Quantum Computing Architectures

Justin Clark

Table Of Contents

Introduction

Throughout [1], Princeton and Harvard computing architectures are conceptually developed. It is also clear from [1] that strategies such as parallel processing have the potential to dramatically increase the performance of these classical computing architectures. As an example for how performance might increase with additional processors, let us compare the processing power of a single CPU system to that of a fifty CPU system. All else being equal, one would expect the performance of the fifty CPU system to be fifty times faster than single CPU system. In other words, the performance of these classical systems should increase linearly. However, this is not the case in practice. There is something called Amdahl's Law that accounts for the fact that not every part of a program is parallelizable [2]. The equation is given in the following equation,

$$speedup = \frac{1}{f + \frac{1-f}{P}}$$

where f defines the proportion of the program that can be parallelized and P is the performance increase of the non-parallelized part of the program. If 100% of a program is parallelizable, we would, in fact, see a perfectly linear performance increase as the number of processors increased as hypothesized above. Otherwise, we would see asymptotic performance increases after a certain point. The point of bringing up Amdahl's Law is to illustrate that the ideal performance increase is not necessarily always possible.

Another famous law to consider is called Moore's Law [3]. This law states that the number of transistors on computer chips(which could be considered a proxy of computational power) increases about once every two years. Of course, transistors can not decrease in size forever. Eventually, once they are small enough, transistors will be so small and densely packed together that they will be affected by quantum mechanics and susceptible to thermodynamic effects (among other issues). Similar to Amdahl's Law, the performance increases will be asymptotic and are predicted to stall sometime in the next few decades [4]. Ultimately, the limits of Moore's Law can be boiled down to the fundamental limits of physics of our current computing paradigm.

What if, on the other hand, we were able to change the fundamental method of computation so the limits of classical computing did not apply? This is where quantum computing comes into play. In fact, the idea of quantum computing was really solidified when David Deutsch built upon the Church-Turing Thesis and some ideas of physicist Richard Feynman [5] to develop the foundation of the theory of quantum computing [6]. The basic idea is that changing the fundamental method of computation can change the time complexity of certain algorithms. Further (as we will discuss in a later section), quantum computing has the potential to exponentially speed up certain algorithms as well as greatly improve upon others.

THE SCOPE OF THIS PAPER

Before proceeding too far into the depths of quantum computing, it is worth noting that I will not be covering the necessary quantum physics to truly understand how to build a quantum computer. Instead, I will overview the basic concepts of quantum computing and quantum computing architectures (while ignoring how

counter-intuitive some of the concepts are). The discussion will also skip advanced details in quantum computing such as quantum teleportation, superconductors, quantum photonics, etc. After building the motivation as to why quantum computing is a potential solution to the stalling of Moore's Law in the previous few sections and then outlining the scope of this research paper, the sections of the rest of this paper are as follows:

- A basic conceptual overview of quantum computing.
- An overview of quantum gates and quantum circuits.
- General practical requirements of quantum hardware.
- A discussion of actual quantum (or supposedly quantum) computing architectures being developed in industry.
- An overview of popular quantum algorithms and their applications.
- A final summary and look into the future of quantum computing.
- An appendix with my experimentation designing quantum circuits and programs.

BRIEF CONCEPTUAL OVERVIEW OF QUANTUM COMPUTING

As discussed in a previous section, quantum computing is a complete redesign of how computations should be performed. Classical computing builds off of Information Theory in the sense that all bits are represented at a hardware level by voltages; 0 or 1, off or on. In quantum computing, bits are thrown out the window because they are restrictive. Instead of bits, quantum computers use something called a qubit. These qubits can be thought of as both a 0 and 1. More specifically, a qubit is typically represented by a complex number, which has a real part and an imaginary part. Instead of thinking about a qubit as having a real part and an imaginary part, the qubit can be thought of as having a zero part and a one part [5]. This has elegant

mathematical implications since complex numbers, in turn, are very clearly represented and transformed by vectors, matrices and their operations. Since qubits can be be abstracted to linear algebra, there is much less quantum physics knowledge (practically zero) required for a programmer to develop quantum algorithms than one might imagine [7]. However, we will save the bulk of this discussion for the penultimate section of this paper.

So, what are the practical computational implications of using qubits over bits? In the beginning of this paper, the discussion began with parallel processing in classical computing architectures. Recall that the ideal speedup of introducing a new CPU is linear. That is, a system with 50 CPUs (not all that uncommon these days) processing an algorithm that is 100% parallelizable is 50 times faster than a system with a single CPU. With a quantum computer, the theoretic speedup is *exponential* as the number of qubits increases. So, let's say a quantum computer with a single qubit can process an algorithm in x seconds. A quantum computer with 50 qubits is sped up by a factor of 2^{50}, i.e. it takes $\frac{x}{2^{50}}$ seconds. Let this sink in for a moment. A classical computer with 50 CPUs can theoretically speed up processing by a factor of 50 over a single CPU system. A quantum computer with 50 qubits can theoretically speed up processing by a factor of $1.126\mathrm{E}+15$ over a single qubit system.

How is this even possible? First, let us focus on a single qubit system. There is something in quantum physics called superposition where particles are actually in many different places at once. In the context of quantum computing, superposition is

applied to the particle in the single qubit. In other words, the qubit represents all possible combinations of 0- and 1- parts. Geometrically, it makes sense to represent the possible states of the qubit as the surface of a sphere (specifically, the *Bloch Sphere*) represented in the Figure 1 [8].

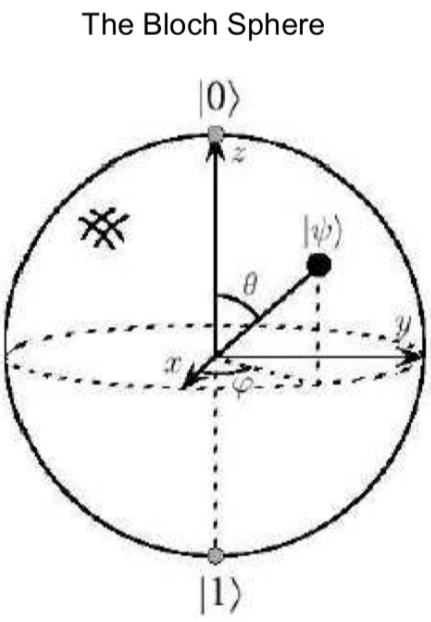


Figure 1: The "north" and "south" poles of the Bloch Sphere represent the 0- and 1-parts respectively.

From this point forward, the notation for the 0- and 1-parts of the state of a qubit will be represented by $|0\rangle$ and $|1\rangle$. The overall state of a qubit is typically represented by $|\psi\rangle$.

The next important concept to cover in the conceptual overview of quantum computing is something called *entanglement*. This is important for the following reason: a single qubit certainly represents more information than a single bit, but modern classical computers can operate on *billions* of bits with relative ease. How can a quantum computer take advantage of the power of *multiple* qubits? This is where entanglement comes in. There is an aspect of quantum physics where multiple particles can be in a state of entanglement, which simply means that a measurement or transformation to one entangled particle affects the other entailed particle(s). Additionally, all the states of the qubits must be in a state of *coherence* which you might think of as particles speaking the same language, i.e. they can actually communicate with each other. Qubits that are not cohered will be unable to communicate and the quantum computation is no longer effective. Keeping qubits in

coherence has proven to be one of the most difficult problems in quantum computing and quantum information. As we will discuss later, the leading quantum computing research groups are all focused on designing more stable qubits that can remain cohered for a longer period of time.

Quantum Gates and Circuits

CLASSICAL GATES

Many textbooks, such as [5] and [10] build the intuition behind quantum gates and circuits by first focusing on classical gates. Of course, classical gates such as `OR` and `NAND` are very familiar to anyone in the field of computer science. The power of these two-bit input, one-bit output gates is that with some combination of `AND` and `NOT`, or `OR` and `NOT`, *any* Boolean function can be represented. Practically, this allows computer engineers to construct entire computers consisting entirely of these few simple gates. This property is called *universality*. However, `AND` and `OR` gates (among other two-bit input, one-bit output gates) also have an undesirable property called *irreversibility*. This means that they lose information when converting input bits to output bits. Consider the following truth table of an `AND` gate:

A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

There is a clear function mapping the input bits **A** and **B** to the output bits, but there is no way of taking the output bits and determining the input bits. Why is this important? As discussed in [4] and [10], when information is lost, the laws of thermodynamics dictate that energy must be dissipated according to the following equation: $kTln(2)$ per bit erased, where k is Boltzman's constant and T is the absolute temperature in Kelvin. This idea was touched on in the previous section when discussing the limits of Moore's law. Fortunately, classical gates *can* also have a property of reversibility in gates such as `SWAP` and `NOT`. A truth table demonstrating a `SWAP` gate is below, which simply switches the **A** and **B** bits:

A	B	A'	B'
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

It is clear that outputs can be mapped back to inputs in this case. `SWAP` gates do have the property of reversibility, but they do *not* have the property of universality. There are, in fact, reversible gates that are also universal, but these gates require 3-bit inputs and 3-bit outputs. In practice, this is much more inefficient since complexity of classical gates is exponential. That is, dealing with 4 permutations from 2-bit inputs is much simpler than dealing with 8 permutations of 3-bit inputs, and so on. This has led to the use of 2-bit irreversible, universal gates as opposed to 3-bit reversible, universal gates in most computer systems. Another important concept, to at least mention briefly, is

that classical states can be described as permutation matrices. This allows for easy manipulation of classical bits in terms of matrix operations [10].

QUANTUM GATES

Reversibility and universality of classical gates are important considerations in developing quantum gates. There is one major difference that is the core advantage of quantum gates: while classical gates only manipulate 0 or 1 values, quantum gates have much more flexibility and can operate on qubit superpositions and are not forced to output either 0 or 1. It is beyond the scope of this paper to cover the quantum physical laws that lead to this behavior, so major details are omitted. However, it suffices to know that quantum states can be described as *unitary* matrices which means that they are *always* reversible. This is extremely important because there is no information lost in quantum operations, thus making them thermodynamically efficient.

With the necessary background of classical gates and understanding the power of reversibility, we can appreciate quantum gates. Quantum gates give us the flexibility to create something like a $\sqrt{NOT}$ gate. As mentioned above, gates can be represented a matrices. The matrix representation of such a gate is the matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}^{\frac{1}{2}}$ which can result in a non-binary output. This example of the $\sqrt{NOT}$ gate in [10] is one of the simplest gates and is not necessarily implemented in practice. However, the authors go on to discuss something called a *Hadamard* gate which they purport as possibly the most important single qubit gate in practice. The Hadamard

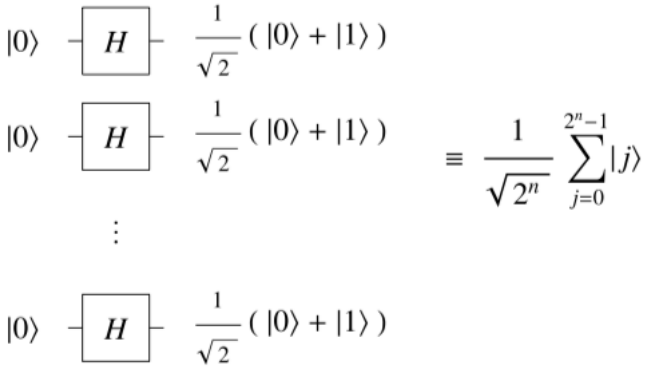


Figure 2: Initialized to the zero (ground) states, the number of representations by the Hadamard gate increases exponentially as the number of gates increases linearly [10]. The Hadamard gate can be thought of as assigning equal probability to 0 and 1.

gate is described by $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\frac{1}{2}}$. Figure 2 illustrates the power of this gate.

The above figure is the justification behind the claim that quantum computing offers an exponential speedup with a linear increase in qubits. There are many other important quantum gates; particularly multiple-qubit gates. One of the most important things about quantum gates is that almost every *two*-qubit gate is universal, unlike classical gates which require three bits. This allows relatively simple implementation of reversible and universal gates.

QUANTUM CIRCUITS

The discussion on quantum circuits will be brief since they largely follow from quantum gates, but quickly move beyond the scope of this paper. I would like to show one example of a block diagram from [10] that illustrates how several gates may be strung together and how the qubits are measured or *read-out* at the end of a computation (note that the description is also taken from [10]). Figure 3 shows a more complicated circuit of quantum gates. However, it is worth noting that there are some great resources for those wanting to learn quantum computing and how to design quantum circuits. In [38], the IMB Quantum Experience™ is introduced and there are

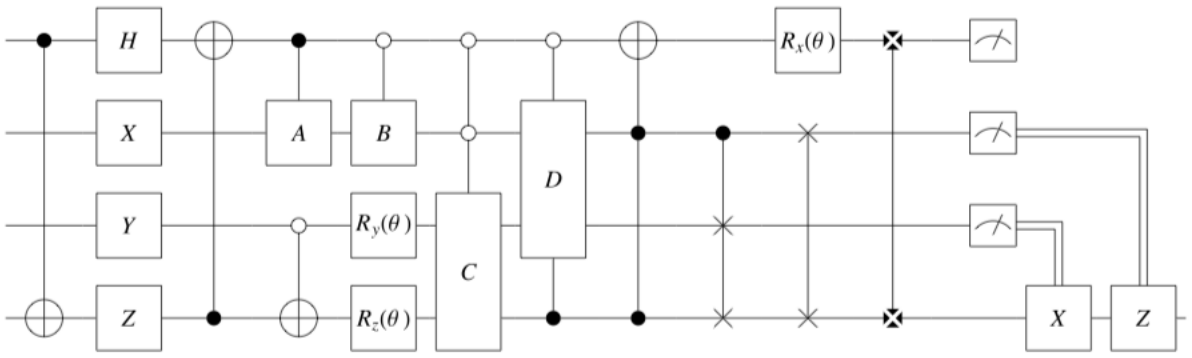


Fig. 3 A quantum circuit diagram illustrating different types of quantum gates. Single qubit gates are drawn as boxes on the qubit on which they act labeled with their gate name (H for Hadamard gate, X and Y for Pauli gates, $R_x(\theta)$ for a rotation gate about the x-axis through angle θ etc.). Controlled gates are drawn with their controls depicted as circles (white for control-on-$|0\rangle$ and black for control-on-$|1\rangle$) and the operation they perform on the target is show as a labeled box. At the end of the computation certain qubit values are read out. Also shown are the special 2-qubit gates, CNOT, SWAP, and iSWAP

several open source tools for actually creating quantum circuits and even writing quantum algorithms. Two examples will be presented in the Appendix.

Quantum Hardware

In the previous section, the idea of quantum gates and circuits was presented, but they were presented in an abstract and mathematical way. How does one think about actually designing the hardware behind quantum computing? There are many things to consider. First, as discussed in the introductory section, qubits must remain entangled *and* cohered. Maintaining these two states is easy. That is, if you have some way of designing a system completely undisturbed by external forces like heat, radiation, electromagnetic frequencies, sound, etc. Upon second thought, maybe this is quite a difficult task. In reality, this is not possible which means there are bound to be errors associated with entangled and cohered qubits in quantum computations. Fortunately, one of the pioneers in the field of quantum computing, Peter Shor, formulated a method for these errors called *quantum error correction*. This idea is

discussed in sections 1.7 and 1.8 of [11], which are lecture notes from Physics 229 at Cal Tech.

Before discussing Shor's proposed method here, it is important to first consider the types of errors that can be corrected. Quantum error correction is a method that accounts for the unique errors encountered in real-world quantum computers that are absent in classical computers. There are four types of errors discussed in [11§1.7]. The first type of error encountered in quantum computers is a phase error. Phase errors occur when qubit states are flipped to their orthogonal counterparts. For example, the state $\frac{1}{\sqrt{2}}[|0\rangle + |1\rangle]$ could switch to $\frac{1}{\sqrt{2}}[|0\rangle - |1\rangle]$. The second error has to do with the continuity of qubits. While classical bits are discretely 0 or 1, quantum information is continuous and there could be minor errors in measuring quantum states. The error term is usually referred to as ε. The third type of error has to do with measuring qubit states. It has not been explicitly stated up to this point, but when a measurement or read-out of a qubit state occurs, the *wave function* (i.e. the probability curve of where the particle *could* be at a given time) collapses and fails to exhibit quantum properties such as superposition. This measurement causes a disturbance to the other entangled and cohered bits, which must be taken into account. Lastly, the fourth error that must be corrected is related to the idea that quantum information cannot be copied exactly as it can be with classical information.

Quantum error correction has similarities to classical error correction methods such as *checksum* [12], but is, of course, adapted to suit the world of quantum

physics. The method proposed by Shor is to encode a single qubit in 9 qubits given the following equations from [11]:

$$|0\rangle \Rightarrow |\bar{0}\rangle \equiv \frac{1}{2^{3/2}}[|000\rangle + |111\rangle][|000\rangle + |111\rangle][|000\rangle + |111\rangle]$$
$$|1\rangle \Rightarrow |\bar{1}\rangle \equiv \frac{1}{2^{3/2}}[|000\rangle - |111\rangle][|000\rangle - |111\rangle][|000\rangle - |111\rangle]$$

The basic idea behind these equations is to build redundancy into quantum information. With these 3 clusters of 3 qubits, we are able to compare the relative phases of each cluster without actually taking a measurement. There are additional unitary matrix transformations that can correct for the small, continuous errors that will map a quantum state back to its original form (most of the time). With these (simplified) steps, all four error types discussed above can be corrected for. Of course, beyond a certain point, any sort of error correction is hopeless and qubits will cease to be cohered.

Now that we have covered quantum error correction and why we need it for practical quantum computers, it is time to discuss how the hardware is actually implemented to take advantage of quantum effects while also accounting for quantum errors. There are five necessities for practical quantum computer implementation discussed in [11§1.9]. 1) The ability to store qubit states long enough for them to be cohered, 2) isolating the computing environment from the external environment to minimize decoherence errors, 3) the ability to measure the states of qubits efficiently, 4) to facilitate meaningful state interactions and manipulations between qubits (i.e. quantum gates and circuits), and 5) precision.

While the field of quantum computing is largely in the research phase, there have been three major proposed architectural implementations to meet these goals which are also discussed in [11§1.9]. Note that all three of these methods require a high-level understanding of particle/energy physics, so I will be covering these implementations at the highest level possible. The first of these is called an *Ion Trap* which requires lasers to target qubits in the ground state, excite them (i.e. apply quantum "gates" to them), and then the excited qubits will emit light back out. When an ion absorbs energy from the photon, information is is stored directly in the atom. The quantum computers under this scheme are very well insulated, but the speed of the computations are limited by a bottleneck called the *energy-time uncertainty relation*. This relation is beyond the scope of this paper, but it suffices to know that the ion trap implementation is not as fast as other methods. The second implementation scheme is called *Cavity QED*. This is another method that uses lasers, but instead of information being stored in the internal states of ions (atoms), the information is encoded in the polarization of a photon (the nature of how its wave behaves). Lastly, the lecture notes describe something called *NMR* (nuclear magnetic resonance). As of the last update of [11§1], this is the newest and most popular method in coherent quantum processing. The basic idea of NMR architectures is to store the quantum information of qubits into the nuclear spin of particles. This is advantageous because the spins of particles remain cohered for a relatively long period of time, which means that there is enough time to do meaningful computations.

There are a few more important topics to cover before diving into some real-world quantum systems developed by large companies. The first is a recent extension

of the NMR technology described above. In [13], the authors describe a method for implanting qubits into silicon chips, and are excited via magnetic forces (as opposed to lasers). This approach allows qubits that are far apart to seamlessly communicate, whereas previous designs only allowed qubits to interact (remain in coherence) if they were spread over short distances. Since [13] was only recently accepted to *Nature* in June of 2017, many of the large companies working on quantum computers have not used this method for architectures currently in production.

The next thing to note is that practical quantum computing systems take many precautions to keep external forces from interacting with fragile qubits and their states. The most obvious precaution is to keep the system as cold as possible because heat can drastically affect how long qubits remain cohered. Temperatures are usually measured in *micro-Kelvin*. Another, related, precaution is that excess particles near the system (e.g. "air" molecules such as O_2, CO_2, etc.) can also interfere with qubits, so systems are often running in a (near) vacuum.

Lastly, it is important to discuss the count of qubits advertised by many of the systems that have been created. At around 50 qubits, it is widely expected that quantum computers will start to surpass modern supercomputers. However, there are some systems that have been able to reach 50 qubits and still vastly underperform classical computers. This is because many qubits are required to correct quantum errors (which is discussed above). In other words, there has yet to be a quantum computer created (as far as the public is aware) which is equivalent to a "pure" 50-qubit system.

Real-World Quantum Architectures

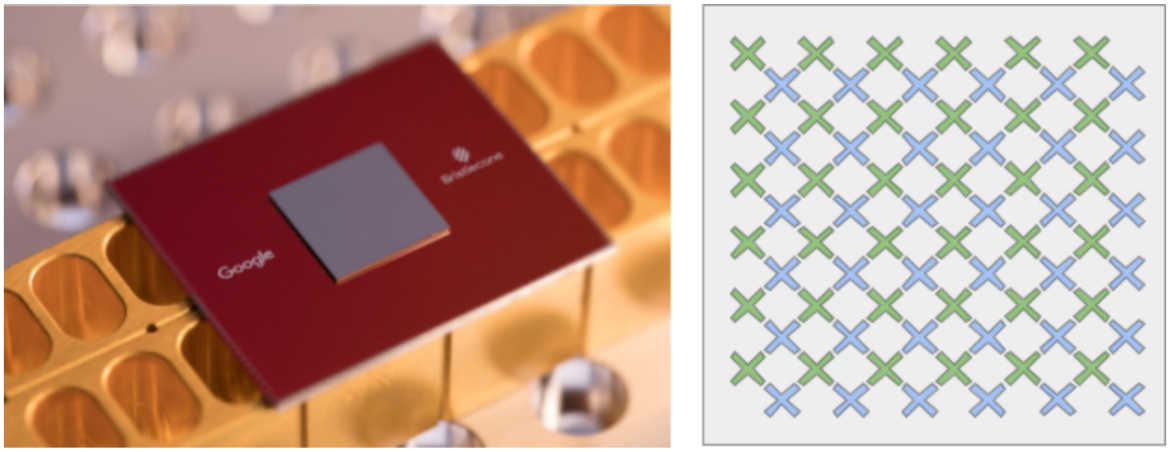
The previous sections have covered most of the necessary material to fully appreciate what companies like Intel, Google, IBM, Microsoft, and D-Wave Systems are doing. The first part of this section will overview the most recent Google quantum processor *Bristlecone*, which was done in partnership with Intel. The second part of this section will cover the 50-qubit quantum computer designed by IBM. The third part of this section will describe Microsoft's unique approach to quantum computing called topological quantum computing. Finally, the fourth part of this section will cover a more controversial "quantum" computer created by a company called D-Wave Systems. One preface I would like to give for this section is that quantum computers are still very much in the research and development phase. This means that there is quite limited literature on detailed designs for these systems.

GOOGLE BRISTLECONE™ PROCESSOR

This new 72-qubit processor was first announced on the Google Research Blog in March of this year (2018). To understand the motivation for building the Bristlecone™ processor, we must first understand its 9-qubit predecessor. The idea behind the 9-qubit processor was to achieve the absolute lowest possible error rate (which was extensively discussed in the previous section) on a silicon chip. Google designed the 9-qubit processor based on the ideas of [16] which proposes a linear array of qubits to minimize error with a superconducting circuit. Additionally, [17] provided a method for digitizing quantum computing with the same superconducting circuit. The approach of [17] largely defines the potential applications of a system that uses the digitized method quantum computing which are primarily physics and chemistry simulations.

The Bristlecone™ extends the 9-qubit processor into a 2-dimensional array of qubits (specifically, a 9-by-8 qubit array). The communication between qubits is a little different in this processor than in the linear array of the 9-qubit processor. The “grid” layout of qubits means that each qubit can interact with its four nearest neighbors. Figure 4 below, from [14], shows an image of the processor and the arrangement of the qubits.

Figure 4



Bristlecone is Google’s newest quantum processor (left). On the right is a cartoon of the device: each “X” represents a qubit, with nearest neighbor connectivity.

Notice that this does *not* have the long-range qubit connectivity that is described in [13], but the grid does allow for effective communication with relatively small error rates. Another thing to note about the Bristlecone™ is that it uses superconducting circuits to create qubits [19]. This is in contrast to the methods discussed in the previous section where quantum information is stored in ions, entire atoms, and photons. In [19], the authors discuss that superconducting circuits can closely mimic behavior of particles at the quantum level, but they do so at a macroscopic scale. Further, superconducting circuit qubits allow quantum-like behavior to be created in a silicon chip. At this point, it is also worth mentioning that almost any approach to

designing qubits that involves magnetic fields is also likely combined with superconducting circuits. However, as stated in the introduction, the science behind superconductors are well beyond the scope of this paper.

IBM Q™

In the above section on the Google Bristlecone™, we talked mostly about a new processor Google designed in an attempt to minimize error correction. In this section, we will discuss the approach taken by designers of the IBM Q™. Interestingly, IBM was one of the first companies to release a commercial quantum computer. The first computer they released was a 20-qubit computer and have recently released a 50-qubit computer. Most of the details of the overall architecture of their IBM Q™ system is in the form of conference presentations and press releases; most notably, [18]. However, there are a few papers and patents which discuss the details of the qubits used in the IBM Q [20, 21, 22].

The IBM Q™ takes a similar approach as Google and creates their qubits with superconducting circuits. However, the IBM Q™ uses something called *transmon qubits* which are actually a hybrid with superconducting circuits and the QED cavity approach discussed in the previous section. The authors of the original paper, [39], describe the transmon qubit in the following way: “The transmon benefits from the fact that its charge dispersion decreases exponentially (…) while its loss in anharmonicity is described by a weak power law. As a result, we predict a drastic reduction in sensitivity to charge noise (…)”. In other words, the basic idea of the transmon qubit is to create a robust qubit for the QED cavity architectures. Further details of the transmon qubit and Figure 5 below are from [21]. Additionally, details on error correction mechanism of

transmon qubits are in [20] and one of IBM's patents on the transmon qubit design is [22].

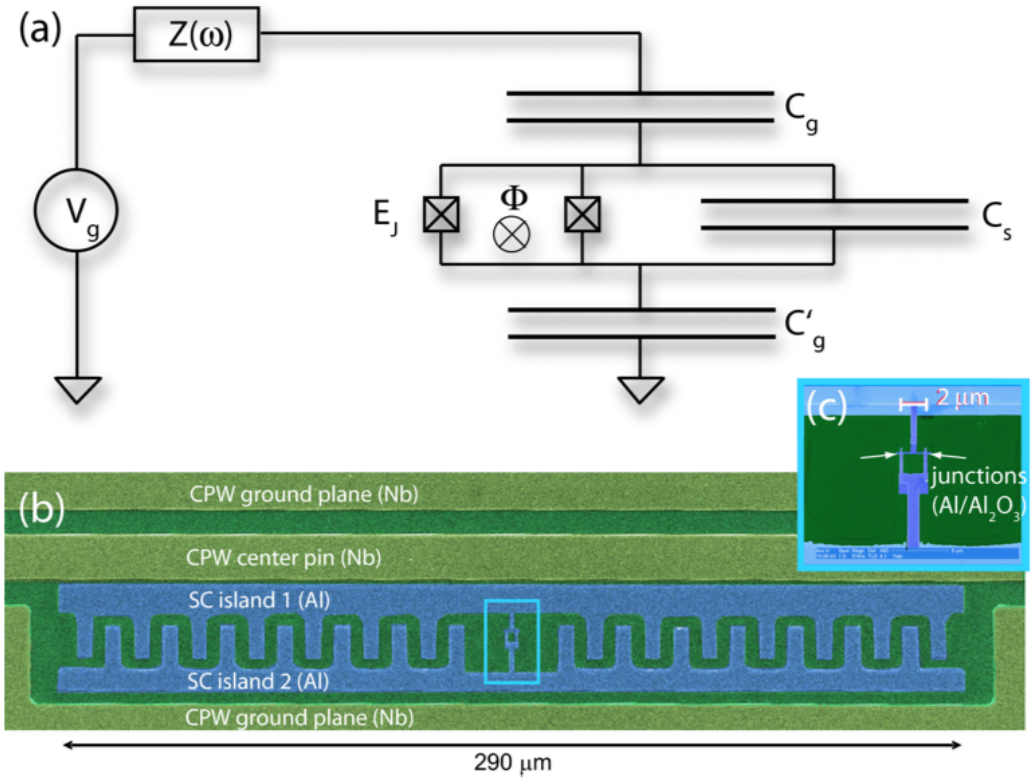


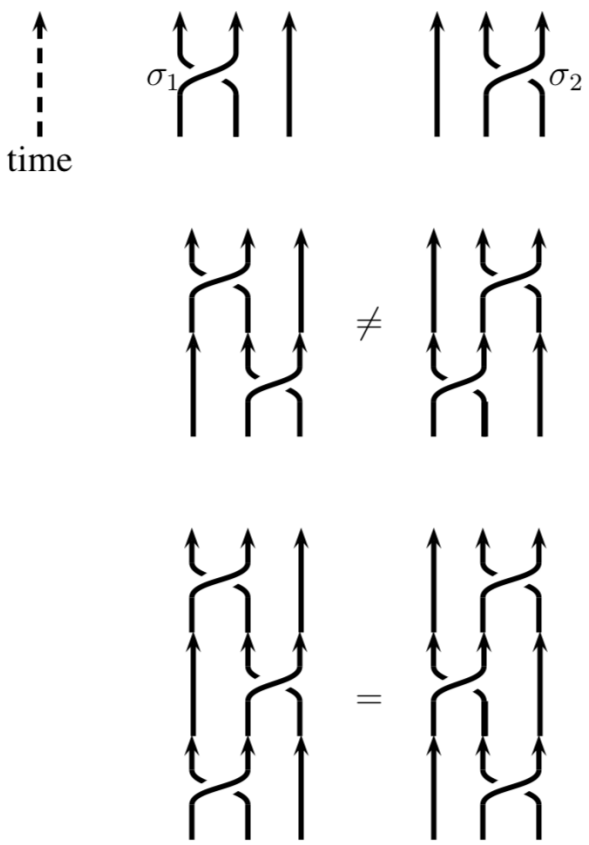
Figure 5 The transmon qubit. (a) The circuit of the transmon is identical to the circuit of a differential single Cooper pair box, consisting of two superconducting islands coupled by two Josephson junctions. The coupling to ground is purely capacitive. The use of two Josephson junctions allows for tuning of the effective Josephson energy via the external magnetic flux penetrating the superconducting loop. (b) and (c) show the optical and SEM image of a transmon device positioned inside a coplanar waveguide. While the size of the junctions and the superconducting loop is very similar to CPB devices, the inter-island capacitance C_s is increased drastically due to the large size of the islands and the interdigitated finger structure. This capacitance is matched by comparably large capacitances C_g and C'_g to the ground plane and centerpin of the transmission line resonator.

MICROSOFT QUANTUM COMPUTER

As with Google and IBM, Microsoft is working towards creating reliable quantum computers whose qubits can remain cohered for a longer period of time than what is currently possible on existing systems. The approach Microsoft is taking in designing quantum computers is fundamentally different from other systems and is rooted in a mostly theoretical branch of mathematics called topology and is aptly named topological quantum computing. The main idea of topological quantum computing is to create qubits with "quasiparticles" called a *non-abelian anyons,* which is extensively discussed in Microsoft's white paper, [23].

One major difference between this topological approach to quantum computing and the methods being pursued by Google and IBM is that the approach Microsoft is taking has yet to produce a viable prototype. However, topological quantum computing holds major promise due to its inherent stability in how qubits are encoded. When non-abelian anyons are used to encode quantum information, they are in a topological object called a *braid*. While sources of quantum errors might directly affect the quasiparticles (as with all previous quantum computational methods discussed), the information of the braid as a whole is actually preserved and is more robust to external noise. Another advantage of topological quantum computing that is described in [23] is the fact that encoding and measuring information in a braid requires very low energy. There are many details of topological quantum computing that are beyond the scope of this paper, but Figure 6 shows a simple illustration from [23] on how braid operations work and why they might be considered non-abelian. The reason these non-abelian braid objects are robust is because they can easily be corrected.

Figure 6



Top: The two elementary braid operations σ_1 and σ_2 on three particles. **Middle:** Here we show $\sigma_2\sigma_1 \neq \sigma_1\sigma_2$, hence the braid group is Non-Abelian. **Bottom:** The braid relation (Eq. 3) $\sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1}$.

If Microsoft can pull off practical topological quantum computing, they have the potential to dominate the market, assuming the method really is as robust and scalable as they have marketed on their website and in their white paper.

D-WAVE SYSTEMS 2000Q™

The previous three systems are all being designed in a way to create the most optimal qubits (i.e. qubits that are robust to coherence disruptors). On the other end of the spectrum, we have a company like D-Wave Systems (one of the original companies marketing of the power of quantum computers) that is focused primarily on designing computers with the largest number of “qubits”. I put quotes around the word qubits

because D-Wave computers have been very controversial and many experts claim that it is not actually a quantum computer. Instead, many say it should be considered a quantum annealer. Like Google and IBM, D-Wave uses superconductors to minimize resistance in their system. Another hallmark of all D-Wave systems is that the “qubits” encode information via magnetic fields [24]. One distinct disadvantage in D-Wave Systems computers is that their qubits are inherently extremely fragile. This is partly because of how qubits are interconnected, which is discussed more below.

In all of the previous sections of this paper, we have discussed entanglement and coherence between qubits, which has largely been a property that is exhibited by particles themselves. However, “qubits” in the D-Wave Systems quantum annealers are “cohered” in a different fashion. Instead of relying directly on the quantum behavior of particles, each qubit is connected via hardwiring with the other qubits in the system. This is one of the largest points of contention with D-Wave Systems and quantum computing experts since they don’t use “real” quantum coherence. This method of “coherence" is one of the reasons their qubits are so fragile [24, 25, 26]. However, the entanglement is achieved in a similar fashion as the nearest neighbor grid approach taken by Google in their Bristlecone™ processor.

D-Wave Systems has recently announced a computer called the 2000Q™ which is advertised as having 2048 qubits. As described in the *Nature* article, [25], there has been even more backlash than usual from scientists in the quantum computing field because the system had to sacrifice interconnectivity to be able to increase the number of qubits as much as they did. The reason the 2000Q™ is not as interconnected as previous generations of their quantum annealer is because the

number of connections increases exponentially with the number of qubits. Figure 7 shows how a 4-qubit grid would have to be fully interconnected.

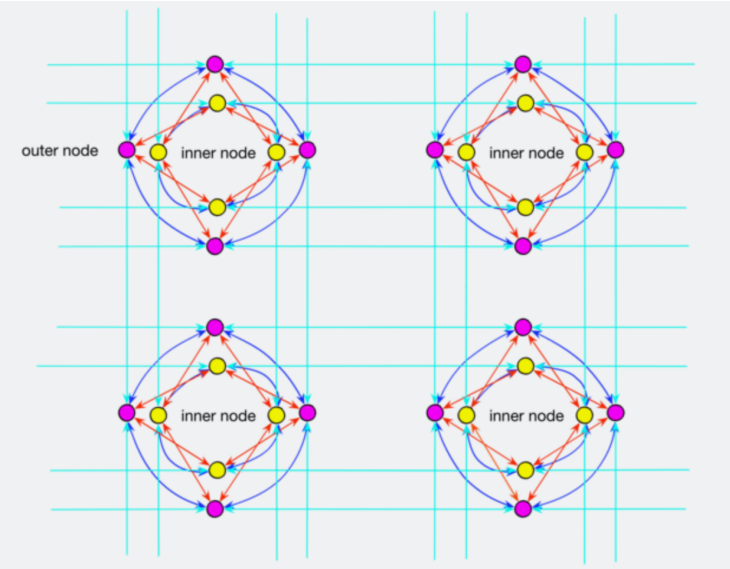


Figure 7: This diagram from [25] illustrates how the interconnectivity is designed in a grid of 4 qubits. At a scale of 2048 qubits, this becomes too hard to handle and some pruning of connectivity was necessary.

However, it is worth noting that D-Wave Systems has not created an entirely worthless pseudo-quantum computer. Their quantum annealers are still capable of solving some optimization problems *much* quicker than classical computers. One example that is (unsurprisingly) much faster to solve on the D-Wave 2000Q™ than in classical systems is the simulated annealing algorithm. Simulated annealing is a combinatorial optimization problem that can find good solutions in a very large search space with heuristics and other simplifications. It is often times described as searching for the lowest point in a landscape with many peaks and valleys. Of course, it is not "simulated" on the 2000Q™, but is instead called quantum annealing. The advantage quantum annealing has over simulated annealing is that the algorithm can essentially "look through" hills to see if there is a lower valley on the other side. This method leads to a dramatic improvement to this sort of combinatorial annealing problem. D-Wave Systems wrote a paper (though it is not yet peer-reviewed), [27], which claims that they can solve equivalent annealing problems on the 2000Q™ up to *100 million times faster*

than on classical counterparts. So, while there is certainly limited quantum functionality in D-Wave systems, they have designed systems that are extremely efficient at solving a narrow band of combinatorial optimization problems which could prove practically useful well before “real” quantum computers see the light of day.

SECTION WRAP-UP

There are many more quantum architectures being designed in laboratories around the world that have not been discussed in this section. The reason for mentioning the four that I did is because they are each quite different from one another and to illustrate how diverse the approaches to solving the problem of practical quantum computing are. It is interesting to see that there is no agreed upon method for designing quantum computers in a way that leads to reliable and scalable coherence between qubits. In a way, I draw parallels between the current state of quantum computing and what I know of the early stages of classical computers. Maybe since designing the hardware of the systems being pursued by Google and IBM is easier (as they are silicon based), it is a more short-term solution given the current state of materials science and manufacturing. So, these architectures might be compared to CISC architectures as classical counterparts. On the other hand, topological quantum computing being pursued by Microsoft is not quite feasible given the current state of technology, but in the future it might come out on top as the dominant method of quantum computing (as was the case with RISC architectures).

Algorithms and Applications

While the actual implementation of quantum computing continues to remain in its infancy, quantum algorithms have been under development for decades now. In this

section we will cover some of the most prominent algorithms as well as give insight into current research and practical applications of quantum algorithms. There will be five subsections in this section. The first will cover the first quantum algorithm, Deutsch's Algorithm and its generalization, the Deutsch-Jazsa Algorithm. The second subsection will cover Shor's Algorithm and its application to factoring integers (plus the implications of such an algorithm). The third subsection is about Grover's quantum search algorithm and applications. Fourth, we will cover quantum walks as well as how they can be combined with Grover's algorithm. Finally, we will cover some current research that is combining many of the concepts of previous algorithms (and many more) to extend the fields of machine learning, chemical modeling, and cryptography. Note that the first four subsections are inspired by [7], but this paper will not dive into the formal proofs covered in the book. Instead this section will focus on the high-level concepts while drawing analogies to classical algorithms wherever possible.

DEUTSCH'S ALGORITHM AND THE DEUTSCH-JOZSA ALGORITHM

The first quantum algorithm ever conceived was by David Deutsch. The main idea of this algorithm is to determine if the Boolean function $f : \{0,1\} \implies \{0,1\}$ is a *constant* or *balanced* function in one pass as opposed to the classical algorithm that requires two steps. A function is constant if it has the same output for all input values and conversely, a balanced function does not exhibit such behavior. This is a very simplistic algorithm since the function f is so small. However, in the early 1990s, David Deutsch and Richard Jozsa collaborated to generalize the algorithm to be able to handle a function that is exponentially larger in the following form: $f : \{0,1\}^n \implies \{0,1\}$. The interesting result of the Deutsch-Jozsa Algorithm is that it

can determine whether the Boolean function is constant or balanced in one pass for *any* value of n. This was the first quantum algorithm to prove that there is an exponential speedup over its classical counterpart [7].

SHOR'S ALGORITHM

Shor's Algorithm is one of the most popular quantum algorithms. In chapters 11 and 12 of [7], Shor's Algorithm is described first in terms of the core quantum component of the algorithm and then the full algorithm is described in terms of integer factorization. It is also stated that Shor's Algorithm has classical and quantum components. The core quantum component of the algorithm is something called a *quantum Fourier transform* (QFT). In the most basic sense, the QFT is a method for detecting periods of a function (or, in the case of integer factorization, a string of bits). Once the periods are detected, the qubits are readout and fed back into a classical computer to do the rest (which is largely based off of a known method called the *Euclidian Algorithm* [28]). The quantum circuit of the quantum component of the alorithm is shown in Figure 8:

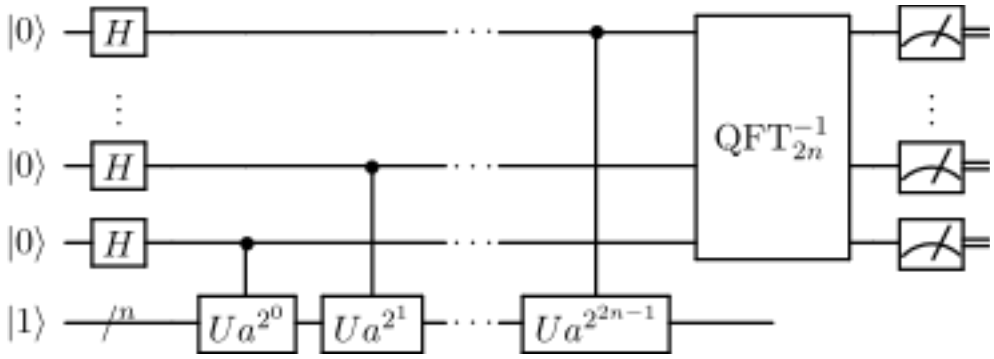


Figure 8: The quantum circuit of the quantum component of Shor's algorithm. Recall familiar features such as the *Hadamard* gate *H*. Graphic taken from [29]

There are two important concepts I would like to highlight from Shor's Algorithm. First, the QFT is prevalent in many popular algorithms and will likely continue to be one of the key features in designing useful quantum algorithms. The second is that quantum computers and classical computers can work together to solve problems that neither one could solve alone. In the case of Shor's Algorithm, the runtime complexity is roughly $O(n^3)$ whereas there has not been a classical algorithm developed that can solve the problem in polynomial time. Thus, the problem is in class NP [7§12]. This is another case where the speedup is expected to be exponential. The reason Shor's Algorithm is so practically important is because many modern-day encryption protocols assume that integer factorization is not feasible for computers to solve. If a quantum computer is designed which can factor integers with $O(n^3)$ complexity, the security problem would be seemingly insurmountable. The only solution to such a problem would be to use quantum encryption, which are discussed in [29], [30], and [31] but are beyond the scope of this paper.

GROVER'S ALGORITHM

Grover's Algorithm is an interesting problem that is about quantum search. The basic idea is described in [7] as finding a needle in a haystack. In classical algorithms, search is $O(n)$ in the worst case, which is certainly not optimal when the search space is enormous (as is the case in many enterprise database systems). Grover's Algorithm is a search algorithm that has a runtime complexity of $O(n^{1/2})$, which is a major improvement to classical systems. Also note that this search algorithm assumes an unsorted data structure; not a tree-based data structure.

Unlike Shor's Algorithm, Grover's Algorithm is purely quantum, but the use cases would extend beyond the quantum realm. The original idea outlined in Grover's original paper, [32], is to use the algorithm for increasing search speeds in databases. There are, however, still many steps that would need to be taken for Grover's Algorithm to be implemented in real-world systems (even if practical quantum computers existed). An interesting thing about Grover's Algorithm is that it does *not* promise an exponential speedup as many of the other popular algorithms. Instead, it offers a *very good* improvement to a problem that is used frequently by anyone that owns a computer.

QUANTUM WALKS

This subsection is based off chapters 14 and 15 of [7]. The classical analog to quantum walks is something called *random walks* which take place on graphs. Often times, these random walks are expressed through matrices that represent graphs. The basic idea of classical random walks is to determine the answers to three basic questions: 1) what is the expected number of steps to reach a starting node s to an ending node t, 2) what is the expected number of steps that need to be taken to reach every node in the graph, and 3) if stopped after n steps, what is the probability of ending on node t.

Given what we know about superposition, it might be almost intuitive that with the right number of qubits, we can represent a superposition of all the nodes on the graph. Then, the state of each qubit is evolved via quantum gates and eventually ends up at a solution. Of course, the type of gates used in evolving states of qubits will differ

based on the problem at hand. One such problem is quantum graph search, which has common classical counterparts such as depth-first search and breadth-first search. The method for solving these graph search problems is to use Grover's Algorithm *and* quantum walk algorithms together. This points to an interesting fact that quantum algorithms are not so stringent that they must be standalone.

FURTHER ADVANCEMENTS IN APPLICATIONS OF QUANTUM COMPUTING

Some of the most exciting advancements in recent memory (in my opinion) have been either related to machine learning or quantum computing. As we will see in this section, these advancements are not necessarily independent. Figure 9 below shows how the fields of quantum information processing and machine learning overlap to create a field called quantum machine learning. The figure is taken from [33].

Another one of the exciting applications of quantum computing has been previously mentioned. In the section about Google's Bristlecone™ processor, we discussed how their processors have been based on papers focused on quantum processors that can best model actual quantum systems (like chemical reactions in the body). There have been many researchers that are working on quantum algorithms in this field because the implications related to drug discovery and human health are almost endless. A couple of papers that develop quantum algorithms for modeling chemical systems are [34] and [35].

Finally, it is worth mentioning again that quantum cryptography is one of the most pressing applications of quantum computers. Quantum machine learning and chemical modeling are very nice to have, but if the quantum decryption is used by a

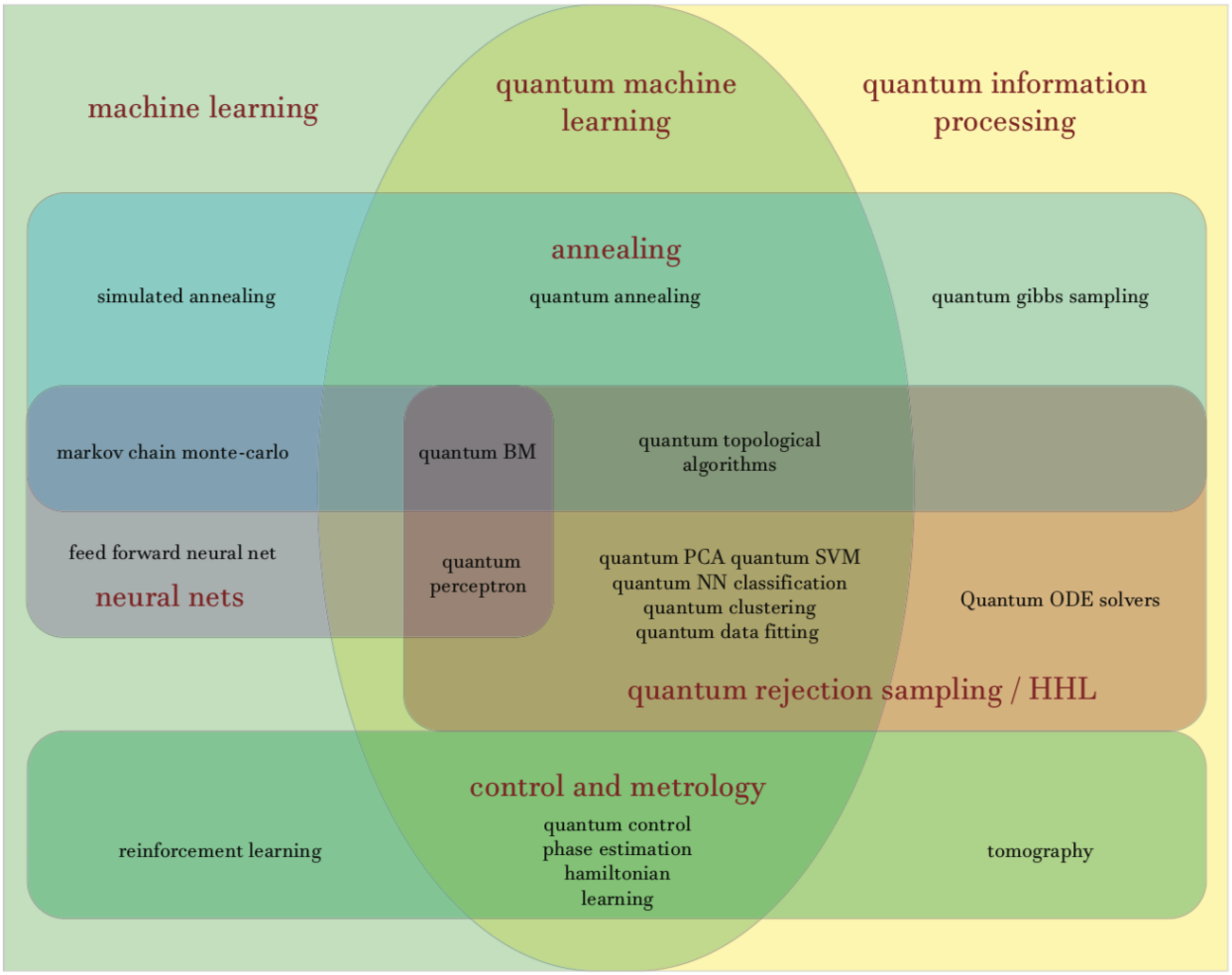


Figure 9: This chart shows machine learning algorithms and their equivalent quantum machine learning counterparts. There is obviously much relation to quantum information processing, but my understanding of the field is vastly limited. However, the algorithms described in that section are direct applications of quantum computing systems.

malevolent party before quantum encryption is implemented, there are many problems that are likely to arise. Fortunately, there are many researchers that have been focused on solving the problem for years. One of the most prominent papers related to quantum cryptography is [36], and [37] offers a nice treatise on the subject.

Summary and Closing Thoughts

Throughout this paper, we have built a basic understanding of quantum computers and computation by first discussing the concept of using qubits instead of bits as a way of processing quantum information. We also briefly touched on the ideas superposition, entanglement, and coherence in the introductory section. The purpose of the first section was to build some of the intuition behind the very unintuitive nature of quantum computing.

In the second section, we moved to the ideas of quantum gates and quantum circuits. We learned that quantum gates are much more flexible than classical gates. One of the simple examples given was the $\sqrt{NOT}$ gate. Though, the Hadamard gate (which basically says that there is a 50/50 chance that a qubit is either 0 or 1) is one of the most important single qubit gates since these gates can be chained together to increase the state space exponentially over a chain of classical bits. We also learned that it is easier to achieve sustainable, universal computation in quantum gates because reversible and universal gates can be created with 2-qubit gates as opposed to the 3-bit gates required by classical computers. This is important because quantum computers have lower thermodynamic dissipation and less (relative) complexity of circuit design. The last part of the section briefly touched on quantum circuits, but is deferring some of the discussion to the tutorial followed in the Appendix.

The third section gave an overview of quantum hardware in general. While the previous two sections had covered very theoretical concepts, the third section noted that in order for quantum computers to be practical, the hardware must account for the

seemingly endless sources of noise that could break quantum coherence (e.g. heat, photons, stray molecules, etc.). The method for accounting for this was developed by none other than Peter Shor and it is called quantum error correction. This section then went on to discuss the needs of a practical quantum computer and three of the classic methods for creating qubits. Lastly, we covered a newer method proposed that can theoretically allow qubits to remain entangled at larger distances.

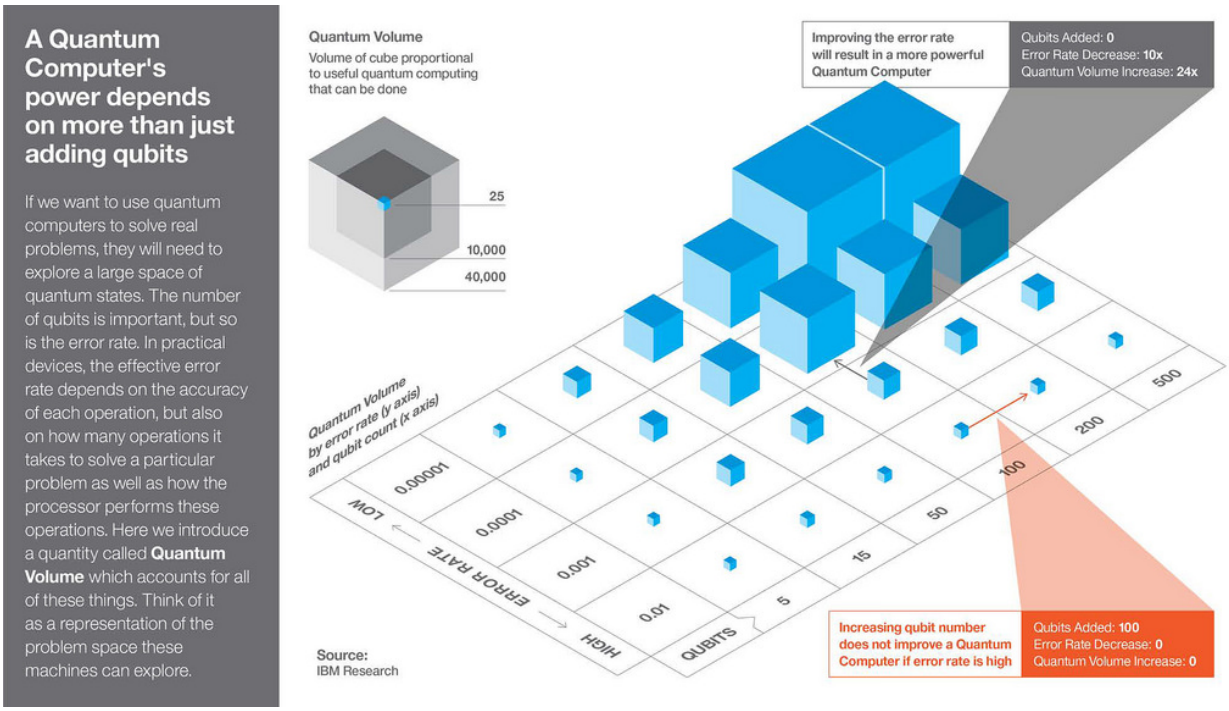
In the fourth section, we moved from general architectures to those that are actually being developed in practice by companies such as Google, IBM, Microsoft, and D-Wave Systems. Google's new processor called Bristlecone™ is a method that builds from a design of a linear array of qubits to a 2D array of qubits. The method they employed was an attempt to create a processor on a silicon-based chip that is robust to quantum errors. The IBM Q™ is a 50-qubit system where the qubits are designed as a hybrid between superconducting circuits and one of the classic three qubit method described in the third section called the cavity QED. Next, we covered Microsoft's more theoretical approach to quantum computing with something called topological quantum computing which is completely different from the more common approaches. While they have no (public) prototype of a topological quantum computer, their motivation behind such a method is that the qubits are inherently more stable because they are stored in a robust topological object called a braid (though, the quasiparticle that might exhibit such a behavior is not known for sure to exist). Lastly, we covered the controversial computers created by D-Wave Systems. There are a few things keeping these computers from being definitely classified as quantum computers, namely, their method of "coherence". Instead, D-Wave computers are widely

considered quantum annealers. While there is some controversy, these systems have proven to be the first practical computer that can solve a narrow class of quantum algorithms in the realm of combinatorial optimization.

The fifth and final section covered some of the concepts and applications behind a few of the most popular quantum algorithms today. The first was Deutsch's Algorithm and its generalization of the Deutsch-Jozsa Algorithm which was the first algorithm to explicitly show an exponential speedup over its classical counterpart. The second algorithm was the famous Shor's Algorithm for factoring integers. Shor's Algorithm has major implications in the field of cryptography. Next, we covered Grover's Algorithm which showed a polynomial speedup over classical (unsorted) search algorithms. We then moved to quantum walks and how Grover's Algorithm could be applied to quantum walks to improve upon classical graph search algorithms like depth-first search. The last part of this section overviews a few of the fields that might be the most impacted (at least in the near-term). These fields were machine learning, chemical systems modeling, and, unsurprisingly, cryptography.

So, where is quantum computing going? How long before we actually have practical quantum computers? And, more importantly, when will they start to outperform classical supercomputers (a term called quantum supremacy)? The short answer is: we have no idea, but this doesn't lead to an interesting discussion. As it was briefly mentioned earlier, some popular science publishers have purported 50-qubits as being the magic number of qubits. In a quantum computer that is completely immune to error, 50-qubits would, indeed, begin to outperform the fastest supercomputers on some tasks. However, in doing research, I found IBM to have a balanced view. Instead

of thinking purely about the number of qubits, they think about practicality along two dimensions: the number of qubits and the error rate (since the qubits required to correct errors are not being used in the meat of the actual computations). The graphic below was presented in [18]. IBM coined the term *quantum volume* because it accounts for both of these dimensions. As we can see, neither the number of qubits nor the error rate is effective alone. Instead, a combined approach to achieving the maximum number of qubits with the minimum error rate will lead to quantum supremacy. It is yet to be seen which of the many approaches to quantum computers will lead here, but it should be clear that there are major strides being made on the front of quantum computing.



References

[1] Dumas, Joseph D. II. Computer Architecture: Fundamentals and Principles of Computer Design, Second Edition, CRC/Taylor & Francis, 2017. ISBN 978-1-4987-7271-6.

[2] Encyclopedia of Parallel Computing. (2011). Boston, MA: Springer US. doi: 10.1007/978-0-387-09766-4

[3] Moore's law.(Definition). (2008, January 1).The Columbia Encyclopedia, 6th ed., 2008 Online Supplement. The Columbia University Press.

[4] Kumar, S. (2015). Fundamental Limits to Moore's Law.

[5] Perry, R. (2012). Quantum computing from the ground up. Singapore: World Scientific.

[6] Deutsch, D. (1985). Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences, 400(1818), 97–117. doi:10.1098/rspa.1985.0070

[7] Lipton, R. (2014). Quantum algorithms via linear algebra : a primer. Cambridge, Massachusetts: The MIT Press.

[8] Inkoom, P. (2011, January 1). A connection of quantum computation and DNA computation using the Bloch sphere. ProQuest Dissertations Publishing. Retrieved from http://search.proquest.com/docview/881636220/

[9] Pötz, W. (2006). Quantum Coherence From Quarks to Solids. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/11398448

[10] Williams, C. (2011). Explorations in Quantum Computing (2nd ed.). London: Springer London. doi:10.1007/978-1-84628-887-6

[11] Preskill, J. (1998). Lecture notes for physics 229: Quantum information and computation. *California Institute of Technology*, *16*.

[12] Braden, R. T., Borman, D. A., & Partridge, C. (1988). *Computing the internet checksum* (No. RFC 1071).

[13] Tosi, G., Mohiyaddin, F. A., Schmitt, V., Tenberg, S., Rahman, R., Klimeck, G., & Morello, A. (2017). Silicon quantum processor with robust long-distance qubit couplings. *Nature communications*, *8*(1), 450.

[14] Google (2018, March 5). *A Preview of Bristlecone, Google's New Quantum Processor*. *Research Blog*, research.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html.

[15] Google Unveils Advanced 72-Qubit Processor. (2018, March 8). Rockaway: Advantage Business Media. Retrieved from http://search.proquest.com/docview/2012237725/

[16] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, et al. (2015). State preservation by repetitive error detection in a superconducting quantum circuit. Nature, 519(7541), 66. doi:10.1038/nature14270

[17] R. Barends, A. Shabani, L. Lamata, J. Kelly, A. Mezzacapo, U. Las Heras, R. Babbush, et al. (2016). Digitized adiabatic quantum computing with a superconducting circuit. Nature, 534(7606), 222. doi:10.1038/nature17658

[18] IBM Research (2017, December 08). *Experimental quantum computing at IBM*, https://www.youtube.com/watch?v=T-8uuq7Izl8

[19] You, J. Q., & Nori, F. (2006). Superconducting circuits and quantum information. *arXiv preprint quant-ph/0601121*.

[20] Houck, A. A., Schreier, J. A., Johnson, B. R., Chow, J. M., Koch, J., Gambetta, J. M., ... & Schoelkopf, R. J. (2008). Controlling the spontaneous emission of a superconducting transmon qubit. *Physical review letters*, *101*(8), 080502.

[21] Houck, A. A., Koch, J., Devoret, M. H., Girvin, S. M., & Schoelkopf, R. J. (2009). Life after charge noise: recent results with transmon qubits. *Quantum Information Processing*, *8*(2-3), 105-115.

[22] Gambetta, J. M., Ketchen, M. B., Rigetti, C. T., & Steffen, M. (2014). *U.S. Patent No. 8,642,998*. Washington, DC: U.S. Patent and Trademark Office.

[23] Nayak, C., Simon, S. H., Stern, A., Freedman, M., & Sarma, S. D. (2008). Non-Abelian anyons and topological quantum computation. *Reviews of Modern Physics*, *80*(3), 1083.

[24] Irigoyen, I. M., Manuel, L. L., & Villanueva, E. S. D-Wave quantum computer.

[25] Lee, Chris. (2017, January 5) Ex*plaining the Upside and Downside of D-Wave's New Quantum Computer. Ars Technica*, arstechnica.com/science/2017/01/explaining-the-upside-and-downside-of-d-waves-new-quantum-computer/.

[26] Boixo, S., Rønnow, T. F., Isakov, S. V., Wang, Z., Wecker, D., Lidar, D. A., ... & Troyer, M. (2014). Evidence for quantum annealing with more than one hundred qubits. *Nature Physics*, *10*(3), 218.

[27] King, J., Yarkoni, S., Raymond, J., Ozfidan, I., King, A. D., Nevisi, M. M., ... & McGeoch, C. C. (2017). Quantum annealing amid local ruggedness and global frustration. *arXiv preprint arXiv:1701.04579*.

[28] Motzkin, T. (1949). The euclidean algorithm. *Bulletin of the American Mathematical Society*, *55*(12), 1142-1146.

[29] Boykin, P. O., & Roychowdhury, V. (2003). Optimal encryption of quantum bits. *Physical review A*, *67*(4), 042317.

[30] Xi-Han, L., Chun-Yan, L., Fu-Guo, D., Ping, Z., Yu-Jie, L., & Hong-Yu, Z. (2007). Quantum secure direct communication with quantum encryption based on pure entangled states. *Chinese Physics*, *16*(8), 2149.

[31] Zhang, Y. S., Li, C. F., & Guo, G. C. (2001). Quantum key distribution via quantum encryption. *Physical Review A*, *64*(2), 024302.

[32] Grover, L. K. (1996, July). A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (pp. 212-219). ACM.

[33] Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum machine learning. *Nature*, *549*(7671), 195.

[34] Aspuru-Guzik, A., Dutoi, A. D., Love, P. J., & Head-Gordon, M. (2005). Simulated quantum computation of molecular energies. *Science*, *309*(5741), 1704-1707.

[35] Kassal, I., Jordan, S. P., Love, P. J., Mohseni, M., & Aspuru-Guzik, A. (2008). Polynomial-time quantum algorithm for the simulation of chemical dynamics. *Proceedings of the National Academy of Sciences*, *105*(48), 18681-18686.

[36] Ekert, A. K. (1991). Quantum cryptography based on Bell's theorem. *Physical review letters*, *67*(6), 661.

[37] Gisin, N., Ribordy, G., Tittel, W., & Zbinden, H. (2002). Quantum cryptography. *Reviews of modern physics*, *74*(1), 145.

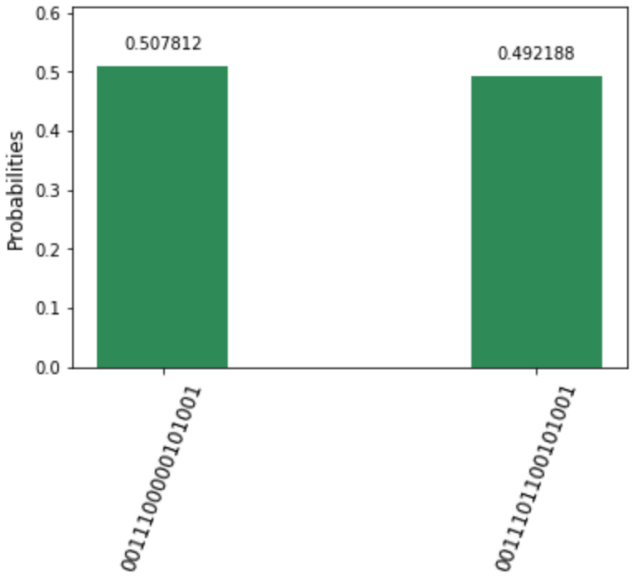
[38] Castelvecchi, D. (2017). IBM's quantum cloud computer goes commercial. *Nature News*, *543*(7644), 159.

[39] Koch, J., Terri, M. Y., Gambetta, J., Houck, A. A., Schuster, D. I., Majer, J., ... & Schoelkopf, R. J. (2007). Charge-insensitive qubit design derived from the Cooper pair box. *Physical Review A*, *76*(4), 042319.

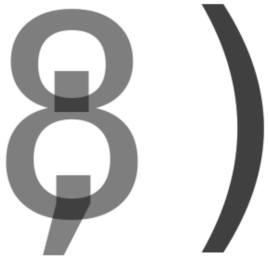
Appendix

The IBM Quantum Experience™ [18, 38] was designed to let the public begin designing quantum circuits and algorithms. The purpose of this appendix is to show a few examples of their Python SDK called QISKit [https://github.com/QISKit/qiskit-tutorial]. One of the most enticing things about their API is that anyone can run their algorithms on a real quantum computer. Of course, simulation is also possible, but the simulations do have any quantum errors. The first example I would like to highlight is their “Quantum Hello World” program. Example code and the probability distribution of the final state of the bits are shown below:

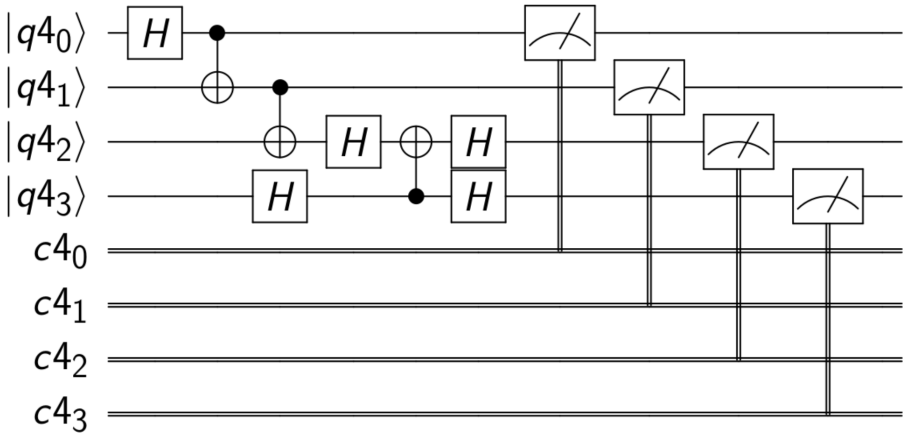
```
from qiskit import QuantumProgram
from qiskit.tools.visualization import plot_histogram

# set up registers and program
qp = QuantumProgram()
qr = qp.create_quantum_register('qr', 16)
cr = qp.create_classical_register('cr', 16)
qc = qp.create_circuit('smiley_writer', [qr], [cr])

# rightmost eight (qu)bits have ')' = 00101001
qc.x(qr[0])
qc.x(qr[3])
qc.x(qr[5])

# second eight (qu)bits have superposition of
# '8' = 00111000
# ';' = 00111011
# these differ only on the rightmost two bits
qc.h(qr[9]) # create superposition on 9
qc.cx(qr[9],qr[8]) # spread it to 8 with a CNOT
qc.x(qr[11])
qc.x(qr[12])
qc.x(qr[13])

# measure
for j in range(16):
    qc.measure(qr[j], cr[j])
```



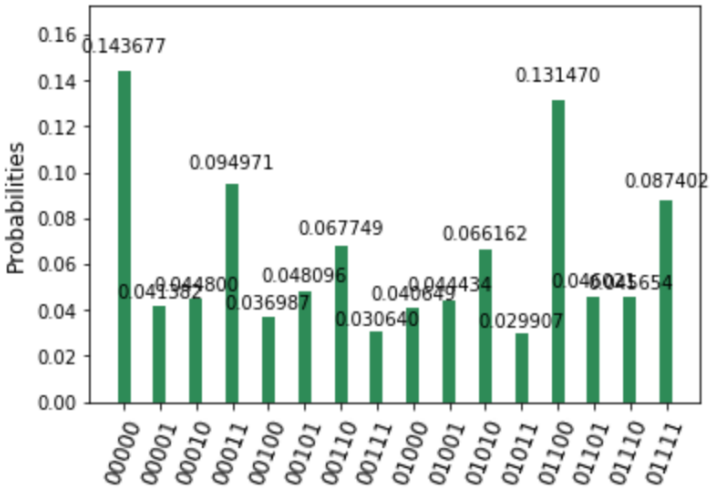
Which leads to the visualization:



Of course, there is not much of a practical application of this “Hello World” program, but it is a nice introduction into creating quantum circuits. Another nice thing about QISKit is that there is also a method for visualizing the circuits created. Let us take a look at the following circuit (which is designed in a similar fashion as above):



After running this circuit on the actual 50-bit IMB Q™, the probability distributions of final states are shown below:



The QISKit gives anyone the ability to start learning how to program quantum computers and designing quantum circuits. This effort by IBM could be just the catalyst that is needed to move to a phase where practical quantum computers are a reality.