Quantum Computing:

A Mathematical Analysis of Shor's Algorithm

Stephen Clarke

Fall 2019

MA 398, Dr. Molitierno

Faculty Mentor: Tina Romansky

Abstract

This paper will explore how quantum computers work from a base level and look at mathematical functions, specifically finding prime factors of large integers. Methods optimized for quantum computers such as Shor's Algorithm exploit quantum properties and result in solutions that are notably more efficient than the best algorithms executed on classical computers. Finally the paper will define the parameters of an experiment with quantum algorithms and provide quantitative results comparing traditional computers to simulated and physical quantum computers. This will lead to understanding real world effects of quantum algorithms on fields such as Cybersecurity, Computer Science, Optimization and Mathematics.

1 Introduction To Quantum Computing

Quantum Computing is a relatively new form of thinking using quantum properties of subatomic particles. These ideas have been studied since the 1980s, but it is only recently that quantum hardware has been turning these ideas into reality. This introduction will cover the differences between classical and quantum computers, and some of the ideas that make quantum computers what they are.

DEFINITION 1.1 *The smallest unit of information in a classical computer is called a* ***bit*** *which can be represented as two values:* 0 *and* 1 *(sometimes also treated as T and F).*

A bit can be operated on using Boolean algebra in such a way as to change its value. In Table 1 there is a truth table with some examples of basic algebra performed on two bits c_1 and c_2.

c_1	c_2	c_1 and c_2	c_1 or c_2	not c_1
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Table 1: Classical Bit Boolean Algebra

In the world of quantum computers, *bits* are used in combination with quantum

bits, or *qubits* to perform tasks and create algorithms.

DEFINITION 1.2 *A **qubit** is the smallest unit of memory in a quantum computer.*

In this paper, qubits will be represented by $a|0\rangle + b|1\rangle$ with $a, b \in \mathbb{C}$ which satisfy $|a|^2 + |b|^2 = 1$ [3]. $|0\rangle = \begin{pmatrix} 1 \\ \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ \\ 1 \end{pmatrix}$ a notation which will be described only at a high level for the purposes of the thesis of this paper. What is important is that $|0\rangle$ represents a classical bit 0 and $|1\rangle$ represents a classical bit 1 which will be expanded in Definition 1.3.

DEFINITION 1.3 ***Measuring** a qubit results in the quantum state collapsing and the qubit being recorded into a classical bit.*

If a qubit q_1, represented by $a|0\rangle + b|1\rangle$, is measured and assigned to a classical bit c_1, the probability of c_1 equalling 0 is $|a|^2$ and probability of c_1 equalling 1 is $|b|^2$.

To start to understand how qubits function, some quantum concepts need to be defined.

DEFINITION 1.4 *A **superposition** is a quantum state where a qubit can have different probabilities, when measured, to be 0 or 1.*

Illustrated in Figure 1, also referred to as the Bloch Sphere, there is a superposition of three qubits X, Y and Z, which are just the axes. The resulting superposition is represented by $|\psi\rangle$. Superposition can be though of as vector addition of different qubit states.

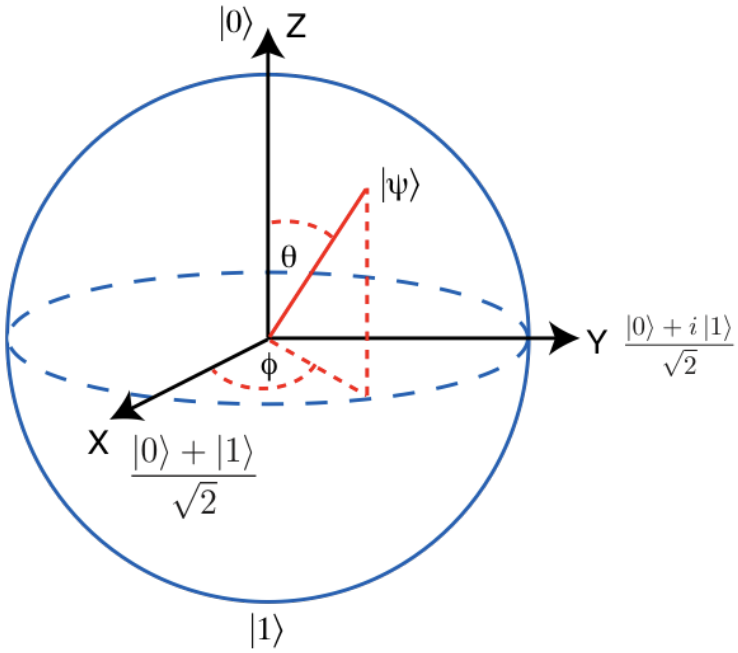
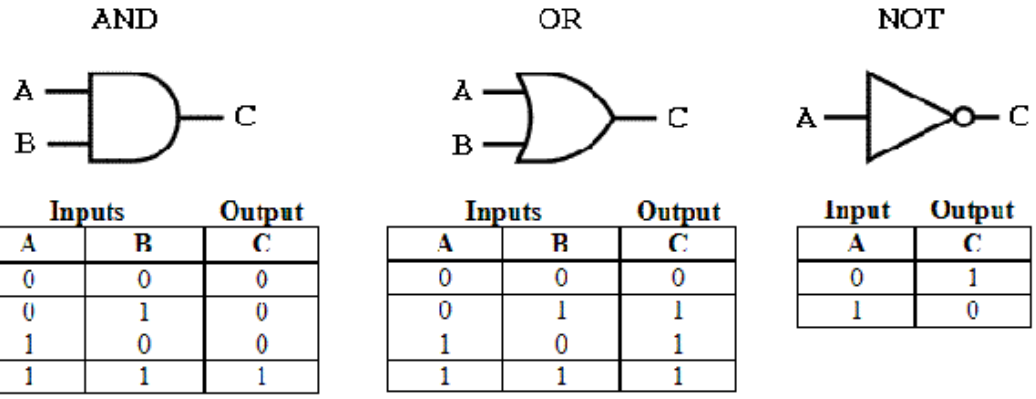


Figure 1: Bloch Sphere

As illustrated, each qubit can have a value 0 or 1 along each axis, and then a rotation in the complex plane modeled by its *phase*. For example a qubit can be in the position $i|1\rangle$ which will have the value 1 but rotated 90 degrees. Different rotations will result in different probabilities of outcomes when the qubit or state of qubits is measured.

DEFINITION 1.5 *A **gate** is a physical component that performs Boolean Algebra on bits or qubits.*

Gates in quantum computing act similarly to the gates in traditional circuit design. In traditional circuits, there are different gates that correspond with the *and, or, not,* functions that are operations of Boolean algebra (Figure 2).



Inputs		Output
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Inputs		Output
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Input	Output
A	C
0	1
1	0

Figure 2: Classical Gates

In quantum computing there are also gates, but most of them work a little differently than traditional ones.

DEFINITION 1.6 *The **Hadamard** gate, a quantum gate, puts a qubit in a state where there is a probability of* 0.5 *for the qubit to be 0 or 1 when measured.*

In reality, the Hadamard gate effectively rotates the qubit by π about the z-axis and $\frac{\pi}{2}$ about the y-axis as seen in the Bloch Sphere in Figure 1 (when done, this is physically represented by a phase shift).

This probabilistic effect happens by the pure nature of quantum computers. The qubit being modified is being rotated in the complex plane so it cannot be concluded what the real value will be before measured, only that it may be a 0 or 1. After measurement, the qubit collapses into a real state and value can then be assigned to a classical bit.

The Hadamard gate also has the property that when applied twice, the qubit is rotated back into its original position.

DEFINITION 1.7 *The **X gate** switches the real value of a qubit between 0 and 1.*

The X gate is simply a NOT gate in traditional computers. This also has the effect of flipping the probabilities of getting 0 or 1. A simple quantum circuit can be seen in Table 2 showing this effect. In the first row, a rotation is applied so that the state becomes (about) 85% 0 and 15% 1. In the second row, an X gate is applied after the rotation and the probabilities flip.

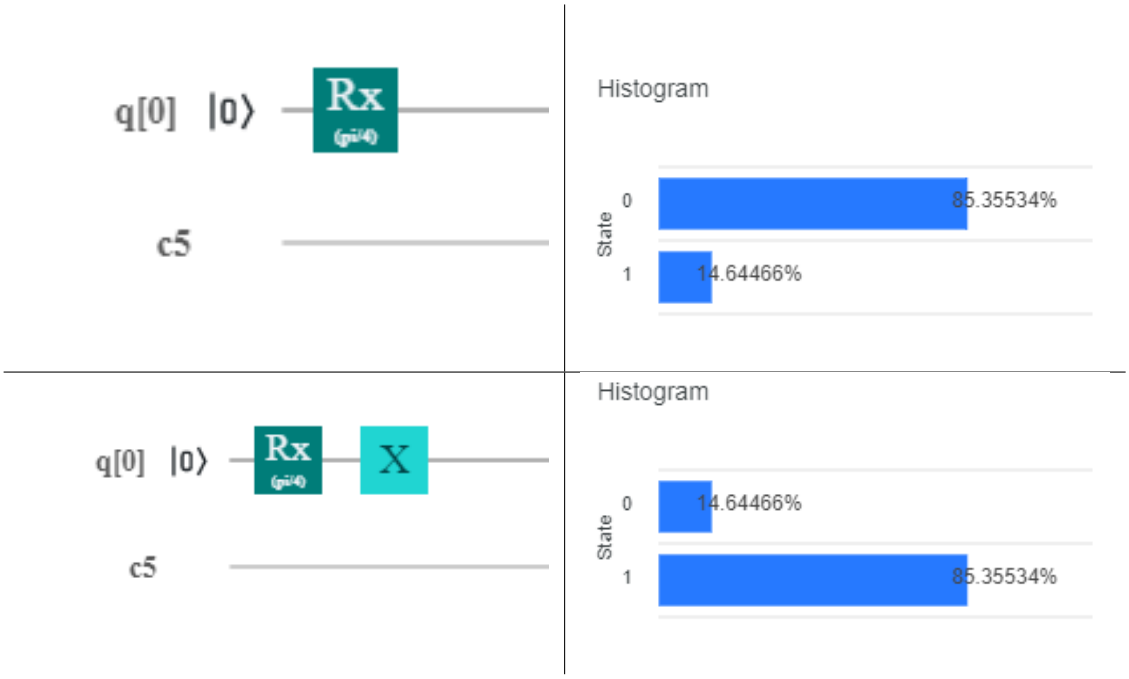


Table 2: X Gate

In Table 3 there are some simple gates being applied to a single qubit. This is only a brief overview of a few of many gates and other gates will be explained as they are introduced throughout the paper.

q_1	Hadamard q_1	Hadamard q_1 twice	X q_1
$\|0\rangle$	$\frac{1}{\sqrt{2}}\|0\rangle + \frac{1}{\sqrt{2}}\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\frac{1}{\sqrt{2}}\|0\rangle + \frac{-1}{\sqrt{2}}\|1\rangle$	$\|1\rangle$	$\|0\rangle$

Table 3: Quantum Bit Algebra

It can also be verified that the Hadamard gate results in a probability of 0.5 by referring back to Definition 1.3.

$$|0\rangle\tfrac{1}{\sqrt{2}} \implies (\tfrac{1}{\sqrt{2}})^2 = (0.07071\cdots)^2 = 0.5 \text{ chance}$$
$$|1\rangle\tfrac{1}{\sqrt{2}} \implies (\tfrac{1}{\sqrt{2}})^2 = (0.07071\cdots)^2 = 0.5 \text{ chance}$$

Thus there is a probability of 0.5 of the qubit being measured as 0 and the same percent of being measured as 1.

DEFINITION 1.8 *An **entangled** state of qubits is a where an operation on one qubit will apply immediately to every other qubit in the state.*

During traditional quantum entanglement, two particles will be measured to have the same spin and phase as each other, no matter how far or what process is done to either. The particles are completely dependant on each other. In Quantum Computing, this same property can be used where if two qubits are entangled, any measurement made to one will cause the other qubit's value to be known.

This has applications to performing the same operation on different qubits to quantum teleportation: sending information instantaneously between entangled qubits. Entangling qubits can be done in code but will be explained later in the paper.

In all current quantum systems, there is some level of error. This is true even when running quantum code. When running code on IBM's quantum computer, there will be slight mistakes in the results. If, in a simulated quantum computer, two qubits are measured to be 00 or 11 50% of the time, when the same code is

run on the actual quantum computer, there may be results such as those seen in Figure 3. Notice that this error happens on ALL quantum computers and there is a big effort in reducing it, making them more reliable and capable of performing larger operations.

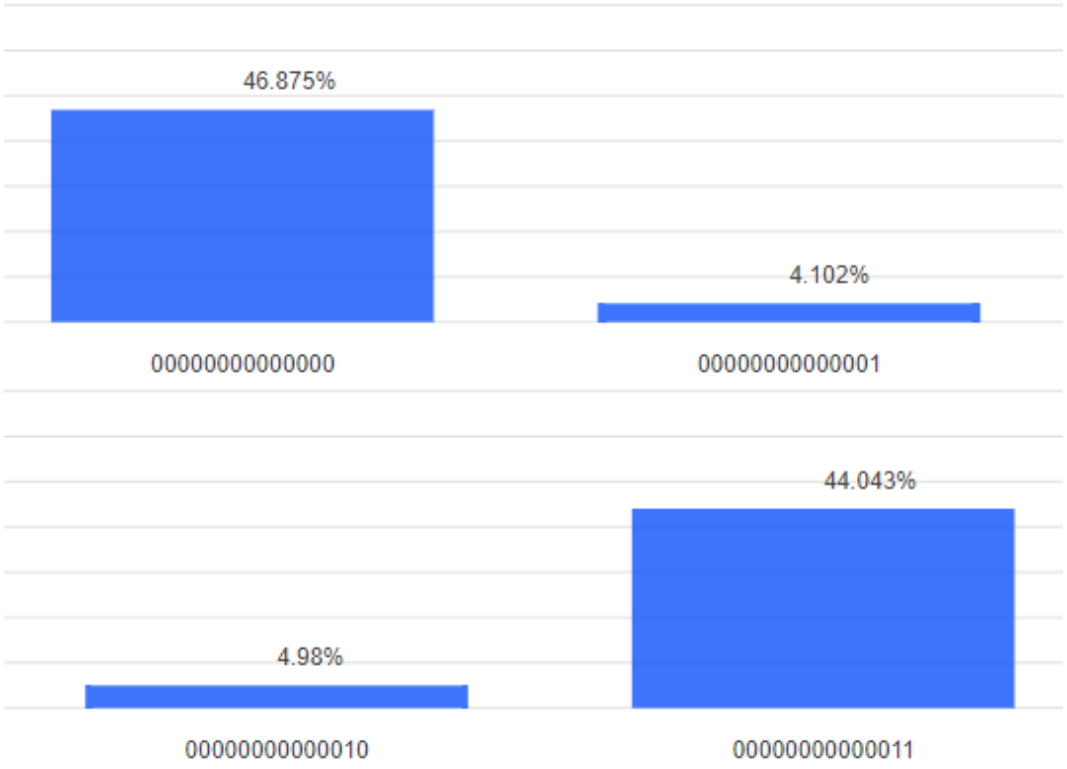


Figure 3: IBMQ Results

The goal of the rest of this paper is to describe, implement in software, and analyze the results of Shor's Algorithm, which is a Quantum Algorithm to find the prime factors of an integer.

Shor's Algorithm in steps

- 1. guess integer a as a prime factor
- 2. use quantum computer to find number r

  such that a^r = multiple of our number+1
- 3. $a^{r/2} \pm 1$ is a better guess

A high level overview of the algorithm follows. After the first step in guessing an integer as a prime factor, the best way to check if it is in fact prime is to use the Euclidean Algorithm. Finding the number r in the second step is where the bulk of the algorithm lies. Superposition is used to calculate powers on up to an infinite string of numbers, as well as applying a Quantum Fourier Transform to find the a period between those numbers. That period will lead to a calculation of r. After that, r must be checked to see if it is a better guess (is a prime factor), again using the Euclidean Algorithm to verify. The next few sections will discuss the prerequisite processes that happen inside of Shor's Algorithm such as the Euclidean Algorithm and the Quantum Fourier Transform, and then Section 4 will compile all of the methods into the final algorithm.

2 Euclidean Algorithm

The purpose of the Euclidean algorithm is to find the greatest common divisor of two integers a and b. The first step of the algorithm is to fit a and b into the following equation where q_0 is the amount of times b fits into a and r_0 is our remainder.

$$a = q_0 b + r_0$$

This step then continues in this fashion by using b and the remainder r_0, and then with r_0 and the next remainder r_1, until there is a remainder of 0.

$$b = q_1 r_0 + r_1$$

$$r_0 = q_2 r_1 + r_2$$

$$r_1 = q_3 r_2 + r_3$$

$$\vdots$$

$$r_n = q_{n+2} r_{n+1} + 0$$

Once the remainder is zero, the answer is given as r_{n+1}.

3 Traditional and Quantum Fourier Transforms

3.1 Traditional Fourier Transforms

One common use of the Fourier transform is to identify the dominant frequency components in a complex signal. This is done by transforming a wave from a space defined by time to another space defined by the frequency of the function, as illustrated in Figure 4.

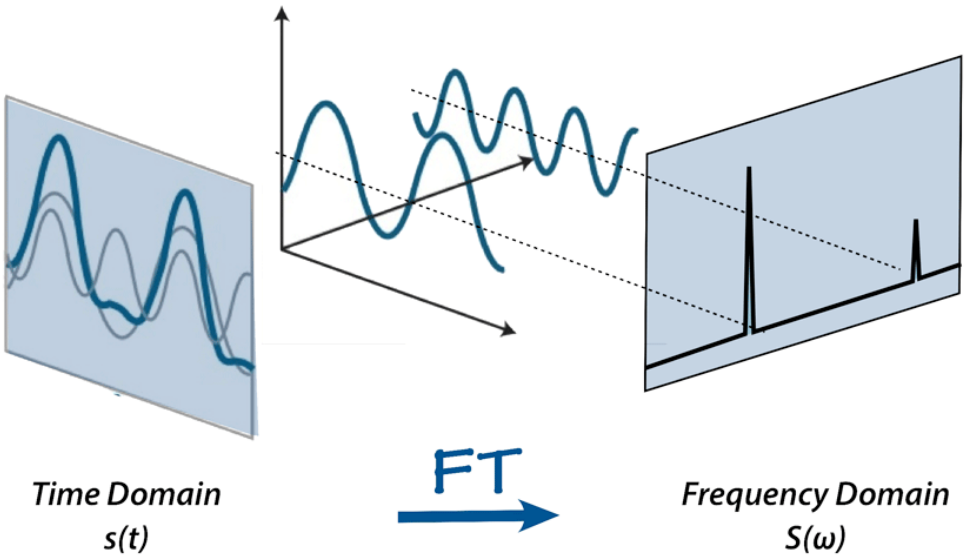


Figure 4: Fourier Transformation

In order to get from the time domain to the frequency domain, the following transformation is used.

DEFINITION 3.1 *The* ***Fourier Transformation*** *is defined as:*

$$\hat{F} = \int_{-\infty}^{\infty} F(x)e^{-ikx}dx$$

Conversely, to get from the frequency domain back to the time domain we can use the following transformation.

DEFINITION 3.2 *The* ***Inverse Fourier Transformation*** *is defined as:*

$$F(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{F}(k)e^{ikx}dk$$

When creating a computer algorithm for calculating Fourier Transforms, there are many different approaches, but the most famous ones are the Discrete Fourier Transform(DFT) and Fast Fourier Transform (FFT). Both work on discrete functions. The FFT follows similar steps as the DFT but uses a few shortcuts when the size of the input is a multiple of 2^n for $n \in \mathbb{Z}$. This effectively reduces the number of steps for the function to perform.

3.2 Quantum Fourier Transforms

The Quantum Fourier Transformation (QFT) is not all that different from the traditional transformations, however it uses quantum properties like superposition to make the process go much quicker on a quantum computer compared to a traditional computer. It is also similar to the FFT in that it works on discrete functions and performs more efficiently when the size is a multiple of 2^n.

Following is a brief overview of the QFT. The QFT takes in a list of numbers, N, preferably with the cardinality of N being close to or equal to 2^n. The closer the cardinality is to this, the more accurate the transformation. The transform performs the operation

$$\sum_x a(x)|x\rangle \rightarrow \sum_x A(x)|x\rangle$$

where

$$A(x) \neq 0 \implies x \in j\frac{N}{r} \text{ with } j \in \mathbb{Z}$$

This represents the list transforming from a physical space to our frequency space. All of the x that are not the frequency of the system will collapse to zero and everything left behind will be a multiple of the frequency, r, of the list. When the quantum state is measured, the state will collapse and can be represented as $|j\frac{N}{r}\rangle$.

4 Shor's Algorithm

Now a more detailed analysis of Shor's Algorithm will be explored. The premise of Shor's algorithm is to take a hard problem of factoring a number and convert it into another hard problem of finding a period of a function. This will be explained in the Preface. The input of the algorithm will be $M \in \mathbb{Z}$

4.1 Preface

DEFINITION 4.1 *The **order** of an element $a \in \mathbb{Z}$ in a group equals the smallest integer $r > 0$ that satisfies*

$$a^r = 1 \ mod \ M$$

If r does not exist, then the group is considered to be infinite. [4]

If a and M are co-prime, then the order of a is finite. Now consider the function

$$f(x) = a^x \mod M$$

Then we know that

$$a^x \mod M = a^{x+r} \mod M$$

if r is the *period* of the function (as it would be the *order* of a group). Now referring back to Definition 4.1, and assuming that r is *even*:

$$a^r = 1 \mod M$$

$$\implies (a^{\frac{r}{2}} + 1)(a^{\frac{r}{2}} - 1) = 0 \mod M \ [4]$$

This means that $a^{\frac{r}{2}} \pm 1$ can both be non-trivial factors of M as long as they are not 1 or M itself and it can be verified with the Euclidean Algorithm. This changes the problem of just finding factors in a classical sense to a problem of finding a the period r which will lead to the factors needed. Fortunately there is the QFT to figure that out which will be explored in Step 2 of the algorithm.

4.2 The Algorithm

Step 1: Guessing a number

Initially choose a random integer a such that $1 < a < M$ (M being the integer input). Immediately, the Euclidean Algorithm described in Section 2 is applied to a to determine if it is co-prime with M. If not, then a factor has been found and the algorithm is complete. However, this case is *very* unlikely when factoring very large numbers.

Step 2: Using the Quantum Computer

Now that there is a number with which to work, a two register quantum state is created. This state will be

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle|f(x)\rangle$$

where

$$f(x) = a^x \mod M \text{ and}$$

$$n \text{ must exist such that } M^2 \leq 2^n < 2M^2$$

The two registers are independent of each other. $|x\rangle$ is in the first register and $|f(x)\rangle$ is in the second register.

This state is similar to a group where each qubit in the state is an element of the group. Another good thing to note is the size or cardinality of this state because

that is how many qubits a quantum computer will need to perform the processes that continue. If M is 15, then

$$(15)^2 = 225 \leq 2^{(8)} < 450 = 2(15)^2$$

resulting in $8 - 1 = 7$ qubits required to perform the operation. Now take M as a 10 digit number, 1264759385. Then

$$(1264759385)^2 = 1,599,616,301,945,578,225 \leq 2^{(61)} <$$

$$3,199,232,603,891,156,450 = 2(1264759385)^2$$

Here, $61 - 1 = 60$ qubits are required to factor a number this large. However, for practical cases using Shor's algorithm, numbers being factored will be at least 100 digits long. Currently, the largest quantum computer available has 53 qubits so there is a long way to go before Shor's algorithm becomes practical.

Returning to the algorithm, $f(x)$ is now calculated. If one could peek into the quantum state at this point, they could find out when $f(x) = 0$, and that would be an answer to the algorithm. However, this is impossible since measuring the state collapses onto a random point, not necessarily when $f(x) = 0$. Fortunately, no matter what $f(x)$ is, it is guaranteed that all of the solutions for $f(x) = k$ for any integer k, are some period (the same distance) apart from each other. The register is then measured, leaving behind a random solution to $f(x)$. Also notice that calculating $f(x)$ requires taking a to the power of $1, 2, \ldots 2^n - 1$. This can be

done classically, but quantum computers allow all of these calculations to be done in one step through *entanglement*, just one of many processing optimizations of the algorithm.

The QFT is now applied to the remaining register to find a period. After that is applied, the final state is measured to find a value, highly likely to be a value j which is a multiple of $\frac{2^n}{r}$. Remember that r is the period of the function.

Step 3: Verifying an Answer

If r is even, then the Euclidean Algorithm may be used again to check if $a^{\frac{r}{2}} \pm 1$ has a common factor with the input M. If r is odd or the common factor is trivial (1 or M), then the process restarts at Step 1, picking a new random number. Once common factors are found with $a^{\frac{r}{2}} \pm 1$ and M, those two factors will be the prime factors of the input M.

4.3 Example

Let the input $M = 15$.

Step 1

Guess a random number a. Take $a = 7$. Immediately, use the Euclidean Algorithm to find the Greatest Common Divisor(GCD). GCD$(15, 7) = 1$ Thus 7 and 15 are

co-prime.

Step 2

Now create a quantum state. As calculated before, the size of the state will be 7.

$$|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle + |2\rangle|f(2)\rangle + |3\rangle|f(3)\rangle + |4\rangle|f(4)\rangle + |5\rangle|f(5)\rangle + |6\rangle|f(6)\rangle + |7\rangle|f(7)\rangle$$

Calculating $f(x)$ will result in

$$|0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + |3\rangle|13\rangle + |4\rangle|1\rangle + |5\rangle|7\rangle + |6\rangle|4\rangle + |7\rangle|13\rangle$$

Observe that the second register has a repeating pattern, 1,7,4,13. 1 happens every four numbers, 7 happens every four numbers, and the same follows with 4 and 13. In a quantum system a Quantum Fourier Transform would be used to find this period but these are small enough numbers to notice. It can be concluded that the period $r = 4$.

Step 3

Now verify

$$a^{\frac{(4)}{2}} \pm 1 = 48, 50$$

Use Euclidean Algorithm to find the common factors of $M = 15$ with 48 and 50.

$$\text{GCD}(15, 48) = 3$$

$$\text{GCD}(15, 50) = 5$$

And therefore, the prime factors of 15 are 3 and 5.

5 Remarks

5.1 Efficiency of Shor's Algorithm

Some of the best classical implementations of finding prime factors is guessing random numbers and using the Euclidean Algorithm to check if they are factors of an input. Although this sounds terrible, classical computers are fairly good at doing it. However, with very large numbers the computational power required grows exponentially. Thus it is said that the efficiency of traditional methods doesn't pass 2^n computations.

Shor's Algorithm surpasses this traditional method by completing the task in n^3 computations. This means that if a 100 digit number needs to be factored, it would take a classical computer up to $2^{100} = 1.267 \times 10^{30}$ computations while Shor's Algorithm on a capable quantum computer could complete it in roughly $100^3 = 1,000,000$ computations. One million computations is not an easy feat on modern quantum computers, however it is much more viable than the classical option which could take years to calculate on traditional computers.

5.2 Writing Quantum Code

There are multiple ways of writing quantum code, but for the purposes of this paper, IBM's Quantum Circuit Designer was used. The circuit designer is a base level quantum gate composer where quantum gates can be added to a series of up to 5 qubits. An example of the circuit designer can be seen in Figure 5. There are other methods of writing quantum code such as Qiskit. Qiskit is a library on top of Python which can be compiled into code that the IBM Quantum Computer can read. This allows for more versatility in writing code over the Circuit Composer, but ultimately they can both do the same thing.

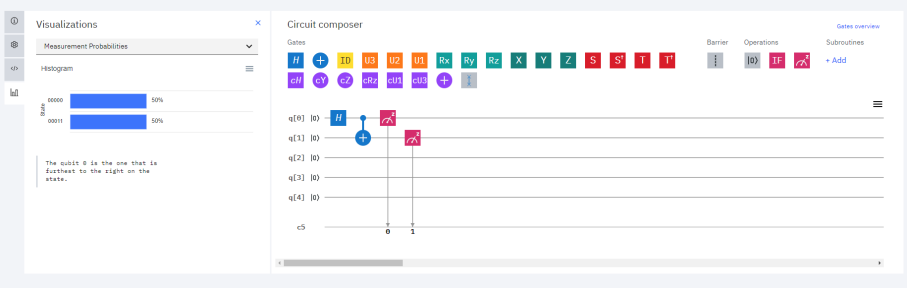


Figure 5: Sample Quantum Circuit

Both codes can currently be run on IBM's 5 qubit quantum computer but since it is open to the public, there are long wait times before code can be executed. The circuit shown in Figure 5 was run and took 20.5 hours to complete. The results can also be seen in Figure 3.

5.3 Thoughts on the Future of Quantum Computing

Throughout the time writing this paper, Quantum hardware has made a few giant leaps. At the beginning, IBM owned a 11 qubit computer, only able to do a few tests inconsistently. About a month later, Google started doing research on a 53 qubit computer. They made a lot of headway and some even claimed that they reached quantum supremacy, where they completed a virtually unsolvable problem by traditional computers (one that would take more time to solve than the universe will exist). Finally, Microsoft and Amazon (a couple hours prior to writing this) announced that they are both releasing cloud quantum computing for businesses and consumers to use. Some are aiming for 1,000 qubit devices but that may be a bit of a way off.

Some claim that quantum computers will never replace everyday computers like cell phones since they are so hard to create and maintain. Others are optimistic and believe quantum computers will grow like traditional computers did. Regardless if they end up in people's pockets or not, people will still have access to them through cloud services that are already being proposed. Creating quantum algorithms is still very hard and their uses are not quite known yet, but all that can be done now is to find new ways that they can improve everyday life.

References

[1] Elizabeth Ellen Parsons Simulation of a Quantum Prime Factoring Algorithm *University of Colorado*, 2016.

[2] Andrew Steane Quantum computing *Department of Atomic and Laser Physics, University of Oxford Clarendon Laboratory, Parks Road, Oxford, OX1 3PU, England*, July 1997.

[3] Terr, David Qubit *MathWorld–A Wolfram Web Resource, created by Eric W. Weisstein*, http://mathworld.wolfram.com/Qubit.html.

[4] Eleanor Rieffel and Wolfgang Polak Quantum Computing, A Gentle Introduction *The MIT Press*, 2014.