2\right)$ qubits. Compared to the Fast Fourier Transform which takes $O\left(n2^n\right)$ bits. [4]

Using the symmetry of the Fast Fourier Transform as a basis, we can apply the Fourier Transform as an operation using the "*controlled phase shift*, where the least significant bit is the control, to multiply only the odd terms by the phase without doing anything to the even terms". [3] In general, this helps the Quantum Fourier Transform provide an exponential speedup over the Fast Fourier Transform. [4]

Lastly, looking at the Quantum Fourier Transform for period finding. Using a classical algorithm, we query our function iteratively until we find that the function repeats. This takes $O\left(2^n\right)$ operations. Using the Quantum Fourier Transform, we can query our function "with $N=2^n$ inputs for each *n* qubits at the same time". [3]

One thing to note about the Quantum Fourier Transform is that it is really just an abstraction of the Hadamard gate. While the Hadamard gate operates on a single qubit, and can put it into superposition, the Quantum Fourier Transform operates on an arbitrary number of qubits. [1]

VI. Methodology

Using an article from Qiskit as a reference, we used IBM Quantum Experience to construct our circuits. Figure 1 shows the circuit. [5]

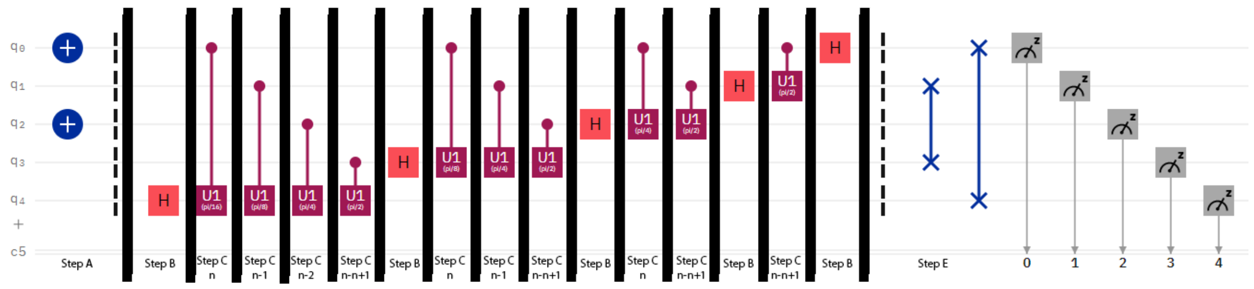


Figure 1. The QFT circuit.

The QFT circuit as shown in Figure 1, can be dissected into its individual parts below:

A. We first prepared the state (5, or 00101) that is to be transformed.
B. Next, we applied the Hadamard gate to the qubit with the highest index.
C. The index now becomes n. We then apply a turn based on the equation $(\pi/((n)\bullet 2))$ down to $\pi/((n-n+1)\bullet 2)$ where n is larger than 1.
D. With the most significant qubit taken care of, it can now be ignored, and then the process for the other qubits, using the same logic, needs to be repeated.
E. To complete the QFT circuit, swap the qubits outer most qubits with each other then recurse down, and measure the results.

Similarly, the circuit for the inverse QFT was also constructed using an article from Qiskit as a reference. Figure 2 shows the circuit. [5]

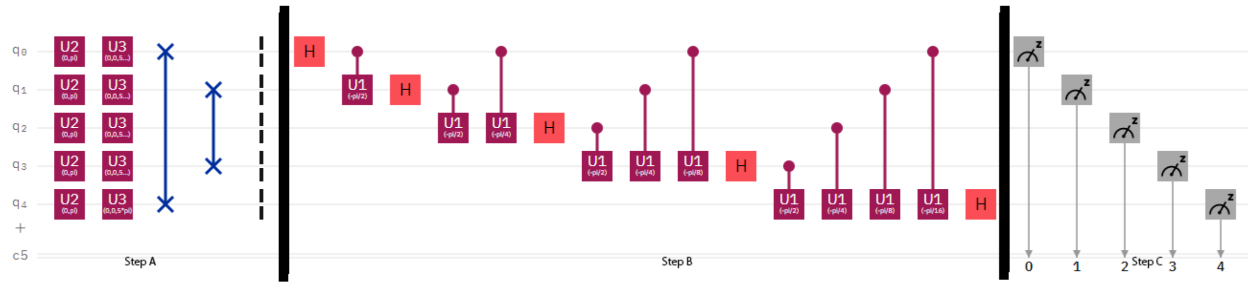
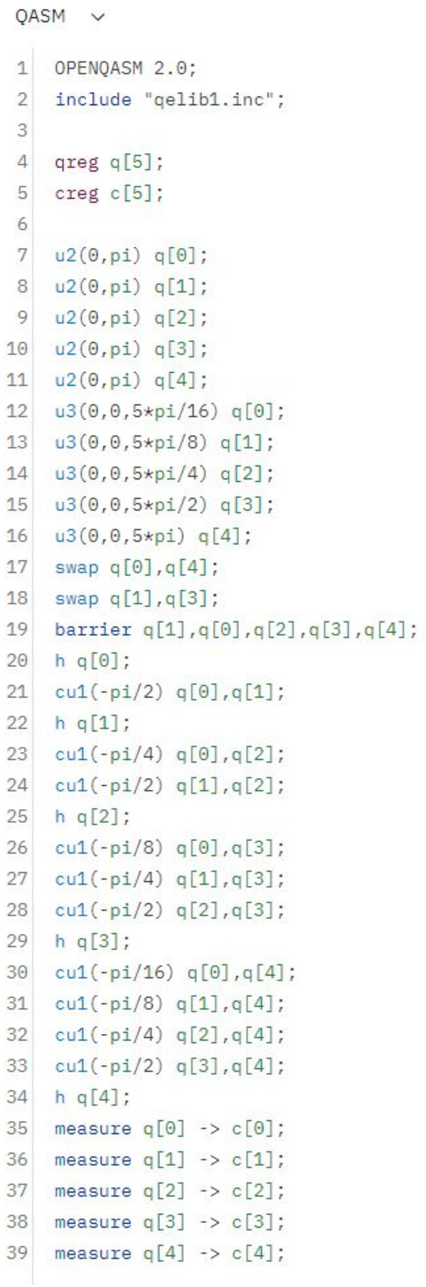


Figure 2. The Inverse QFT circuit

The QFT inverse circuit, as seen in Figure 2, can be dissected into its individual components below:

A. First, prepare the circuit with the results of the QFT circuit. [1]
B. Then, repeat every step in the QFT circuit but in reverse, starting from the index with the lowest index:
C. To complete the inverse QFT circuit, measure the results.

The code for a 5-bit QFT circuit in QASM is as given in Figure 3.



```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg c[5];

u2(0,pi) q[0];
u2(0,pi) q[1];
u2(0,pi) q[2];
u2(0,pi) q[3];
u2(0,pi) q[4];
u3(0,0,5*pi/16) q[0];
u3(0,0,5*pi/8) q[1];
u3(0,0,5*pi/4) q[2];
u3(0,0,5*pi/2) q[3];
u3(0,0,5*pi) q[4];
swap q[0],q[4];
swap q[1],q[3];
barrier q[1],q[0],q[2],q[3],q[4];
h q[0];
cu1(-pi/2) q[0],q[1];
h q[1];
cu1(-pi/4) q[0],q[2];
cu1(-pi/2) q[1],q[2];
h q[2];
cu1(-pi/8) q[0],q[3];
cu1(-pi/4) q[1],q[3];
cu1(-pi/2) q[2],q[3];
h q[3];
cu1(-pi/16) q[0],q[4];
cu1(-pi/8) q[1],q[4];
cu1(-pi/4) q[2],q[4];
cu1(-pi/2) q[3],q[4];
h q[4];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
measure q[3] -> c[3];
measure q[4] -> c[4];
```

Figure 3. 5-bit QFT circuit in QASM

VII. Results and Discussion

Running our QFT circuit on the server at Valencia gives this result as shown in Figure 4.

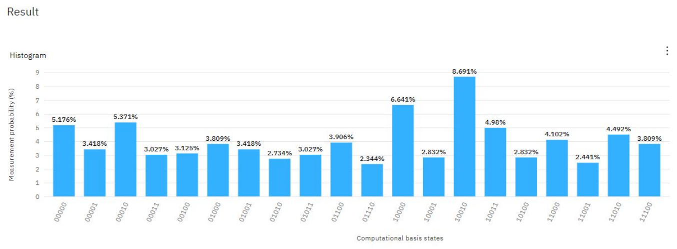


Figure 4. QFT results on Valencia.

As expected, the results are non-deterministic.

To have a more deterministic result, we need to use our inverse QFT circuit to see if state 5 was transformed correctly. The results are as seen in Figure 5 and Figure 6.

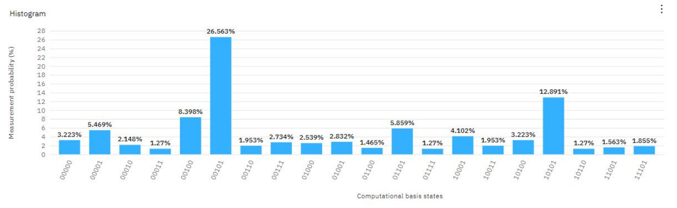


Figure 5. Inverse QFT results on Athens.

Figure 4 shows that our state was indeed transformed correctly.

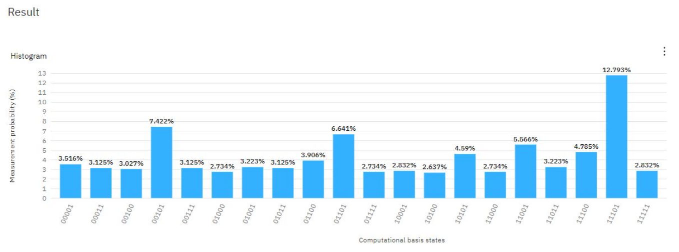


Figure 6. Inverse QFT result on Valencia.

Interestingly, while Athens gave us the result that we were expecting, Valencia is adding an extra bit at the most significant bit's position. We suspect that this can be attributed to interference.

VIII. Conclusion

In conclusion, we have managed to create a successful Quantum Fourier Transform circuit that does indeed transform a qubit into its constituent frequencies correctly by using an inverse circuit to verify our results. The Fourier Transform is a useful algorithm but the Quantum Fourier Transform is even more useful because it takes considerably less time and is only limited by the number of qubits available and the potential interference that quantum computers are subject to. The best Quantum Fourier Transform circuits take only $O(nlog\,(n))$ time when the classical Fourier Transform takes $O\,(2^n)$ time. The issue with quantum computers of course lies in the possibility of interference muddling the results. But considering the run time, it should be easy enough to compare results of two quantum computers to right any potential wrongs and show the usefulness of this algorithm.

IX. Future Work

For our future work, more research is needed to obtain an approximation of the QFT algorithm with the computation complexity of $O(nlog\,(n))$. Whilst this study tackled how to construct a Quantum Fourier Transform circuit, it did not take into account the amount of time the experiments took. We could also do more work on how QFT is relevant in Shor's algorithm by displaying how QFT is used within it. With Shor's algorithm, one could theoretically break the cyber security systems used today given a quantum computer that is big enough to handle the task.

X. Acknowledgements

We would like to acknowledge the creator of the algorithm, Don Coppers, our instructor, Jaime Raigoza for the guidance, and lastly, Qiskit, for providing both useful lectures and documentations.

References

[1] *IBM Quantum Experience*. quantum-computing.ibm.com/.

[2] *Operations glossary*. (n.d.). Retrieved from IBM Quantum Experience: https://quantum-computing.ibm.com/docs/iqx/operations-glossary

[3] Osgood, B. (n.d.). *The Fourier Transform and its Applications.* Retrieved from Stanford Engineering Everywhere: https://see.stanford.edu/materials/lsoftaee261/book-fall-07.pdf

[4] *QFT, Period Finding & Shor's Algorithm.* (n.d.). Retrieved from EdX: https://courses.edx.org/c4x/BerkeleyX/CS191x/asset/chap5.pdf

[5] *Quantum Algorithms.* (n.d.). Retrieved from CNOT: https://cnot.io/quantum_algorithms/qft/

[6] *Quantum Fourier Transform*. (n.d.). Retrieved from Qiskit: https://qiskit.org/textbook/ch-algorithms/quantum-fourier-transform.html