The University of Akron
IdeaExchange@UAkron

Williams Honors College, Honors Research Projects

The Dr. Gary B. and Pamela S. Williams Honors College

Spring 2021

Understanding the research and applications of quantum computing

Joshua Foss
jmf185@uakron.edu

Follow this and additional works at: https://ideaexchange.uakron.edu/honors_research_projects

 Part of the Other Computer Sciences Commons, Quantum Physics Commons, and the Theory and Algorithms Commons

Please take a moment to share how this work helps you through this survey. Your feedback will be important as we plan further development of our repository.

Recommended Citation

Foss, Joshua, "Understanding the research and applications of quantum computing" (2021). *Williams Honors College, Honors Research Projects*. 1410.
https://ideaexchange.uakron.edu/honors_research_projects/1410

This Dissertation/Thesis is brought to you for free and open access by The Dr. Gary B. and Pamela S. Williams Honors College at IdeaExchange@UAkron, the institutional repository of The University of Akron in Akron, Ohio, USA. It has been accepted for inclusion in Williams Honors College, Honors Research Projects by an authorized administrator of IdeaExchange@UAkron. For more information, please contact mjon@uakron.edu, uapress@uakron.edu.

Understanding the Research and Application of Quantum Computing

Joshua Foss

The University of Akron

March 25, 2021

Abstract:

This paper provides a brief overview of the principles and applications of quantum computing. Quantum computing utilizes the probabilistic state of quantum bits to enable the system to provide faster computation. Unlike classical bits, quantum bits can exist in a probabilistic state between 0 and 1; however, upon measurement, the state is collapsed and either 0 or 1 is return based on the probability. This begins one of the many hurdles of quantum computing, including lack of noise correction and limitation of quantum bits in the systems. Despite these, quantum computing has many applications, one being cryptography, first breaking current cryptography with Shor's algorithm and also providing a more secure method. So while quantum computing may not fully replace the classical computer, when it becomes practice, it will be a strong aid in data processing.

Introduction

Quantum computing is a powerful technology with the potential to completely impact the future of computer science and many other fields. While this technology is still ways away from consumer use, because of current limitations that are discussed later in this paper, it is essential to understand this young technology so that one does not get left behind when it does leap

forward. That is why I begin this project, so I could start to understand the basic concept of quantum computing and have a part in the technology when it becomes practical.

Currently, there are a handful of quantum computers being developed. Some of these computers are being created by companies such as Microsoft, IBM, Google, and Honeywell. As these companies began developing their systems, they also needed an interface for their computers. Various companies created their own: Microsoft with Q#, IBM with QisKit, and Google with Quantum Engine API. Other companies have the option of incorporating these already made interfaces into their own systems. Later in this paper, Q# will be discussed more in depth.

As I began my research into quantum computing, I started with one of the most important quantum computing concepts, the quantum bit or qubit. In the paper "Quantum Computing: Progress and Prospects," a qubit is described as being similar to a classical bit in that it has the states 0 and 1 but also that it can be a superposition of these two states (Horowitz & Grumbling, 2019). Because of this, the qubit can store exponentially more data in each qubit compared to a bit. For example, in a classical 2-bit system, in order to compute each possible output of a function, one would need to generate 4 different outputs; however, using qubit, just 2-qubit could store all 4 four outputs. This is because when the qubit is in a superposition, it is in a probabilistic state between 0 and 1. So traditionally qubits are represented as a vector, |0> being [1,0] and |1> being [0,1]. These vectors grow by a factor of two as one begins adding qubits to the system. So along with 0 and 1, the qubit also has two coefficients, with one corresponding to 0 and the other to 1. These two coefficients squared and added together will equal 1. Another way the value of a qubit can be described is through something called a Bloch sphere, depicted below. A Bloch sphere is a form of a unit sphere and represents all possible values of the qubit.

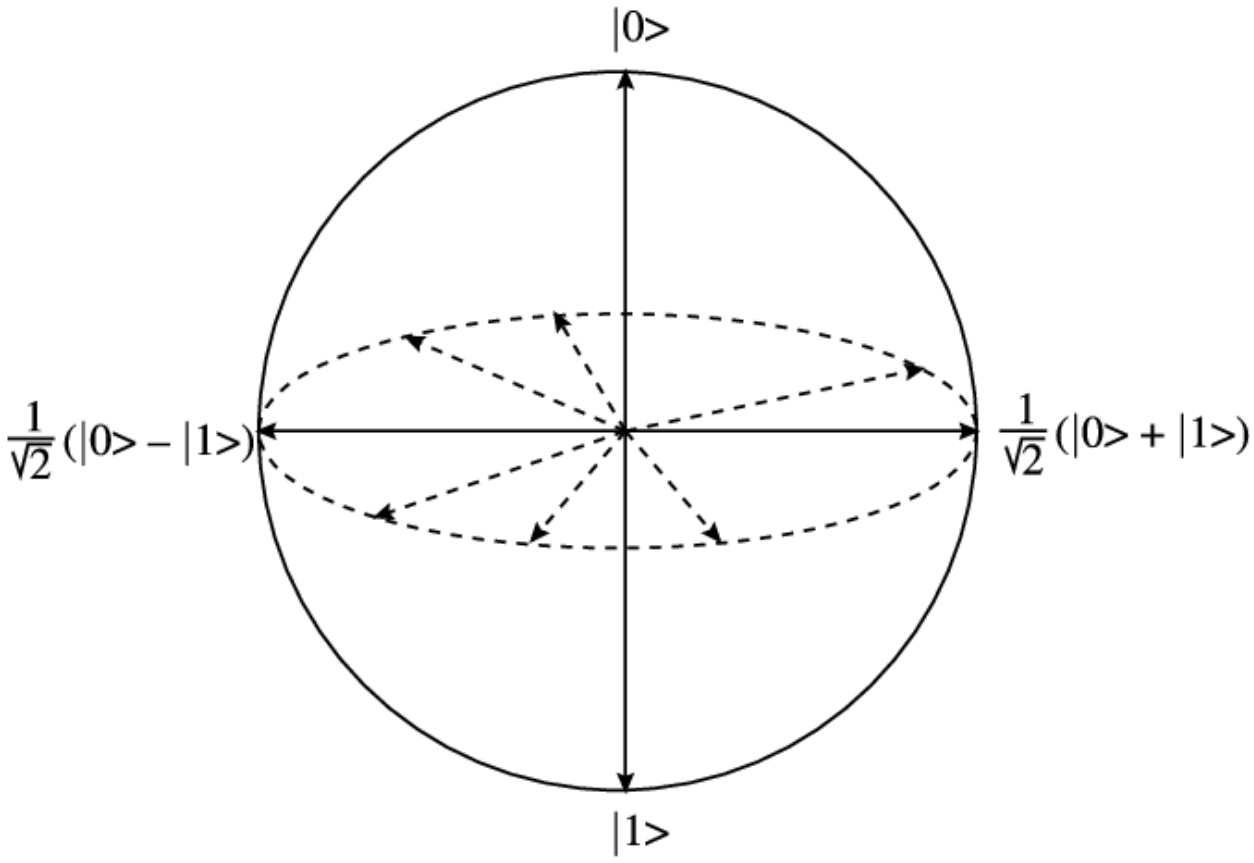


Figure 1

However, to move a qubit into a superposition state gates are required. The first gate use is a gate to move the qubit 90 degrees on the Bloch sphere. This would be from |0> to $1/\sqrt{2}(|0> + |1>)$. This gate is called the Hadamard gate and enables the superposition of the qubit. Along with this gate, several other gates move the qubit in different amounts around the Bloch sphere. For example, the X gate functions as a not gate that will rotate a qubit 180 degrees across the Bloch sphere. These gates are also represented by vectors and when multiplied by the qubit's vector, the product is the result of the gate operation on the qubit. The quantum gates are depicted below

along with their respected vector.

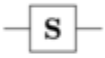
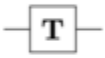
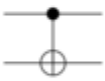
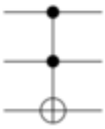
Operator	Gate(s)	Matrix
Pauli-X (X)		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
π/8 (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

Figure 2: The name, representation, and matrices of quantum gates
https://en.wikipedia.org/wiki/Quantum_logic_gate

It is from this that so much of the power of quantum computing is derived. It is also one of the biggest hurdles. Even though the qubit can contain so much information because of this property,

upon measurement, only one value is recorded, 0 or 1, based on their probabilities. So, if a qubit were at the value of $1/\sqrt{2}(|0> + |1>)$ there would be a 50/50 chance that 0 or 1 would be recorded. Multiple measurements must be taken to make up for this, and then the superposition state can be exposed by variations in the output. However, there is another property of a qubit that makes this difficult. Once a qubit is measured, the probabilistic state is collapsed to a deterministic state of the value recorded. Therefore, the program must run multiple times with multiple qubits since the same qubit cannot be measured multiple times.

Another concept of quantum computing that provides it one of the biggest upper hands over classical computing is qubit entanglement. In the paper "Quantum Computing: Progress and Prospects," this is described as two or more qubits that are intrinsically linked together causing the measurement of one to dictate the measurement of the others, regardless of their separation (Horowitz & Grumbling, 2019). In other words, the measurement of one qubit collapses the state of another even if the qubit has zero interaction. Entanglement has many applications, such as uses in cryptography, with two users having one piece of an entangled pair, and since these two qubits will determine the other, messages can be encrypted and decrypted with the values returned by the qubits. However, this can be difficult to achieve because of the need to get a qubit to each user.

Qubits need a system to work within, and there are a few different types of quantum computers: analog quantum computer, AQC, Noisy intermediate-scale quantum gate based, Digital NISQ, and fully error-corrected based. First, AQC operates through coherent manipulation of qubits, by changing analog values without the use of gates. (ADD)

Problems

Now, I have covered a couple of the benefits of quantum computing and its potential power once fully implemented. However, there are many constraints currently on the functionality of the technology. First, a qubit cannot be copied, its state can only be moved from one qubit to another, and this creates a difficulty for debugging since multiple qubits of this state are not easily generated. A more significant hurdle is the lack of noise immunity in quantum operations. These are slight imperfections in the signal sent through the quantum gate that can accumulate and results in loss of coherence (Horowitz & Grumbling, 2019). Currently, noise in quantum systems is a significant issue that prevents the scaling of these systems.

Noise comes from two different places, loss of coherence and cross talk. Loss of coherence can affect all aspects of a qubit, being the magnitude and phase. One way to prevent this is to isolate the system from the environment. However, this cannot be completely done, and eventually, the quantum system will trend toward the environment. Meaning an excited state will lose energy over time if the environment of the system is low energy. Also, loss of coherence can come from physical processes adding random phase shifts to the quantum state over time. The other noise is cross talk which is when qubits used to perform gate operations affect other qubits in that system. This adds to the noise because there is a chance the output of a gate operation is wrong, and along with this operation increasing the chance of this for future operations. Both metrics, loss of coherence and cross talk, are accounted for in the error rates of a gate, and the error rate is determined from randomized benchmarking.

Application

Quantum computers have many possible applications, with one of the most talked about being cryptography. This is the case for two reasons: breaking the current cryptography and then the need for creating better cryptography. Currently, most encryption methods on classical computers use the fact that massive integer factorization is nearly impossible. This provides extremely secure encryption. If integer factorization becomes easy and possible for these numbers, most current encryption will fail, security will be breached, and a different solution will need to be found. This is precisely what quantum computing has done, providing an easy way to conduct integer factorization. Shor's algorithm is a theoretical application of this for quantum computers. Because of this disruptive technology, research and development are being poured into this field, attempting to overcome noise and limited qubit constraints.

In order to promote the development of quantum software, even though the hardware still needs improvement, Microsoft created a toolkit for quantum development, along with a programming language Q# (Quantum development kit - Quantum Programming: Microsoft Azure) . This simulates the quantum system so that one can develop quantum programs without the need for physical hardware. However, since it is a simulation, actual benefits of a quantum computer will not occur, such as speedup. With this in mind, I implemented Shor's algorithm for integer factorization. I did this by creating a C# driver function that called the Q# implementation of Shor's algorithm. The driver function, displayed in figure 3, first picks a random number, *x*, between 1 and the number to factor, *N*. Once a is determined, it is checked if it is trivial. If not, *x* is sent into the quantum program to find the period of *x* and *N*. The period, *r*, returned, and the algorithm ensures that the gcd(*x*^*r*/2 +- 1, *N*) are nontrivial factors (Lu,

Browne, Yang, & Pan, 2007). Those two values are the factors. Because of the properties of quantum computing, the system can easily find the period of the values.

```
using (var qsim = new QuantumSimulator())
{
    int N = 15;
    int p = 0;
    int q = 0;
    bool trivial = true;
    while (trivial) {
        Random rnd = new Random();
        int x = rnd.Next(1, N);
        Console.WriteLine("x: " + a.ToString() + "\n");
        if (GCD(x, N) == 1)
        {
            // answer should be 5
            var restored =  await QuantumPeriodFinding.Run(qsim, N, x);
            int r = int.Parse(restored.ToString());
            if ((r % 2) == 0 && (Math.Pow(x, r/2) % N) != (N - 1))
            {
                p = GCD((int)Math.Pow(x, r/2) - 1, N);
                q = GCD((int)Math.Pow(x, r/2) + 1, N);
                if ( p != 1 && q != 1)
                {
                    trivial = false;
                }
                else
                {
                    Console.WriteLine("Trivial\n");
                }
            }
            else
            {
                Console.WriteLine("Trivial\n");
            }
        }
        else
        {
            Console.WriteLine("Trivial\n");
        }
    }

    Console.WriteLine("p: " + p.ToString() + "\nq: " + q.ToString());
}
```

Figure 3: Shor's algorithm C# driver function

When developing this program, I had in mind to compare the time taken to find the factorization of a few numbers using the Q# program and a classical C++ program. First, I used the Q# program to find the factorization of 15, which took 11.6 seconds to run. I then also attempted to factor 143. However, this required too many qubits and caused the program to crash. So instead of checking a larger number, I tried 9. Similarly, this did not work, and was too trivial, sending the program into an infinite loop. Because of this, I stuck with 15 and began to compare the quantum time with the classical time; however, the classical time was less than 1 millisecond, so there was not much to compare.

The factorization of 15 on the quantum program was significantly slower than the classical program for 2 main reasons. The first apparent reason is that this Q# program is running on a simulation. The properties that make the quantum computer more efficient are simulated on a classical computer and are then limited by the efficiency of the classical computer. The other reason is that 15 is a small number and the speed up may not be evident with small numbers. Shor's algorithm will not increase the time taken as quickly as a classical computer since its complexity is polynomial, compared to the classical algorithm complexity being sub-exponential. So, quantum factorization may be only preferable with dealing with large numbers. However, this will require many qubits, which the current system does not have. When the system begins to support a large number of qubits, with little noise or noise correction, most current encryption will become insecure, and an alternative will be needed, also known as post-quantum cryptography.

Discussion

Over time, quantum computing will continue to progress because of the threat of breached security. The fact that it will break encryption drives the research forward so that one

can get there first and create a solution or crack someone else's encryption. Without this, funding would be limited because there is no consumer benefit currently in sight. However, as we learn more about this technology, we will learn to more easily create higher qubit capacity systems, and with this, consumer applications may arise. Currently, quantum computers are thought to be used as aids for classical computers, being called by a driver function, from a classical computer, complete some computation, and return a value to the driver function. One way these could aid classical computers is their use in parallel processing. While this currently may not be possible, because of the current limitations of quantum systems, one may be able to use entangled qubits to allow two or more systems to communicate nearly instantly in order to pass information or synchronize between processes. While this may not be a consumer application, it could benefit industries that use parallel processing and would also add funding to the field, allowing for more growth.

Conclusion

Quantum computing will eventually take the floor as a dominant technology and completely change the world. Once our understand and use of the technology increases, allowing for more complex systems with better error correction, and a larger number of qubits, we will be able to disrupt the current technology industry. This will require a change in the way we process data and encrypt data. Currently, quantum systems are thought to only be practical for applications that can leverage quantum properties, but just like how graphics processing units eventually became practical for non-graphical applications through CUDA, this may also happen with quantum systems allowing for anything to benefit from the powerhouse within quantum systems.

References

Horowitz, M., & Grumbling, E. (2019). Quantum computing: Progress and prospects. Washington, DC: The National Academies Press.

Quantum development kit - Quantum Programming: Microsoft Azure. (n.d.). Retrieved April 07, 2021, from https://azure.microsoft.com/en-us/resources/development-kit/quantum-computing/

CUDA toolkit. (2021, March 29). Retrieved April 07, 2021, from https://developer.nvidia.com/cuda-toolkit

Lu, C., Browne, D. E., Yang, T., & Pan, J. (2007). Demonstration of a compiled version of Shor's quantum factoring algorithm using photonic qubits. Physical Review Letters, 99(25). doi:10.1103/physrevlett.99.250504