Quantum-Computation and Applications

Bhupesh Bishnoi[†]

Abstract

In this research notebook on quantum computation and applications for quantum engineers, researchers, and scientists, we will discuss and summarized the core principles and practical application areas of quantum computation. We first discuss the historical prospect from which quantum computing emerged from the early days of computing before the dominance of modern microprocessors. And the re-emergence of that quest with the sunset of Moore's law in the current decade. The mapping of computation onto the behavior of physical systems is a historical challenge vividly illustrate by considering how quantum bits may be realized with a wide variety of physical systems, spanning from atoms to photons, using semiconductors and superconductors. The computing algorithms also change with the underline variety of physical systems and the possibility of encoding the information in the quantum systems compared to the ordinary classical computers because of these new abilities afforded by quantum systems. We will also consider the emerging engineering, science, technology, business, and social implications of these advancements. We will describe a substantial difference between quantum and classical computation paradigm. After we will discuss and understand engineering challenges currently faced by developers of the real quantum computation system. We will evaluate the essential technology required for quantum computers to be able to function correctly. Later on, discuss the potential business application, which can be touch by these new computation capabilities. We utilize the IBM Quantum Experience to run the real-world problem, although on a small scale.

1 The era to Quantum Computation and Information

First and foremost, quantum computation and information have been a scientific curiosity implemented in the academic laboratory and a few governments and industrial laboratories around the world [1–13]. It started as a discipline in physics. But over the last five years to ten years it is transitioning from scientific curiosity to technical reality [14–16]. In that transition, many problems need to be addressed. Many of them are falling on the engineering side. So, going forward, people who come from traditional engineering disciplines will have significant roles to play. There is a new term that is being coined called "quantum engineering." Moreover, it is bridging quantum science and the traditional engineering disciplines. Both will have to pivot to meet the needs of building quantum testbeds, which will lead to future quantum systems. In terms of background, many people today, whether trained as quantum physicists or trained as quantum engineers, are discussing a lot about the other discipline [17]. The hope is that as we move forward that we begin to define the engineering aspects of quantum information, that we begin to abstract it in the way that engineers abstract concepts from physics to make it an engineering and systems problem, and as this happens in the current decade, people will be able to contribute more without knowing the underlying, in-depth quantum physics. So, the motivation

[†]**bishnoi[At]ieee[Dot]org**

is to begin bridge quantum science and quantum engineering. In this sense, quantum computers are becoming available commercially for larger people outside of the academic laboratory. Now, it becomes more the domain of computer sciences rather than physicists and mathematicians. Traditionally physicists and mathematicians have been at the forefront of this field and pushing it. Now the goal is to bring computer scientists and electrical engineers up to the same level as we go forwards to tackle systems engineering challenges. With the advent of access to these computers will certainly bring in more computer scientists, software programmers, and electrical engineers in the domain. However, it does not mean that physicists and mathematicians stop doing work in quantum science and theory. so, the physicists and mathematicians will continue to be actively involved in this new computational development. Also, related to cost-effective ways to build a quantum computing experience, in-house familiarity, and expertise, there are examples of online quantum computers available to open source to use and operate. These include the IBM Quantum Experience [18]. Rigetti has an online quantum computer [19]. D-Wave has an online quantum computer. Google is also preparing an online quantum computer [20]. The impact and advantage of these open-source availability are many discussions started about the algorithms that might run on these existing systems and have an impact on the society with the current state of the art quantum hardware capability. We also get to interface with faculty and graduate students at the university and institutions. Access to talent is a huge part of the development. if we have some good ideas to explore and would be the solution, then we can engage directly with places like IBM, Rigetti, D-Wave, and Google to implement on their systems. So, there is a natural way via academia to ramp into the quantum information field. It starts with education, and we hope those are some useful ideas and a beautiful way to go ahead. With the current state of art quantum computation technological capability, it looks like there is no big relation between we can calculate today and the future quantum computational potential. So, should we need to study and worried about quantum computing today? There is some area which will affect today's world, one is information security, and it is important to everyone. we want to keep information secure not just today but for the next few decades. So, we write down and share the information, e.g., business strategy, trade-secret, government communique, personal information with our partner, but do not want to reveal everyone else and get hacked. So, we encrypt the information and share on electronic channel worldwide. currently, state of the art encryption scheme is strong, and we do not have a quantum computational capability yet, so decryption of information is not possible for others, this is true, but in future, we will have the quantum computational capability and with that, today shared and communicated information can be decrypted in the future. So, how long do we want today's information to be secure in the future. We need to understand the implications of quantum computing and now realize our information encryption vulnerability. So, that is one important aspect. Another important aspect is that most economic problems can be boiled down to optimization problems. We need to optimize that if it is doing routing and layout of an electrical circuit in a complicated chip. It could be as simple as home deliveries. we want to pick the shortest path possible. There are many examples of optimization problems. we will be able to identify certain problems within our organization, which might benefit from quantum computing, which is manifestly not related to factoring but more closely related to optimization [21] or quantum simulation or quantum emulation. These are problems that we want to think about and explore the solution because, in the future, with quantum computers, it will have an impact on our life.

Quantum computing is not going to replace classical computing. We will always need a classical computer to run beside it and control it. At least as we understand it today, quantum computers solve specific problems that we know of today much more efficiently than classical computers. However, those problems are not necessarily the types that we would want to run on our classical computer. It may be better said that a quantum computer can run any algorithm

that we can run on our classical computer. However, it may not do so, any better than we can currently do with our classical computer. So, for the foreseeable future, we believe that quantum computers will, for example, remain in the cloud, or we could think of them as these large mainframe-type computers. we use it for problems, like quantum simulation. in the nearer term, particularly before we can make fully error-corrected, fully error-resistant quantum computers, we will be using qubits that are faulty. We, therefore, cannot, by themselves, do a very, very long computation. so, it is likely that we will first see quantum computers that are viewed as accelerators for a core processor with a classical computer [22, 23]. The classical computer is running and orchestrating an overall algorithm. Nevertheless, it pings or pulls the quantum node periodically to get an answer to a sub-problem of a small part of the problem, takes that answer back, and then incorporates it into a larger algorithm that is being run. So, indeed, It is true that quantum computers, very likely, will be working alongside classical computers, at least for the foreseeable future. On classical digital computers, digital words and gates are applied several billion times a second. what is the analogous fundamental operation on a quantum computer? One starts with qubits. Then how are operations commanded? In classical computers, arbitrary Boolean logic can be performed with a universal set of one-bit and two-bit gates. There is not a unique arrangement of single and two-bit gates that will give us a universality. However, with just a handful of gates or even one gate, one can perform universal Boolean logic. then in computers, there is a clock that runs at gigahertz rates and just clocks the application of these gates throughout some algorithm or computation. As in a quantum computer, it is a bit different, and there are also some similarities. let us start with the similarities. First, quantum computers also have the concept of universality and the fact that there is a combination of single-qubit and two-qubit gates, which can reach, as we say, any point in the Hilbert space. We can take any quantum state and turn it into any other quantum state spanned by the qubits that we have. so, that is the concept of universality. Again, it is a handful of gates, as we have seen, not unique. However, with one and two-qubit gates, we can perform arbitrary quantum logic. Now, how it works, is that we will take a massive quantum superposition state of N qubits. That gets fed into a computer, or that is the starting point of the computer. Then through single-qubit and two-qubit gates, according to a prescription that the algorithm designer decides we implement and set up quantum interference to occur in such a way that by the end of the computation, ideally, all of the probability amplitude resides in one of these states that is, in fact, the answer to our problem. It encodes the answer to the problem so that when we measure with a very high probability or even unit probability, the system collapses onto that single state. The probability that we measure a given state goes as the magnitude squared of the coefficient in front. That is why it must approach unit value. When we make that measurement with high probability, we will get the right answer. so, that is how these two computers work. Now, in a classical computer, running faster and faster is always desirable. It is also true that we do want to have a fast gate speed in a quantum computer. For example, if one type of quantum computer can run at 100 megahertz and another type of quantum computer can do the same calculation in the same way. However, it only runs at 100 kilohertz, then, the one that is running at 100 megahertz will run 1,000 times faster. we can celebrate that a quantum computer can run or can solve problems exponentially faster. so, something that might take the age of the universe now it only takes a day on the 100-megahertz quantum computer. Nevertheless, on the 100-kilohertz quantum computer, it would take 1,000 days. In comparison to it found the age of the universe, it is a fantastic achievement. However, 1,000 days that is around three years. so, from a human time frame timescale, that is not as useful as something that runs in a day. So, there are two aspects. There is the exponential improvement or the strong polynomial improvement, the quantum enhancement, that happens [24]. However, on top of that, it is still important that we do not forget the prefactor, that we run quickly, and we have a faster clock

and gating. The next challenge is addressing large-scale quantum computing from sub-modules. so, would it be possible to use quantum networking or other methods [25, 26]? Like a cluster computer does, to simulate a system of 1,000 qubits, but by only using 20 such sub-units with every 50 qubits, and this is the concept of modularity. It is an active area of research. It is expected that we will add something that looks like modularity in any large qubits system that we build. In classical computer systems, we always see some degree of modularity in building those systems because they are complicated. As in the early stages of quantum computing, we are working with qubits that are individually faulty. There is a whole research line on trying to address that through fault tolerance and error correction [27]. However, that is going to take some time. In the meantime, we are looking at these noisy systems, and this is the concept of noisy, intermediate-scale quantum computing, which is quantum computing using the qubits we have today or in the foreseeable future, which are quite good, but not good enough to run a computation from start to finish in its entirety. so, these NISQ [28], quantum computers very likely will be clusters of, 50 to 100-qubits and very likely will be a co-processor that runs alongside a classical computer where the classical computer is going to pull these quantum subsystems or ask a question to this sub-module many times and get answers back and aggregate these answers in order to address and solve an overall problem. That is certainly one way we can imagine modularity being incorporated into a quantum computer. There are other examples of this in trapped ions, superconducting qubits, for example, to take the trapped ion example, a single ion trap can hold on the order of 100 qubits [29, 30]. However, then to go to 1,000 qubits, that is beyond what any single trap would be able to accommodate and hold. So, the idea would be that we build 100-qubit ion traps and then communicate between them using lasers and entanglement to run a larger scale computer built out of these smaller modules. A similar concept is being developed at QCI for superconducting qubits, where we have a microwave cavity. Inside this cavity, we can encode multiple qubits worth of information, whether that is several modes within that cavity, or the information is being stored in complicated `Ket` states. However, then these resonators would then be coherently coupled to one another to make a larger scale quantum computer. So, modularity will be a part of future quantum computing as we build the larger systems.

Now, other than quantum system hardware, there is also active development in the quantum programming languages. There is much activity going on into the development of the software stack. Each of the major industrial players who are looking at quantum computing is developing it, IBM has their Qiskit [31], Google or Rigetti [20], and Microsoft, all are developing software platforms and addressing various levels in the software stack, whether that is software which immediately controls the physical qubit layer, the hardware layer, or whether It is software that is interpreting a program that is written by a user who is trying to implement a quantum algorithm and translating that program and compiling that into the types of instructions that a quantum computer needs in order to operate. So, there is a lot of research activity in the programming domain, not just in the industry, but also in the university. There are academic programs that are focused on these very tasks. We expect that these research activities will continue both because there is much work to do, but also because quantum computers themselves are going to mature with every new generation. Moreover, as they mature, the quantum programming languages will also mature [32]. There is another aspect of software development, which is not directly related to implementing an algorithm but is equally important and related to the development of very basic hardware iteration. The electronic design automation (EDA) type design software that is needed to design quantum computers. There are existing simulators to simulate the electromagnetic of the superconducting chip or semiconducting chip or whether It is layout optimizer or whether it is taking a GDS file design of the quantum circuit and then simulating that quantum circuit to understand if it is at least electromagnetically behaving the

way that It is supposed to. All of these are essential aspects of designing a real system. So, that is another area where software development is being done. It would be fair to say that with the launch of the IBM Quantum System One, the first commercially available quantum computer, that quantum computing has emerged from the laboratory and will more become the domain of computer scientists, software programmers, electrical engineer off course with physicists and mathematicians. The IBM quantum system is the first, commercially available universal gate model type quantum computer. The D-Wave system has also been around for a few years. That is also commercially available to perform quantum annealing.

We will start with an introduction to the types of quantum computing devices that exist. We will also look at the history of classical electronic computing. Then we compared that to where quantum computing is today. We look at quantum gates, single-qubit gates, two-qubit gates, and how they are used in universal quantum algorithms. We discuss quantum interference and quantum parallelism and how that underlies the power of a quantum computer. We look at examples of quantum simulation [33] or emulation, quantum annealing devices [34], and the universal gate model quantum computer. We will also discuss qubit modality and their performance. Thus, we start with the DiVincenzo criteria for quantum computers. We will discuss qubit robustness and the coherence time. We will also discuss how the gate time is critical and introduced the metric for qubits called the gate fidelity. Then, we compare different modalities against one another. We will also investigate several of them, including defect centers, ion traps, superconducting qubits, semiconducting qubits. We will focus on trapped ions and the superconducting qubits. We are looking at the promises of quantum computing, and the promise of quantum communication, and looking at quantum advantage and algorithms. We look at circuit models and look at the Deutsch-Jozsa quantum algorithm. At the end, we will discuss about various industry player in the quantum computation domain to discuss about their research and perspectives on quantum computing, we discuss IBM [35–38],Google [20, 39, 40],Microsoft [41–43],IonQ [44],Rigetti [45, 46],QCI [47],and D-WaveSystems [48]. In the next, we are going to look at circuit models and discuss the Deutsch-Jozsa quantum algorithm. We will then apply the Deutsch-Jozsa algorithm and run it on the IBM Quantum Experience.

2 Quantum Computing

We discuss quantum computers every day in the news and the popular press. There is excitement about quantum computing. It is said that quantum computers will solve certain types of problems. Ones of tremendous importance to humankind are problems that today are prohibitive or even impossible to solve with current computers. We will discuss pharmaceuticals and drug discovery. We are gaining a better understanding of new materials like high-temperature superconductors, New methods for machine learning [49–52], artificial intelligence, optimization problems, and financial services in technology. Quantum computers will even challenge and change the way we securely communicate information. It certainly sounds like a fantastic and exciting future, which leads us to a few fundamental questions. What exactly is a quantum computer, and what is its suitable application? More importantly, when will we have one? Quantum computers are not just smaller, faster versions of classical computers. They are fundamentally different. Whereas in the digital computer world, a bit, which is one fundamental element of information, is a zero or a one. In a quantum computer, we can have a quantum bit, or qubit, that is in a superposition of zero and one. We can design and control them. We are engineering and manipulating quantum mechanics. So, when We have a quantum computer is a fascinating and nuanced question. Moreover, the answer will, therefore, be finicky. We have been saying that quantum computers are ten years away. We have been saying that for decades. Depending on the

definition, we already have quantum computers, but they are just small. It is not decades away or a century away from that quantum age has now arrived. Quantum computers are not merely faster, smaller versions of the conventional computers we have today. Nor are they another incremental step in the evolution of Moore's Law. Instead, quantum computers stand for a new, fundamentally different type of computing paradigm, one that carries tremendous advantage for certain types of problems of importance. Quantum computing could transform industries where there are significant optimization problems. We have a lot of discrete or binary decisions to make to figure out, do we do this first or that first. Another way to understand the difference between classical and quantum computers is to look at quantum systems of quantum simulation. A quantum processor is a suitable tool for modelling other quantum systems. Biomolecule systems and tons of other systems that we use, material systems fundamentally work on based on those quantum mechanical properties. We need a quantum machine to simulate quantum effects. When we can manipulate individual molecules and understand what is going on in those molecules, how they bond, then We will be able to have an excellent handle on generating new things and novel materials that might be very useful. Still, we are just at the very beginning of quantum computing development. Assembling and testing the prototype processors. It is a bit like being in the 1950s at the dawn of transistor-based computing. Furthermore, just as integrated circuits led to an information processing revolution last century, driving economic growth and productivity, many people today believe that quantum computing will have a similar impact on this century. Quantum computing and quantum algorithms present fundamentally new programming and algorithm design paradigms. How do we fundamentally unlock new ideas in computing? We are still discussing a lot about how to improve the individual components and connect them. We are looking to enable the increased complexity and functionality of these qubits. We are here at the very beginning of the new revolution. We find that it is tremendously exciting. The goal here is to separate the promise from the hype and to technologically understand the basics of the quantum computation working principle and its applications. We will begin by focusing on those basics.

Quantum computers are not merely smaller, faster versions of today's computers. Instead, they represent a fundamentally new paradigm for processing information. They can exceed the performance of conventional computers for problems of importance to humankind and businesses alike, in areas such as

- Cybersecurity,
- Materials science,
- Chemistry,
- Pharmaceuticals,
- Machine learning,
- Optimization, and more.

3 An overview of Quantum Computer

Classical computers have changed dramatically over the past 80 years, from the room-filling vacuum-tube-based computers like ENIAC (Figure 1)

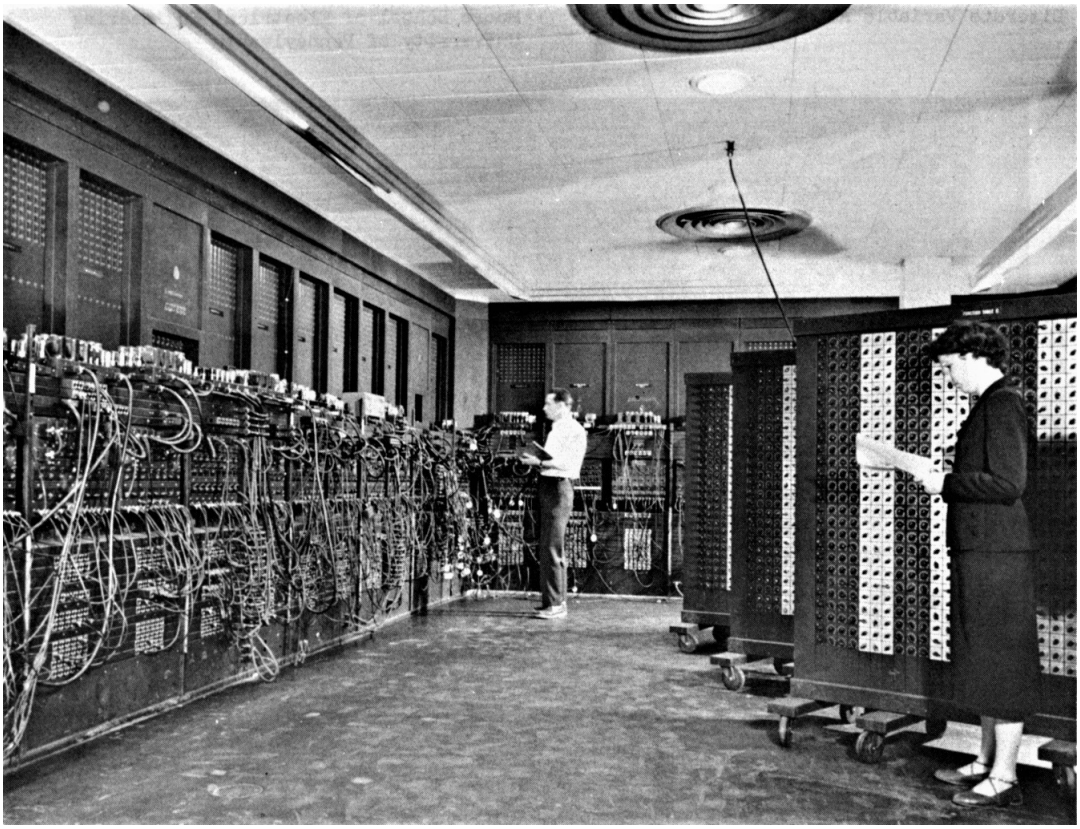


Figure 1: ENIAC; Source:U.S. Army

Currently, quantum computers are at the research and prototyping stage, looking more like an ENIAC than a laptop or tablet. They often occupy an entire laboratory space with a variety of machines and tools to house and operate the core of the quantum computer. A portion of this infrastructure surrounding the quantum computing "core" is necessary to shield the quantum computer from sources of electromagnetic noise, mechanical vibration, heat, and other noise sources, which tend to degrade performance. Another portion, comprising conventional "classical" computers, electronics, and optical systems, is used to control the quantum computer, implement an algorithm, and read out the result.

Figure 2: A research-grade superconducting quantum processor. The processor is located inside the white dilution refrigerator hanging from the support structure. The refrigerator cools the processor to the milliKelvin temperatures required to operate it;Source:IBM

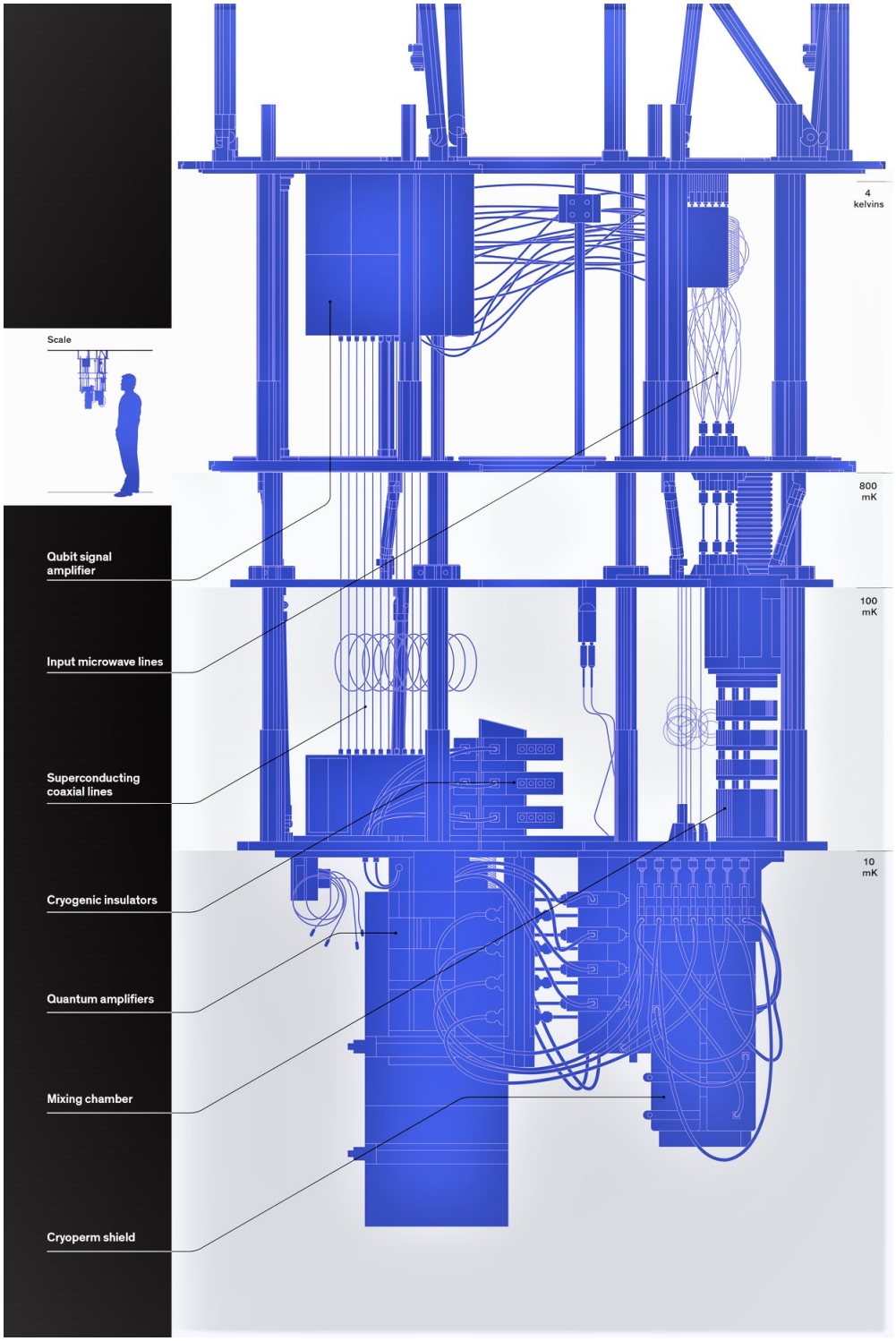


Figure 3: Dilution refrigerator;Source:IBM

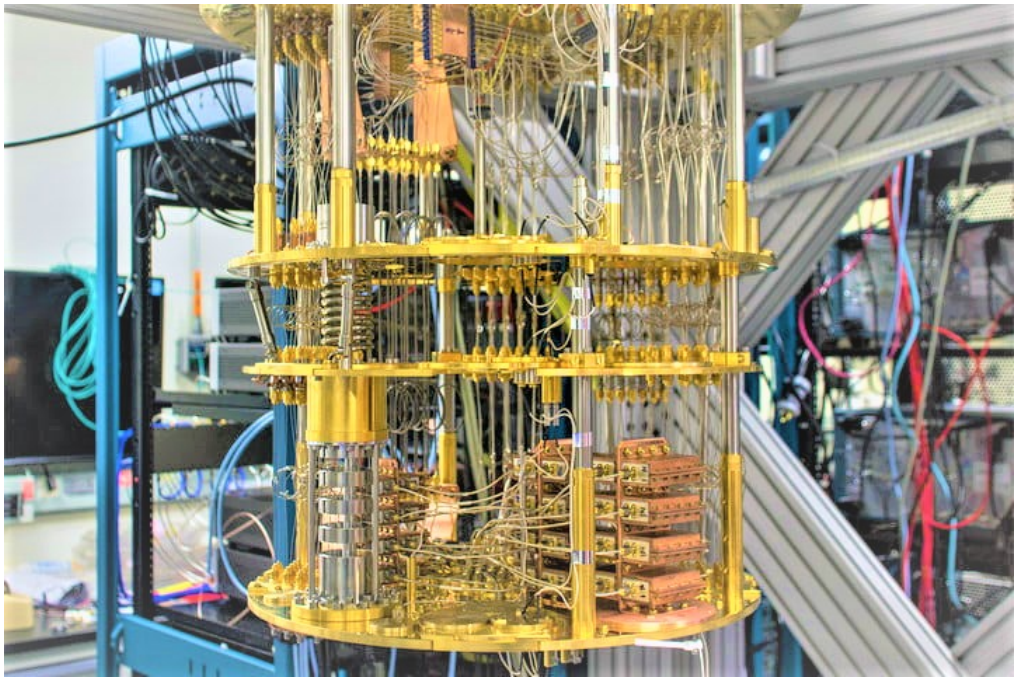


Figure 4: superconducting quantum processor;Source:IBM

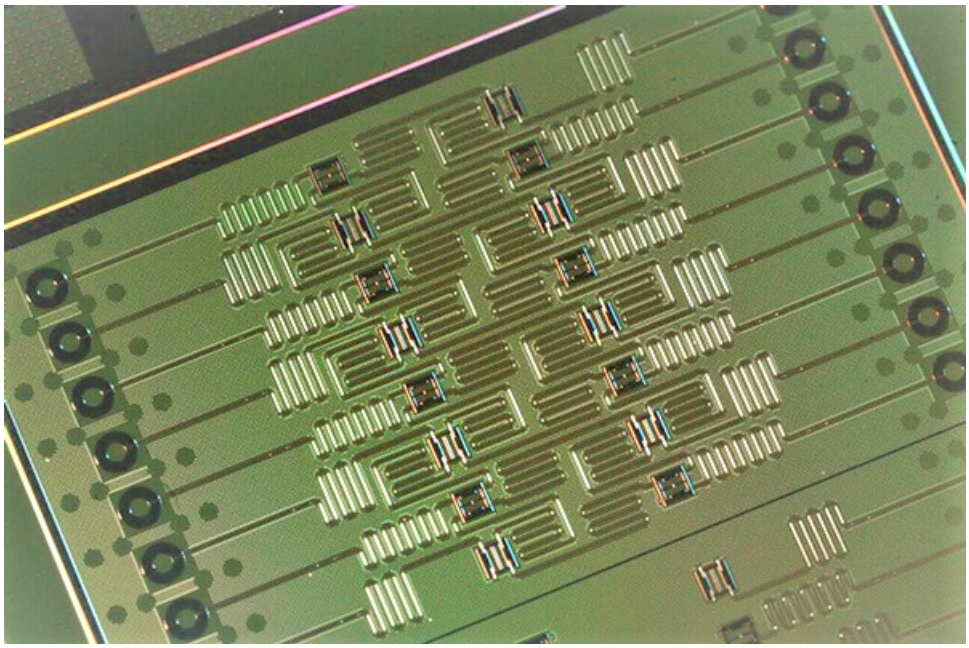


Figure 5: superconducting qubit citcuit;Source:IBM

Figure 6: superconducting quantum annealer;Source:D-Wave

In the picture above, we see a large research-grade "dilution refrigerator" used to house and cool a prototype superconducting quantum processor. Refrigeration is required to cool the quantum chip to its operating temperature of less than 20 milliKelvin, a temperature more than 100 times colder than outer space. The refrigerator also serves to reduce the thermal load and noise that would otherwise degrade performance, arising in large part from the room-temperature electronics connected to the chip through various types of electrical cabling. To the left of the refrigerator are racks of such electronics, including arbitrary waveform generators, microwave signal generators, and current sources used to control the processor.

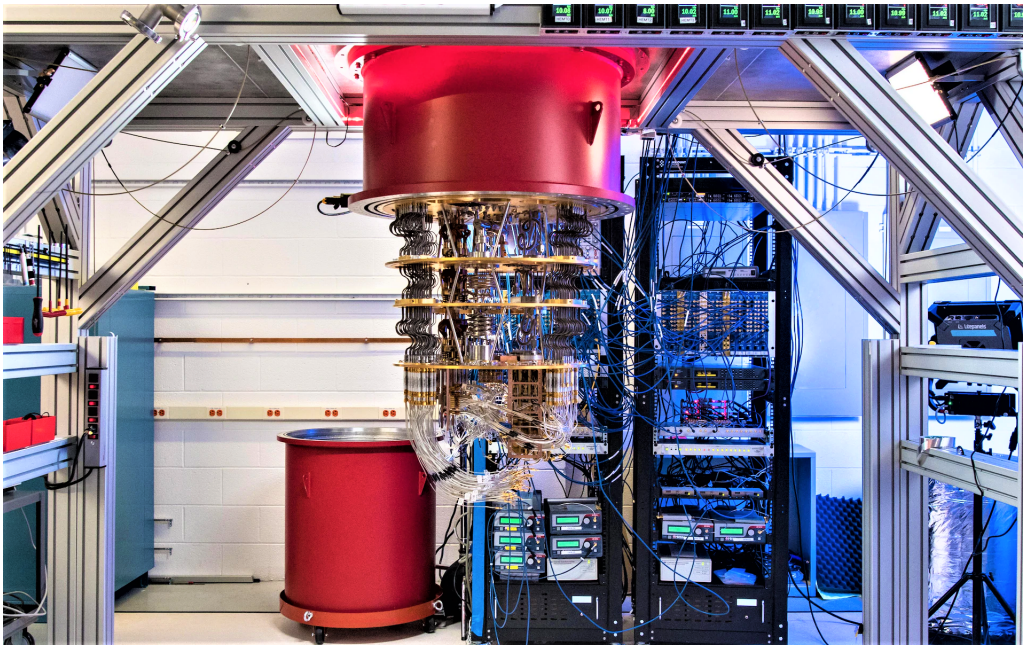


Figure 7: Google System;Source:Google

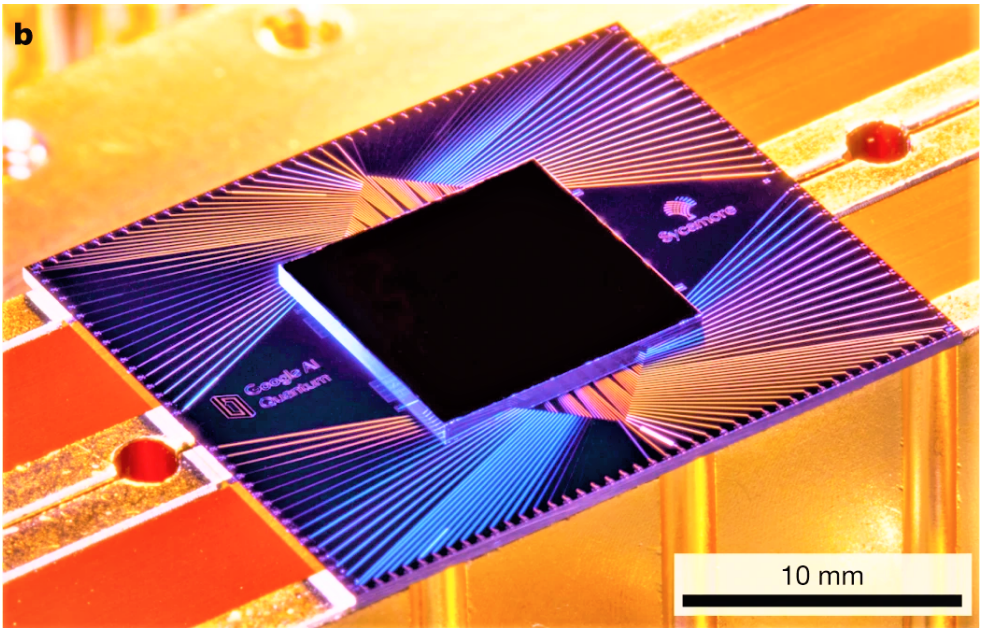


Figure 8: Google superconducting quantum processor;Source:Google

In the figure below, we see an optical table, on which stands a large black box housing the optical system used to control and measure a trapped-ion quantum computer [53]. The trapped ion computer "core" itself may reside in a cryogenic chamber at a temperature around 3-4 Kelvin,

a temperature similar to outer space to obtain and maintain an ultra-high vacuum (although this is not required). High-stability lasers send light through a variety of mirrors, beamsplitters, optical modulators, and the like to address and read the individual ions that comprise the quantum computer.

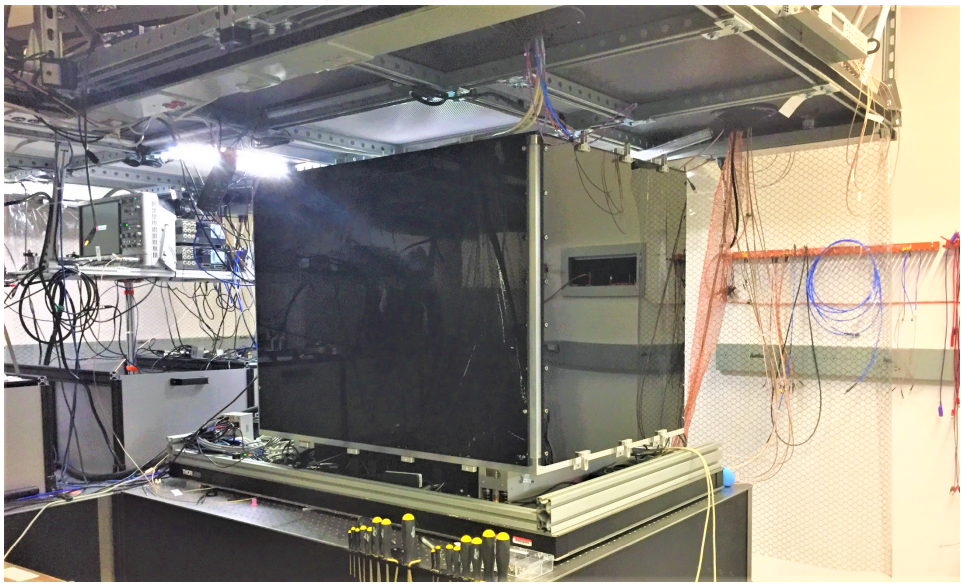


Figure 9: Trapped-ion quantum computer; Source: IonQ

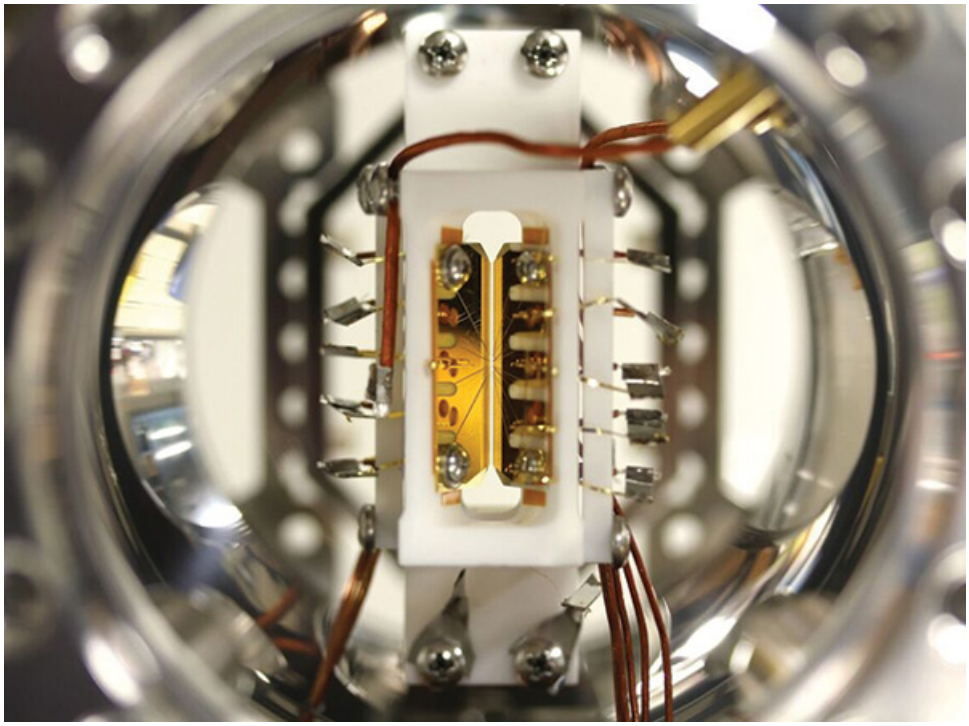


Figure 10: Trapped-ion quantum computer; Source: Joint Quantum Institute

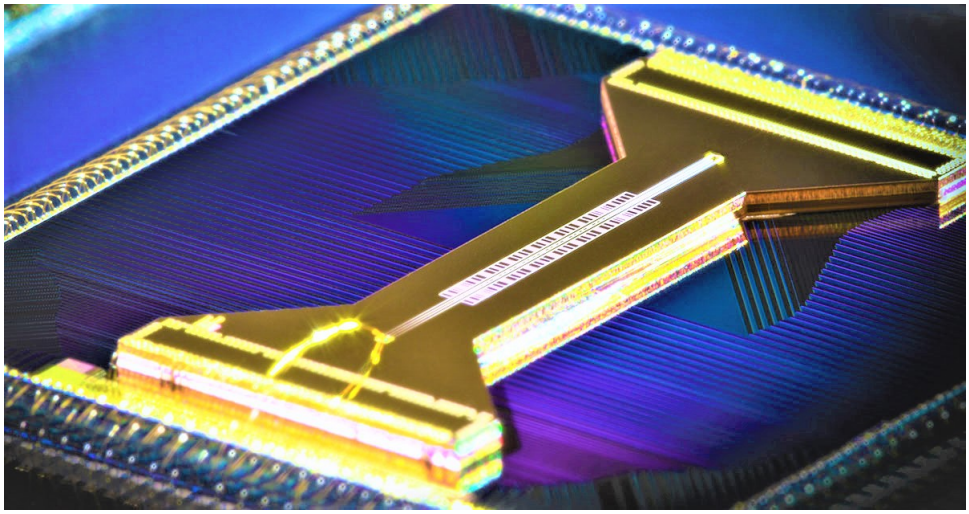


Figure 11: Honeywell's Ion Trap Quantum; Source: Honeywell

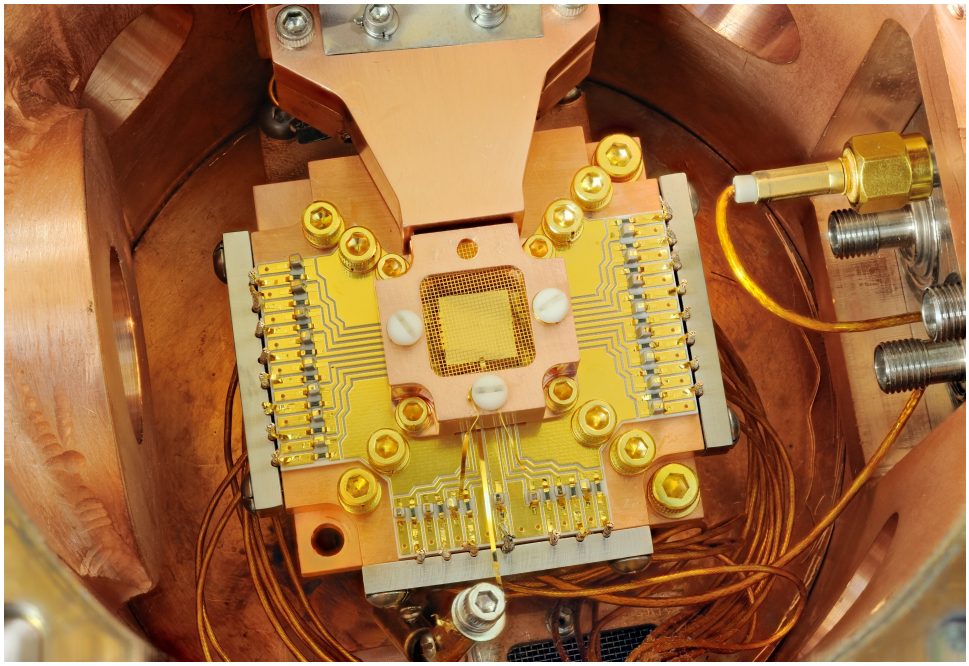


Figure 12: Trapped ion quantum computer; Source: NIST

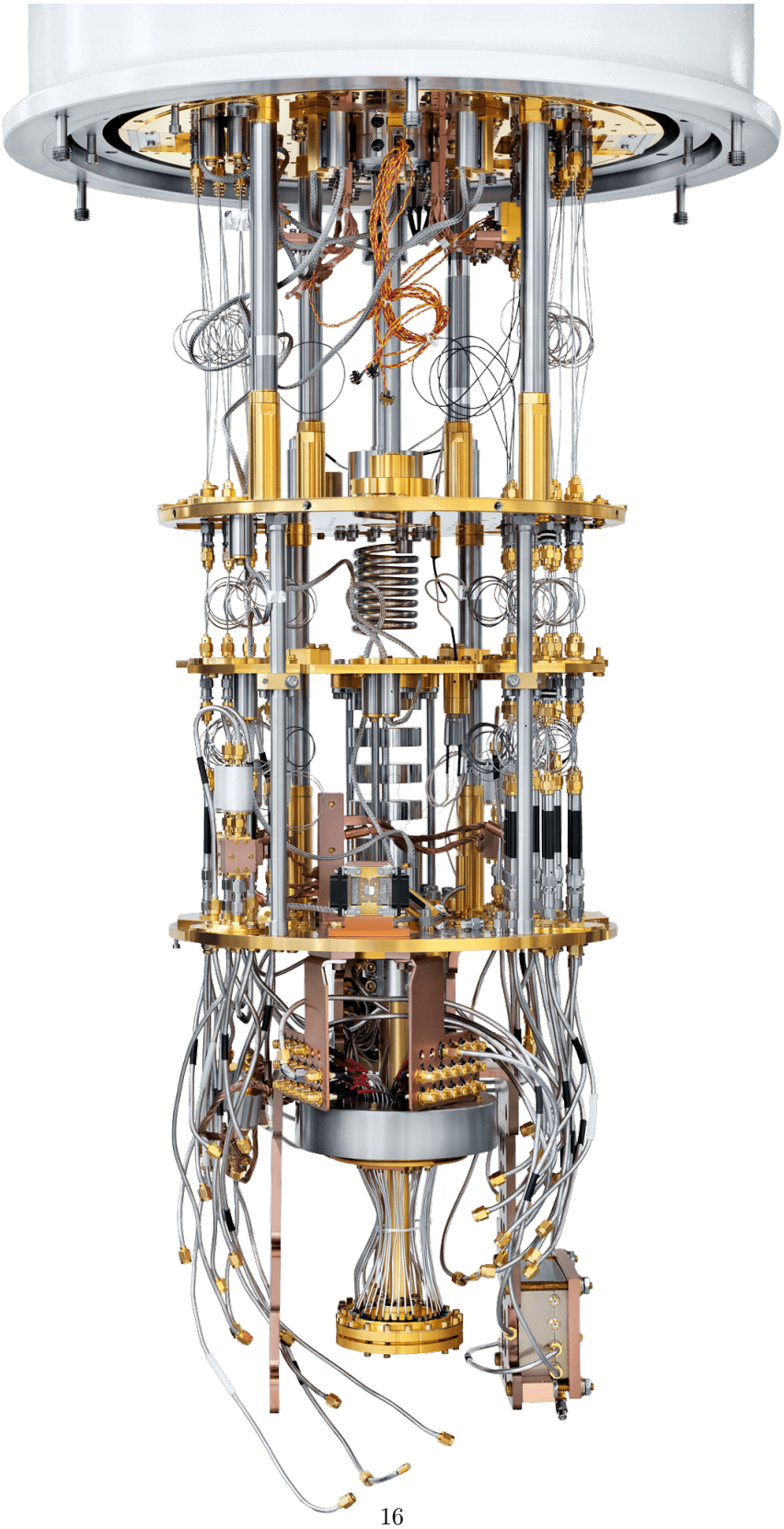


Figure 13: QPU (quantum processing unit) ; Source: Rigetti

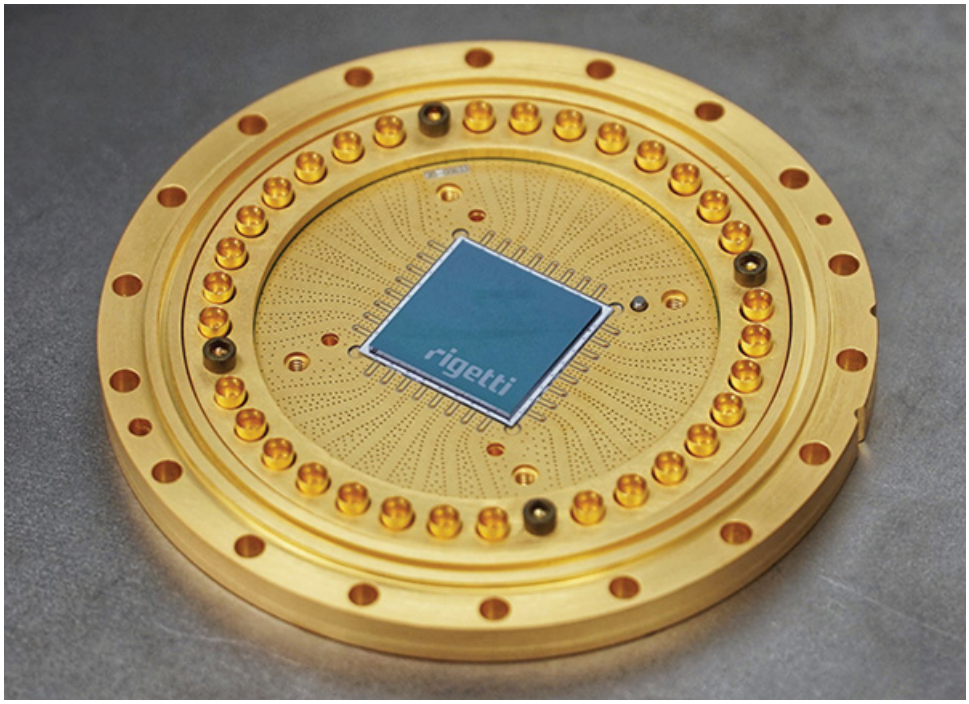


Figure 14: gold-plated copper disk with a silicon chip; Source: Rigetti

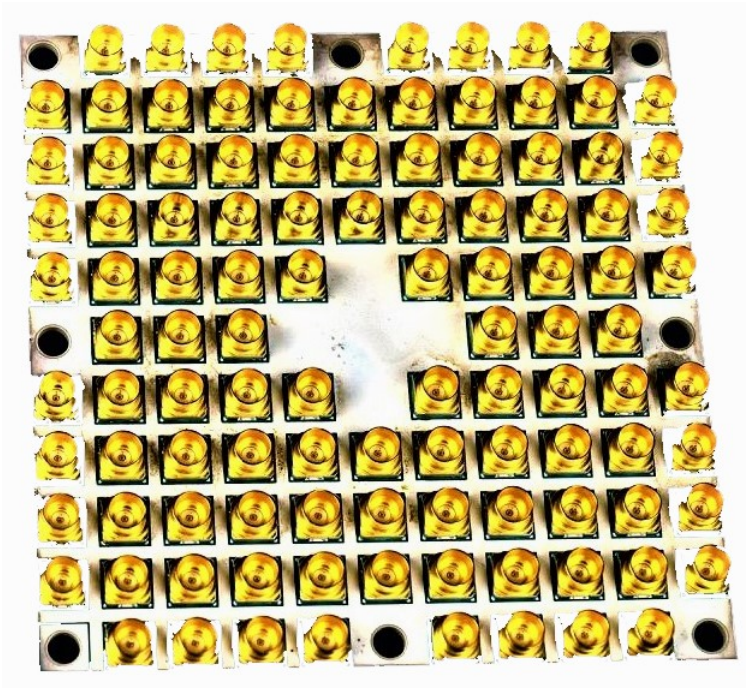


Figure 15: 49 qubits superconducting quantum processor Tangle Lake; Source: Intel

4 Physical and Conceptual Models of Classical Computation

To understand quantum computing, we first need to understand how information is processed today and, more generally, what constitutes "a computer" and "computation."
The most common approach to classical information processing today uses a conventional electronic computer comprising a memory and a transistor-based computational processing unit. However, this is not the only physical manifestation of a classical computer. For example, human beings and our brains process information with different physical methods and architectures [54]. As these two examples suggest, there are many physical models of classical computation.
Physical Models of Classical Computation:
Mechanical: A computational system built with primarily mechanical components is a mechanical computer. Adding machines used for bookkeeping during the first part of the last century is an example of such computers. The input system is composed of numbered key buttons. After entering the numbers, a user pulls the crank, gear-wheels start turning, and the sum is mechanically computed and displayed. As in the case of an adding machine, mechanical computers are generally designed to implement application-specific tasks.
Electrical: Electrical computers use electrical elements that switch on/off electrical currents or voltages. Today's personal computers are in this class, and they use transistors as the fundamental switching elements. Transistors enable the construction of a universal classical computer, one that can, in principle, tackle any computable problem. However, it may not be able to do so efficiently (in a reasonable amount of time or using a reasonable amount of physical hardware).
Optical: Systems that use photons the fundamental particles of light to perform computation are optical computers. The gates used to perform logic with photons can be engineered using nonlinear optical materials. As of today, existing optical computers tend to be application-specific.
Biological: Biological molecules, for example, proteins or DNA, can be used to process information. The individual, necessary elements for a fully operational biological computer, such as biological transistors, have been demonstrated. However, present biological, computational systems are hybrids that require the addition of electrical or mechanical components to operate.
There are also several conceptual models and architectures of classical computation, which in principle, any of the above physical systems can be used to process information.
Conceptual Models and Architectures of Classical Computation:
Turing Machine: A Turing machine comprises a memory tape and reads/write head. The memory tape is divided into discrete cells that store data. The head successively manipulates the cells. According to a set of rules, cell data may be altered depending on the cell's prior information, which is also accessible by the head. This scheme provides an architecture to construct universal computational systems.
Cellular Automaton: A cellular automaton comprises an array of cells, each connected to several of its neighboring cells. After the cells are set to initial values, the cellular automaton evolves according to a set of rules that governs how the state of each cell changes in response to the states of its neighboring cells. Depending on the rules and connectivity of the cells, the result may be a uniform, oscillating, chaotic, or another intricate pattern. This computational concept is used to simulate or mimic the behavior of biological or chemical systems. For example, a global function emerges from a large number of seeming independent agents that interact with one another in a specified way.
Von Neumann Architecture: Architectures comprising a central processing unit and a memory unit are called von Neumann architectures. The central processing unit contains a controller, an arithmetic logic unit, and registers. The controller manages the computational processes: it

requests data from memory, stores it in a register, directs the arithmetic logic unit to process the data in the register, and then sends the result back to the memory. This model is used for most present-day computational devices.

We are all familiar with laptop computers, desktop computers, even servers in the cloud that we interact with daily. In this section, we will refer to these as classical computers to contrast them with quantum computers. We will be discussing in detail later in the section. Classical computers use transistor-based integrated circuits-computer chips to process information and solve problems, whether for implementing a financial transaction, simulating a weather pattern, developing a CAD design, or even just drafting an email to a colleague. However, it is important to remember that there are many alternative ways in which one can process information. So, before we get started, let us discuss about what constitutes a computer and computation. There are many models of classical computation, and we would like to illustrate a wide variety of models, some of which we might find unexpected. One model is mechanical. So, for example, the thermostats on the walls, are driven by pneumatic pressure, and they actuate a little switch. So, they are a little kind of computer that controls temperature, but it is an analog computer. So mechanical computers do not need to be digital. They can come in all kinds of shapes and forms. the one that is the most famous in history is Babbage's difference engine. We can have electrical circuits, which are ways of building a universal classical computer. We can also have optical computers that are made from information carriers that are photons and not electrons. Biological. In many ways, we and We are walking, discussing computers. this idea of biology and biological systems as computers is currently going through a renaissance because of the notions of neural networks [55, 56] and deep learning and these kinds of networks of neurons that act as computation. We also want to make a distinction between these models and some conceptual models. These are models by which we might realize computation, and these are the conceptual models that we might want to realize, one of them, the Turing machine. It is a machine that has a head and tape, and the tape has slots on it, which may have ones and zeros. it is something which has an extent to left and right, which goes off, in principle, to infinity. the tape is a kind of memory. the thing which is ostensibly doing the computation is a head that can read and write to this tape, but the only thing inside this head is a finite state machine. So, there are different states, and there are transitions between the states which happen to depend on what is read at what time and the earlier state that it used to exist. Turing machines like this come in many different flavors. There are probabilistic Turing machines, and there are universal Turing machines. So, given a certain kind of structure of a finite state machine here, we find that this Turing machine, then, can simulate any other Turing machine. Here are another model cellular automata. here, the model of computation is a world which is a grid in two dimensions in n dimensions where the point is that we have some kind of state in a local cell of this grid, and it undergoes transitions based on the state of its neighbors. we may have a local Cartesian neighborhood. we may have super Cartesian, but depending on what we are surrounded by, we change our state. we change ourselves to become empty or filled or different colors and so forth. these rules of patterns and pattern changes can give rise to computation. There is Von Neumann architecture. It is, again, a conceptual model of computation. here, the idea is to split memory from something which does arithmetic. So, we have an arithmetic logic unit, for example, some registers. Memory reads out data and feeds it into this ALU. Then the ALU feeds data back into the memory, and this is the model that is used by all processors today, including the ones on all our phones. There is DNA-based computation. This is the idea that we have strands of bases in AGCT, and then, A and T associate each other, which is called ligation. then G and C also ligate together. when we have two different strands of DNA, they will pattern match other strands in the right locations to produce base-pair ligations. this has been shown to allow a kind of computation using polymerase chain reaction tools. If we have a beaker, for example,

with just one DNA strand, with PCR, we can amplify the number of DNA strands there so we can detect a certain DNA sequence. Thus, it is elegant that we can think of using such tools to do computation. we want us to be open to such models of computation, especially today, because we are starting to reconsider what it means to do computation. In many ways, we are at the end of the road of silicon today. We cannot rely on Moore's Law much longer to provide increasing scaling. That is exponential of capability and size, power, and weight to build the computers. We need to look at different physical mechanisms to build computation. that is why all these different approaches, where we represent information in different ways, is so appealing to think about because the next thing beyond silicon, could be something different very different that utilize these ideas. we might discover that it is already happening all around us, or within us, if we only know how to think about it in the right way. So, although this section is about quantum computation, ostensibly, what We want to think about is how we are also thinking about a broader question of what is the physics of computation? How do we think of physical mechanisms as doing computation? Moreover, how do we think we might be able to exploit physical mechanisms and biological mechanisms that exist to realize the computations that we want to achieve?

5 Origins of Quantum Computing

How long has the idea of quantum computing been around? When did it start, and what have been the key milestones in its development? Before answering these questions for quantum computing, it is worth looking back at the history of classical electronic computing and how those technologies developed over the past 100 years.

The development of classical computers did not jump directly from the vacuum tube to laptops and smartphones. Commercial demand for intermediate products throughout the 1900s incentivized companies to develop and advance the technologies that, over time, led to the ubiquitous classical computers we have today. Early examples of such "off-ramp" products include:

Radar: Before being replaced by transistors, vacuum tubes were used to modulate radar signals.

Frequency mixers: Some of the first research on transistors developed out of an attempt to build frequency mixers for radio receivers during World War II. It was the starting point for Bell Lab's work on transistors.

Transistor radios: The development of the bipolar junction transistor lead to the creation of transistor radios sold by companies like Texas Instruments, IDEA, and Sony. Unlike vacuum-tube radios, which could not output sound while the tubes were warming up, transistor radios could turn on and output sound immediately.

Amplifiers: Transistors were used (and are still used today) in all manner of products requiring electrical amplification, including sound speakers, hearing aids, radios, and telephones.

Along these lines, it will be challenging to sustain intense commercial interest and funding for quantum computing technology development if the first useful quantum computer is a 1,000,000-qubit fault-tolerant machine that is still 20-30 years in the future. Nearer-term commercial applications of quantum information [1] technologies will be needed to seed and maintain the virtuous cycle of technology development needed to realize large-scale quantum computers. Some of these quantum information technologies "off-ramp" applications could be, for example,

Noisy, intermediate-scale quantum simulation: Small, error-prone quantum computers may find use in simulating small-scale quantum systems perhaps as a co-processor to a classical computer.

Noisy, intermediate-scale Optimization: Noisy, error-prone quantum computers may

also have applications to optimization or classification problems. For more examples of noisy, intermediate-scale quantum (NISQ) applications of quantum computing [28].

Besides, the various components needed to build these quantum systems will likely generate new business opportunities and expand existing ones. For example, the optics, electronics, refrigeration, software, and services are likely " dual-use" beyond solely quantum computing. These products will benefit from the enhancements required for quantum computing and, in this enhanced state, support customers with applications beyond solely quantum computing.

Before diving into quantum computing, revisiting the history of classical electronic computing in the last century is worthwhile. We will see a few takeaways from this history that are relevant to the current and future development of quantum computing. Lee De Forest invented the first three-terminal triode vacuum tube in 1906. Vacuum tubes, much like the transistors that would later follow, are essentially faucets for electricity. The application of a small voltage on one terminal effectively opens a valve, which allows current to flow between the other two terminals. As such, vacuum tubes were used primarily as amplifiers for radio receivers, but they could also be used as on/off switches to implement logic gates. Thus, it was about 40 years after that first invention. We had the first large scale computers based on vacuum tubes, such as the electronic numerical integrator and the computer, or ENIAC, developed at the University of Pennsylvania in the 1940s. Also, around that time in 1947, the first transistor was invented at Bell Laboratories, and the first fully transistor-based computer soon followed. That computer, the transistor experimental computer number 0, or TX0, was built at MIT and Lincoln Laboratory in the mid-1950s, and it featured discrete transistors and magnetic core memory. Quite different from the computers we know and use today. Shortly after that, in 1959, the first integrated circuits using silicon were demonstrated. However, still, it was a good 20 or 30 years before we had the types of integrated circuit chips and memory chips that we now use in the computers daily. The first commercially available monolithic processor, the Intel 4004, appeared in 1971. It was a 4-bit processor, featured 2100 transistors, and clocked in at around 740 kilohertz. Within a year or two, however, Intel came out with another processor, the 8008. An 8-bit processor with nearly double the number of transistors. This doubling of the number of transistors approximately every two years was exemplary of what became known as Moore's Law. by the 1990s, following Moore's Law, the number of transistors had increased into the millions. today, we have computer chips with five billion transistors or more with multicore processors and graphical processing units, GPUs with close to 20 billion transistors. Although performance increases had previously simply followed from this Moore's law type scaling, these straightforward improvements have significantly waned over the past decade. Nonetheless, with the development of high k dielectrics, low resistance interconnects, multicore processors, 3D integration, and the like, we can expect continued improvements in the performance of classical processors for years to come. In contrast to these 100 plus years of classical computing development, quantum computing is much more recent. In the early 1980s, Richard Feynman suggested that if we want to simulate a quantum system, a task that is very hard for a classical computer, we should use a quantum system to perform that simulation [57, 58]. He was suggesting we should build a quantum computer. He also noted that it is a fascinating problem, because it is not so easy, and he was right. Over the next decade or so, researchers thought about what kinds of algorithms could potentially give a quantum advantage for real-world problems of significance, and how fragile quantum states could ever be used to implement such an algorithm. The answers started in the mid-1990s, including two significant milestones in the history of quantum computing. The first was the discovery of Shor's algorithm, developed by Peter Shor [59]. Shor's algorithm was not the first quantum algorithm to show the quantum advantage. However, it was the first that also addressed an important practical problem, namely the factorization of large numbers. Now that is an important problem because the difficulty of factorization is a pillar for the present-day encryption

schemes that protect the information. Essentially factoring large numbers is a very challenging problem on a classical computer, which is why it is used for public-key encryption. Peter Shor showed that factorization could be done efficiently on a quantum computer. Also, around that time, Peter Shor and his colleagues Robert Calder bank and Andrew Steane developed the first quantum error-correcting codes [60], which, once fully implemented, will enable quantum computers to continue to operate robustly in the presence of errors. Since then, researchers have focused on both the underlying physics, as well as the hardware that we can use to build quantum computers. Starting at the single-qubit level, researchers have explored numerous qubit modalities, from superconductors to trapped ions, semiconductors, and more. Today we have processors operating with 10 to 20 qubits and reports of 50 qubits being available soon. There is also a marked transition from prototype demonstrations in the early 2000s to where we are today, which is engineering larger and larger quantum systems. We are even now seeing examples of cloud quantum computers [61] on the web that can be used by people worldwide to try out algorithms, and We will use one in this section. So, what does this all mean? Well, we think there are a couple of takeaways from this brief historical discussion. The first is that technology development takes time. It took over a century to get from the first triode vacuum tubes to the computers we have today. That development is not over. It continues today. there are many changes along the way. The right approach to building a classical computer changed many times over the years. There was no single right answer. The right technology in the 1940s was different from the 1980s, and that was also different from today. Nonetheless, in hindsight, all these steps were crucial to the overall development. Similarly, we can expect that going forward, and quantum computing will likely go through technology evolution. The best qubit modalities today are not necessarily the ones that will excel in the future. However, the observation is that in the absence of effort, we should not expect the right technology to appear if we wait long enough. Technology is developed; it is not bestowed. The road to future quantum computers, whatever they may end up looking like, is paved with the technologies we develop today. Finally, we should not underestimate the crucial role that commercialization played in the development of classical computing technologies. From the very beginning, transistors had commercial applications that generated revenue long before computers were available, including radio amplifiers, hearing aids. Although governments played a key role in seeding the development of transistor-based computers and are playing an equally crucial role in the development of quantum computers today, it was commercial development of transistors and computer chips that ultimately enabled the virtuous cycle of development that made possible Moore's law like scaling that led to the computers we use today. a major challenge for quantum computing is to identify these kinds of commercially useful applications. For qubits and small-scale quantum processors that can kickstart a similar virtuous development cycle. One that will be needed if we are to realize large scale quantum computers.

6 How is a Quantum Computer Different

How is a quantum computer different than a classical computer? In this section, we will compare and contrast classical and quantum computers to gain a better understanding of the unique ways in which a quantum computer represents information.

So how is a quantum computer different? We can begin to answer that question by comparing it with a classical computer. Classical computers are the computing devices that we use every day at work and home, and they process information using transistors, each of which can store one bit of information. We will call this a classical bit, which is binary, and it can take on one of two states. It can either be in state 0, let us say the absence of a voltage on the transistor

gate, or it can be in state 1, the presence of a voltage on that gate. These are discrete, robust states, and when we measure the state of that transistor, we will see either a 0 or a 1, depending on where that bit was set. We can contrast that with a quantum computer, which is built from logical elements called qubits, which is short for quantum bits. A qubit is binary in the sense that it is realized using a quantum coherent two-state system, and so it can be set in state 0 or state 1, but because it is quantum mechanical, it can do much more. A qubit can also be at a quantum superposition state. It is a single state, but it carries aspects of both state 0 and state 1 simultaneously, and this is a manifestly quantum mechanical effect. We can represent a qubit state on what called a Bloch Sphere, which for this discussion, we can think of as the planet Earth, where state 0 is at the North Pole, and state 1 is at the South Pole. In this representation, a classical bit can be either at the North Pole or the South Pole, but that is it.

Figure 16: The two logical states of a classical system correspond to north and south on a globe

In contrast, a qubit can exist anywhere on its surface. Now, a qubit can also be at the North Pole or the South Pole. That is fine, but when it is anywhere else, the qubit is in a superposition state, again, a single state that takes on aspects of state 0 and state 1 simultaneously, as shown in the notation here. Superposition states result in probabilistic measurements, meaning that if we had said an equal superposition of 0 and 1, and we measure the qubit, we have a 50/50 chance of measuring in state 0 and a 50/50 chance of getting state 1. If we identically prepare that same state and measure it, and do that repeatedly, Half of the time, we will get a 0, and Half of the time, and We will get a 1. As a result, quantum computers rely on encoding information in fundamentally different ways than classical computers. So, on a classical computer N, classical bits represent a single N-bit state. For example, if N equals 3, we have 3 classical bits, and they can represent the state 000 or 001, all the way up to 111. There are eight different combinations, but the three classical bits can represent only one of them at a time. Thus, when we want to process the information on a classical computer, we pick one of those states as the input, we process the information, and we get a result as an output. However, if we also want to process information using a different input state, we have two choices. We can either process in parallel by using additional copies of the hardware, or with added time, we can process sequentially on the same piece of hardware. It is classical parallelism, and in both cases, we needed more resources, either more hardware or more time. The qubits in a quantum computer, on the other hand, can be set into a single superposition state that simultaneously carries aspects of all these 2 to the N components. So, for example, with three qubits, a quantum computer can represent aspects of all eight different components in a single quantum superposition state. Consequently,

we have a quantum version of parallelism, and importantly, we also have quantum interference between those constituent components. Quantum parallelism and quantum interference make a quantum computer different, and in the next section, we will give examples of how they work in a quantum processor.

In transistor-based classical computers, a transistor represents a classical binary bit that can store one bit of information. In one of two distinct logical states, classical bits are found in logic state 0 or logic state 1. "State 0" corresponds to the transistor switch being "off" (e.g., no voltage is applied to the transistor gate, and so no current flows in the transistor channel), and "state 1" corresponds to the transistor switch being "on" (e.g., a voltage is applied to the gate, and so a current flows through the transistor channel). These discrete states are robust and can be measured with near certainty.

The fundamental elements of quantum computers are "quantum bits", typically referred to as "qubits." Qubits are quantum-mechanical two-level systems. They are binary in the sense that they can be initialized in classical states 0 or 1. However, as quantum mechanical objects, qubits can also be prepared in a quantum superposition state: a single quantum state that embodies aspects of both state 0 and state 1.

Quantum superposition states are succinctly represented using Dirac notation [62–67]. In this notation, quantum states are expressed as "kets," where $|0\rangle$ and $|1\rangle$ represent the quantum states 0 and 1 respectively. A qubit that is in a superposition of these two states is then written as $|\psi\rangle = a|0\rangle + b|1\rangle$. The coefficients a and b are called "probability amplitudes," and they are related to the relative "weighting" of the two states in the superposition. An obvious special case occurs when either is zero, in which case the state$|\psi\rangle$ is no longer in a superposition. For example, if $a = 0$, then $|\psi\rangle = |1\rangle$. More generally, both a and b can be complex numbers and therefore must satisfy $|a|^2 + |b|^2 = 1$, a normalization to unity that ensures the "weights" being compared are of a standard, consistent size. This is analogous to the convention that probabilities are set to sum to 1.

Both classical and quantum bits can be visualized on a "Bloch sphere," a tool which can be thought of as the planet Earth, as pictured in Fig. 5. By convention, the "north pole" of the sphere represents state 0, and the "south pole" represents state 1. A classical bit is either at the north pole or the south pole, but nowhere else. In contrast, a qubit may exist anywhere on the surface of the sphere. When the qubit state is anywhere except for the north and south poles, it is in a superposition state.

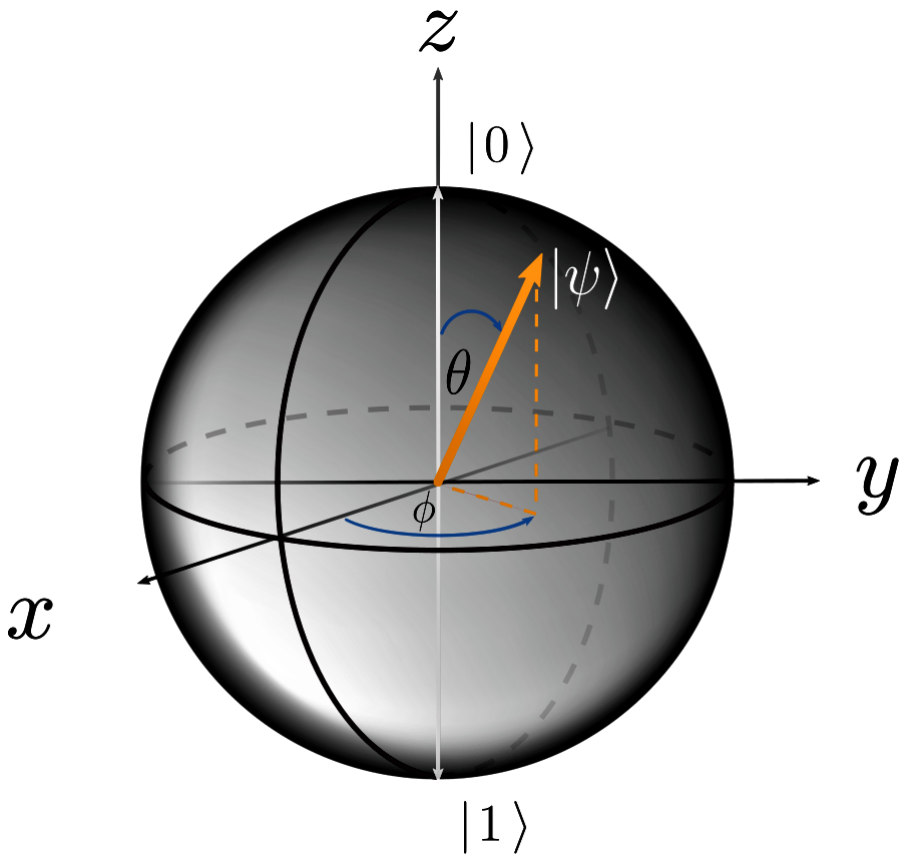


Figure 17: The Bloch Sphere

To better illuminate the connection between the state of a qubit and the Bloch sphere, the coefficients of the state $|\psi\rangle$ can be expressed as $|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle$. It is a straightforward exercise to confirm that these coefficients satisfy the normalization condition mentioned above. The two angles θ and ϕ, determine the point on the surface of the Bloch sphere corresponding to the state $|\psi\rangle$, as pictured in Fig. 6. Intuitively, these angles relate to the globe in Fig. 5, because θ moves the state $|\psi\rangle$ in the north-south direction and corresponds to the qubit "latitude," while ϕ moves the state along the east-west direction and corresponds to its "longitude."

Ideal projective measurement of a qubit occurs along a single axis of the Bloch sphere, for example, the z-axis (which on the globe would be the line connecting the north and south poles). It is called the measurement basis, and measurement will yield a classical result either "state 0" or "state 1" along this axis. The measurement process itself is probabilistic, and the probability of obtaining either $|0\rangle$ or $|1\rangle$ is related to the qubit's projection onto the measurement basis. As an example, consider when the qubit is an equal superposition of states $|0\rangle$ and$|1\rangle$. It occurs whenever $\theta = \pi/2$ and corresponds to the states along the equator of the globe. In these cases, the state, when measured along the z-axis, is equally likely to result in the outcome $|0\rangle$ or $|1\rangle$, because their probability amplitudes are the same. Intuitively, any point on the equator when projected onto the z-axis is at the center of the Earth, equally "far" from the north and south pole.

As a result, quantum computers rely on encoding information in fundamentally different

ways than classical computers [68]. For N bits, there are 2^N possible classical states. However, a classical computer can represent only one of these N-bit states at a time. Processing multiple N-bit states can either be performed sequentially in time or parallel using additional copies of the hardware. It is classical parallelism. In contrast, the qubits in a quantum computer can be set into a single superposition state that may simultaneously carry aspects of all 2^N components. As we will see shortly, this enables two uniquely quantum mechanical effects: quantum parallelism and quantum interference.

In the context of the Bloch sphere, we have discussed about making measurements along what we call the qubits quantization axis, or the z-axis. the question was, basically, are there ways in which we can make measurements along different axes of the Bloch sphere? And the answer is yes. it depends a bit on the modality that we are discussing about. But generally speaking, we could always keep our measurement apparatus measuring along the z-axis. right before we make that measurement for example, if we wanted to measure the x-axis what we could do is a rotation that basically brings the x-axis to the z-axis and then make our measurement. so, we are still measuring along the z-axis. But by doing this rotation right before the measurement, we are effectively rotating the qubit states along x up to the z-axis and then making a measurement. so, this is a common way to do state tomography [69] and process tomography measurements when It is more convenient to just have our measurement basis be just along the x-axis, just along the z-axis is to do a rotation right before measurement to measure effectively in these different bases.

Bloch Sphere, The same way that a person's position can be defined by a point on the Earth, a quantum state can be defined by a point on the Bloch sphere. While a point on the globe always refers to position, a point on the Bloch sphere refers to a qubit's state. For example, a qubit state can be spin-up (north pole), spin-down (south pole), or in a superposition of spin-up and spin-down (anywhere else). Identify the qubit states on the Bloch sphere.

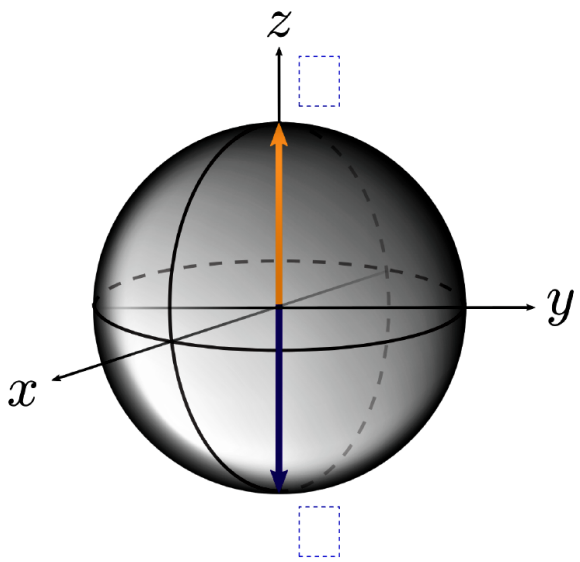


Figure 18: Bloch Sphere

On the Bloch Sphere, the z-axis runs from the south to the north pole of the sphere. Axes x-axis and y-axis are perpendicular to one another in the plane of the equator of the sphere. The north pole corresponds to state 0 (orange vector), and the south pole corresponds to state 1 (blue vector). The equator corresponds to equal superposition states. $(|0> + |1>)/\sqrt{(2)}$ is at

the surface of the sphere in the +x direction. $(|0> -|1>)/\sqrt{(}2)$ is at the surface of the sphere in the -x-direction.

7 Dirac Notation

We discussed that a quantum bit qubit for short is the name given to a quantum-mechanical two-level system. The particular physical system we viewed was the spin of an electron in an atom, where the two states were spin-up and spin-down. Although qubits are realized using physical systems, it is advantageous to think of them as mathematical objects [70], because it will be easier to work with them using mathematics [71]. The approach is technology agnostic, independent of a particular physical system. In this unit, we will discuss basic concepts from linear algebra, necessary for understanding how quantum states and gates operate using Dirac notation [62–67]. At the end of this unit, we will see the concept of measurement at a very high level; more details about it will be given in the following section.

As an introductory example, consider the state space representation of four light bulbs. In this classical system, each light bulb is a classical two-state system and can be either: OFF $\rightarrow$ state 0, or ON $\rightarrow$ state 1. It means that our classical system can be in $2^4 = 16$ possible configurations. Suppose that for some reason, we decide to communicate our ATM PIN (which is 1248) to our neighbor in an extremely insecure manner using light bulbs. To do this, we would first write each digit of the ATM number using binary representation, and then turn ON/OFF the lights according to the predefined state-space definition:

1=0001 $\rightarrow$ OFF OFF OFF ON,
2=0010 $\rightarrow$ OFF OFF ON OFF,
4=0100 $\rightarrow$ OFF ON OFF OFF,
8=1000 $\rightarrow$ ON OFF OFF OFF.

To send the decimal number 1, we will keep the first three bulbs OFF and the last one ON. To send the number 2, we will keep OFF the first 2 bulbs, ON the third bulb, and OFF the fourth bulb.

Quantum bits and classical bits both represent two-state systems, as in the section, qubits have unique quantum-mechanical properties. Thus, to represent the state of a qubit, people use a standard notation called Dirac notation, or "bra"-"ket" (read: bracket) notation. The representation uses vectors, which can then be manipulated using linear algebra concepts, such as matrix multiplication. If it has been a while, the following text and links will serve as a refresher.

1. States 0 and 1 are represented as kets $|0\rangle$ and $|1\rangle$ (the ket in bra-ket), and correspond to column vectors. In particular, ket $|0\rangle$ and ket$|1\rangle$ are usually written as:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{1}$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2}$$

2. Bras (the bra in bra-ket) are the Hermitian conjugate of kets. Operationally, a Hermitian conjugate is found by transposing a vector (or matrix) and taking the complex conjugate of each element. Since the states $|0\rangle$ and $|1\rangle$, as written above, contain only real numbers, the Hermitian conjugate is equivalent to the transpose and results in the following row vectors $\langle 0| = \begin{pmatrix} 1 & 0 \end{pmatrix}, \langle 1| = \begin{pmatrix} 0 & 1 \end{pmatrix}$.

The use of the Hermitian conjugate may be more evident after the next point.

3. The inner product between two states, say $|\phi\rangle$ and $|\psi\rangle$, is written as the bracket (as in, bra-ket) $\langle\phi|\ \psi\rangle$, and in general results in a complex number. This is evident through an example. Consider the quantum state

$$\begin{aligned} |\psi\rangle &= \alpha|0\rangle + \beta|1\rangle \\ &= \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \end{aligned} \tag{3}$$

3. Taking the inner product of two vectors shows that $\langle 0|\psi\rangle = \alpha$ and$\langle 1|\psi\rangle = \beta$. In this example, the inner product of the general state$|\psi\rangle$ with each of the basis states$|0\rangle$ and $|1\rangle$ returns a number which corresponds to the "probability amplitude" of $|\psi\rangle$ in each of those states. Hence, the Hermitian conjugate is a mathematical tool used to calculate the projection of one state onto another. Finally, it should be noted that since the inner product is a complex number in general, it can be decomposed into a product of its modulus (it is magnitude) represented as $|\langle\psi|\ \phi\rangle|$and a phase factor, $e^{i\theta}$, where θ is the angle between the vectors representing the states $|\psi\rangle$ and $|\phi\rangle$.

4. The norm or "length" of the vector representing a state $|\psi\rangle$ is given by the square root of the inner product:
$|\langle\psi|\ \psi\rangle| = \sqrt{\langle\psi|\ \psi\rangle}.\sqrt{\langle\psi|\ \psi\rangle}$

5. Physical states represented in ket notation have a norm equal to one, that is $\langle\psi|\ \psi\rangle = 1$. Checking and ensuring that the norm of a state has unit value is procedure called "normalization". Since $|0\rangle$ and $|1\rangle$ are physical states with unit norm, they must also satisfy the following condition (since $1^2 = 1$):

$$|\langle 0 \mid 0\rangle| = \sqrt{\langle 0 \mid 0\rangle}, \ \ |\langle 1 \mid 1\rangle| = \sqrt{\langle 1 \mid 1\rangle}. \tag{4}$$

States $|0\rangle$ and $|1\rangle$ are orthogonal, i.e $\langle 0 \mid 1\rangle = \langle 1 \mid 0\rangle = 0$. This means there is no projection of state $|0\rangle$ on to state $|1\rangle$ and visa versa. They are independent vectors, and so there is no way to write $|0\rangle$ in terms of $|1\rangle$ or vice versa; this is called linear independence.

When a quantum state is the sum of linearly independent states, such as $|0\rangle$ and $|1\rangle$, it is said to be in a superposition state. This is the case for the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ defined above. The coefficients α and β are referred to as probability amplitudes and, as we have discussed, are in general complex numbers. The hermitian conjugate of $|\psi\rangle$ is $\langle\psi| = \alpha^* \langle 0| + \beta^* \langle 1|\,, = \begin{pmatrix} \alpha^* & \beta^* \end{pmatrix}$, where α^* and β^* are the complex conjugates of α and β respectively.

To better understand the probability amplitudes α and β of $|\psi\rangle$ represent, Let us think more about the superposition concept. A light bulb is either ON or OFF, and that is it. When we look at it, or "measure" it, we know precisely which state it had been in and continues to be in. On the other hand, while a quantum system can certainly be in the classical states $|0\rangle$ or $|1\rangle$, it can also be in a superposition state: a single state that carries aspects of both $|0\rangle$ and $|1\rangle$. What does this mean?

Let us take a qubit prepared in the superposition state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$. When this qubit is measured, quantum mechanics tells us that the qubit state will be projected onto our measurement basis. In the examples in the section, we are measuring along the z-axis, that is, the axis which represents states $|0\rangle$ and $|1\rangle$. Measurements must give us a classical result, and so any given measurement will result in one of the classical states: either state $|0\rangle$ or state $|1\rangle$. We never measure a superposition state directly. However, if we identically prepare and measure the state $|\psi\rangle$ many times, we will find that we will obtain state $|0\rangle$ with probability $|\alpha|^2$ and state $|1\rangle$ with probability $|\beta|^2$. We call the coefficients α and β probability amplitudes, since their magnitude squared will yield the probability that we measure their respective states. As shown

in the example in (3) above, these probability amplitudes can be found by projecting the vector representing state $|\psi\rangle$ onto the vectors representing the states $|0\rangle$ and $|1\rangle$.

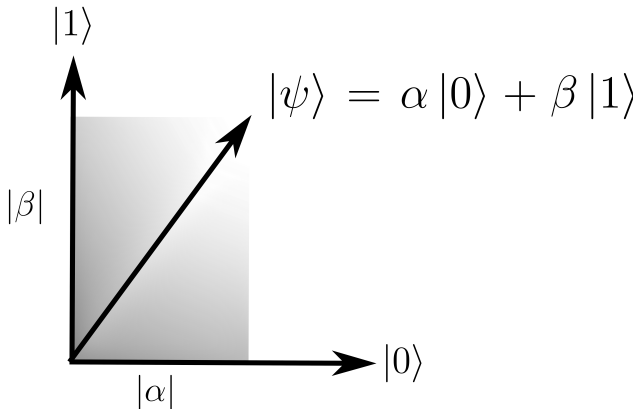


Figure 19: 0 and 1 states

This is represented in the last figure, which shows the projection of the superposition state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ on to the measurement axis corresponding to states $|0\rangle$ and $|1\rangle$. Notice that the closer the state $|\psi\rangle$ is to $|0\rangle$, the larger the projection $|\alpha|$ and thus the probability $|\alpha|^2$of measuring state$|0\rangle$. In fact, when $|\psi\rangle$ coincides with $|0\rangle$ the value of $|\alpha|$ becomes equal to 1, and we will measure state $|0\rangle$ with certainty (probability $|\alpha|^2$ equal to 1).

To summarize, a superposition state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ satisfies the normalization condition $|\alpha|^2 + |\beta|^2 = 1$ (this ensures that the probabilities of measuring all states add to unity), and the probability of measuring the states $|0\rangle$ and $|1\rangle$ are $p(0) = |\langle 0|\ \psi\rangle|^2 = |\alpha|^2andp(1) = |\langle 1|\ \psi\rangle|^2 = |\beta|^2$ respectively.

8 Bloch Sphere

The Bloch sphere is a useful tool for visualizing single-qubit states. Using Dirac notation, as we have discussed, one can write an arbitrary single-qubit state $|\psi\rangle$ as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{5}$$

where α and β are the probability amplitudes and $|\alpha|^2 + |\beta|^2 = 1$. In general, probability amplitudes are complex numbers, and can therefore always be written as the product of a real number and a complex exponential phase factor. For example, the probability amplitudes α and β can be expressed as

$$\alpha = |\alpha|(\cos\phi_\alpha + i\sin\phi_\alpha) = |\alpha|e^{i\phi_\alpha} \rightarrow ae^{i\phi_a}, \tag{6}$$

$$\beta = |\beta|(\cos\phi_\beta + i\sin\phi_\beta) = |\beta|e^{i\phi_\beta} \rightarrow be^{i\phi_b}, \tag{7}$$

where $a = |\alpha|$ and $b = |\beta|$ are the magnitudes of α and β, and $\phi_a = \phi_\alpha = \arg(\alpha)$ and $\phi_b = \phi_\beta = \arg(\beta)$ are the arguments α and β referred to as "phases". Using this convention, the single-qubit state $|\psi\rangle$ becomes

$$
\begin{aligned}
|\psi\rangle &= ae^{i\phi_a}|0\rangle + be^{i\phi_b}|1\rangle \\
&= e^{i\phi_a}\left(a|0\rangle + be^{i(\phi_b-\phi_a)}|1\rangle\right) \\
&\equiv e^{i\phi_a}\left(a|0\rangle + be^{i\phi}|1\rangle\right)
\end{aligned} \tag{8}
$$

where we have factored out the phase $e^{i\phi_a}$, referred to as the global phase, and defined a relative phase $\phi = \phi_b - \phi_a$ with ϕ defined from 0 to 2π.

We do this because it is only relative phases that play a role in quantum interference or the values of physical observables based on measurements. Any phases that sit out front may be omitted without harm.

To see this explicitly, remember that the probability of measuring the state $|\psi\rangle$ in another state $|\mu\rangle$ is given by $p(\mu) = |\langle\mu|\psi\rangle|^2$. Defining $|\mu\rangle = e^{i\phi_g}\left(c|0\rangle + de^{i\phi_r}|1\rangle\right)$, a straight forward calculation yields:

$$
\begin{aligned}
p(\mu) &= |\langle\mu|\psi\rangle|^2 \\
&= \left|\left(e^{-i\phi_g}\left(c\langle 0| + de^{-i\phi_r}\langle 1|\right)\right)\left(e^{i\phi_a}\left(a|0\rangle + be^{i\phi}|1\rangle\right)\right)\right|^2 \\
&= \left|e^{i(\phi_a-\phi_g)}\left(c\langle 0| + de^{-i\phi_r}\langle 1|\right)\left(a|0\rangle + be^{i\phi}|1\rangle\right)\right|^2 \\
&= \left|e^{i(\phi_a-\phi_g)}\left(ac + bde^{i(\phi-\phi_r)}\right)\right|^2 \\
&= \left|e^{i(\phi_a-\phi_g)}\right|^2\left|\left(ac + bde^{i(\phi-\phi_r)}\right)\right|^2 \\
&= \left|\left(ac + bde^{i(\phi-\phi_r)}\right)\right|^2.
\end{aligned} \tag{9}
$$

Two important properties of complex numbers are used in this calculation. First, for any two complex numbers w and z, it is always true that $(wz)^* = w^*z^*$. And, second, $|z|^2 = z^*z$ where $*$ denotes the complex conjugate. The latter property is useful because it means that $|e^{ix}|^2 = e^{-ix}e^{ix} = 1$ for any real x, and this leads to the removal of the global phases when calculating the measurement probability. This absence of both ϕ_a and ϕ_g from the final result, regardless of their value, indicates that the global phase has no physical relevance. Therefore, it is conventional to omit these phase factors from calculations.

Removing the global phase from $|\psi\rangle$reduces the number of variables needed to specify a state from four $(a, b, \phi_a, \phi_b)$to three (a, b, ϕ). One further degree of freedom can be removed by directly incorporating the normalization condition $|\alpha|^2 + |\beta|^2 = 1$into the coefficients. Following convention, this is performed by parameterizing a and b using the trigonometric functions

$$
\begin{aligned}
a &= \cos(\theta/2), \\
b &= \sin(\theta/2),
\end{aligned} \tag{10}
$$

where θ goes from 0 to π. The reason for selecting trigonometric functions is due to the natural geometric interpretation of the angle θ, as will be discussed in more detail below. Therefore, the state of a single qubit can be represented in complete generality by

$$
|\psi\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)e^{i\phi}|1\rangle \tag{11}
$$

which has only two free variables.

Geometrically, θ and ϕ can be mapped to a point on a sphere, referred to as the "Bloch sphere", using a spherical coordinate system. The angle θ is called the "polar angle", and it is measured from the positive z-axis to the Bloch vector representing the state $|\psi\rangle$. The angle ϕ is called the "azimuthal angle," It is measured from the positive x-axis to the projection of state $|\psi\rangle$ onto the x-y plane (see the figure for the correct orientation).
Let us consider the polar and azimuthal angles for a few standard quantum states.
First, Let us consider the "poles" where the z axis meets the surface of the sphere, corresponding to $\theta = 0$ and $\theta = \pi$ and representing the states $|\psi\rangle = |0\rangle$ and $|\psi\rangle = |1\rangle$ respectively. Note that for $\theta = \pi$, corresponding to $|\psi\rangle = e^{i\phi}|1\rangle \rightarrow |1\rangle$, the angle ϕ becomes a global phase factor and is therefore not needed.
$-(\theta = 0, \phi) \rightarrow |0\rangle$: this is the point where the z axis meets the north pole in the positive-z direction ($z = +1$).
$-(\theta = \pi, \phi) \rightarrow |1\rangle$: this is the point where the z axis meets the south pole in the negative-z direction ($z = -1$).
Next, Let us consider the equal superposition states $|\psi\rangle = |0\rangle + e^{i\phi}|1\rangle$ on the "equator" in the x-y plane. These states all share $\theta = \pi/2$, and are uniquely identified on the equator by the angle ϕ. Let us further look at four specific examples as we work our way around the equator:
$-(\theta = \pi/2, \phi = 0) \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$: this is the point where the x axis meets the equator in the positive-x direction ($x = +1$).
$-(\theta = \pi/2, \phi = \pi/2) \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$: this is the point where the y axis meets the equator in the positive-y direction ($y = +1$).
$-(\theta = \pi/2, \phi = \pi) \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$: this is the point where the x axis meets the equator in the negative-x direction ($x = -1$).
$-(\theta = \pi/2, \phi = 3\pi/2) \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$: this is the point where the y axis meets the equator in the negative-y direction ($y = -1$).

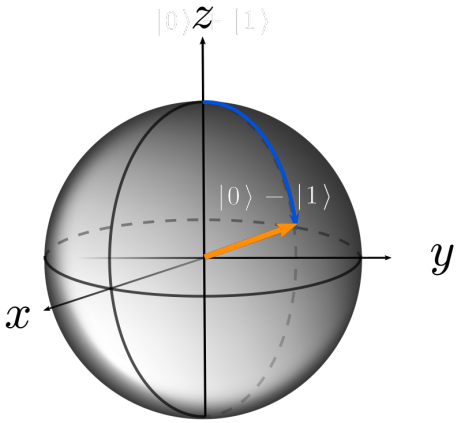


Figure 20: Bloch Sphere

9 Quantum Parallel and Interference

How does a quantum computer process information? How can quantum logic operations lead to quantum advantage? In this section, we will be introduced to two quantum-mechanical phenomena quantum parallelism and quantum interference that are fundamental to quantum information

processing [1, 72, 73]. We will be given visceral, intuitive examples that allow us to "see" directly how the quantum versions of parallelism and interference efficiently manipulate the weighting coefficients probability amplitudes within a quantum superposition state, process quantum information.

Quantum parallelism and quantum interference are what make a quantum computer different than a classical computer. However, what is these quantum effects? Furthermore, how do they work in a quantum computer? To gain some insight, we will consider an example of a small quantum computer with three qubits. Here we have three atoms. Each of which has an electron with a spin. Moreover, each of these spins can either be pointed up, which We will call spin-up, or pointed down, which We will call spin-down. We will use these three electron spins as the qubits. There are eight different spin combinations that we can have, from all three pointed up, to all three pointed down. We place these qubits in a single quantum superposition state, comprising all eight of these spin configurations. It takes eight complex numbers, C1 through C8, to specify the weighting of each of these components. The superposition state can then be represented as a state register with all eight spin configurations and their respective coefficients. Let us now imagine that we want to perform an operation that flips the spin of Atom 1. We can do this by applying an electromagnetic pulse with the right strength and the right duration, such that it rotates the spin of Atom 1 by 180 degrees. It is called a π pulse. Furthermore, it acts to flip the spin. Spin-up rotates to spin-down, and spin-down rotates to spin-up. So, when we flip the spin in Atom 1, it acts to flip the spin on each of the spins in the configurations that make up the superposition state. For example, the coefficients C1 through C4, associated initially with spin-up in Atom 1, are now associated with spin-down. Similarly, the coefficients C5 through C8, associated initially with spin-down, are now associated with spin-up. It happens simultaneously across all the spin configurations that make up the quantum superposition state, even though we are performing only a single operation on a single qubit, and this is an example of quantum parallelism. Let us now look at quantum interference between these states. In this case, we will address Atom 3. We will consider a type of pulse called a $\pi/2$ pulse. A $\pi/2$ pulse does takes a spin-up and rotates it to a superposition state of up plus down. If we have taken quantum mechanics before, we remember that there is a normalization factor 1 over square root of 2 sitting in front. It maintains the length of the vector on the Bloch sphere. However, here we will omit those factors as they are not crucial for this discussion. So, a $\pi/2$ pulse rotates a spin-up to a superposition of up plus down. We can visualize that on the Bloch sphere. The spin-up is pointed at the North Pole, and we rotate it to the equator by rotating it $\pi/2$, or 90 degrees. We will associate the direction of the vector that it is now pointing with a plus sign. Thus, the superposition state is up plus down. In the state space, for the moment, let us just look at coefficient C5. C5 is associated initially with a spin-up on Atom 3. After the $\pi/2$ rotation, it is now associated with both a spin-up and a spin-down. Next, let us look at what happens to spin-down. A $\pi/2$ pulse will rotate a spin-down pointed at the South Pole up to the equator, but now in the opposite direction. We will associate this new direction with a minus sign. Thus, the resulting superposition state is up to minus down. In the state space, the coefficient C6, associated initially with a spin-down in Atom 3, is now associated with both up and down, but with a minus sign for the spin-down. So, we find plus 6 for spin-up and minus 6 for spin-down. So, what does it all mean? Well, if C5 equals C6, for example, then C5 minus C6 is zero. Moreover, there is no longer any weighting to the up-up-down state. It is an example of destructive quantum interference. At the same time, there is constructive quantum interference that increases the weighting of the state with C5 plus C6. Furthermore, this is also an example of quantum parallelism because this quantum interference process also happens simultaneously to all the other states in the register. So, quantum parallelism and quantum interference form the foundation for how a quantum computer processes information. As, with even a single gate

operation, quantum parallelism and quantum interference allow us to simultaneously manipulate and change the values of the many weighting coefficients that comprise a superposition state. At a fundamental level, we can efficiently implement quantum algorithms on a quantum computer [74].

Quantum parallelism and quantum interference are two quantum mechanical concepts that distinguish a quantum computer from a classical computer.

Quantum Parallelism:

Let us revisit the concept of quantum parallelism introduced in the section. We looked at three qubits; here, Let us look at two qubits.

Suppose we have two qubits, realized by two separate electrons and their associated spins. Each electron spin can either be pointed up the "spin-up" state $|\uparrow\rangle$ or it can be pointed down, the "spin-down state$|\downarrow\rangle$." As qubits, they can also be in superpositions states of $|\uparrow\rangle$ and $|\downarrow\rangle$.

A system of $N = 2$ spins can be found in $2^N = 4$ possible spin configurations. An equal superposition of these configurations results in four complex probability amplitudes (weighting factors) c_i:

$$|\Psi\rangle = c_1|\downarrow\downarrow\rangle + c_2|\downarrow\uparrow\rangle + c_3|\uparrow\downarrow\rangle + c_4|\uparrow\uparrow\rangle \tag{12}$$

A π-pulse applied to the first qubit (left-most spin in the bra-ket) will flip its spin. This rotation is implemented using an electromagnetic pulse with a precise amplitude and duration such that it rotates the spin by 180 degrees.

$$|\Psi\rangle \xrightarrow{\pi\text{-pulse on left spin}} |\Psi'\rangle = c_3|\downarrow\downarrow\rangle + c_4|\downarrow\uparrow\rangle + c_1|\uparrow\downarrow\rangle + c_2|\uparrow\uparrow\rangle \tag{13}$$

As we can see, a single π-pulse on a single qubit effectively shuffles the individual probability amplitudes amongst all of the $2^N = 4$ spin configurations making up a quantum superposition state. It is an example of quantum parallelism.

Quantum Interference:

Let us now explore what happens when a $\pi/2$-pulse applied to the second qubit. There are two cases to consider:

If the second qubit is in the spin-up state, a $\pi/2$ pulse applied along the y-axis will rotate the spin from the north pole down to the equator. This aligns the spin with the +x direction, creating the equal superposition state $(|\uparrow\rangle + |\downarrow\rangle)/\sqrt{2}$ with a " + " sign.

If the second qubit is instead in the spin-down state, a $\pi/2$ pulse applied along the y-axis will rotate the spin in the same counter-clockwise direction, bringing it from the south pole up to the equator. This aligns the spin in the x-direction, creating the equal superposition state $(|\uparrow\rangle - |\downarrow\rangle)/\sqrt{2}$, this time with a corresponding " − " sign.

The resulting state is:

$$\begin{aligned} |\Psi\rangle \xrightarrow{\pi/2\text{-pulse}} |\Psi''\rangle = & \\ & \frac{1}{\sqrt{2}}(c_2 - c_1)|\downarrow\downarrow\rangle + \frac{1}{\sqrt{2}}(c_2 + c_1)|\downarrow\uparrow\rangle \\ & + \frac{1}{\sqrt{2}}(c_4 - c_3)|\uparrow\downarrow\rangle + \frac{1}{\sqrt{2}}(c_4 + c_3)|\uparrow\uparrow\rangle \end{aligned} \tag{14}$$

The probability amplitudes now add and subtract one another. Suppose there are two coefficients with equal values, for example, $c_3 = c_4$. In this case, there is a complete cancellation of the probability amplitude for $|\uparrow\downarrow\rangle$. Such a reduction of the probability amplitude is called "destructive quantum interference." On the other hand, there is a doubling of the probability amplitude in

front of $|\uparrow\uparrow\rangle$. Such an enhancement of the probability amplitude is called "constructive quantum interference." Furthermore, since the constructive and destructive quantum interference happens across the entire state space, this is also an example of quantum parallelism.

10 Quantum Gates

What are quantum logic gates? How are they visualized? In this section, we will discuss the single-qubit and two-qubit gates. We will see an example of each X gate and the CNOT gate and contrast them with their classical analogs. A small set of such single-qubit and two-qubit gates forms a universal gate set that can be used to implement any algorithm on a circuit-model quantum computer.

Classical computers can perform arbitrary Boolean logic with a small set of single-bit and two-bit gates. For example, the NOT gate, combined with the AND gate, is considered universal in that it can, in principle, implement any classical algorithm that uses binary logic. Similarly, quantum algorithms can be run on quantum computers using a small, universal set of single and two-qubit gates. Let us begin with an example of a single-qubit gate called the X-gate and its classical analog, the NOT gate. A NOT gate takes one bit as its input, and it inverts it. For example, a 0 at the input is inverted to state 1, and a 1 at the input is inverted to state 0. The quantum analog of this gate is called the X-gate, which takes a quantum state as its input, in this case, state 0, and rotates it to state 1 or it takes state 1 and rotates it to state 0. We can represent this operation on the Bloch Sphere with state 0 at the North Pole and state 1 at the South Pole. We see an envelope of the pulse that we are using to drive the X-gate. We call this a π pulse because it will rotate the Bloch vector representing the qubit's state from the North Pole to the South Pole, a rotation of 180 degrees. The red arrow that comes in and out of the Bloch Sphere screen represents the envelope of the pulse that we are using to drive this operation. Moreover, visually, much like the spokes of an umbrella will rotate around the umbrella's central axis when we twist its handle. The Bloch vector rotates around the axis to which we are applying the pulse. In this case, since this pulse is applied along the x-axis of the Bloch Sphere, therefore we call the operation an X-gate. Now, as we show it here, we are rotating from the North Pole to the South Pole from state 0 to state 1. In this configuration, this is simply a classical operation, but the X-gate can do much more. The X-gate can take as its input any superposition state, that is any starting point on the Bloch Sphere and rotate it around the x-axis by 180 degrees. It is a quantum mechanical operation. The qubit starts in a superposition state, and it ends in a superposition state. What the X-gate essentially does is take the input quantum state, α 0 plus β 1, and swaps the coefficients to generate an output state, β 0 plus α 1. The X-gate is one of a handful of standard single-qubit gates that rotate the qubit state around a few different axes on the Bloch Sphere. Let us now consider an example of a two-qubit gate, and the controlled-NOT gate, or CNOT-gate, and its classical analog, the exclusive OR, or the XOR-gate. XOR takes two bits as inputs, bit x, and y. We will call bit x the control bit and bit y the target bit. The truth table for XOR shows the output states of all four possible input state combinations. For example, when the control bit x is in its 0 states, the target bit y, whether a 0 or a 1, remains unchanged, and its value is just passed to the output. However, when control bit x is set to state 1, the target bit y is inverted. 0 becomes 1, and 1 becomes 0. The quantum analog of this gate is the CNOT-gate, and it takes as inputs qubit x and qubit y. Again, we will call x the control bit and y the target bit. When qubit x is in state 0, qubit y remains unchanged. When qubit x is in state 1, qubit y undergoes a π rotation, a rotation of 180 degrees. It means that the rotation of qubit y depends on the state of qubit x. We can consider an interesting example where x, the control qubit, is in an equal superposition of 0 and 1, and y, the target qubit, is in state 0.

To determine the output, let us take it one piece at a time. When x is in state 0, y remains unchanged. When x is in state 1, the qubit y undergoes a rotation of 180 degrees that flips state 0 to state 1. The resulting output state, 0, 0, plus 1, 1, is a fascinating state because it cannot be factorized into an x component cross a y component. This type of state is entangled, and an entangled state is a manifestly quantum mechanical state. Universal quantum computation can be built from a small subset of these types of single and two-qubit gates [75]. A universal gate set allows us to perform any type of quantum algorithm on a gate model quantum computer [76].

Classical computers perform arbitrary Boolean logic with a small set of single-bit and two-bit logic gates. The NOT gate (a single-bit gate) in conjunction with the AND gate (a two-bit gate) are together one example of a universal gate set. Such a universal gate set can, in principle, implement any arbitrary classical algorithm based on boolean logic.

The quantum analogue of a NOT gate is the X-gate. A NOT gate inverts its input (NOT $0 \rightarrow 1$, and NOT $1 \rightarrow 0$) . Similarly, the X-gate would swap states $|0\rangle$ and $|1\rangle$. The swap can be visualized on the Bloch sphere as a 180-degree rotation around the x-axis (this is why it is called an "X-gate"). Because the rotation is 180-degrees, the signal we send to the qubit in order to perform an X-gate is referred to as a π-pulse. In general, the X-gate can be applied to any arbitrary quantum superposition state, and it acts to swap the probability amplitudes on the states $|0\rangle$ and $|1\rangle$:

$$\frac{1}{\sqrt{2}}(\alpha|0\rangle + \beta|1\rangle) \xrightarrow{\text{X gate}} \frac{1}{\sqrt{2}}(\beta|0\rangle + \alpha|1\rangle) \tag{15}$$

In addition to the X-gate, one may rotate the qubit state around the y-axis or the z-axis. A π-rotation around the y-axis is called a Y-gate, and a π-rotation around the z-axis is called a Z-gate.

Not all classical gates have a direct quantum analog. It is because quantum circuits must be reversible [77]. In a reversible circuit, one can precisely reconstruct the input state(s) given the output state(s). For example, a NOT gate is reversible, because, given the output 0/1, we know the input was 1/0. However, the AND gate is not reversible, because, for example, given the output 0, we cannot tell if the input had been 00, 01, or 10. The reason quantum circuits have to be reversible has to do with the fact that coherent quantum states ideally always undergo unitary evolution. So a quantum gate can be "undone" by applying the inverse of that unitary evolution.

There is a quantum analog to the classical exclusive-OR gate (abbreviated XOR), called the controlled-NOT gate (abbreviated CNOT). The CNOT gate is a "conditional gate" comprising two qubits: a "control qubit" and a "target qubit". When the control qubit is in state $|0\rangle$, the target qubit remains unchanged. However, when the control qubit is in state $|1\rangle$, an X-gate is applied to the target qubit: it undergoes a π-rotation around the x-axis of the Bloch sphere.

Consider an interesting example where the control qubit is in an equal superposition of $|0\rangle$ and$|1\rangle$, and the target qubit is in state$|0\rangle$:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \xrightarrow{\text{CNOT}} \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle). \tag{16}$$

(Note: We are here introducing "tensor notation," [78] where $|x\rangle \otimes |y\rangle$ indicates a two-qubit state with the first qubit in-state $|x\rangle$, and the second qubit in-state $|y\rangle$. Occasionally, we may drop the $\otimes$ and write a two-quit state as simply$|x\rangle|y\rangle$.) The resulting output state is remarkable because it can be no longer separable into two single-qubit components such as $(\ldots)_x \otimes (\ldots)_y$. It is known as an entangled state, and it is a manifestly quantum mechanical state.

Universal quantum computation can be built from a small subset of these types of single and two-qubit gates. A universal gate set enables us to perform any type of algorithm quantum or classical on a gate model quantum computer. However, universality does not imply quantum

advantage. Many algorithms can be implemented on a quantum computer in principle but feature no quantum advantage.

11 List of Classical Gates

Classical computing is performed using a universal set of Boolean logic gates. The table below introduces several examples of single-bit and two-bit gates. Subsets of these gates form a universal gate set. For example, the NOT gate and the AND gate together can implement any Boolean logic function.

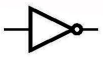
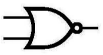
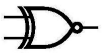
GATE		CIRCUIT REPRESENTATION	TRUTH TABLE
NOT	The output is 1 when the input is 0 and 0 when the input is 1.		Input: 0, 1 — Output: 1, 0
AND	The output is 1 only when both inputs are 1, otherwise the output is 0.		Input: 0 0, 0 1, 1 0, 1 1 — Output: 0, 0, 0, 1
OR	The output is 0 only when both inputs are 0, otherwise the output is 1.		Input: 0 0, 0 1, 1 0, 1 1 — Output: 0, 1, 1, 1
NAND	The output is 0 only when both inputs are 1, otherwise the output is 1.		Input: 0 0, 0 1, 1 0, 1 1 — Output: 1, 1, 1, 0
NOR	The output is 1 only when both inputs are 0, otherwise the output is 0.		Input: 0 0, 0 1, 1 0, 1 1 — Output: 1, 0, 0, 0
XOR	The output is 1 only when the two inputs have different value, otherwise the output is 0.		Input: 0 0, 0 1, 1 0, 1 1 — Output: 0, 1, 1, 0
XNOR	The output is 1 only when the two inputs have the same value, otherwise the output is 0.		Input: 0 0, 0 1, 1 0, 1 1 — Output: 1, 0, 0, 1

Figure 21: Classical Gates

The table above comprises a set of single-bit and two-bit Boolean logic gates. Logic gates perform Boolean functions on the inputs to yield output. The second column contains the graphical symbols used to represent the gates introduced in column one, and the third column

contains the truth tables for these gates. Truth tables comprise all possible combinations of input states, 2^1 for single-bit gates and $2^2 = 4$ for two-bit gates, and the corresponding output state.

The most basic Boolean logic gate is the NOT gate, a single-bit gate, which inverts the input bit, logic $\rightarrow$ logic 1, and vice versa. Single-bit gates alone are insufficient to form a universal gate set; universal computation requires some form of logical operation on multiple bits, such as a two-bit gate.

The table presents six two-bit Boolean logic gates. The AND gate outputs a logic 1 only if the two inputs are both in the logic 1 state. A NAND gate is an inverted AND gate, that is, an AND gate followed by a NOT gate. The OR gate outputs logic state 1 if at least one input bit is 1. The NOR gate is an OR followed by a NOT gate. The exclusive OR, the XOR gate, outputs a logic 1 only if the two input bits differ. An inverted XOR gate, an XOR gate followed by a NOT gate, is called an XNOR gate and outputs a logic 1 if the two input bits are the same.

As these six two-bit gates foretell, gates are not unique and can be generated using combinations of other single-bit and two-bit gates.

12 Single-Qubit Gates

Universal quantum computing is performed using a small set of single-qubit and two-qubit gates. The table below introduces several single-qubit gates used in the circuit model of quantum computation.

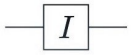
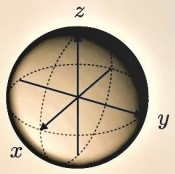
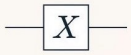
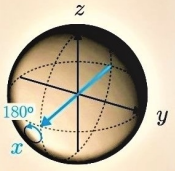
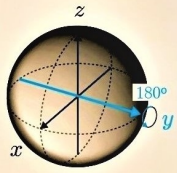
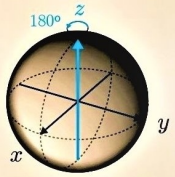
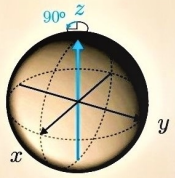
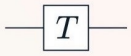
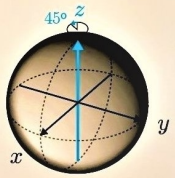
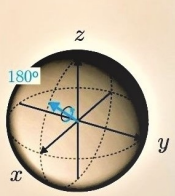
GATE	CIRCUIT REPRESENTATION	MATRIX REPRESENTATION	TRUTH TABLE	BLOCH SPHERE
I Identity-gate: no rotation is performed.		$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	Input / Output $\|0\rangle$ / $\|0\rangle$ $\|1\rangle$ / $\|1\rangle$	
X gate: rotates the qubit state by π radians (180°) about the x-axis.		$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	Input / Output $\|0\rangle$ / $\|1\rangle$ $\|1\rangle$ / $\|0\rangle$	
Y gate: rotates the qubit state by π radians (180°) about the y-axis.		$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	Input / Output $\|0\rangle$ / $i\,\|1\rangle$ $\|1\rangle$ / $-i\,\|0\rangle$	
Z gate: rotates the qubit state by π radians (180°) about the z-axis.		$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	Input / Output $\|0\rangle$ / $\|0\rangle$ $\|1\rangle$ / $-\|1\rangle$	
S gate: rotates the qubit state by $\frac{\pi}{2}$ radians (90°) about the z-axis.		$S = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{pmatrix}$	Input / Output $\|0\rangle$ / $\|0\rangle$ $\|1\rangle$ / $e^{i\frac{\pi}{2}}\|1\rangle$	
T gate: rotates the qubit state by $\frac{\pi}{4}$ radians (45°) about the z-axis.		$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$	Input / Output $\|0\rangle$ / $\|0\rangle$ $\|1\rangle$ / $e^{i\frac{\pi}{4}}\|1\rangle$	
H gate: rotates the qubit state by π radians (180°) about an axis diagonal in the x-z plane. This is equivalent to an X-gate followed by a $\frac{\pi}{2}$ rotation about the y-axis.		$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	Input / Output $\|0\rangle$ / $\frac{\|0\rangle + \|1\rangle}{\sqrt{2}}$ $\|1\rangle$ / $\frac{\|0\rangle - \|1\rangle}{\sqrt{2}}$	

Figure 22: Single Qubit

The first column of the table introduces the gate and a short definition of its action. The second column shows the graphical representation of each gate, and the third column shows the corresponding matrix representation. The fourth column provides the truth table for the input states $|0\rangle$ and $|1\rangle$. The last column indicates the rotation on the Bloch sphere that the gate performs. We will walk through the I, X, Z, and Y gates to provide a sense of how single-qubit gates work, in general.

The first gate is "identity", an operation that does not alter the input state and is used to represent a lossless quantum channel [79]. It is represented by the identity matrix.

$$I|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle \tag{17}$$

$$I|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \tag{18}$$

There are two items to note here. First, we use "operator notation" to mathematically represent the application of a gate operation on a qubit state. For example, the identity operation applied to qubit state $|0\rangle$ is written: $I|0\rangle$. we may sometimes see a "hat" added to an operator, e.g., $\hat{I}$, to make its role clear. The corresponding matrix and vector can then replace the operator and state, respectively, to calculate the gate's action. Second, we note that the identity operator leaves the state of the qubit unchanged.

The "X-gate" can be visualized as performing a rotation around the x-axis on the Bloch sphere. As discussed previously, the X-gate is the quantum analog of the classical NOT gate. See below how the X-gate acts on states $|0\rangle$ and $|1\rangle$:

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \tag{19}$$

$$X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle. \tag{20}$$

The X-gate is said to perform a "bit flip", because the qubit states $|0\rangle$ and $|1\rangle$ are "flipped" to$|1\rangle$and $|0\rangle$ respectively. As in the section, more generally, the X-gate swaps the probability amplitudes in a quantum state: X $(\alpha|0\rangle + \beta|1\rangle) \rightarrow \beta|0\rangle + \alpha|1\rangle$.

The Z-gate "adds a phase" of -1 to the $|1\rangle$ state and leaves $|0\rangle$ unchanged.

$$Z|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle \tag{21}$$

$$Z|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = -\begin{pmatrix} 0 \\ 1 \end{pmatrix} = -|1\rangle \tag{22}$$

Although the net result is to multiple that state $|1\rangle$ by -1, the language "adds a phase" is often used. The terminology arises from the fact that the Z-gate will rotate a state about the Z-axis by π radians (180 degrees). Phases are additive, and so applying a π phase shift to a starting phase ϕ_0 results in $e^{i(\phi_0+\pi)} = e^{i\phi}e^{i\pi} = -e^{i\phi}$. Thus, "adding a phase" in the exponent leads to a factor $e^{i\pi} = -1$.

The Y-gate is a rotation around the y axis by π radians (180 degrees). This operation may also be written in terms of the X-gate, the Z-gate, and a global phase. That is, the Y-gate may be viewed as performing both a bit flip and a phase flip, as well as introducing a global phase factor of i:

$$Y = iXZ = i \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{23}$$

Applying the Y-gate to qubit states $|0\rangle$ and $|1\rangle$ yields:

$$Y|0\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ i \end{pmatrix} = i|1\rangle \tag{24}$$

$$Y|1\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = -\begin{pmatrix} i \\ 0 \end{pmatrix} = -i|0\rangle \tag{25}$$

Rotations around the z-axis by an arbitrary angle ϕ causes state $|1\rangle$ to acquire a phase $e^{i\phi}$ and leaves state $|0\rangle$ unaffected. Rotations around the z-axis by the specific angles $\pi/2$ and $\pi/4$ are referred to as the S-gate and T-gate, respectively.

Finally, the Hadamard gate, or H-gate for short, induces a π rotation around an axis exactly in between the x-axis and z-axis of the Bloch sphere. For example, it takes states located on the z-axis and rotates them on to the x-axis, creating equal superposition states: $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle$ and $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle$.

13 Two-Qubit Gates

Two-qubit gates are unitary quantum operations on two qubits. When these gates cannot be written as the product of two single-qubit gates, they are called "entangling gates". They may also be "controlled" or "conditional" gates. The word controlled arises from the conditional implementation of these gates, in which a unitary operation U is conditionally applied to a target qubit depending on the state of the control qubit. A two-qubit controlled gate, denoted here as $U^c_{A,B}$,with the first qubit as control (A), and the second qubit as target (B) can be written in Dirac notation as

$$U^c_{A,B} = |0\rangle\langle 0|_A \otimes I_B + |1\rangle\langle 1|_A \otimes U_B, \tag{26}$$

Where I_B and U_B are single-qubit operations applied to qubit B, depending on the state of qubit A. If qubit A is in state $|0\rangle$, then the identity operation is applied to qubit B, and its state remains unchanged. However, when qubit A is in state $|1\rangle$, the unitary single-qubit operation U_B is applied to qubit B:

$$|0\rangle_A \otimes |0\rangle_B \rightarrow |0\rangle_A \otimes I_B|0\rangle_B, \tag{27}$$

$$|0\rangle_A \otimes |1\rangle_B \rightarrow |0\rangle_A \otimes I_B|1\rangle_B, \tag{28}$$

$$|1\rangle_A \otimes |0\rangle_B \rightarrow |1\rangle_A \otimes U_B|0\rangle_B, \tag{29}$$

$$|1\rangle_A \otimes |1\rangle_B \rightarrow |1\rangle_A \otimes U_B|1\rangle_B. \tag{30}$$

The specific operation U_B depends on the type of two-qubit gate.

In quantum circuits, a controlled gate is indicated by a vertical line that connects two qubits: a solid black circle indicates the control qubit, and the target qubit is indicated by a symbol representing a single-qubit unitary operation U that is conditionally applied, as shown in the circuit below.

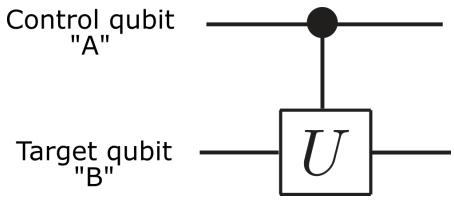


Figure 23: control unitary AB

Note that the roles of target and control qubit may be swapped [80]. A controlled unitary, $U^c_{B,A}$, with the B as the control qubit and A as the target qubit is denoted:

$$U^c_{B,A} = I_A \otimes |0\rangle\langle 0|_B + U_A \otimes |1\rangle\langle 1|_B, \tag{31}$$

Where I_A represents the identity gate operating on qubit A, and U_A is the single-qubit gate that operates conditionally on qubit A. The quantum circuit of this controlled gate is shown in the following figure.

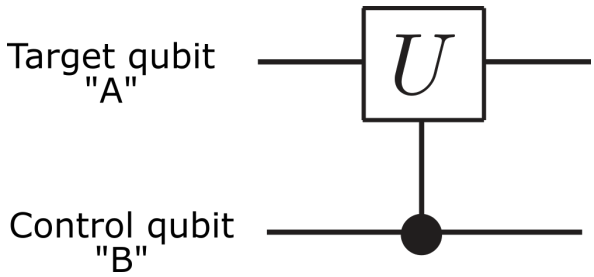


Figure 24: control unitary BA

Two common examples of conditional gates are the controlled-phase gate referred to as the CZ-gate and the controlled-NOT gate most commonly referred to as the CNOT-gate (it could also be called a CX-gate). Both gates take their name from the single-qubit operation U that is conditionally applied to the target qubit.

In the case of the controlled-phase gate, the conditional single-qubit operation is the Z-gate. The word "phase" refers to the phase factor ($\exp\{i\pi\} = -1$) that results from an application of the Z-gate on state $|1\rangle$,

$$\begin{aligned} Z|0\rangle &= |0\rangle, \\ Z|1\rangle &= -|1\rangle. \end{aligned} \tag{32}$$

If qubit A is the control and B is the target, the controlled-phase gate can be written in Dirac notation as

$$Z^c_{A,B} = |0\rangle\langle 0|_A \otimes I_B + |1\rangle\langle 1|_A \otimes Z_B, \tag{33}$$

and its quantum circuit is given by

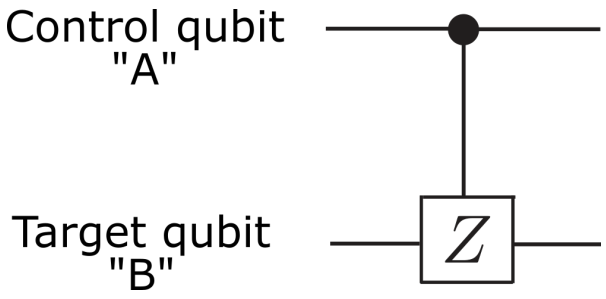


Figure 25: CZ gate

In the case of the CNOT gate, the single-qubit operation is the X-gate. The word "NOT" comes from the effect of the X-gate on the state $|0\rangle$ and $|1\rangle$,

$$\begin{aligned} X|0\rangle &= |1\rangle, \\ X|1\rangle &= |0\rangle. \end{aligned} \tag{34}$$

If qubit A is the control and B is the target, CNOT operator and its corresponding quantum circuit are:

$$CNOT_{A,B} = |0\rangle\langle 0|_A \otimes I_B + |1\rangle\langle 1|_A \otimes X_B. \tag{35}$$

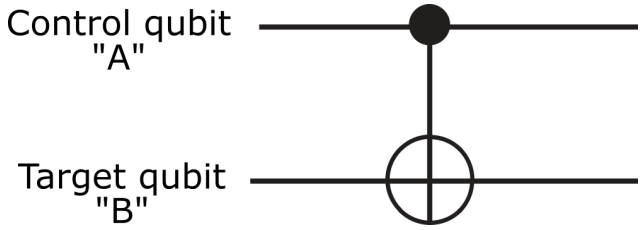


Figure 26: CNOT AB

Similarly, if the role of the control and target qubits are interchanged, the CNOT operator and its quantum circuit are:

$$CNOT_{B,A} = I_A \otimes |0\rangle\langle 0|_B + X_A \otimes |1\rangle\langle 1|_B. \tag{36}$$

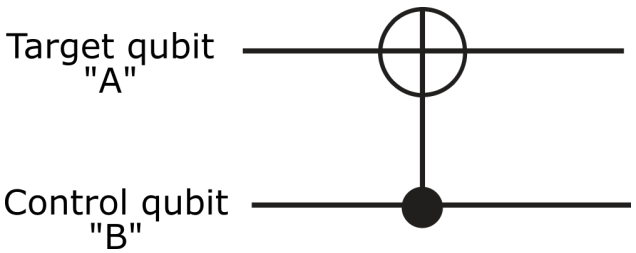


Figure 27: CNOT BA

A CNOT gate can be implemented using a Z-gate by applying a Hadamard gate, H, before and after a Z-gate, since X=HZH. The figures below represent this identity.

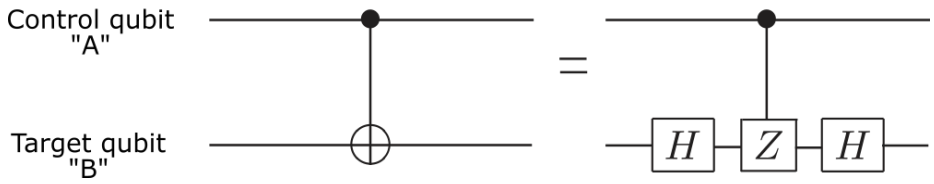


Figure 28: CNOT CZ Identity

The table below shows a summary of the two controlled-gates CZ and CNOT. Note that the sub-indexes A and B in $cZ_{A,B}$ are usually omitted. In their absence, it is generally assumed that the first qubit is the "control qubit," and the second qubit is the "target qubit." It is not only valid for the controlled-phase gate (CZ) but also the controlled-unitary gate (cU) and the controlled-NOT gate (CNOT).

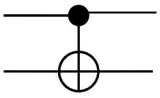
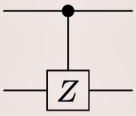
GATE	CIRCUIT REPRESENTATION	MATRIX REPRESENTATION	TRUTH TABLE
Controlled-NOT gate: apply an X-gate to the target qubit if the control qubit is in state $\|1\rangle$		$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	Input: $\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $\|11\rangle$ — Output: $\|00\rangle$ $\|01\rangle$ $\|11\rangle$ $\|10\rangle$
Controlled-phase gate: apply a Z-gate to the target qubit if the control qubit is in state $\|1\rangle$		$cZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$	Input: $\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $\|11\rangle$ — Output: $\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $-\|11\rangle$

Figure 29: Two Qubit

14 How Universal Algorithm Works

How is a universal quantum algorithm implemented on a quantum computer? In this section, we will have an intuitive introduction to how a universal quantum computer uses single-qubit and two-qubit gates to implement an algorithm.

Let us get an intuitive picture of how a quantum algorithm works. A universal quantum algorithm is built from a small set of single and coupled qubit gates. The input to a quantum computer is a massive superposition state, in general. Then, we apply the types of single-qubit gates that we have just discussed. The single-qubit gate operates on all the states simultaneously through quantum parallelism. It is followed by quantum interference, which modifies the coefficients in front of those states. We will also apply the types of coupled qubit gates that we have discussed, for example, the rotation of qubit y-depends on the state of qubit-x. Again, through quantum parallelism, these operations apply to the entire state space, followed by quantum interference, which will again modify the coefficients. Moreover, the goal of an algorithm designer

is to ensure that, by the end of the algorithm, one of these coefficients has a value that is unity or very close to unity. It corresponds to the state that gives us the answer to the problem. It is a crucial point because we need to measure the output from the quantum computer to find the answer to the problem. The measurement process itself, in a quantum system, is probabilistic and leads to a classical result. So, although the output of the quantum computer is a massive superposition state, in general, the measurement process will project those qubits on to only one of the classical states that makes up the superposition. The probability that we get any given state corresponds to the magnitude squared of that coefficient. So, having a coefficient that is close to one or unity will give us a very high probability, the correct answer.

A quantum algorithm comprises a sequence of single-qubit and two-qubit gates. Quantum parallelism and quantum interference are used by the algorithm designer to take an input state typically a massive superposition state and, in a step-by-step manner, modify the weighting coefficients (probability amplitudes) until the quantum mechanical state evolves into an output state that encodes the answer to the problem. Since a projective quantum measurement will yield a single, classical state, it is imperative that the algorithm results in a final state with a probability amplitude near unity such that near-unity probability, the measurement will lead to the correct answer.

15 Universal Quantum Algorithm

What types of algorithms can be run on a universal quantum computer? What are the advantages, requirements, and challenges associated with universal quantum computation? In this section, we present a high-level introduction to universal quantum algorithms and quantum computation.

There are a couple of different kinds of quantum computers. The most general is a Universal Fault-Tolerant Quantum Computer. The gate model algorithm that we discussed earlier runs on this type of computer. Building a Universal, Fault-Tolerant Quantum Computer [81], we think, is one of the most significant scientific and technological endeavors today. Such a machine, when we build it, will have all the power that quantum has to offer. A Universal Fault-Tolerant Quantum Computer is a holy grail, in some sense, as it means we will be able to program, implement, and reliably run complex, large-scale quantum algorithms. We use a small set of single and two-qubit gates to implement universal quantum computation. In principle, this computer can run any type of algorithm-quantum or classical. However, there are only specific algorithms that are known to exhibit a quantum advantage. Something that We will refer to as quantum speed up. One example is Shor's Factoring Algorithm. Shor's algorithm is used to break public-key cryptography. For example, RSA encryption a scheme that is used for secure communications [82]. The security of RSA encryption is based on the premise that factoring a large integer number into two smaller prime numbers is a challenging computation for a classical computer to perform. The computational complexity scales poorly with the length of the encryption key. So, for example, if we are dissatisfied with our current security level, just double the length of our public key, and it will become exponentially harder for a classical computer to break that encryption scheme. In contrast, Shor's algorithm can perform the same tasks, with only a marginal increase in difficulty, as the key's length has increased. Other examples include Grover's algorithm for searching an unsorted database [83], or sampling solutions to linear equations. For these types of algorithms, quantum speedup of some degree exists over known classical algorithms when run on a universal quantum computer [77]. The actual amount of speed up for these algorithms we will discuss later in the section. The advantage of a Gate Model Quantum computer is that it is universal. In principle, it can run any algorithm. Now, in practice, not all

algorithms will exhibit a quantum speedup, but many will present. The challenge lies in making a computer that is both large enough and operates for long enough to complete an algorithm and obtain the answer. Even though the fundamental building blocks, the qubits themselves, are prone to errors. In a quantum computer, each operation has a probability of being noisy, That is of adding an error to the computation. We need a way to combat the build-up of all these errors so that the output we get out of the algorithm, or the computation is reliable. We will see later in the section; such errors can be overcome by something called Quantum Error Correction. However, it will require additional resources to implement it.

The power of a universal computer is that it can implement any algorithm that can be expressed in terms of a circuit comprised of logical gates. A universal, fault-tolerant quantum computer will enable people to program, implement, and reliably run arbitrarily complex quantum algorithms. In fact, a universal quantum computer can run any type of algorithm quantum or classical. Determining when it is advantageous to implement an algorithm on a quantum computer, as opposed to a classical computer, is one of the principal aims of quantum information science. The study of how efficiently a computer, whether a classical or a quantum computer, can solve a given problem results in the formal segregation of algorithms into what are called "computational complexity classes." This efficiency is determined based on how the computational resources needed to solve a problem, grow, or "scale" with the size of the problem instance. For example, how the size of the memory or the number of computational steps scales with the problem size.

Two important computational complexity classes are P and NP [84, 85]. An example of a problem in P is the multiplication of two numbers of length n, which requires n^2 time steps to complete. For example, this means multiplying the binary numbers 01 and 10 requires 4-time steps since each number has n=2 digits, but multiplying 100 and 110 requires 9-time steps, since n=3 for these numbers. This time scaling, and any time scaling of $a \times n^b$ where both a and b are constants, is referred to as "polynomial scaling" and is considered efficient.

These problems should be compared with those in the complexity class NP, where the time required to solve a problem is exponential in the number of problem inputs and is therefore not considered to be efficiently solvable. An example of such scaling would be 2^n. One common point of confusion is that time steps required to solve both P and NP problems can be comparable for small n. For example, when n=1,2,3, the given polynomial scaling requires time steps of $n^2 = 1, 4, 9$, as compared to the exponential scaling, which requires $2^n = 2, 4, 8$ time steps. However, as n increases, these two scalings rapidly diverge, and this is why they are categorized into different complexity classes. Considering n=10, these two scalings are already an order of magnitude apart, with $10^2 = 100$and $2^{10} = 1024$. Go to n=100, and $100^2 = 10,000$, whereas $2^{100} > 1 \times 10^{30}$that is, 10 followed by 30 zeros! That is the power of exponential scaling. Finally, it is important to note that problems are classified into these categories based on the best-known algorithm for solving them and are therefore subject to change as discoveries are made.

One of the main reasons for the current interest in quantum computers is that specific problems can be solved in polynomial time on a quantum computer even though the best known classical algorithm for the same problem requires exponential time. A prominent example of this is Shor's algorithm. Shor's algorithm effectively factors large integer numbers into two constituent prime numbers, a computationally hard task for classical computers. The difficulty of Factorization is that it is used as the foundation for public-key cryptography [86], for example, RSA encryption. It is even using the best-known classical algorithm, the resources required to break RSA encryption using a classical computer increase exponentially with the length of the public key. Hence, the problem is considered to be in NP (although it has not yet been formally proven to be NP). Practically, this means that doubling the number of bits in an RSA encryption key makes it exponentially more difficult for a classical computer to break the scheme. In

contrast, Shor's algorithm can perform the same task efficiently on a quantum computer, with only a marginal increase in difficulty as the length of the key increases.

16 Quantum Simulation Emulation

Quantum computers can simulate a quantum system using both digital gates and analog evolution. They can also emulate a quantum system by tailoring the qubits and their connectivity. In this section, we will take a look at quantum simulations [87] and a specific example, nitrogen fixation. Another type of quantum computer is a quantum simulator [88], a processor that simulates a physical system's behavior, a chemical reaction, or a biological process. Simulations can be implemented using single and two-qubit gates as on a universal quantum computer, and we will call these digital simulations. Alternatively, the qubits themselves, the way that they connect, and the strengths of those connections can be designed to emulate a system's behavior. We will call this an analog simulator. There are also examples of hybrid simulators [89]that use aspects of both digital and analog approaches [90]. Nature is quantum mechanical. So, with quantum computers, we get this ability to take a step closer to being able to computationally model aspects of physical systems. So, we think this is a pretty fantastic potential first application of quantum computers. As the first business application, with the knowledge of today, we would say the simulation of molecules in materials [51] is the most likely, which, in turn, can have a broad impact on science, technology, and society. One example is the simulation of chemical reaction mechanisms. It is estimated that somewhere between 1% and 2% of worldwide energy production is used to produce ammonia for agricultural fertilizer. The critical step is a process called nitrogen fixation. In industry, nitrogen fixation is performed using the Haber Bosch process, which requires both high pressure and high temperature. In contrast, there exist bacteria that use an enzyme called molybdenum nitrogenase, which, even at room temperature, can catalyze atmospheric nitrogen into ammonia. How do the bacteria do it? we do not know precisely. We do know the critical component is an iron-molybdenum cofactor, a chemical compound that acts as a helper molecule for the enzyme. However, it is not known how the reaction works in detail. Simulations with classical computers, exist but only provide approximate answers. If we could study that process and understand it, we could potentially engineer a catalyst that makes this reaction efficient. Now to study this with classical computers, this is the lifetime of the universe time scale to get a solution. On the other hand, quantum simulation of the chemical reaction steps would provide the reaction energies involved in the nitrogen fixation. This type of quantum chemistry simulation has many applications [91–93]. For example, developing new pharmaceutical drugs or tailoring a material to have unique properties. For these types of simulations, it has been shown that a quantum speedup can exist over known classical algorithms.

Quantum computers can be applied to many types of simulation problems. The challenge is that many of these problems require large numbers of well-behaved qubits, and such large-scale quantum computers are still likely a decade or more away. Before we get to the point where we can simulate systems entirely on quantum computers, there may be room for hybrid classical-quantum systems to hold a quantum advantage over purely classical algorithms [77]. One example is called a variational quantum eigensolver, another hybrid quantum-classical algorithm is Variational Quantum Fidelity Estimation [94],variational quantum factoring (VQF) algorithm [95].

A variational quantum eigensolver [96–103] determines the lowest-energy state of a quantum system, a difficult task of importance to quantum chemistry. The protocol goes as follows:

1. The quantum processor is set so that it simulates the dynamics of the quantum system of interest

2. The classical computer proposes a trial ground state, which is then loaded into the quantum

processor

3. The quantum processor calculates the energy of that state and passes it back to the classical computer

4. The classical computer takes that value for the energy and uses it to generate a new ground trial state, one which should give a lower energy

5. This process repeats until the VQE cannot find a new trial state that gives yet lower energy. This state is then likely the lowest-energy state of the system.

A vital aspect of this process is that the classical computer is not merely being used to initialize a quantum computation; the classical and quantum computers are passing information back and forth throughout the simulation. Therefore, the quantum computer is acting as a co-processor. It need not be coherent over the entire duration of the simulation, but only over the period of time, it takes to simulate the energy of a single trial state. This approach may allow useful computations to be done in the nearer term using less reliable qubits.

The VQE method has been shown to successfully determine the molecular spectra [104]of a hydrogen molecule H_2. To simulate the hydrogen molecule comprising two electrons, a quantum processor with 2 superconducting qubits is sufficient [105]. It has also been used to simulate other small molecules, such as BeH_2. All of these simulations to date are prototype problems that demonstrate the mechanics of the quantum simulation, but the problems themselves are easily solved on a classical computer. As the number of qubits increases, VQE can be applied to larger, more challenging simulations.

17 Quantum Annealing

A quantum annealer is a different type of computer that addresses classical optimization problems by mapping them onto a set of interconnected qubits and then searching for a solution (or solutions) that minimizes the total energy. In this section, we will get a high-level introduction to quantum annealing computers and how they work.

The third type of quantum computer is called a quantum annealer, and it is used solely to solve classical optimization problems. Optimization problems are those that we face every day. What is the best way for us to be able to drive home when there is traffic? What is the best way to route aircraft around an airport? All of those are optimization problems. If we are planning a very complicated mission, space mission with many moving parts that involve materiel, people, gasoline, if we can optimize and improve the efficiency of something like that, quantum computing could have a considerable impact. Quantum annealers do not use digital gates. Instead, an optimization problem is encoded directly into the qubits and their connectivity to one another, and by the strength of those connections. Finding the qubit states that then minimize their total energy is equivalent to optimizing and finding an encoding problem. However, how do we anneal these qubits into states that minimize their energy? So, the name quantum annealing is related to making a sword, where we take a piece of metal, and then we heat it, and in the old days, a swordsmith would beat the sword into some shape, and they would cool it, quench it in water heat it up again, and then beat it into shape and do that multiple times. Furthermore, that was called an annealing process. So, heat it, cool it down, heat it, cool it down. Quantum computing uses a similar technique. To perform annealing, we start by setting the qubit states and their couplings to one another in a configuration where we know the ground state. Let us call this the starting configuration or its starting Hamiltonian. We then slowly change the qubits and their couplings from those in the starting Hamiltonian to those in the encoded problem Hamiltonian [106]. Now, if we make these changes slowly enough, it is likely that We will remain in or very near the ground state of the system so that by the end of this evolution, the

qubits are in their ground state, and that represents the solution to the optimization problem. If we have a problem that we can craft as a landscape like the Alps, for example, what the computer does is, without adding or subtracting, it would find the lowest valley or valleys in that landscape. It is probabilistic, and again, does not add or subtract, but rather, it is not a three-dimensional version of the Alps. However, it is an n-dimensional energy landscape, and it is finding a low energy solution to that problem. The challenge is that for problems of unusual size and complexity, it is almost certain that no matter how slowly we change these parameters, the annealer will, at some point, leave its ground state. It is due to the more significant number of energy states and their proximity to one another and as a result, quantum annealers must find additional mechanisms to return the computer to the ground state, for example, through quantum tunneling or by introducing excess relaxation, a loss of energy that returns the computer to its ground state. If we hit a barrier like a mountain range, as we described earlier, what the machine does is rather than we have to put more energy to climb the mountainside and go down the other side, we use the quantum mechanical property of tunneling. We tunnel through the barrier to get to what should be a lower energy solution in the next valley if we will. On the other hand, add too much relaxation and the computer may not even be quantum mechanical anymore. As a result, it is currently unknown if a quantum annealer can exhibit quantum enhancement for a general class of optimization problems. It may just represent another type of classical computer. Although, as a classical computer, it would not scale well with the size of the problem. However, it may still be interesting if it were substantially faster than any of today's transistor-based classical computers. Volkswagen, in March of 2017, announced the optimization of about 500 taxis going from downtown Beijing to the airport, always congestion used the machine to see if we could come up with optimized routes for each of those taxis to get everyone moving more quickly. Worked, but it is a prototype application. Could not yet handle all the vehicles in Beijing, for example. There are technical challenges facing quantum computing. For the quantum annealing computers, how do we scale them up to be able to start attacking more real-world problems instead of subsets of real-world problems? There has a keen interest in quantum annealers because classical optimization problems are everywhere, from supply transport optimization to sensor and satellite tasking, pattern recognition [107], needle in a haystack problem. Many problems can be reduced to optimization, and so there is an extensive application pull to understand quantum annealers better.

The third type of quantum computer is a quantum annealer, an "application-specific" computer that is used solely to solve classical optimization problems. Quantum annealers do not use digital gates. Instead, an optimization problem is encoded directly into the qubits and their interactions (qubit interaction is referred to as coupling). Minimizing the total energy of the system is equivalent to optimizing and finding an encoding problem.

To quickly explain some physics jargon: Every quantum system has some amount of energy that is determined by the state of each object (e.g., each qubit) in the system. The state of each individual object, in turn, determines the global state of the entire system, and each global state corresponds to the system having particular total energy. The Hamiltonian is a function that matches each state to an energy value. So, for our purposes, the Hamiltonian is a function that tells us "if our qubits are in the state $|\psi\rangle$, our system has energy E_ψ. The state that gives the lowest energy for a system is called the ground state.

The idea behind quantum annealers is that if we have a quantum system governed by one Hamiltonian, we can transfer it to another Hamiltonian by changing the system's parameters.

Quantum annealing starts with a qubit state configuration for which the ground state is well known. It is the starting configuration or starting Hamiltonian. By gradually turning off the starting Hamiltonian, while turning on the problem Hamiltonian, one can transition to the configuration that encodes the problem. If this transition is performed slowly enough and the

system has no noise, then the computer will remain; it is ground state throughout the evolution. Measuring the state of the computer in the final configuration yields the answer to the problem. As described here, this is an "adiabatic quantum computer." [108, 109]

The challenge is that for problems of unusual size and complexity, it is almost certain that no matter how slowly the coupling parameters are changed, the annealer will leave its ground state. There are two primary reasons for this: first, the "minimum gap" between the ground state energy and nearby excited-state energy levels gets very small as the size of the problem increases; and second, there is always noise in the system due to non-zero temperature. Although operation in a cryogenic environment reduces noise, it does not eliminate it [110]. as the minimum gap gets smaller eventually, even cryogenic temperatures begin to look relatively "warm."

Consequently, quantum annealers must find additional mechanisms such as quantum tunneling or excess relaxation to keep the computer in the ground state. On the other hand, too much relaxation will make the computer manifestly classical. As a result, it is presently unknown if a quantum annealer can exhibit quantum enhancement for a general class of optimization problems or whether it is merely another type of classical computer. Nevertheless, there is intense research into quantum annealers today, because the application pull is strong. Many real-world can be cast as optimization problems (e.g., financial portfolio optimization [111], product distribution, routing of autonomous vehicles [112]), and so there is a strong motivation to understand quantum annealers better. Quantum annealing is a method for finding solutions to classical optimization problems. The quantum annealer encodes the problem on to a set of qubits and biases those qubits into a starting state with a known ground state (that is not the solution to the problem). The system is then slowly evolved towards a set of bias points that do represent the problem being solved, and the ground state of the system is the answer to the problem. However, a quantum annealer is likely to leave its ground state during this evolution. The concept is that, though a combination of quantum tunneling and relaxation processes, the system should find its way back to the ground state (or a lower-energy state) to find the encoded (approximate) solution to the optimization problem [113].

18 The DiVincenzo Criteria for Quantum Computers

There are a variety of physical systems that have been proposed as the qubits for quantum computers. What properties must a given technology possess in order to build a quantum computer? In this section, we will discuss the DiVincenzo Criteria [114, 115], the minimum set of requirements a qubit technology must have to be considered a viable candidate for quantum computing, and for communicating quantum information.

There are numerous examples of quantum mechanical two-level systems in nature that could potentially serve as qubits. For example, the electronic states of an ion, or the electron spin of a phosphorus atom implanted in silicon [116] or a nuclear spin of a defect in diamond. We can even manufacture electrical circuits that behave as quantum mechanical two-level systems or artificial atoms. However, what makes a useful qubit? Furthermore, how do we assess these qubit modalities and compare them to one another? Furthermore, how can we determine if a modality is well-suited for quantum computing? We can begin to answer these questions by first drawing on the intuition from classical computing. What makes an excellent classical logic element? Furthermore, why did we end up with transistors? If we want to build a computer at a scale large enough to perform new problems, we should start with a technology that scales well, one where we know how to define and characterize the logic elements and where we can manufacture them in large numbers. These devices must also accurately represent and process classical information. We need to set their states to provide input to the computer, and we need

to be able to measure the result to get an answer. Finally, these devices must be robust against failure to complete the computation and reliably obtain the answer. Transistors satisfy all these requirements very well, so it is no wonder that today's computers are built from transistors and not, vacuum tubes, or mechanical switches. So, what makes a useful qubit? Around the year 2000, David DiVincenzo, then a researcher at IBM, articulated five necessary conditions that any qubit technology must at least possess for a suitable physical implementation for large scale quantum computation [115]. First, it should be a scalable physical system with well-defined and characterized qubits. Second, we must be able to set the input state and, third, measure the resulting output state. Fourth, we must be able to perform a universal set of gate operations. For example, the single and two-qubit gates that we discussed previously that are needed to run an algorithm. Fifth, the qubits must robustly represent quantum information. In many ways, these requirements are like those for classical computers. It is only in how the requirements are met that we can identify differences—for example, performing a universal set of one-qubit and two-qubit gates rather than universal Boolean logic. Alternatively, to robustly represent quantum information, qubits must have long coherence times [117], a concept that loosely translates to the meantime to failure for a transistor. In addition to these five requirements for the qubit technology, David DiVincenzo added two conditions related to the communication of quantum information between qubits. So, continuing from number five, number six is that the technology must support the interconversion of quantum information between a stationary qubit and a flying qubit. Moreover, seven, there must be a way to transfer flying qubits faithfully between two locations. These two requirements describe a quantum version of an interconnect that is, the means to take the quantum information encoded in one qubit, convert it to an object that can move, like a photon, provide the means to guide a photon without loss to another qubit at a distant location, and then hand back that quantum information. Again, analogous to requirements for routing signals within a classical computer but following the rules of quantum mechanics. These criteria, today referred to as the DiVincenzo Criteria, articulate the basic requirements that any qubit technology must possess if it is a viable physical implementation for quantum computation.

Several quantum mechanical two-level systems potentially could serve as qubits. For example, the electronic states of an ion, the electron spin of a phosphorus atom implanted in silicon, a nuclear spin of a crystal defect in diamond, and even manufactured "artificial atoms," electrical circuits can be described as quantum mechanical two-level systems. What are the basic requirements any one of these modalities must at least possess to be a candidate for a quantum computing technology?

In 2000, David DiVincenzo, then a researcher at IBM, articulated five fundamental requirements for any qubit technology to be a suitable physical implementation for large scale quantum computation (see the paper here [115]).

In addition to these five criteria for qubit technology, David DiVincenzo added two conditions related to the communication of quantum information between qubits.

DIVINCENZO CRITERIA

REQUIREMENTS FOR THE PHYSICAL IMPLEMENTATION OF QUANTUM COMPUTATION	
D1: Scalable qubits	Scalable physical system of well-defined, characterized qubits
D2: Initialization	Prepare a simple, fiducial input state
D3: Measurement	Measure the qubit state
D4: Universal gate set	Perform a universal set of gate operations with high fidelity
D5: Coherence	Robustly represent quantum information (long coherence times)
REQUIREMENTS FOR ROUTING QUANTUM INFORMATION	
D6: Interconversion	Ability to interconvert stationary and flying qubits
D7: Communication	Ability to transmit flying qubits faithfully between two locations

Figure 30: DiVincenzo Criteria

These criteria, known as the DiVincenzo Criteria, are, in many ways, adapted from the conditions for classical operational computers and summarize the fundamental requirements qubit technologies need to fulfill at a minimum to be a candidate quantum computing technology.

19 Qubit Coherence and Gate Time

In this section, we will discuss two types of errors that can occur in qubits energy relaxation and decoherence and their corresponding characteristic lifetimes T_1 and T_2. We will also consider the clock speed at which qubit operations can be performed [118].

The average number of operations that can be performed within a qubit lifetime is a proxy for a more rigorous metric called gate fidelity. Implementing more operations before an error occurs is good. As we will see in this section, longer coherence times do not necessarily translate to more operations per gate time, as the gate operations themselves are generally slower in long-lived qubit modalities. However, just as with classical computers, there is a distinct advantage to faster clock speed, as that means we obtain results faster.

The DiVincenzo criteria articulated the requirements that a qubit technology must have to be a viable candidate for the physical implementation of a quantum computer. In this section, we will build on two of those criteria, related to qubit robustness and quantum gates, to define metrics that will allow us to compare qubit modalities with one another. To do this, let us first look, in more detail, at the qubit coherence time, the analog of the meantime to failure for a transistor. Quantum computers, like classical computers, must be built from robust elements. The coherence time is one metric that quantifies the robustness of a quantum bit. Essentially, it is the amount of time, on average, that a qubit state is maintained before the quantum state is lost. As an illustration, let us consider a qubit that we set into a quantum state psi, and consider what happens to that qubit over time. At first, the state is well-defined, we just put it in that state, we are confident that we did a good job, and so we know what the state is. Over time, however, the qubit begins to interact with its environment. When it does so, the qubit experiences noise that alters the qubit state in ways that we did not anticipate. Intuitively, we

can imagine that the state begins to blur. as time goes on, and the qubit is subject to more of this environmental noise. Eventually, we can no longer recognize the state, and the quantum information is fully lost. The qubit is still there, but it is no longer in the state that we would have expected it to be in after this amount of time. The noise has altered the behavior of the qubit, and it is now in some other state. Let us now look at how we quantify qubit lifetimes, by looking at errors on the Bloch Sphere. There are two fundamental ways in which a qubit loses quantum information. The first is energy relaxation. Imagine that we put the qubit in its excited state, state 1. Unfortunately, it probably will not stay there, due to noise at the qubit frequency, the qubit will, at some point, lose its energy to the environment and return to the ground state. This loss of energy is called energy relaxation, and on average, it occurs after a time called T_1. The second way a qubit can lose quantum information is through a loss of phase coherence. Imagine that we intentionally set the qubit at one point on the equator of the Bloch Sphere. It likely will not stay in that direction forever, due to interactions with the environment. there are two ways that phase coherence can be lost. First, the qubit state may move around on the equator, due to the environmental noise. If we repeat the experiment often, the noise will sometimes drive the Bloch vector east along the equator, sometimes to the west, and over time, the Bloch vector fans out more and more. It does this until, eventually, we can no longer tell which direction, or equivalently, which phase, the Bloch vector has. the average time it takes for this to happen is called the pure dephasing time, T_ϕ. Now there is another way things can go wrong on the equator. Remember that, on the equator, the qubit is in a superposition state of 0 and 1, with 1 being the excited state. If that component of the superposition state loses its energy to the environment, then the 1 state flips to 0, and the superposition state is lost. Essentially, the qubit is relaxed from the equator to the north pole. Energy relaxation is also a phase breaking process since once the Bloch vector points to the north pole, we can no longer tell which way it had been pointing on the equator, that phase information is lost to the environment. Thus, the average amount of time that a qubit remains coherent is related to both the dephasing time, T_ϕ, and the energy relaxation time, T_1, which together give a time T_2, over which phase coherence is lost. Thus, a qubit loses its quantum information by two mechanisms, energy relaxation and loss of phase coherence, characterized by the times T_1 and T_2. Now another important metric for quantum computers, just as with classical computers, is the clock speed, the time required to perform a quantum operation. It is called the gate time, and although it generally differs for single and two-qubit gates, we can use a typical time, or conservatively, use the slowest gate time, to define the clock speed with which we can operate the quantum computer. as with all computers, faster is better. Even if we have an exponential speedup from a quantum algorithm, it still takes some time to perform that algorithm. All else being equal, a faster clock speed will translate to obtaining the answer more quickly. A key figure of merit, then, is the number of gates one can perform within the qubit lifetime. The more gates one can implement before an error occurs, the larger an algorithm one can run. This metric also illustrates an interesting trade-off with qubits. In general, qubits with long lifetimes have less interaction with their environment. That is good because they have less sensitivity to noise. However, the trade-off is that they respond more slowly, even when we intentionally try to control them. It is because They are not just weakly interacting with their environment. They are also weakly interacting with the control fields. Similarly, qubits that more strongly interact with their environment may have shorter coherence times, but they generally respond faster. The number of gates one can perform, on average, before an error occurs, may not differ much between these two cases. However, one of these qubit modalities may have a much faster clock speed than the other one, and that is a good thing. This figure of merit, the average number of gates one can perform before an error occurs, is a proxy for a more general and rigorous concept, called gate fidelity, which We will discuss next.we would

like to understand if a quantum processor will have the same limitation of classical processors regarding the gigahertz and clock speed hardware. It is possible to have a or is it possible to have a quantum CPU with much more than with many more gigahertz than is typical nowadays at 3 gigahertz? that the clock speed matters whether we are discussing about classical or quantum computers from the point of view of it will take a certain amount of time to solve a problem. if we want to solve that problem faster, then we need to run the computer faster. But we think making a comparison between classical clock speeds and quantum clock speeds is not really the right comparison because, as we know, a classical computer can really get bogged down on some of these hard problems that a quantum computer can solve very efficiently. so, a problem like factorization, which might take the age of the universe on a classical computer so, just completely impractical on 100-megahertz quantum computer might be able to do this factorization could be done in about an hour or a couple of hours. that is for, a 2,000 or 4,000-bit number. So, that has nothing to do with clock speed. That has to do with quantum computing being a fundamentally different computing paradigm than classical computing. But then what we discuss and we think this is an important point, is that if we have a quantum computer and it can do that calculation in a day at, 100 megahertz, fantastic. It did not take the age of the universe. It took a day much more efficient, exponentially more efficient. But if we run on another quantum computer that is 1,000 times slower, then that will take 1,000 days. so, there, the speed actually matters. So, within quantum computing paradigm [119, 120], comparing different computers, it makes sense to discuss about gate speed. But we also want to emphasize that It is not just gate speed. So, it is gate speed, assuming that all else is equal, meaning that we have two computers and they have the same connectivity, the same overhead for error correction. If they are identical in and along all of those axes, then clearly, running one 1,000 times faster is going to give we an answer at 1,000 times sooner. But gate speed is not the only thing that determines how fast we achieve a solution. the degree to which qubits can be connected to one another for example, do we only have in array of qubits, we only able to connect to a nearest neighbour or do we have an all-to-all connectivity? Can every qubit discuss through a two-qubit gate directly to any other qubit in the system? That is a huge difference. That makes a tremendous difference in the execution speed. So, it is not just clock speed that matters. But clock speed does matter, as explained.

There are two processes by which a qubit loses quantum information. The first is called energy relaxation, and it is characterized by time T_1. It is characteristic time it takes for a qubit in its excited state to relax back to its ground state by emitting energy to the environment. It can be visualized on the Bloch sphere as a qubit prepared in the excited state (south pole) switching to the ground state (north pole) through energy loss. T_1 also characterizes the time over which a qubit will absorb energy from the environment. However, most leading qubit technologies today operate in a regime where such energy absorption processes are negligible compared with energy decay.

The second way a qubit loses information is through decoherence. There are two ways for a qubit to lose coherence. The first is that due to environmental noise, a Bloch vector on the equator might move along the equator away from it is the original position in a manner we cannot predict, and over time, we lose track of which way the vector is pointing. It is called dephasing, and it occurs over a characteristic time T_ϕ. The second is that the excited-state portion of the qubit superposition state could lose its energy to the environment, and the qubit "falls off" the equator and back to the ground state (north pole). It is also a phase-breaking process since once the vector is at the north pole, we can no longer tell which way it had been pointing on the equator, and it occurs over the same characteristic relaxation time, T_1. Together, these two dephasing mechanisms lead to the decoherence time T_2, which is a function of the dephasing time and the energy relaxation time: $1/T_2 = 1/T_\phi + 1/2T_1$.

Mitigating the resulting errors can be quite challenging on quantum computers, much more

so than correcting classical errors on conventional computers. For example, one can periodically error-correct a bit by simply measuring and resetting its state on a classical computer. For example, a classical bit might be stored as a voltage, where a voltage of 5V denotes a 1, and 0V denotes a 0. Even though we cannot assume that a bit set to 1 will stay at 5V forever, we can periodically measure it is voltage and reset it to 5V if it starts to decrease.

In contrast, because of the way measurements affect quantum systems, this option is not directly available to us in quantum computers. It is not possible to measure a state and then reset it precisely, because making that measurement disrupts the quantum state of the system in a manner that is not entirely reversible. It is why we work very hard to shield qubits from environmental noise in the first place. Still, there are quantum versions of error mitigation and error correction that can fix residual errors, provided the underlying qubits are "good enough".

20 Gate Fidelity

How does one confirm that a gate operation is working as intended? In this section, we will discuss the gate fidelity of a quantum operation. Gate fidelity quantifies the quality of gate operation, and it is used to compare qubit modalities of varying types.

In the last section, we focused on two of the DiVincenzo criteria, qubit coherence, and quantum gates. We developed an intuitive figure of merit, the average number of gate operations that can be performed before an error occurs. Such metrics are important because they allow us to compare different qubit modalities, even when those modalities have remarkably different properties. However, the definition we introduced only accounted for errors due to qubit decoherence. That is fine for qubits dominated by decoherence, but other sources of errors limit some qubit modalities. For example, control errors, imperfections in the pulses that are used to drive a gate operation. For these qubits, even if their coherence times were practically infinite, their gates would still be subject to control errors. So, we need a more rigorous way to characterize the robustness of gate operation, one that is sensitive to a broad set of error sources. that leads us to this section's subject, a more general concept called gate fidelity. Gate fidelity is a rigorous means to define how well a gate operation works. Essentially, it is a measure of how closely the actual gate operation matches, on average, a theoretically ideal version of that operation. For example, if we apply an X-gate to a qubit prepared in state 0, how close do we get to state 1? Now intuitively, errors could occur along any direction of the Bloch Sphere, and so we need to check for errors along with all directions after the operation is complete. Until this point in the section, we have only measured the qubit along the Z-direction. In principle, we can measure the qubit along any axis that passes through the center of the Bloch Sphere, including the x-axis or the y-axis. Besides, gate errors may be manifest differently, depending on the starting point of the qubit, so we need to check how the gate performs against different initial states, and not just the north pole, for example. Now at this point, we may be asking ourselves, how is this even feasible? The qubit state can be anywhere on the surface of the Bloch Sphere, representing an infinite number of possible initial and final states. How can we possibly check them all? Well, we do not have to. That is because we can define a basis set that spans the entire qubit state space. To draw an analogy, think of global positioning. the position anywhere on the globe can be represented by projecting it onto a convenient basis, in this case, a coordinate system that spans the globe. We often use latitude and longitude, but we could equally use Cartesian coordinates, x-, y-, and z. Similarly, we can define a basis that spans the entire qubit state space, and then use it to specify any qubit state at the input. We can also use it to characterize any state at the output, by projecting back onto this basis. It works because quantum mechanics is described by linear algebra. It is enough to understand projections to and from the basis set to know what

will happen globally. We will see in the section later, and such projective measurement plays a crucial role in the digitization of errors and the ability to correct them. Although the coefficients of a superposition state will dictate the probability of obtaining a 0 or 1 when the measurement is done, for any given measurement, we will get a 0 or a 1. In this way, quantum states and their errors, which appear to be analog, can be digitized through projective measurement. Now we discuss previously that n-qubits could represent 2^N states. Here, the basis can be represented in a matrix form, and the number of matrix elements that span the state space is the square of the number of states. So, 2^N, quantity squared. For a single qubit, n equals 1, and we need four elements. For two qubits n equals 2, and the number quickly increases to 16. So, determining the gate fidelity is essentially a standard black box type problem. We have a gate operation, and theoretically, we know how it should perform. However, due to errors, its actual operation is not precisely known. That is the black box. Thus, we need to characterize, on average, how well the gate operation performs. To do this, we probe the gate operation using the input basis states, perform the actual gate operation, and then, for each input, project the resulting output state onto the entire basis set. We then compare the results to what we would expect from a theoretically ideal operation to determine the gate fidelity. The described approach is called process tomography [121], and it represents a complete description of the errors during a gate operation. It also requires many steps to implement, as the product of input and output elements is 2^{4N}. Now, although there are 2^{2N}-th constraints, due to the properties of these matrices, this only reduces the total number of measurements by a small amount. Thus, implementing process tomography scales very poorly with the number of qubits, and becomes impractical as n gets large. This is an issue, because although the gates only act on one or two qubits, the errors may leak to nearby qubits, qubit 3, qubit 4, or qubit 5. Thus, to account for this, the basis we choose must encompass all n qubits. Besides, process tomography is sensitive to all errors, including initialization errors, when preparing the input state, and measurement errors at the output, which, although certainly real errors, are unrelated to the quality of the gate operation itself. As a result, an alternative approach, called randomized benchmarking [122–125], has also been developed. Randomized benchmarking essentially interleaves the gate operation being characterized by a random assortment of other gate operations [126]. Although half of the gates are chosen at random, we know what they are, and so we can predict the expected output state, assuming all the gates were ideal. It is then compared to the same experiment, but with only the random assortment of gates, to see how much the error rate has changed when adding the interleaved gate. This change in error rate is then attributed to the interleave gate itself. This approach is repeated for increasing numbers of pulses to obtain a refined estimate for the average error per gate and, thereby, the gate fidelity. Randomized benchmarking is much more efficient than process tomography, and it is also insensitive to initialize and measurement errors. However, it only provides a net error rate without revealing specific error channels. However, it is measured, a gate fidelity of 100% means that the actual operation perfectly matches the ideal operation. For example, no matter where the qubit starts on the Bloch Sphere, the actual x-gate would perfectly rotate the input state to the correct output state. Now, as we might expect, it is generally not possible to achieve a perfect gate fidelity. There will always be some level of error, whether due to qubit decoherence during the operation, imperfections in the control pulse itself. The goal is to see how many nines, two nines, 99%, three nines, 99.9%, that one can achieve. The higher the fidelity, the closer it comes to an ideal gate. We will see later in the section, achieving high fidelity is critical, because it translates directly to two important aspects of quantum error correction. First, the fidelity must at least reach a minimum value, called the threshold, for the error correction to give us a net improvement in the error rate. second, once above this threshold, the higher the fidelity, the less the resource overhead required to implement the error correction.

Gate fidelity is a metric that characterizes how well a gate operation works. Intuitively, we are asking the question: how close does the actual implemented gate operation come to an ideal, theoretically calculated one?

To answer this question, we need to consider all potential errors that may impact the gate operation and the resulting qubit evolution. These errors can, in principle, arise from any direction on the Bloch sphere, so we need to check the fidelity for different starting states of the qubit. Similarly, we need to assess the qubit's final state in all directions along which errors may occur.

To do this, we define a basis set for the qubit that spans all of the possible qubit states and qubit operations. Global positioning is an analogous concept, where we can identify our position on the planet earth using a basis of longitude and latitude, or, equivalently, we could use the Cartesian coordinates x-, y-, and z. Similarly, since quantum mechanics is governed by states and operators that follow linear algebra rules, we can define a basis for the qubit states and operations.

Qubit state tomography is the projection of a qubit state onto this basis [69], and it fully characterizes a qubit state (its position on the Bloch sphere, for example). Similarly, process tomography fully characterizes a gate operation. Since a gate may be applied to any qubit input state, we must first prepare the qubit in each of the basis states that span the possible input state-space. Then, for each of these inputs, the gate operation is applied, and the resulting output state must then be projected and measured against the entire set of basis states at the output. This process is then repeated for each input state. Ultimately, the net results are then compared to a theoretical gate operation, and the "distance" between the two results quantifies the gate errors.

The number of elements in the input matrix that represents the basis states for N qubits is 2^{2N}, and there is the same number of elements in the output matrix. Since the gate operation must be applied to each input state, and the result in turn projected on to each of the output states, the total number of input-output combinations is $2^{2N} \times 2^{2N} = 2^{4N}$. For a single qubit, this is 16; for two qubits, the number quickly increases to 256. while there are a few constraints that will reduce these totals by 2^{2N}, the net number of combinations that must be measured is only reduced to 12 and 240, respectively. It is another example of the power of exponential growth, but, in this case, the rapid expansion in the number of states works against us. We must make all of these measurements to characterize the gate entirely. Besides, process tomography is sensitive to state preparation and measurement errors commonly referred to as SPAM errors. While SPAM errors are real errors, they are independent of the gate operation and should not be included in the gate fidelity.

An alternative approach is called randomized benchmarking. Randomized benchmarking characterizes a gate operation's aggregate performance by interleaving it with a random (but known) assortment of other gates [127,128]. The random gates are first characterized by themselves to assess a baseline level of error, including SPAM errors. Then, the same measurement is performed with the desired gate operation interleaved into the sequence. The outputs are then compared, and the net additional error is attributed to the addition of the desired gate operations. This process is repeated while increasing the number of gates to refine an estimate for the net error per gate. In this way, randomized benchmarking provides an aggregate fidelity that is somewhat more efficient to implement and is ideally independent of SPAM errors. The trade-off means that it averages all error channels without quantifying each individually (as is possible with process tomography).

Gate fidelity is a crucial metric that allows us to compare remarkably different qubit technologies on an equal footing. Gate fidelity is a benchmark that determines the feasibility, efficacy, and resource overhead required for implementing quantum error correction.

21 Qubit Modalities: Electron and Nuclear Spins

There are several physical manifestations of qubits. In the next three sections, we will discuss several qubit modalities. In this first section, we will be introduced to physical qubit modalities based on electron and nuclear spins. In the second section, neutral atoms and trapped ions. In the third section, superconducting qubits and other modalities.

Until this point in the section, we have generally discussed qubits as quantum mechanical two-level systems. However, how do we physically realize a qubit in practice? Moreover, what are the leading physical realizations today? In this section, we will have a general introduction to several candidate qubit technologies. Later, we will present an in-depth look at two of the leading qubit modalities. Let us start with electron spin. As in the introduction to quantum parallelism and quantum interference, the electron spin has two states, spin up and spin down. The question is how to isolate a single electron. In this first example, the electron spin is trapped in a quantum dot, a small region of semiconductor material where a single electron can be trapped. We start with a two-dimensional sheet of electrons called a two-dimensional electron gas, which can be realized at the interface of a slab of silicon and a slab of silicon germanium. Metallic gates are then defined on the surface of the device to define the quantum dot region electrostatically, and by applying the appropriate gate voltages, a single electron spin can be trapped there. Combinations of microwaves and baseband pulses are then used to implement quantum gates. Now we will find several key properties of quantum dot qubits shown here. We will not read through them all, as we can find them in the table associated with this section. In the next section, we will make comparisons between qubit modalities based on these numbers. Now, in addition to the single dot shown here, there are also more complex designs, including double dots, triple dots, and even dots based on CMOS devices. Each of which adds complexity to gain certain advantages. In general, the main attraction of quantum dots is that they leverage advanced silicon fabrication technology [129], two, they are relatively small in area and so in principle can be integrated into large numbers. Three, much like CMOS, they are controlled using gate voltages. The main challenge is that multiple gates are required to define a single qubit. Thus, 3D integration techniques will certainly be required to realize large scale circuits. Another example of an electron spin qubit is in phosphorus-doped silicon [130]. Here phosphorus atoms are implanted in the silicon substrate, and the spin-off of its outermost electron serves as the qubit. As with quantum dots, electrostatic gates control the qubits using a combination of the baseband, RF, and microwave pulses. The main advantages of phosphorus-doped silicon are that it also leverages silicon fabrication technologies. The electron spin qubit has very long coherence times [130]. The primary challenge is that the dopants must be close to one another, around 10 nanometers, to have a sufficiently large two-qubit coupling to implement a two-qubit gate. It makes it extremely challenging to place the gates and route the wires required to control the qubits. It also means that crosstalk between control lines and qubits will be significant. Again, 3D integration will be required here. A second challenge is that the phosphorus dopants must be implanted in the silicon with very high precision. This remains an outstanding challenge. Phosphorus dopants also have a nuclear spin that can be used as a qubit controlled using radiofrequency or RF pulses. The main advantage of nuclear spin qubits is their extremely long coherence times, which arise because the spins are largely decoupled from their environment. The trade, though, is that the gate times are rather slow. A second system that affords both an electron spin and a nuclear spin is the nitrogen-vacancy center in diamond. Diamond is formed from a tetrahedral lattice of carbon atoms. Occasionally, however, a defect interrupts this lattice, and one example is called a nitrogen-vacancy [131], in which a nitrogen atom is injected into the lattice and causes the carbon vacancy to form. The result is an extra pair of electrons that form a spin 1 system, which is the lowest 2 spin levels that can be used as the qubit. Alternatively, the nuclear spin of the nitrogen

atom, or the surrounding carbon atoms, may also be used as the qubit. The electron spin is addressed by a combination of microwave pulses to implement the quantum control and lasers to initialize and measure the qubit. The nuclear spin has a magnetic field dependent splitting that is controlled at radio frequencies. The advantage of NV centers is that they are rather well-suited to the interconversion and communication of quantum information. They also have reasonably long coherence times. They can even be operated at room temperature, albeit with some lower coherence. The primary challenge for NV centers is their scalability. While few qubits spin clusters local to a single NV center have been demonstrated, currently, it is not possible to place high coherence nitrogen vacancies in precise locations to create large qubit arrays.

22 Qubit Modalities: Atomic States

Another qubit technology is based on the internal states of atoms, or neutral atoms, as they are called, to contrast them with the ions that we will discuss next. Neutral atoms can be trapped by cross propagating optical beams, which combine to form an egg carton like potential. The qubit states are hyperfine states, resulting from an interaction between the electron spin and the nuclear spin. Such hyperfine transitions are driven at very well-defined microwave frequencies, commonly used for atomic clocks [29, 30]. These are highly stable qubits, and as such, their coherence times are very long. Thus, the gate fidelity in neutral atoms is generally limited by control errors. The main advantage of neutral atoms is their long coherence times, and the Ability to trap these neutral atoms in two dimensional, and even three dimensional, arrays. Arrays with up to 49 qubits have been demonstrated, although not yet adequately controlled. There are a few challenges, however. One is the high laser power required to trap and control these neutral atoms. Another is that loading the trap is a stochastic process. Atom re-arrangement can later be made to fill in the gaps, but it requires extra steps. Furthermore, third, neutral atoms will require integrated optics to ultimately be scalable, something that is not yet been implemented, although concepts exist for its implementation. The next example, trapped ions, is a leading qubit modality today. Trapped ions have been used as atomic clocks for decades. These systems are stable and very well-characterized. Ion qubits generally start as atoms with two electrons in their outermost shell. Then, one of those electrons is removed through ionization. The qubit is realized as either an optical transition between orbital states of this outermost electron or a microwave transition between hyperfine states. A primary advantage is that many of the DiVincenzo criteria are satisfied for trapped-ion qubits [132]. To date, arrays of 10 to 20 trapped-ion qubits have been demonstrated, and surface traps in silicon are now being developed and used to both capture and control these ions in a scalable manner [133]. The primary challenge is the 3D integration of optical and electrical technologies into the surface traps to make them scalable. We will discuss more trapped ions and their relative advantages and challenges at the end of this section.

A trapped-ion is a charged particle that is confined in all three spatial dimensions by electromagnetic fields. The initial goal of trapping ions stemmed primarily from the desire to perform high-precision spectroscopy and mass spectrometry on the ions. The hope was that by suspending them free in space, the ions could be interrogated for long periods of time (which improves precision), while at the same time being as weakly perturbed by their confinement as possible (which improves accuracy). However, it turns out that it is not trivial to construct such a trap due to a well-known result from electromagnetism, known as Earnshaw's theorem, which states that three-dimensional trapping cannot be achieved simply by electrostatic fields alone.

Despite this complication, two ingenious methods were developed in the late 1950s to trap ions: the Penning trap and the Paul trap, invented by Hans Dehmelt and Wolfgang Paul, respec-

tively. The Penning trap gets around Earnshaw's theorem by utilizing a combination of static electric and magnetic fields, and the Paul trap does so by utilizing radiofrequency electric fields. For this work, Dehmelt and Paul shared the 1989 Nobel Prize in physics.

In 1975, David Wineland, who studied electrons confined in Penning traps with Dehmelt and who would receive a Nobel Prize of his own in 2012, first proposed with Dehmelt a method to cool the motion of an atomic ion in a trap to very low temperature with the use of lasers. The ultimate motivation of this laser cooling was to realize even higher precision atomic spectroscopy due to the reduction of spectroscopic shifts that result from the motion. Wineland subsequently moved on from his postdoc to the National Institute for Standards and Technology (NIST), and in 1978, he along with his colleagues there, implemented laser cooling of a cloud of trapped magnesium ions. At nearly the same time, Dehmelt and colleagues at the University of Heidelberg in Germany demonstrated the cooling of barium ions using the same method.

The demonstration of laser cooling of atomic ions, and the high-accuracy and precision spectroscopy it enabled, led to another idea: using laser-cooled trapped ions as a frequency standard, or atomic clock. In 1984, Wineland and his colleagues at NIST demonstrated this concept using beryllium ions in a Penning trap. In order to realize this high-performance clock, ion trapping and cooling of the motion were not the only essential things. Instead, having two internal (electronic) quantum states of the ion that had long coherence times was essential. This is because the certainty in the splitting between these two levels sets the accuracy and precision of the clock, and this certainty is degraded by decoherence. In addition to long coherence times, it was necessary to be able to prepare accurately, or initialize the internal quantum state of the ion before interrogation by the radiation field used as the clock's master oscillator, as well as to read out the final state of the ion after this interrogation had occurred.

In 1994, Peter Shor developed an algorithm that could be run on a quantum computer and perform the cryptanalytical task of factorization of numbers into their constituent primes exponentially faster than classical ones [59]. Later that year, motivated to some degree by the importance of factoring algorithms to the security of many modern cryptosystems, Ignacio Cirac and Peter Zoller at the University of Innsbruck laid out a framework for physically implementing a quantum computer using cold ions confined in a Paul trap [134]. They recognized that most of the critical ingredients for quantum computing with ions were already in place due to work on atomic clocks, namely ion-motion cooling, high coherence, high-fidelity internal ion state preparation and readout, and the ability to put ions in superpositions of two internal states accurately. This last technique had been demonstrated as part of the method of clock interrogation, and it was evident that it represented the presence and control of a qubit in the trapped-ion system [135]. The chief innovation of Cirac and Zoller was their development of a method to use the motion of the ions in the trap, coupled to one another via their Coulomb repulsion, as a way to implement a so-called controlled-NOT (CNOT) gate [136]. The CNOT gate creates quantum entanglement between the ions [137], which is essential for the implementation of Shor's algorithm, by universal quantum computing.

Over the next few years, the CNOT gate and ion-ion entanglement were demonstrated experimentally by Wineland's team at NIST and by a group at the University of Innsbruck led by Rainer Blatt. Since then, motivated by the success of these early demonstrations and by the general interest in quantum computing, there has been an explosion of groups around the world developing and implementing techniques and technology for more sophisticated control of trapped-ion qubits. This includes a proof-of-principle demonstration of Shor's algorithm by Blatt's group using five trapped ions to factor the number fifteen. While this work has mostly been carried out by academic research groups, it has recently begun to be explored in the private sector as well, with a startup company called IonQ focused on developing a commercially-available trapped-ion quantum computer. One of IonQ's founders, Chris Monroe, worked on Wineland's

team to implement the first trapped-ion CNOT gate.The Honeywell has also recently reported an effort to develop quantum computers with trapped ions. The field of trapped-ion quantum computing also relies heavily on other specialty industries, such as high precision optics and laser companies, since laser light is one of the primary means of trapped-ion quantum control, and it must be highly stable in both its absolute frequency as well as its direction of propagation.

The future appears bright for trapped ions since they offer a platform for high-coherence qubits and high-precision, universal quantum control. The main challenges going forward will be scaling systems of trapped ions to a large enough that they can move beyond the science experiments and proof-of-principle demonstrations of today to quantum processors that can outperform classical computers for practically useful tasks. In order to achieve this, there will be a need for the further development of ion control technology and techniques that do not suffer performance degradation as the system size grows. Besides, the development of quantum algorithms that make efficient use of trapped-ion qubits will be required. New academic research groups and industrial efforts will likely spring up to meet these challenges.

23 Qubit Modalities: Superconducting Qubits

The next example, superconducting qubits, are manufactured artificial atoms. Unlike the previous examples based on spins and naturally occurring atoms, superconducting qubits are electrical circuits that behave like atoms. Necessarily superconducting qubits are nonlinear oscillators built from inductors and capacitors [138, 139]. The inductor is realized by a Josephson Junction [**?**, 140–142], a nonlinear inductor that makes the resonator anharmonic. We will discuss this in the following section. Anharmonic oscillators feature in the addressable two-level system that We will call the qubit. The qubit states can either be states of phase across the Josephson Junction, the flux in a superconducting loop, or even charge on the Junction island. The main advantages of superconducting qubits are that the gates are fast compared with the other qubits, and They are manufactured on silicon wafers using materials and tools common to CMOS foundries. To date, arrays of 10 to 20 qubits have been demonstrated, including the cloud-based quantum processors that We will use in this section. The main challenge is the integration of control and readout technologies that maintain qubit coherence, even at millikelvin temperatures. We will discuss the challenges, advantages, and the current state of the art in the following section. Finally, several other qubit modalities are being pursued that are at various stages of development and maturity. The most developed and mature include linear optics quantum computing, where the presence or absence of a photon constitutes the qubit. Quantum information is processed using linear optical components like beam splitters [143], phase shifters, mirrors, and interferometers. Effective nonlinear interactions are achieved using single-photon sources, photodetectors, and the like. The main challenge with linear optical quantum computing is that it is tough to make photons interact with one another. Besides, high fidelity memory is a big challenge, and many schemes rely on probabilistic sources and gates to process information. Another type of quantum computer based on neutral atoms is a quantum emulator. A quantum emulator uses Rydberg atoms [144] or Bose-Einstein condensates to emulate a condensed matter or atomic system using tailored atomic energies and coupling terms [145]. A more exotic qubit modality that is generating significant interest today is based on something called a Majorana fermion [146], a fermion that is its own anti-particle. Several efforts worldwide are trying to realize a Majorana qubit [147], using a combination of superconducting and semiconductor materials [148, 149], which feature a strong spin-orbit interaction. If successfully realized, Majorana qubits have been theoretically shown to exhibit topological protection, resilience to noise [150]. That is not unlike the resilience afforded by quantum error correction. The challenge is that

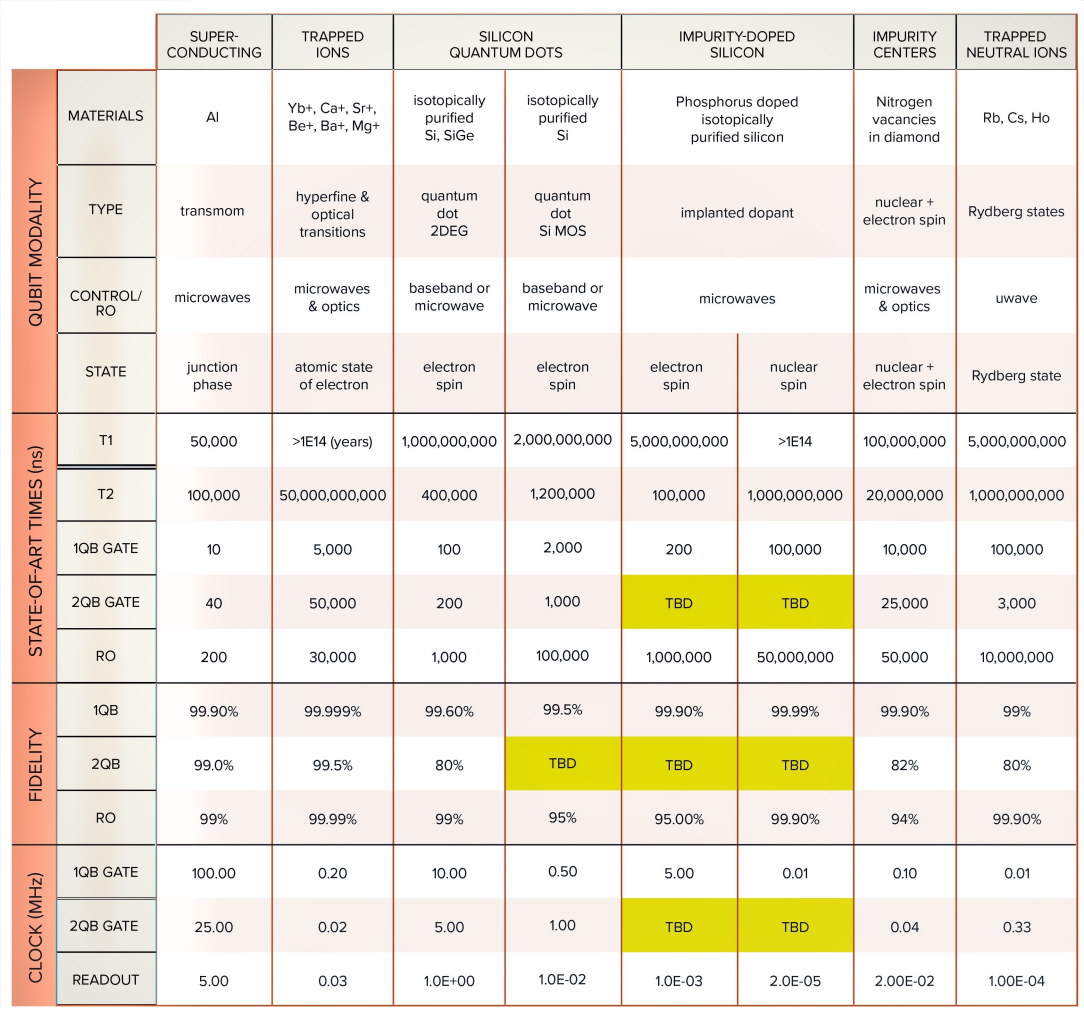
They are challenging to realize. To date, there has been no definitive demonstration of a Majorana qubit featuring this topological protection. Other qubit candidates include molecular ions, which are trapped and controlled in a manner like atomic ions, and electrons on liquid helium, which are free electrons that form an electron lattice on the surface of liquid helium and can be controlled using electron spin resonance techniques. In superconducting qubits, it appears that there really is only a move up and down in an energy level. how is this useful for quantum computation? From the Bloch sphere, it appears that spin and superposition play an important role in storing information that would be used in the final answer after the gate operations. How is this working? Whether it is the superconducting qubit or trapped ion qubit, whether it is a physical electron spin in a semiconductor qubit or what we call a pseudospin, a system that behaves like a spin, in all of these cases, a qubit has two energy levels that we care about, state 0 and state 1. But because it is a spin-1/2 system or a pseudospin-1/2 system, a system that behaves like a spin-1/2 system, quantum mechanically, the states of that system are described within this Bloch sphere picture that we have discussed about. so, classically, states 0 and 1 would be the North Pole and the South Pole. the excited state would have an energy that is generally higher by some value than the ground state energy is state 0. But the point is that we can put the qubit into superpositions of these states, 0 and 1. when they are in a superposition of 0 and 1, the state or the Bloch vector is pointing to any position on the earth, any position except the North and South Pole. Now, it is important to remember that It is a bit hard to describe these states in the vernacular because the words we use describe the world around us, which is classical. so, we often say that the qubit is in state 0 and state 1 simultaneously. There is some truth to that. But the thing to keep in mind, and this is what is hard to describe in words, is that It is one state. The qubit is in a state. that state could be state 0, that state could be state 1 those are classical states or it could be in a superposition state. It is a single state. But it carries aspects of both 0 and 1. so, with some weighting coefficients that we call probability amplitudes could be $\alpha 0 + \beta 1$ and so, it is a single state that carries aspects of both 0 and 1. So, even though we say, look, It is in a superposition of 0 and 1, that does not imply that It is in both states and that there are somehow two states. It is a single state. But it carries aspects of both 0 and 1. that is true whether It is a superconducting qubit, which, due to its design, is a pseudospin-1/2 system, or whether It is, an electron spin, literally an electron spin, that is being electrostatically trapped in a semiconducting material.

24 Comparing Qubit Modalities

In this section, we will compare the different qubit modalities that we just discussed. We will first do this in the context of the DiVincenzo criteria, and then compare the gate fidelity and the clock rate for a single qubit and two-qubit gates. Let us begin with a stoplight chart and the DiVincenzo criteria. Now, this is undoubtedly going to be a subjective assessment. With all the ongoing rapid technology development, the actual coloring will undoubtedly change over time. Still, it is useful to take a snapshot of the current state of the art to gain insight into the relative strengths and challenges that different modalities face, and that is the purpose here. The stoplight chart assigns colors to indicate progress. Green will indicate that a modality has made the requisite demonstrations and has sufficiently matured to the point that we can envision it proceeding to 100 plus qubits. Yellow indicates that concepts or first demonstrations may exist, but that the technology is not ready to scale to the 100-qubit level. Red indicates that no realistic concepts are currently known that would enable reaching 100 qubits. However, we are not going to consider any of these qubit modalities. They are still too nascent for the purposes here, so we will not see any red. Starting with the first DiVincenzo criteria, D1, scalable systems, and most mature

are neutral atoms, trapped ions, and superconducting qubits. All have demonstrated systems of 10 to tens of qubits, with 50 qubit prototypes coming to a line soon. Silicon germanium and doped silicon qubits are yellow because they will require high wire counts to address their qubits. Moreover, those wires need to be routed with high density, due to the relatively small qubit geometries. All these issues will need to be addressed with 3D integration, which is only just beginning to be conceived for these modalities. Besides, both doped silicon and NV centers are currently limited by the inability to implant high coherence dopants or defects with the precision required to reach the 100-qubit level. Next, for initialization D2, most technologies are doing well. The one exception is NV centers, where the strong laser used to initialize the qubit will occasionally remove an electron. This makes the defect charge-neutral, effectively destroying the qubit until it is reset. For measurement D3, the measurement of neutral atoms is extremely slow and would certainly limit the ultimate clock speed of an error-corrected system. That is why this one is colored yellow. In terms of D4, universal gates, the doped silicon community has not yet reported a 2-qubit gate fidelity. For neutral atoms and silicon-germanium qubits, although their two-qubit gates have been demonstrated, they still have a relatively low fidelity, around 80%, which is why these are colored yellow. In terms of coherence, D5, all these technologies, grade well. For D6 and D7, the interconversion and communication of quantum information, there has been no published work yet for silicon germanium or doped silicon approaches. However, it is anticipated that conversion to microwave photons should follow from the same approaches used for superconducting qubits. Now, as we can see, both trapped ions and superconducting qubits have made significant progress toward meeting the DiVincenzo criteria. It is one reason they are viewed today as leading candidates. Let us turn now to the gate fidelity and gate speed of these modalities. The number of operations before an error occurs directly related to the fidelity, and they are plotted here against the gate speed for single and two-qubit gates. The upper right corner represents the best performance. Additionally, it is desirable to be above the dashed red line, which is representative of the most lenient threshold for quantum error correction. What we see is that trapped ions have the highest single qubit fidelity. On the other hand, they are about 500 times slower than a superconducting qubit gate. For two-qubit gates, superconducting qubits and trapped ions have similar levels of gate fidelity. However, again, the superconducting qubit gate is about 1,000 times faster than the ion traps. Some technologies, like doped silicon, have measured and reported single-qubit gate fidelities, but They are still working on their two-qubit gate fidelities. Again, we can see that both trapped ions and superconducting qubits are leading modalities today. In the case studies, we will take a detailed look at each of these technologies.

The following table provides a snapshot of the current state of technology for several leading qubit modalities as of 2020. Entries labeled “TBD” have not been reported for a given qubit modality.



		SUPER-CONDUCTING	TRAPPED IONS	SILICON QUANTUM DOTS		IMPURITY-DOPED SILICON		IMPURITY CENTERS	TRAPPED NEUTRAL IONS
QUBIT MODALITY	MATERIALS	Al	Yb+, Ca+, Sr+, Be+, Ba+, Mg+	isotopically purified Si, SiGe	isotopically purified Si	Phosphorus doped isotopically purified silicon		Nitrogen vacancies in diamond	Rb, Cs, Ho
	TYPE	transmom	hyperfine & optical transitions	quantum dot 2DEG	quantum dot Si MOS	implanted dopant		nuclear + electron spin	Rydberg states
	CONTROL/ RO	microwaves	microwaves & optics	baseband or microwave	baseband or microwave	microwaves		microwaves & optics	uwave
	STATE	junction phase	atomic state of electron	electron spin	electron spin	electron spin	nuclear spin	nuclear + electron spin	Rydberg state
STATE-OF-ART TIMES (ns)	T1	50,000	>1E14 (years)	1,000,000,000	2,000,000,000	5,000,000,000	>1E14	100,000,000	5,000,000,000
	T2	100,000	50,000,000,000	400,000	1,200,000	100,000	1,000,000,000	20,000,000	1,000,000,000
	1QB GATE	10	5,000	100	2,000	200	100,000	10,000	100,000
	2QB GATE	40	50,000	200	1,000	TBD	TBD	25,000	3,000
	RO	200	30,000	1,000	100,000	1,000,000	50,000,000	50,000	10,000,000
FIDELITY	1QB	99.90%	99.999%	99.60%	99.5%	99.90%	99.99%	99.90%	99%
	2QB	99.0%	99.5%	80%	TBD	TBD	TBD	82%	80%
	RO	99%	99.99%	99%	95%	95.00%	99.90%	94%	99.90%
CLOCK (MHz)	1QB GATE	100.00	0.20	10.00	0.50	5.00	0.01	0.10	0.01
	2QB GATE	25.00	0.02	5.00	1.00	TBD	TBD	0.04	0.33
	READOUT	5.00	0.03	1.0E+00	1.0E-02	1.0E-03	2.0E-05	2.00E-02	1.00E-04

Figure 31: Quibit Modality

The table below is the reproduction of the "stoplight chart" from the section. It provides us with an overview of several leading qubit technologies and their performance concerning the seven DiVincenzo Criteria.

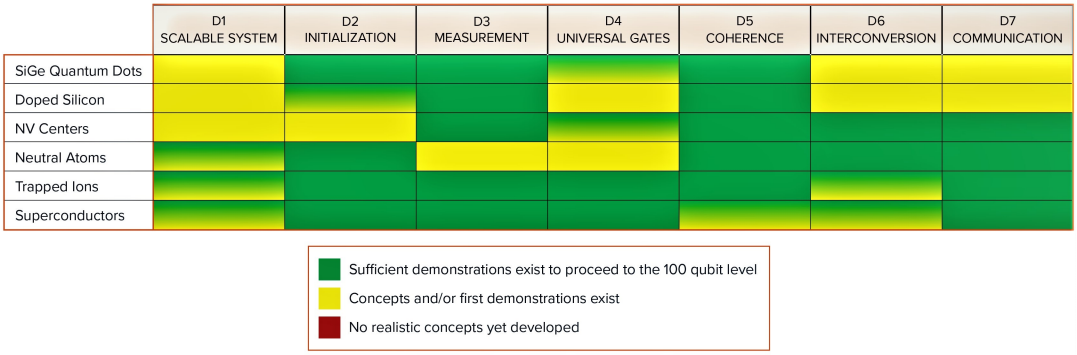


Figure 32: DiVincenzo Criteria

This evaluation is somewhat subjective, but it serves to place these technologies on a common footing to compare progress concerning one another. It indicates where each technology excels and where improvements are needed.

The DiVincenzo criteria, which is, what is the minimum requirement for a particle or a system to be used as a qubit? And among all of the elementary particles in the standard model, which ones can or cannot be used and why? So, we believe, the DiVincenzo criteria specify what is the minimum again, minimum requirement for a technology or modality to be considered for quantum computing in a scalable sense. so there are many engineering problems beyond that to actually truly scale to a large size. But the DiVincenzo criteria look, whatever technology we have, it has to at least meet these requirements. so, for example, it has to be a uniquely identifiable and addressable two-level system. we need to have a universal set of gates typically, a handful of single and two-qubit gates. so, these were the DiVincenzo criteria. It has to have a long coherence time, for example. so, any technology that we might imagine, we wonder if this would be a good candidate to be used in quantum computing, would start with those five criteria. Now, if we look at the elementary particles in the standard model. we think that we can take out some examples of ones that would be good examples and one that would be a bad example. So, first, a good example is the one that is currently being used, which is the electron and its spin. So, the electron is a spin-1/2. whether this is an electron that arises from doping a semiconductor, with phosphorus the phosphorus dopant has one extra electron and that electron is then used as the quantum bit of information or whether that electron is, say starts out in a two-dimensional electron gas and, through electrostatic gating, we either deplete most of the electrons away and leave just one that is a depletion mode semiconductor quantum dot, or There is another version, which is an accumulation mode, and we start with no electrons and then we apply a positive voltage to a gate so, that we pull up one and only one electron however we gather that electron or isolate that electron, it is a spin-1/2. It is been shown that these spin-1/2 electrons are controllable. We can address it with either electric or magnetic fields. So, the spin is magnetic. But through spin-orbit coupling, for example, one can use an electric field to drive a rotation of the spin. There are single-qubit gates. contemporary research now demonstrating two-qubit gates with, doped silicon [151]. so, we think that the electron is a good example of an elementary particle that would make a good qubit. Some other examples electron and helium surface. People have done experiments where they try to isolate single electrons on the surface with the helium. It is less mature. But that would be another example. So, what is an example of a particle which may not be such a great qubit? And we can think of one off the top of my head, which is a neutrino. So, a neutrino we might think, They do not interact with anything. so, their coherence time is presumably infinite. true enough. But it is not enough to have a long

coherence time. we also have to have a reasonable gate time. It is actually the number of gates that we can apply within that coherence time that matters, how many gates per coherence time. How many gates can we apply before decoherence sets in? That is the metric. It is related to the fidelity of the qubit. so, a neutrino, although it lasts forever as we know, it is very hard to detect these neutrinos because It is very hard to couple to them and to control them. So, even if we could grab onto a neutrino, which we think is very difficult, and hold it in isolation, where we could operate on it, It is very challenging to actually perform the analog of quantum operations on such a neutrino because it interacts so, weakly with its environment. so, this is an example of a particle which would not make a good qubit simply because the gate times would be far too long so, perfectly isolated, but useless for quantum computing. NV centers and doped silicon modalities are still working to meet criterion D1, as they are currently limited by the inability to implant high-coherence dopants or defects with high precision reliably. Besides, NV centers are still addressing criterion D2, as the strong laser used to initialize the qubits will occasionally remove an electron and make the defect charge neutral. The doped silicon modality qubits are working towards criteria D4, D6, and D7. Two-qubit gates are just now being demonstrated, and definitive fidelity measurements are being performed. The interconversion and communication of quantum information have yet to be demonstrated.

25 Trapped Ions: Introduction

We will explore todays leading trapped ions qubit modality engineering, science, and technological aspects. Trapped ions were the first qubit technology, primarily due to their historical use in atomic clocks and precision measurements. Ions start as neutral atoms, the chemical elements in the periodic table. For example, we will discuss strontium in this section. A strontium ion is formed when its outermost electron is removed, and a process called ionization. With one less electron, the atom now has a net positive charge, so it is no longer charged neutral, and We will call this an ion. Because the ion is charged, it can be trapped or held in place using oscillatory electromagnetic fields. Although these fields used to be applied using large electrodes arranged in a three-dimensional configuration, today, they are implemented using surface traps electrodes manufactured on silicon wafers that hold the ions just above the wafer surface. Trapped ion qubits are one of the leading modalities being developed to realize a quantum computer, and it has several advantages. From a business perspective, trapped ions are attractive because they leverage a substantial existing technology base, one that goes back three decades to the development of atomic clocks and mass spectrometers. Besides, trapped ions today are fundamentally silicon-based technology. The surface traps themselves are fabricated on silicon wafers, and they use standard semiconductor fabrication tools and techniques. Also, all the critical control and readout circuitry can be integrated with existing CMOS electronics. For example, the electrode voltage is used to hold and move the ions, or the integrated photonic waveguides and gradings used to control the ions optically. Furthermore, even the photodetectors used to read out the ions are all compatible with existing CMOS foundries and materials. In this sense, they are lithographically scalable to large numbers of qubits. Today, there are a growing number of commercial efforts pursuing or supporting the development of trapped ion qubits, including a startup company called IonQ. From an engineering perspective, the surface traps used to capture and hold ions are designed using standard CAD software and layout tools. The electrode control electronics are CMOS circuits, and They are designed in the same way as conventional CMOS electronics. Similarly, the photonic waveguides and couplers are designed and integrated directly into the silicon. Scientifically, state of the art trapped ion quantum processors has demonstrated several prototype quantum algorithms, such as Shor's algorithm at the 10-qubit level, and larger-

scale circuits at the several tens of qubit level are expected soon. Although trapped ions operate more slowly than superconducting qubits, they have demonstrated a more significant degree of connectivity between the ions. It may help offset the relatively slow speed by making problem embedding more efficient. Finally, from a technology standpoint, trapped ion qubits leverage both microwave and data communications technology. The lasers used in trapped ion experiments have an application to atomic clocks, precision navigation, even biology, and optogenetics. Developing compact instrumentation, for example, size, weight, power, even cost, will benefit an array of technologies even beyond trapped ions.

26 Trapped Ions: How They Work

In the following section, we will introduce how trapped ions work, from the ionization of neutral atoms to how they are captured using surface traps. MIT Lincoln Laboratory, is developing technologies to enable practical quantum information processing, including quantum computation, simulation, and sensing, [53, 132, 152, 153]. Starting with neutral atoms with two electrons in the outermost level, we can remove one of these electrons, producing an ion with a positive charge that we can hold onto very tightly using electromagnetic fields. In machines like these, we can trap individual atoms, the smallest amount of an element in the case, strontium, or things like calcium. That is one atom. That is something like smaller than a tenth of a nanometer. we manipulate them with lasers and electromagnetic fields such that we can hold onto them for many hours and manipulate their quantum states to do things like quantum information processing. So, in an individual trapped ion, the amount of time that the quantumness survives, or the coherence time, can be made to be very long. In systems like this, where people regularly manipulate trapped ions, this coherence time can be seconds, even minutes. this allows us to do lots of coherent quantum operations between larger numbers of qubits in the amount of time available to do anything with those quantum systems. Other advantages are that the error rates in the quantum operations that we perform single qubit, double qubit operations that are keys to quantum information processing [154], and the preparation and readout of the quantum states can be done with very low error rate in trapped ion systems. Because these atomic systems are very clean, single quantum systems that we can control very well, we can theoretically predict exactly how They are going to behave. We know where they are, and we can manipulate their states very, very cleanly. We have defined the qubit state within a single ion, but how do we hold on to one atom? Utilizing the positive charge of the ion, we can trap it in a combination of static and dynamic electric fields. By applying a radio frequency electric field to two diametrically opposed rods out of a four-rod square configuration, we can produce a quadripolar field in two dimensions. At any one moment in time, this field will tend to push an ion toward the center of the trap in one dimension and push it away in the other. However, since this field oscillates in time, for the right combination of frequency and amplitude, the ion can effectively be pushed toward the center in all directions as if it were in a two-dimensional bowl. In the third dimension, we can apply a combination of static voltages to trap the ion. This is the same technology used for mass spectrometry, and we see potential offshoots of ion quantum information processing in improvement to small sample sensing and identification. A promising approach to large-scale ion control and manipulation is based on chip-based traps, as shown here. By patterning metal using standard microfabrication techniques onto an insulating substrate, we can produce a flat version of the four-rod trap structure We just described. We will describe in a bit, and ions are trapped in space above the center of the trap where the quadripolar trapping field is produced. To hold onto an individual ion for a long time, a very low-pressure environment is required, as molecules in the air have enough energy to knock the ions out of the trap. Ultra-high vacuum

conditions with background pressures more than 12 orders of magnitude less than standard atmospheric pressure can be maintained in specialized chambers. These extremely low pressures are produced in the laboratory. It employs a cryopump, in which various surfaces in the chamber are made very cold, within a few degrees of absolute zero, utilizing refrigeration. Much as water condenses from the air onto a cold beverage container on a humid day, cryo-pumping results in condensation of almost all the air's constituents quickly and dramatically lowering the pressure in the closed vacuum chamber. Ions can be maintained for hours to days in UHV systems like this. It is important to point out that the ions themselves are not cooled in this manner. They are cooled too much lower temperatures, eventually to only a few tens of millionths of a degree above absolute zero, using lasers. Perhaps counterintuitively, it is possible to remove motional energy from atoms by allowing them to absorb light at a very particular frequency. They re-emit light at a slightly higher frequency, leading to a reduction in motional energy. This laser cooling process developed 30 years ago, is a key enabling technology for quantum computing with atomic systems. This animation shows the process by which we get individual ions into the trap. Inside the vacuum system, we heat a piece of metal to a few hundred degrees celsius, producing hot atomic vapor. Using a combination of laser beams and a magnetic field gradient, we cool approximately a million neutral atoms into what is known as a magneto-optical trap. These cold atoms can then be accelerated toward the ion trap chip using another laser beam. With the radio frequency trapping potential applied to the trap electrodes, neutral and ionized atoms, using other lasers within the trapping volume, will feel the trapping fields and be localized within the trap, where they are cooled with yet another set of lasers, these resonant with transitions in the ion. The result is a single atom held in space approximately 50 micrometers from the surface of the chip. We can image the ions using the light they scatter during cooling, producing images like that shown here.

27 Trapped Ions: Qubit Operations

In the following section, we introduces how to control and measure the states of trapped ions to implement universal quantum computation. It is done using pulses of coherent laser light, of carefully controlled amplitude and phase, applied to the trapped ion. Each ion encodes roughly one qubit, and multiple ions can be trapped next to each other, to allow multi-qubit systems to be controlled and measured.

Once we have the ions, how do we manipulate the qubits? Single-qubit gates are brought about by applying laser pulses resonant with the qubit transition for a certain amount of time. If the qubit starts in the ground state as a function of the laser pulse duration, the qubit state coherently oscillates between the ground state and the excited state, performing what is known as Rabi oscillations. By choosing the appropriate time to stop the Rabi oscillations, we could perform a flip of the quantum state from 0 to 1, producing a quantum version of the classical NOT gate, or the inverter. In this way, we perform a π pulse. Interestingly, if we use a laser pulse half as long as a π pulse to perform a $\pi/2$ pulses, that is a gate with no classical analog, we produce in the ion qubit a superposition state, 0 and 1 at the same time, where the electron is effectively in two states at once. These manipulations form the basis of single-qubit operations on ion qubits. Along with single-qubit gates, 2-qubit gates are required to do arbitrary quantum computations. We bring about these operations utilizing the strong Coulomb interaction between two positively charged ions. Two ions in the same trap share vibrational modes due to their interaction, much like two balls with a spring connecting them. These modes can be excited, depending on their internal qubit states, using laser beams. Thus, the internal qubit states can be entangled through the quantum vibrational mode channel used as a quantum bus. The basic

2-qubit gate is a controlled-NOT gate, somewhat analogous to a classical exclusive OR gate. It can be produced using this bus interaction in combination with a few single-qubit gates, we just described. If one ion starts in an equal superposition state, the state after application of a controlled-NOT gate will be a maximally entangled state, capable of effects like Einstein called "spooky action at a distance." Not only is this spooky action the way entangled systems work, as verified many times in laboratories around the world, it is also a fundamental component that large-scale quantum computers will rely on for their operation. Finally, after a series of single and 2-qubit operations, as we would perform to carry out an algorithm, the quantum state of the ion qubit must be measured. Ions allow a particularly useful mechanism for high-fidelity state readout when compared with other systems [155–158]. By illuminating the ion with light resonant with an auxiliary transition, we cause the ion to absorb and re-emit light on this transition if it is measured in zero states. In contrast, the light will be off-resonant if it is measured to be in the 1 state, and the ion will be dark. The ion state may have been in a superposition before the measurement, but during illumination, the state will be projected to 0 or 1 and remain there. It forms a quantum non-demolition measurement, allowing us to scatter many photons and get useful statistics on the ion state. By setting a threshold on the number of photons we detect from the ion during measurement using a single photon-sensitive detector, we can measure the ion's state with very high fidelity. Therefore, we have established techniques to perform all the operations required for quantum computing using trapped ions, and the properties of the ions themselves allow for very low error in these operations. Researchers in the field of trapped ion quantum computing have demonstrated basic quantum computing primitives in a few ion experiments that approach or surpass the fidelity levels we think we need for useful large-scale systems. Coherence times for a single ion are about 10 minutes long for quantum properties to persist, even becoming comparable to human time scales. Single-qubit gates, with errors at the one-in-a-million-one level, have been achieved using microwave fields. Furthermore, 2-qubit gate fidelities are at the 99.9 percent level in experiments performed by a few different groups. Besides, a few tens of ions have been trapped and individually manipulated as well, though not simultaneously with the highest fidelity gates. The remaining challenge is to maintain the exquisite level of quantum control that is possible while scaling systems to many ions. It must include providing control and readout capability for arrays of ions without simply multiplying up the number and size of the bulk optics and external electronics setups used today equipment that can take up a small room. It is an exciting area of current research.

28 Trapped Ions: Chip-Scale Integration Technology

In the following section, we introduces photonic integration technologies that are being developed to engineer larger-scale surface traps for multi-qubit trapped-ion processors of the future [132, 159, 160]. One of the most significant technical challenges for trapped ion quantum computing systems is providing all of the laser controls needed. It is precisely focused on the many ions held just about a hundred micrometers above the surface of a microfabricated chip. we explains how this can be accomplished by integrating optical waveguides into the chip, and lithographically patterning optical gratings onto the chip's surface, such that light traveling within the chip can exit, then come to a focus, on single atomic ions above the chip. This laser light needs to come in many different colors, ranging from blue wavelengths to red and far-infrared. Surprisingly, the technology exists to guide all these colors within the integrated optics. As we discussed previously, all the required operations for performing quantum computing with trapped ions have been demonstrated in research groups around the world. However, this has been achieved in only a few-qubit system, that is, with around 1 to 10 ions. Perhaps the most significant obstacle

to realizing a practically useful quantum computer is the current lack of ability to control and measure huge numbers of ions. we think We will need thousands to millions, with the same exquisite precision demonstrated in the few-ion systems. It is the direction, the direction of so-called scalability, in which the field is looking. There are four key things we need to do with the trapped ion qubits. We would like to go through them and discuss the technology being developed to address how we might do each one on a large scale. That is, on many ion qubits. First, ions need to be trapped or held in fixed positions. we already discussed that this is done with a combination of both static and oscillating voltages applied to metal electrodes placed around the positions. We would like the ions to be trapped in. These electrodes are typically centimeter to millimeter scale and are made in standard machine shops. It works well for few-ion systems, but to trap huge arrays of ions finely detailed and complex electrode structures are required. this is something challenging to achieve with machining techniques. We would like to utilize micro or nanofabrication techniques, like those employed for making classical computer chips, where metal layers are deposited on wafer substrates, like silicon or sapphire, and are subsequently deposited on wafer substrates finely patterned using what is called optical lithography. Fortunately, we have figured out a way to use these techniques to make ion traps. It turns out it works to unfold the electrodes normally placed around the ion onto a plane that lies below it. By applying a similar set of voltages to these planar electrodes, the ion can be trapped above the surface of the plane, with the surface to ion distance scale being set by the size and spacing of the electrodes. Since these planar electrodes can be fabricated using microchip techniques, they can be made small, and in an arbitrary pattern. It allows us to produce large numbers of zones arrayed in two dimensions for large numbers of ions to be trapped in, as well as to reduce the size of the electrodes to the micrometer scale. In these types of traps, which we call surface electrode traps, ions are typically trapped a few tens to 100 microns above the chip surface. An important additional benefit of these microchip surface electrode traps is that they provide a platform for integration of a host of other key ion control technologies. We have the potential to put anything we can currently put in classical computer chips and more into these ion traps. The second key thing we need to do with the ions is to control their internal quantum states. That is, perform the actual quantum gates or quantum operations. Here, we are including the initialization steps of cooling the ions' motions and setting the internal electronic states in which the ions begin a computation. As we already discussed, this is done primarily with lasers. For the types of ions, we plan to use, we require about a dozen different laser wavelengths, ranging from the near-ultraviolet to the near-infrared, pretty much over the whole visible spectrum. These layers are directed and tightly focused on the ions housed inside a vacuum chamber, through the chamber windows using large numbers of bulk optics, like mirrors and lenses. It is a pretty effective way to control a few ions, but it is nearly impossible to imagine how to use bulk optics to address a large ion array. We need to find a way to focus a dozen different colored laser beams on each ion in a highly controlled manner. For example, we will want the ability to hit one ion that resides in the middle of a large two-dimensional ion array with one laser beam without hitting any of the other ions, which would lead to operation error. A dream would be to plug a fiber for each color of light we need into the chip and have that light routed around the chip, much like a metal trace or wire routes electrical signals. This light will be split up into many paths of the chip, and then directed and focused out of the chip plane to each ion location. Perhaps surprisingly, this technology exists. It is called integrated photonics, and we are beginning to incorporate it into ion traps. we can think of integrated photonics as basically fiber optics on a chip. These tiny fibers called waveguides are made by depositing the right materials, which must be transparent over the visible spectrum, onto the trap chips. These materials are then patterned using the same techniques used to pattern the ion trap electrodes. These waveguide patterns define the paths that the laser light travels along, and we can design them to split the light from one path

into many branches. To get the light out of the chip and onto the ions, we can pattern periodic gaps in the waveguide material that act like a diffractive grating. This grating will bounce and focus the light in the vertical direction. This out-coupling process works in reverse. it turns out to be a very effective way of getting the light into the chip from a fiber optic cable coupled to the laser source. Lincoln Laboratory have demonstrated quantum control of trapped ions with light integrated into a surface electrode ion-trap chip in collaboration with MIT. In this case, the waveguides run below the trap's metal electrodes, and we pattern holes in the electrodes so that the light can get through to reach the ions.

29 Trapped Ions: Leveraging CMOS for Integration

In the following section, We introduces CMOS integration technologies that are being developed to control the ion trap electrodes needed to hold and shuttle ions, and integrated photodetectors for readout in larger-scale surface traps for multi-qubit trapped-ion processors of the future. The third thing we need to do with the ions is to measure or read out their quantum states. As explained earlier, this is done by counting photons that are emitted from the ions when a readout laser illuminates them. This readout needs to be done fast, and the speed is determined by the number of photons we can collect from the ion and how quickly and efficiently the photon detector can give us a click when the photon hits it. Ions emit light isotopically, or in all directions, so it is not very easy to collect it all. For few-ion systems, the light collection is typically done with a huge lens, which, due to its size, is located outside the vacuum chamber. It can collect a few percents of the total emitted light and send it to a detector, such as a photomultiplier tube that converts individual arriving photons into short electrical pulses that we can count on electronics. It works well because we only must collect from a tiny region in space where the few ions reside. For large arrays of ions, this technique is highly inefficient. Instead, we are now working to eliminate the large collection lens and integrate the photon detectors into the ion trap chips with a detector right below each ion [161]. Since the detector is located only a few tens of micrometers away from the ion, they can be made very small and still collect the same amount of light as a big lens placed far away. At the same time, these detectors can be arrayed around the ion trap chip in huge numbers to match the size and pitch of the ion qubit array. It can, in principle, be a very efficient and scalable way to collect and detect light in a trapped-ion quantum processor. The detectors that we are using for this purpose are known as Avalanche Photodiodes or APDs. They are routinely fabricated by academic research groups and by industry using the same facilities and techniques for microchip technology. there is, therefore, a clear path to incorporating them in ion traps. We could detect photons emitted from trapped ions using these integrated APDs. The fourth thing we need to do with the ions is moving them around. We call this shuttling, and it provides a capability that is unique to the ion qubit modality. To create the large entangled states required for practical quantum algorithms, we must find some way to move and distribute quantum information around our quantum computer. Ions can be shuttled by changing the voltages applied to the trap electrodes. The voltages are typically generated with electronics located outside the vacuum system and brought to the electronics via long wires that are connected to the chip around the edges. It is done routinely and is straightforward for few-ion systems because there are few electrodes and corresponding voltage sources required for such a small scale. However, for large arrays of ions, we will require lots of electrodes, and therefore lots of voltages and wires. Think about 10 per ion. Once again, we lean on integration to solve this scalability problem. Lincoln Laboratory have begun to develop tunable voltage sources integrated into ion traps, with an individual voltage source below each electrode and connected through on-chip wiring. Commercial CMOS foundry facilities can be used to fabricate these

integrated electronic devices because they are built from the same transistor technology used in classical computer chips. These individual voltage sources called Digital to Analog Converters, or DACs, can all be programmed and varied at high speed with digital signals that come down just a few wires connected to the trap chip, like the USB communication protocol. We would also like to point out that, as we go forward, there are other electronic devices that we can think about integrating, things like circuits that detect the electrical pulses from the photon counters, or even classical computing processors can store and manipulate classical information. As we might expect, we ultimately need to integrate these different control and measurement technologies together into one chip. It means that we need to use a fabrication process that is compatible with it all. It is not a trivial concern since the performance of the required devices is very sensitive to the fabrication techniques and the materials used. We keep this important constraint in mind as we develop the scalable technology, and make sure we are keeping everything compatible. All the technologies We have discussed are completely compatible with advanced CMOS fabrication techniques and materials. It means that we are truly poised to take advantage of all the incredible scalable technology that has been developed for classical computers, propelled by billions of dollars of investment over decades for the quantum computer. It is the vision of what a scalable trapped ion quantum computer will look like. Ions are trapped above the plane of a surface electrode ion-trap chip. They are individually addressed and controlled with laser light delivered by integrated photonics, and light emitted by the ions is detected with integrated single-photon APDs below the trap electrodes. The ions are shuttled around by varying the voltages on these trap electrodes using integrated electronic circuits. The movie shows a quantum computing algorithm, and we may think to ourselves, this does not look like a computer. We would say that we are trying to build something that computes in a fundamentally different and transformative way. we would argue that we should not expect it to look like anything We have ever seen.

Measuring Ions:

The state of an ion is measured by illuminating it with light that resonates with an auxiliary transition, depending on the qubit's state. It means the transition will "light up" or "stay dark." In the following figure, the arrow pointing up indicates that the electron transitions to a higher-energy auxiliary state. The arrow pointing down indicates that the electron goes back to the initial state. If the electron goes from the auxiliary state back to the ground state, it will emit light (light up); if not, it will stay dark.

30 Superconducting Qubits: Introduction

In this section, we will explore the leading qubit modality superconducting qubits. We will take a closer look at superconducting qubits. Superconducting qubits are electrical circuits built from inductors, capacitors, and Josephson Tunnel Junctions that exhibit quantum mechanical behavior when cooled to millikelvin temperatures. It is often said that superconducting qubits are "artificial atoms." Furthermore, what meant by that is that these circuits can be designed to have properties like atoms. For example, the energy level separation between state 0 and state 1, or their sensitivity to environmental noise, can all be determined by design. Furthermore, this is quite different from qubits based on elements that we find in the periodic table, where we just go with what nature provides. Today, superconducting qubits are one of the leading modalities being developed to realize a quantum computer, and there are several advantages. From a business perspective, superconducting qubits are attractive because they leverage a substantial existing technology base. For example, superconducting qubits are fundamentally silicon technology. They are fabricated on silicon wafers. They use standard semiconductor fabrication tools and techniques. Furthermore, materials like aluminum or titanium nitride are fully com-

patible with CMOS foundries. In this sense, they are graphically scalable to large numbers of qubits. Many major corporations are pursuing superconducting qubits, including Google, IBM, and Intel, as well as startup companies like D-Wave [128]and Rigetti. From an engineering perspective, superconducting qubits are designed in much the same way as classical transistor-based circuits. They use the same kinds of CAD software for layout and simulation and have many of the same considerations as larger-scale computer chips, for example, when routing wires to and from the devices. Scientifically, state-of-the-art superconducting quantum processors are pushing 50 to 100 qubits. Furthermore, this is at a level where even the most massive foreseeable classical computers, let alone the ones we have today, simply would not be able to simulate the quantum computer's behavior or its output. When this happens, we will have reached a point that is often referred to as quantum supremacy [162, 163], a point where a quantum computer has performed a task that could not be calculated precisely on a classical computer. Finally, from a technology standpoint, superconducting qubits leverage and rely on existing technologies, such as microwave control electronics and microwave packaging [164, 165], often using the same frequency bands used in cell phone electronics.

31 Superconducting Qubits: How They Work

We will introduce how quantum mechanical artificial atoms can be built from superconducting electrical circuits, including inductors, capacitors, and Josephson junctions. We will also introduce dilution refrigerators, which cool these artificial atoms to near absolute zero degrees Kelvin, and why they are needed for superconducting quantum computing [166, 167]. Superconducting circuits are a quantum computing modality where quantum information is stored in superpositions of charge and current [168]. In a superconducting metal, as we cool it down below some transition temperature, the electrons pair up into what known as Cooper pairs after one of the inventors of the most successful theory of superconductivity, the BCS theory. The B is for Bardeen, and the S is for Schrieffer. Superconductors have many applications. there is a large market for them far beyond for quantum computing. Superconductors have zero resistance at DC frequencies. So, they are useful for applications where we have large currents, for example, to create large magnetic fields. So, if we get an MRI, the large magnetic fields will probably be generated by currents going around in superconductors. If we tried to make those fields out of regular wire, we would have so much heating that it would not work. Superconductors are also sometimes used for maglev trains and low power classical digital circuits. One of the simplest types of the superconducting circuit is just a linear harmonic oscillator made up of some inductor, L, and a capacitor, C. In an LC oscillator, if we put in energy at the right frequency, it will cycle between charging up the capacitor and putting current through the inductor. How fast this happens depends on the values of the capacitor and the inductor. However, for the circuits we are discussing, it happens a few billion times a second or at gigahertz frequencies. It is also the same frequency that many consumer electronics work at, like our Wi-Fi network at home or our cell phone, which means that we get to use much equipment that is being developed over the years to meet growing consumer and commercial needs. A linear harmonic oscillator has equally spaced energy levels. the energy spectra are quantized, meaning we can only have a discrete number of excitations. we can observe this quantization provided our circuit is cold enough that the energy associated with the temperature is much less than the energy level spacing. Our materials are good enough that we do not lose much energy every oscillation period. It is one of the reasons we need to use superconductors. If the metal were normal, we would lose too much energy per oscillation period. However, there is a problem with using a simple linear oscillator as a qubit. If we have a qubit, we want to be able to move it around the Bloch Sphere

into superpositions of 0 and 1. However, in a linear oscillator, all the energy levels are equally spaced. It means that if we are in the 1 state and try to put in the energy to make the qubit go to the 0 states, we can end up in the 2 states or some higher state instead. Thus, we cannot think of the system as a qubit anymore, because a qubit should only have two energy levels. The way to make a linear oscillator into something that can be used as a qubit is to introduce a nonlinear element. That will make it, so the energy splitting between the 0 and the 1 state is different from the energy levels between the other states. The most common way of doing this is to use a Josephson Junction or JJ. Brian Josephson predicted JJ's in 1962. A JJ is just a very thin insulating barrier between two superconductors. If the barrier is thin enough, the superconducting electrons can tunnel through the barrier without losing any energy. The most important thing about a Josephson Junction for the purposes is that the inductance depends on the current going through the junction. It is different from a loop of wire where the inductance is just a fixed value. the nonlinear dependence of the inductance changes the energy spectra, so the levels are no longer equally spaced. It means we can uniquely address one energy level that we are going to make into the qubit. One of the DiVincenzo Criteria is that we must be able to initialize the system. If we want to be able to put the system in the ground state and have it stay there, thermal excitations must not excite the qubit out of the ground state. That means we need to be cold. Exactly how cold this is depending on what the energy levels are. A typical superconducting qubit frequency of 5 gigahertz corresponds to about 250 millikelvins. So, we need to be much colder than that, around 10 millikelvin or so. we can get to these temperatures using dilution fridges. It is a three-case stage, 3 Kelvin, so 3 degrees above absolute zero. We get to that temperature by using a pulse tube cooler, which we might be able to discuss in the background. then to get colder, we use a dilution refrigerator, which uses a mixture of helium-3 and helium-4. Helium-3 is just a lighter isotope of helium-4. the basic concept behind a dilution refrigerator is that it is like the way we cool a coffee cup. we blow across the top of it. what we are doing is removing the vapor. what must happen is more coffee has to come out of the liquid and into the vapor, and that takes energy, and that is how our coffee cools. So, we can use the same trick with helium-3 and helium-4. Dilution fridges used to be very specialized, hard to use, and required liquid helium. Now, in part driven by the interest in quantum computing, companies are making automated systems that are much easier to operate and to maintain. They also do not require a continuous source of liquid helium to run, only electricity.

32 Superconducting Qubits: "Artificial Atoms"

In the this section, we introduces artificial atoms as they are used for superconducting quantum processing, including their coupling to resonators for control and readout, coherence times, and single-qubit and two-qubit gates. One of the neat things about superconducting qubits is that they are macroscopic things, but they act a lot like atoms. They have discrete energy levels, and we can couple them to cavities just like we can with atoms. With atoms, we have cavity quantum electrodynamics, or cavity QED, that describes how light in a cavity interacts with atoms. With superconducting qubits, we have circuit QED, which describes how light in a cavity interacts with superconducting circuits [169]. The math is the same. However, unlike with atoms where we are limited to whatever is on the periodic table, with superconducting qubits, we can engineer the energy levels to be whatever we want by changing the values of the capacitors and the sizes and numbers of the Josephson Junctions. Therefore, superconducting circuits are sometimes called artificial atoms. Because They are like atoms, but we can engineer their energy levels. Here is a picture of a fabricated superconducting qubit. The two squares of metal make up the capacitor. In between, they are a loop of superconducting material that has JJs in it.

In this qubit, it is easiest to think of the encoded information in currents moving clockwise and counterclockwise. To control the qubit, we can send in a pulse of microwave energy at the qubit is frequency [164]. The phase and length of the control pulse will determine how much the qubit rotates around the x and y-axes of the Bloch sphere. So, this instrument is called an arbitrary waveform generator. Moreover, we can think of as mostly the central command for the experiments. So, these are what we are going to load the qubit pulses, the entangling pulses, and the readout pulses. So, it is going to send, for example, our INQ signals to our qubit. It could also send an INQ signal to our readouts. So, this would be enough to control, at a minimum, a single qubit. By multiplexing, that is, by sending signals at multiple frequencies down the same lines, we can use a unit like this to control multiple qubits simultaneously. Typical single-qubit control pulses are tens of nanoseconds, which is very short compared to the best coherence times of 100 microseconds, and fidelities greater than 99.9% have been achieved. Now, we also need to be able to do two-qubit gates between two superconducting qubits. There are lots of ways to a couple of superconducting qubits to each other, including coupling them directly to each other through a capacitive, or inductive interaction, or mediating the coupling with a resonator or another qubit. In many coupling schemes, it is necessary to change the qubits' energy levels, which we can do by changing the magnetic flux through the loop. Two qubit gate times range from tens to hundreds of nanoseconds. Gate fidelities greater than 99% have been demonstrated. Now, at some point in the quantum algorithm, we also need to be able to get information out of the qubit. One common way of doing this is to couple it to a linear resonator, like the harmonic oscillator we discussed before. In this picture, we see a qubit coupled to a readout resonator. Previously, we discussed using an inductor and a capacitor to make a resonator. However, in this case, we are just taking a microwave transmission line with a distributed inductance and capacitance and is also a resonator. The qubit interacts with a resonator just like an atom interacts with a cavity. Moreover, we end up shifting the resonator frequency by a different amount if the qubit is in the zero states or the one state. So, by sending a pulse of microwave energy to interrogate the resonator, we can discuss the qubit's state. Readout times vary but can be as short as hundreds of nanoseconds.

33 Superconducting Qubits: Manufacturing Artificial Atoms

In this section, we will introduces how superconducting qubits can be manufactured with high coherence using modern CMOS-compatible toolsets. [170–172]. We often say that superconducting qubits are artificial atoms. What we mean by that is that we can build electrical circuits that behave much like the natural atoms on the periodic table. For these artificial atoms, we can design all their properties, their energy level spacing, their sensitivity to noise. It is quite different from qubits based on natural atoms, where we are limited to what we are given by nature. Thus, a key advantage of superconducting qubits is that we design these superconducting quantum circuits and their properties. A second advantage is that superconducting qubits are silicon technology. We manufacture superconducting qubits on silicon wafers using the same tools photolithography, metal deposition, and metal etch. That is used by industry to make Complementary Metal-Oxide-Semiconductor, or CMOS, transistors. The active elements we use, Josephson Junctions, have thin oxide barriers, just like transistors' thin gates. Even the metals we use aluminum, titanium nitride, niobium are all compatible with CMOS fabrication. Thus, in an authentic sense, superconducting qubits are silicon technology. Moreover, this affords us lithographic scalability. It is a straightforward path to increasingly complex circuitry with many interconnected qubits. There are a few differences in the specific process temperatures and other processing parameters used for CMOS and superconducting qubits. This is related

primarily to the need for pristine materials and fabrication in order to maintain high qubit coherence. Superconducting qubits have a variety of ways they can lose quantum information, or what we call decoherence. Several examples of decoherence channels are illustrated here, including charges fluctuating on the device surface [173, 174], trapped magnetic vortices, and stray magnetic or electrical fields. Many of these channels can be enhanced and suppressed by the materials and fabrication choices we make in manufacturing superconducting qubits, as well as by their design. Research within the superconducting qubit community for the last 20 years has focused on identifying and mitigating decoherence sources to improve qubit performance with tremendous success. We have seen more than five orders of magnitude improvement in qubit coherence within these 20 years. In 2018, multiple major modalities of superconducting qubits with unique parameters tuned for different applications had achieved coherence times that exceed the most lenient thresholds for quantum error correction. This is one of the reasons superconducting qubits are at the forefront of demonstrations today.

34 Superconducting Qubits: Fabrication

In the next section, we describes the tool chain used for fabricating superconducting qubits, and the process flow, in this section. The three main steps include the first deposition of superconducting material, a second large-scale patterning of the pristine material, then a third fine-scale patterning of the actual qubit.

Patterning superconducting qubits requires state of the art fabrication technologies. For example, MIT Lincoln Laboratory fabricate superconducting qubits in the 70,000 square foot micro-electronics laboratory. Within this clean room have a full suite of 90 nanometers CMOS tools, and also have dedicated superconducting deposition and etch tools. These dedicated superconducting tools are essential for limiting magnetic contamination sources, which is one of the decoherence within qubits. When we are fabricating superconducting qubits, we first deposit pristine materials. We then work to keep them as pristine as possible by minimizing, processing, and choosing the parameters carefully. Many processing steps have the potential to introduce sources of fabrication induced loss, and We have systematically studied ways to mitigate this potential. There are three main steps in the baseline superconducting qubit fabrication process. First, we prepare substrates and deposit a pristine layer of the superconductor, often aluminum or titanium nitride. Second, we pattern this pristine material into essentially everything for the superconducting qubit circuit, other than the qubit loop. This can include readout and control circuitry, such as resonators that interact with the qubit, the device ground plane, and shunt capacitors, which can store energy from the qubits. Third, we add the qubit loops, which contain Josephson Junctions. Josephson Junctions are thin oxide barriers sandwiched between two layers of superconductor. We often use aluminum as this qubit loop superconducting material. After we fabricate the qubit wafers, we conduct extensive testing of the devices. We then wire-bonded package some chips from the wafers to cool down and measure in the dilution fridges. We fabricate the devices on either 200-millimeter manufacturing style wafers or smaller 50-millimeter prototyping wafers. For the 50-millimeter prototyping wafers, we focus on quickly turning around new designs for rapid testing. For the 200-millimeter manufacturing scale wafers, we focus on the high yield of defect-free complex designs. We now will walk through these main process steps in more detail and highlight at each stage some of the key considerations, starting with the preparation of the silicon substrate and deposition of the high-quality base metal. Silicon substrates have a native surface oxide that is about 1 and 1/2 nanometers thick. The silicon oxide can contain dangling bonds and is a source of loss for superconducting qubits. In order to remove this oxide, we first do a wet chemical etch, and then we load the wafers immediately into the

molecular beam epitaxy or MBE system. Using the MBE, we further prepare the silicon surface by annealing at high temperature and reconstructing the top monolayers of silicon. This is a molecular beam epitaxy deposition system, and this is critical in making superconducting qubits because it forms an essential part of the qubit, the very base metal. the metal we typically use is aluminum. This forms the capacitors, and the wiring, and the connections to the Josephson Junctions, which form the qubit itself. The aluminum comes from these effusion cells, which is a fancy tool, but all it means is that we have a little cone-shaped crucible where we put our aluminum in, and it is melted through these wires around the circuitry. the specific amount of power that goes into it controls the temperature of the aluminum down to under a tenth of a degree. That means we have precise control over the vapor pressure of the aluminum, which means we control how fast the aluminum film is deposited into the tool. we can monitor, in situ, the growth of these aluminum films using those high energy RHEED gun, which stands for reflective high energy electron diffraction What this gives us is a two-dimensional diffraction view of the surface of the wafer. So, we can see the silicon wafer, two, the desorption of the hydrogen on the silicon wafer, and three, the deposition in real-time of the aluminum on the wafer. Next, we pattern the high-quality metal into the shunt capacitor's control and readout circuitry and the ground plane. We pattern a layer of optical resist onto the superconducting base metal to define the features of interest. The pattern is transferred into the underlying superconducting material using either a wet chemical etches or a plasma etch process. Afterward, we strip the resist mask using a chemical stripper. Sources of loss within superconducting qubits can be attributed either to the wafers' materials or to the fabrication process. As a proxy for assessing the loss within superconducting qubits, we can use coplanar waveguide resonators. Lincoln Lab is useing quarter-wave resonators that are capacitively coupled to a center feed line to deposit the same metal as the qubit base metal, and pattern, and etch using identical processes. The standard chip layout has five resonators that are each spaced 200 megahertz apart in frequency by varying the length of the resonators. When cooled to milli-Kelvin temperatures, which have passed the superconducting transition point, we can look at a loss within these resonators as a function of photon number. We assess performance at the single-photon limit, where on average, a single photon is in the resonator cavity. As of 2018, we typically see single-photon quality factors of 500,000 to one million for aluminum and one to 2.3 million for titanium nitride.

35 Superconducting Qubits: High Coherence Qubit Loops and Josephson Junctions

We will describes how high-coherent superconducting qubits are lithographically patterned and fabricated using modern semiconductor fabrication methods. The section includes introduction of cleanroom, where the fabrication process and the actual equipment employed in the process. Next, We will look at the fabrication of high coherence qubit loops and Josephson Junctions. Starting from the patterned base metal, the next step is to lithographically define the region where we will deposit the qubit loops and the embedded Josephson Junctions. We start by depositing a stack of three layers of material. First, we spin on a layer of methyl methacrylate resist that we use as a spacer layer. Second is a layer of germanium, which we use as a hard mask. For the prototyping process on top, we coat a layer of ZEP electron beam or e-beam resist. We pattern the ZEP using an electron beam lithography system to define the qubit loops. Lincoln Lab use a Raith e-beam system. The 100 kilovolts system has a 50-megahertz clock speed, which enables reasonable write times of the nanometer-scale patterns. We say reasonable because we raster an electron beams back and forth across the wafer's surface rather than exposing full dye at a time as we would do in photolithography. This is a slower process

than photolithography, but we gain patterning flexibility for rapid prototyping. Using the e-beam system patterning lines down to less than 10 nanometers on the system are demonstrated. For Josephson Junctions, we routinely pattern sub-150 nanometer features. Alternately, we also can pattern the qubit loops using stepper photolithography. For manufacturing scale wafers, Lincoln Lab use a 193-nanometer wavelength ASML scanner, which enables to pattern features smaller than 100 nanometers. Optical lithography enables orders of magnitude speedup in write time compared to e-beam lithography, since now we are exposing much larger write areas at a time, rather than rastering a nanometer-scale beam across the wafer. After the resist is exposed, by either e-beam lithography or photolithography, we transfer the pattern into the germanium hard mask using a plasma etch. We use plasma etching to pattern features on a smaller scale than what we can do with wet chemistry. we use gas, an ionized gas, and electric field to etch small metal features, silicon features, or oxide features to make the layers in our integrated circuit. After etching the germanium, we etch the spacer methyl methacrylate layer using an oxygen plasma etch. This exposes the silicon substrate and metal contact regions. Besides, this oxygen plasma simultaneously removes the ZEP top layer of resist. In addition to defining the open area for the qubit loops, we also pattern small germanium bridges, where We will pattern the Josephson Junctions. To release the bridges, we over-etch and undercut the methyl methacrylate. Zooming in on a cross-sectional view of the freestanding germanium bridge, we can schematically show the shadow evaporation process used to define the Josephson Junctions. All the shadow evaporation steps happen in situ within different chambers of the same vacuum system. Since we will be making superconducting contact between the qubit loop and the underlying base metal capacitive shunts, we first must prepare that base layer by removing the metal's native oxide. To do this, we sputter away the oxide using argon ions. We then transfer the wafer into the deposition module and put down the first layer of aluminum. Afterward, we move the wafer into the oxidation chamber, flow in oxygen, and allow it to oxidize the aluminum for a specific amount of time to reach the target oxide thickness. The target oxide thickness depends on the desired qubit parameter for the Josephson Junction critical current. Next, we move the wafer back into the deposition chamber, tilt the wafer to the opposite angle, and deposit a second layer of aluminum. We are now located at the PLASSYS shadow evaporation system, a tool that we use to fabricate one of the critical components of the superconducting qubits. This is where we deposit the Josephson Junctions and the associated qubit loops that surround them. So, what we do in here is four steps. First, we load a wafer into the load-lock. Then we load it into the etch chamber once it has pumped down to vacuum. In this etch chamber, we first ion mills the surface as a method to prepare it before we put down the Josephson Junction layers. Now in this etch chamber, we are preparing that top surface by removing the top layer, about a nanometer of aluminum oxide before we come in and deposit these junctions and qubit loops. So, then we move from the etch module over to the deposition module. In this module, we first put down the bottom layer of what will define the Josephson Junctions and qubit loops. This connects directly to that MBE material. From there, we move into the oxidation module. This module's job in life is to be able to put down very pristine, very uniform aluminum oxide layers. It flows in oxygen into the chamber when we have this exposed aluminum film, and it oxidizes to a precise thickness that We have tuned. From there, we move back into the deposition module. by tilting the stage differently, we can use the e-beam defined shadow mask to create a small layer of overlap between the bottom layer of aluminum. that is now being oxidized and a top layer of aluminum. That layer of overlap is specifically the Josephson Junction for the superconducting qubit loop. From there, the process is complete, and we move back out to the load -lock and take the wafers out. After shadow evaporation, we remove the wafer from the deposition system and lift off the resist stack in a chemical solvent. Once the resist is removed, wafer fabrication is complete. The completed wafer contains the superconducting qubits with integrated Josephson

Junctions, as well as base metal pattern and capacitive shunts, the ground plane, and readout and control circuitry.

36 Superconducting Qubits: Testing

In this section, we introduces the use of data-driven process monitoring for assessing fabrication yield and device parameter spreads. The next stage is room temperature testing. we have to do extensive automated testing on every wafer that we fabricate. We conduct data-driven process monitoring to assess device performance and drive the process development. We test each component of the superconducting qubit system. Two examples include checking the critical current density of the Josephson Junctions and measuring the contact resistance between the base metal and the qubit loop shadow evaporated metal. Each wafer is tested using an automated wafer probing station. After we load the wafer, we do thousands of four-point probe measurements using a 26-pin probe card combined with a switch matrix. At each probing location, we apply current to a four-wire test structure and measure the voltage drop. The switch matrix re-assigns the current end voltage locations for all test structures accessible to the 26-pin probe card. Then, the probe card is lifted and transferred to a new position where the measurements continue. After testing is complete, the results are automatically databased. Besides, some devices can be selected for further cryogenic testing, either at liquid helium temperature of 4.2 Kelvin or at millikelvin temperatures in one of the dilution fridges. As one example of room temperature testing, we measure the resistance of Josephson Junctions with varying junction lengths ranging from 100 nanometers to three microns. We plot the inverse of the resistance, which is the conductance, as a function of the junction length. We use the slope of that plot to extract out the low temperature critical current density, JC, of the wafer. On the prototyping 50-millimeter wafers, we measure the critical current density at six identically patterned process monitor chip locations spread across the wafer. In this example, we targeted a critical current density of three micrograms per micron squared. We met that target average critical current density and had less than 2% cross-wafer variation. In a second example of the test structures, we measure contact resistance between the metal layers. We show a false-color scanning electron microscope image of contact between the base MBE aluminum metal in blue and the top shadow evaporated Josephson Junction metal in orange. Additionally, we also check for continuity of millimeter long snaking lines and isolation between interdigitated combs. We use these measurements to check for any particle defects and consistency of lithographic patterning. After testing, the last step is to select wafers for dicing and packaging. Wafers are diced using an automated water-cooled dicing saw. The qubit chips are then packaged, and wire bonded to make connections from the outside cabling to the chip circuitry. From there, the chips are loaded into one of the dilution fridges for measurement.

37 Superconducting Qubits: Why 3D Integration?

We will describe in this section why three-dimensional Integration is needed for superconducting qubit chips and illustrate conceptually how this can be achieved using through-silicon vias and stacked wafer technology [166,167]. So far, people have been working with small arrays of qubits, up to around a few tens of qubits. At some point, however, we run into a fundamental problem of where to put everything. We want many qubits that are coupled together. However, if we look back at the picture, we can see that the actual qubit is only a small portion of what needs to go on the chips. We have bias lines, control lines, readout resonators, for example. those can all be bigger than the qubit. Now, we can imagine making some of these smaller. However, eventually,

if we have a large array of qubits, it will be hard even to get the wiring to the qubits in the center if we are working in two dimensions. most of the demonstrations to date have been limited to geometries where we can laterally access the qubits on the surface of a chip. This problem is not unique to superconducting qubits. Many applications have this same issue. Consider, for example, a large-scale imager with lots of pixels. we need to be able to get our signals out of the pixels. However, we also want to be able to put many pixels on a chip to fill a 2D array. The one-way industry has solved this issue by moving to 3D Integration, where signals are brought vertically instead of laterally. We think it is necessary to move to 3D Integration to make large scale superconducting qubit circuits. However, we need to be careful when doing so because the qubits are affected by things that are not an issue with classical electronics, which is what 3D integration was developed. For example, one way to efficiently route wires in 3D is to build up a multi-layer stack of metal, with a dielectric in between so wiring on different layers can cross each other. However, we know that having lossy dielectric materials near a qubit can cause it to lose its quantum state. So, we have developed a new idea for how to get the benefits of 3D Integration, efficient wiring, and the ability to bring in signals vertically without affecting the qubit. Here is the proposed scheme, which has three separate silicon chips that are held together with indium bump bonds. Within the three stacks, each of the layers is fabricated separately. We have seen that there are process incompatibilities between the processes in separate layers of these three stacks. For example, we must stay at a relatively low temperature once We have deposited shadow evaporated Josephson Junctions on the qubit layer, the interposer layer. However, we need to have processed at a higher temperature for the readouts and interconnect wafer. By separately fabricating each layer, and combining them at the end of the process, we can combine the best processes and optimize the capabilities of the full stack. The top layer of the three stacks contains the qubits. Here we show two examples, capacitively shunted flux qubits [172], that we use for quantum annealing applications [175–179], and transmons, another type of qubit that we use for gate-based quantum computing. As of 2020, state of the art single qubit coherence times for each of these styles are on the order of 50 to 100 microseconds. Simultaneously work is going on to increase qubit coherence further and retain that coherence as we move into the third dimension with increasing interconnect complexity. The second tier in the three stacks is the interposer wafer, where Through Silicon Vias, or TSVs, that are lined with superconducting metal, route signals between the two sides of the wafer. The interposer provides three key benefits. First, it provides an isolated mode volume for each of the high coherence qubits to retain coherence times comparable to those in a planar device geometry. Second, the interposer provides a nearby surface for inductive or capacitive coupling across the vacuum gap between the qubit plane and the interposer plane. This interposer surface can be used for control and readout circuitry, or for coupling qubits that bridge two devices on the qubit plane. The third benefit of the interposer is that it connects the qubits with the readouts and interconnect multi-layer wafer on the bottom layer of the three stacks, while still having the qubits keep a healthy distance from the lossy dielectrics on that multi-layer interconnect module. Moving down to the multi-layer readout and interconnect layer on niobium devices. We previously fabricated tri-layer Josephson Junctions for superconducting qubits, which turned out to have lower coherence than the aluminum shadow evaporated Josephson Junctions that we use today. We also currently fabricate fully planarized multi-layer niobium devices with integrated tri-layer Josephson Junctions for both Single Flux Quantum, or SFQ circuits, as well as near quantum, limited traveling wave parametric amplifiers, or TWPAs. For the three-stack configuration, we envision embedding Josephson Junctions within the multi-layer niobium wiring that could be used for active circuit components, such as on-chip TWPAs. Last, after the fabrication of each of these three separate wafers, we need to assemble the complete circuit. To do this, we are developing a double bump bonding approach with indium thermal compression bonding. Indium, which is superconducting at the millikelvin operational

temperatures, can be used both for the mechanical stability of the wafer stack and for electrical connectivity. We align the chips, using a precision bump bonder system. We use the feature in feature alignment marks, where part of the alignment structure is on each wafer, both during alignment, and as a check afterward of how well we align the two surfaces. We have seen that we routinely achieve better than one-micron alignment between the chips. In addition to lateral alignment, tilt control is also critical for consistent inductive or capacitive coupling across the vacuum gap. Using multiple techniques, we have demonstrated tilt control between wafers of less than 250 micro radians.

38 Superconducting Qubits: How 3D Integration Works

In this section, we will focus on the fabrication procedure used for superconducting qubit systems. How through-silicon vias allow a three-dimensional circuit structure to be realized [166, 167]. Unlike most traditional silicon fabrication processes used for conventional CPUs, multiple wafers are employed for this process. This enables chips made with superconducting qubits to be stacked together with other chips, e.g., for readout, and potentially, classical control circuitry. Such high levels of Integration are likely needed for future quantum computing systems.

Let us now take a deeper dive into the fabrication of the superconducting TSV interposer wafer. First, we fabricate the TSVs, fill them with superconducting material, and pattern the metal on the wafer surface. Next, we mount the interposer wafer to a temporary carrier wafer, flip over the TSV wafer, and remove the excess wafer thickness down to the TSVs. Afterward, we add metal to this revealed surface. Fabricate control circuitry or qubits onto that surface. This revealed side is the surface that ultimately will be near the qubit layer qubits. After all the processing is complete, we dice the wafer and release the individual chips from the temporary carrier wafer. In the end, the chips are available for bump bonding into the three stacks. After the wafers are fabricated, but before we dice and release them, we do extensive room temperature and cold temperature testing to confirm that the TSVs are superconducting. Here, we see a 200-millimeter TSV wafer, which has 52 identical dies. On each die, we have several TSV test structures in addition to the active circuits that we are using for the qubit stacks. We use the process control monitor chips to assess the individual metal properties, as well as four-point probe structures, to look at both single TSVs and chains of TSVs. As one example, we have links of 400 TSVs in a series that we can probe to assess the resistance. When we look at the room temperature resistance of these 400 TSV chains, we see that, on average, we have 37 ohms of resistance per link. Of even more interest to us at room temperature is that we see the standard deviation is only two ohms, which shows a high degree of uniformity across the wafer. Afterward, we took a subset of these devices and cooled them down in the dilution fridge to assess the superconducting transition temperature. The devices go superconducting around 1.6 Kelvin, and that the midpoint is 3.1 Kelvin. It is well above the millikelvin operational temperature. In advance of having the full three stacks available for testing, in 2016 and 2017, we also conducted several experiments looking at components of the eventual three stack. Since qubits are so sensitive to materials and processes, the first experiment we did was to take a regular single-chip qubit and design a flip-chip version of it where all the inductors and capacitors had the same values. However, the qubit chip was flipped and bonded to another chip. We wanted to make sure that the extra processing and the presence of another chip bonded to the qubit chip did not affect the coherence time. Here is what the circuit looked like for the single-chip qubit. The chip has six superconducting flux qubits, and each coupled to a bias line and a readout resonator. For the flip-chip version, we took all the control and readout circuitry and moved it to another chip. The only things left on the qubit chip were the qubits themselves. Then we bonded the

two chips together. The figure on the right shows an infrared image looking through both chips and showing that structures on one chip are well aligned to those on the other chip. We found that the relaxation and coherence times of the single-chip and flipped chip qubits were virtually identical. This is interesting for a couple of reasons. First, it shows that 3D Integration does not significantly degrade qubit performance, which is essential. Second, we have demonstrated that we can move all the control and readout circuitry to another chip, which we are planning to do for the full 3D integration scheme. In this experiment, we stuck with the same general design because we wanted to isolate the effect of 3D Integration. In future work, we are planning to shrink the resonator and other components so they can fit under or between qubits in a 2D array. We are excited to be continuing to develop further and further demonstrations using this three-stack scalable architecture. Within both digital quantum computers and simulators and analog quantum systems, there is a strong need for high connectivity between qubits and significant control and readout complexity. Although there are differences between digital and analog quantum algorithms, both systems have similar 3D scalability needs that are requiring significant engineering developments. We are planning to use this three-stack hardware to scale to testbeds with 100 qubits or more. We will then use what we discuss from these systems as a stepping-stone to future large-scale demonstrations of quantum computers.

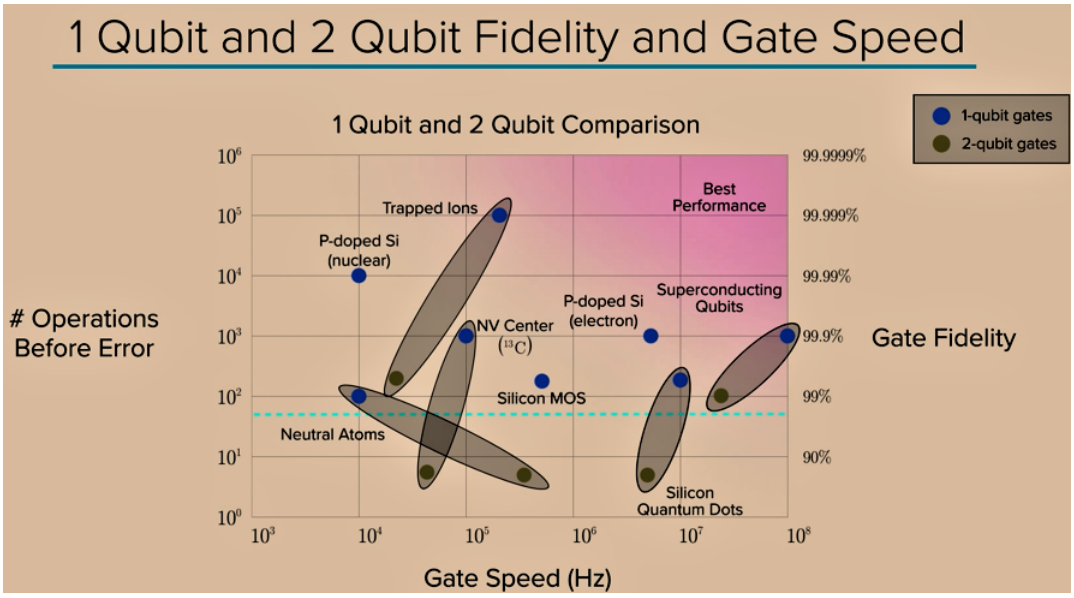


Figure 33: 1-Qubit and 2-Qubit fidelity and gate speed

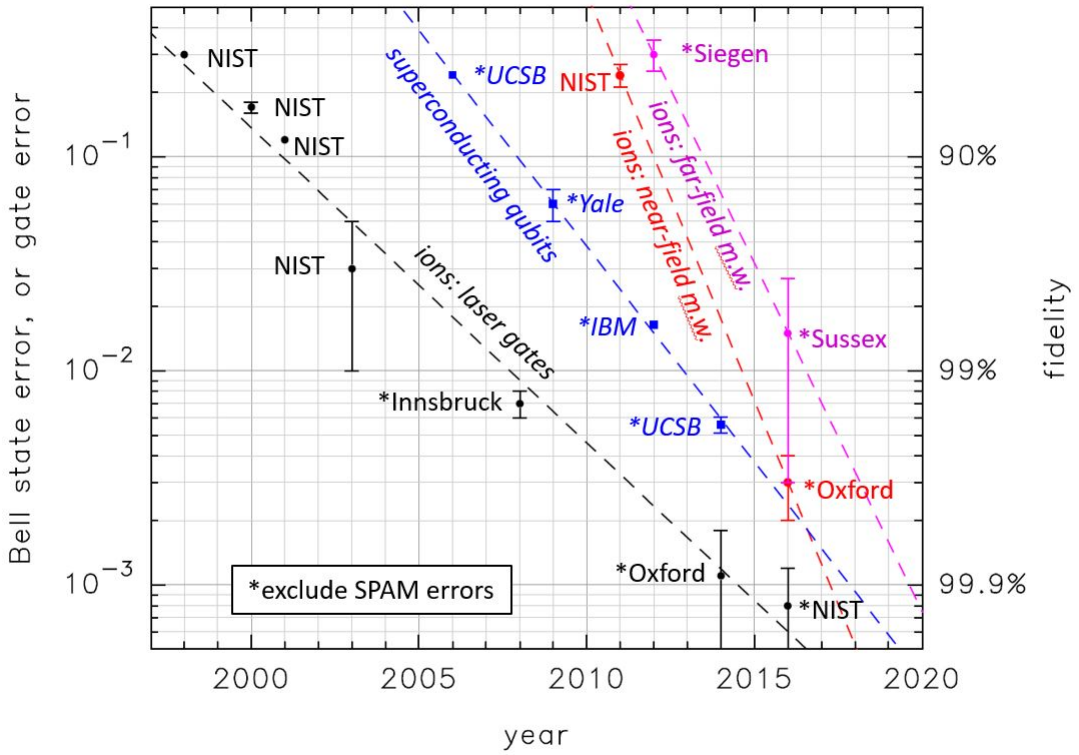


Figure 34: 1-Qubit and 2-Qubit fidelity and gate error

39 The Promise of Quantum Computers

we have discussed how a quantum computer differs from a classical computer, and we have seen examples of the tremendous potential of quantum computing. In this section, we will turn to a more practical question: How can one realize the promise of quantum computing and quantum communication? Let us begin with the promise of quantum computation.

In this section, we will look at quantum algorithms and quantum communication. We will start by considering the promise of these technologies, what is needed to realize that promise, how to get there, and where we are today. Let us begin with quantum algorithms and the promise of quantum computers. As we discussed earlier, quantum computers are not smaller, faster versions of classical computers, nor are they an incremental step in the evolution of Moore's law. Instead, quantum computing represents a fundamentally new approach to processing information. Furthermore, it is the only computing model we know of today. That is qualitatively different from existing computers. This difference and the potential promise of a quantum computer is reflected in the fact that simulating a quantum computer using a classical computer requires exponential overhead. This means that, as we add one qubit to the quantum computer, the size of the classical computer required to simulate it will grow exponentially. Now we do not often experience exponential growth in the everyday lives, and so, to gain an intuition for how powerful it can be, we may remember the old riddle where our employer asks us how we would like to be paid this month, and we have two choices. Either We will start us with one penny, and then each day for one month, We will double the number of pennies and keep the pennies from the last day.

Alternatively, we can have $1 million. Now, it is entirely reasonable to think through the first few days of pennies. For example, two pennies on day one, four pennies on day two, double that to eight pennies on day three. Moreover, we begin to think that $1 million is starting to look pretty good. However, in fact, if we take that $1 million, we have lost a lot of money because the number of pennies is growing exponentially as 2 to the n-th power, where n is the number of days. So, on the 31st day, the last day of the month, we have 2 to the 31 pennies. That is equivalent to about $21 million. That is the power of exponential growth. Returning to computing, if we consider the amount of memory, we would need to store all the available states in the state space of a quantum computer We will similarly see the immensity of exponential growth. For example, representing all the quantum states of just 30 qubits would require at least a very powerful laptop. Increase that to just 40 qubits, and now we need a supercomputer. Double that to just 80 qubits, and we would need all the classical computers we have on Earth. Then double that again to 160 qubits, and now we need all the silicon atoms on Earth. Even small increases in qubits translate to exponential growth in the amount of memory required to represent all 2 to the n numbers that n qubits can store. So, there is a big difference between classical and quantum computers. Furthermore, this suggests the potential for substantial quantum advantage. However, what is needed to realize this in practice? To realize a quantum advantage, we need to identify the intersection of three essential requirements. First, we would like to identify a useful problem, one that has practical importance. Second, this problem should not have a known fast classical algorithm to solve it. Otherwise, we might as well just use a classical computer. Then third, the problem must have a known fast quantum algorithm, one that offers a substantial speedup. Now, this sounds great, but in fact, this promise has been mostly empty, except for a small area of overlap that we know about today. Nonetheless, there are useful problems with a quantum advantage, and this improvement can be quantified using a mathematical expression. One place is in the prefactor of that expression [180], and one place is in the exponential. The prefactor A can be a large fixed number, or it can scale polynomials with the number of qubits n. The exponential scaling of a problem of size n is in the exponential factor. Either of these two types of improvement can indicate quantum advantage. In the following sections, we will compare several algorithms and their respective speedup.

In the section above, an example of the power of exponential growth. In the salary example, if we earn two pennies on the first day, and double that amount each day for 31 days, then we will receive 2^{31} pennies on the last day of the month, equivalent to over $ 21M dollars. This growth rate is termed exponential because the output $y = 2^x$ depends exponentially on the number of inputs x (in this case, the number of days). In computer science, algorithms can be classified according to the scaling of the resources required to run an algorithm as the number of inputs the problem size grows. An algorithm is said to scale exponentially if the resources (time, space) needed to run the algorithm grow following an expression such as $y = c^x$, where c is a constant greater than 1 and x is the problem size. Two other examples of growth functions are linear and polynomial. An algorithm is said to be linear if it can be solved by using$y = ax$resources, where a is a constant, and polynomial if it uses $y = ax^c$ resources where c is a constant greater than or equal to 1. The linear case is itself an example of polynomial scaling with $y = c = 1$.

40 Understanding Quantum Algorithms

Realizing the promise of quantum computing relies on developing new quantum algorithms that offer a quantum advantage. In this section, we will review several algorithms that are known today and present several different approaches to the development of new quantum algorithms.

In the last section, the promise of quantum computing and its potential for quantum ad-

vantage. However, although a quantum computer can do anything a classical computer can do, it often does not improve. we only have a small set of useful algorithms at the disposal today. So, what can we do to realize the promise of quantum computing? Quantum advantage starts with having useful quantum algorithms. Developing new algorithms is very challenging, in large part because quantum computers are based on quantum mechanics. Thus, developing an algorithm for a quantum computer is quite different from doing so on a classical computer. To understand how quantum algorithms work and what they can do. There are currently a couple of different approaches. One is to develop algorithms based on mathematical theorems and their proofs. Shor's algorithm and Grover's algorithm are both excellent examples. Based on either intuition or insight into the problem, a quantum algorithm is first proposed. then, it is theoretically determined if the algorithm is efficient or not. For example, proving that an N-bit number can be factored with high probability, in a time that scales polynomials with the size of that and that N-bit number, rather than exponentially. Again, this type of development generally requires insight into a structure of the problem, that lends itself to enhancement on a quantum computer. A second approach is to use classical computers to simulate quantum computers' behavior, to glean some insight into algorithmic primitives that can then be used to realize quantum enhancement in larger systems. Now, as we might imagine, simulating a quantum computer with a classical computer is not very efficient. After all, if we could easily simulate a quantum computer, we would not need to build one. Currently, classical supercomputers, with significant effort, can simulate about 50 qubits. Various probabilistic and Monte-Carlo techniques have been developed to further increase the qubit number, in exchange for accuracy or completeness in the solution. Again, these kinds of classical simulations can provide insights into how a quantum algorithm works at small scales [181]. So, it can be leveraged in quantum algorithms run on larger scale quantum computers. Beyond small scale simulations, a third approach is just to throw caution to the wind, build a quantum system, and see what happens. The D-Wave system is a great example of this approach. D-Wave has built a quantum annealer with more than 2000 qubits [182], a fantastic engineering achievement [128]. Although there is, yet, no theoretical or experimental evidence pointing to quantum enhancement in quantum annealers, at least for machines and problems studied so far, there remains a lot that we do not know. as we discussed previously, because optimization problems are so important, there is a tremendous application pull to develop quantum-enhanced optimization tools [183]. Thus, getting a real machine into the hands of engineers, computer scientists, and algorithm designers is a great way to start discussing what is and is not possible with the various quantum computing approaches. So, developing quantum algorithms is very challenging, and tremendous efforts have been applied to their development. it is because of this that we have useful, or potentially useful, algorithms today. Examples in the user now category are Shor's factoring algorithm [184]and Grover's search algorithm. Shor's algorithm can be used to break public-key cryptosystems and afford exponential speedup over known classical algorithms. Doing so at a meaningful scale requires a medium-sized quantum computer, with thousands of logical qubits [185]. Although we do not have such quantum computers yet, they are foreseeable. research today is oriented towards developing new, quantum-resistant public-key cryptography standards, as well as a new generation of quantum algorithms to break them. Grover's algorithm is related to a general class of search, collision, and optimization algorithms that afford a quantum advantage over known classical algorithms [186]. However, currently, more research is needed to help make these algorithms practical. There are a couple of reasons for this. One is the data loading problem. For example, we can efficiently load large amounts of data into a quantum computer so it can be efficiently searched. Also, there is currently a rather limited range of problem sizes that admit benefit. Namely, problems that are large enough to make use of the quantum advantage, and yet not so large that it becomes practically prohibitive to operate on a human time scale. For

example, a quantum enhancement that reduces the runtime from a few thousand years to a few decades is obviously a fantastic improvement. However, very few people would find that useful. Although Shor's and Grover's algorithms are perhaps the most well-known examples of a quantum algorithm, it is widely anticipated that quantum simulation will have the greatest economic impact in the future. The simulation would apply broadly to a range of problems, from solid-state and nuclear physics to material science and chemistry. these are problems of importance to both science and industry. Classical simulations of quantum systems are generally limited to very small problem sizes, or an array of simplifying assumptions, such as no entanglement, or semi-classical dynamics [181]. Exact solutions are generally intractable because the number of variables needed to describe a fully quantum system grows exponentially with the problem size. It has been shown that quantum simulations can efficiently model quantum systems. the hope is that some of these problems may see the quantum advantage, on a smaller scale, or even error-prone quantum computers. Whether or not this is achievable remains an open question. Other quantum algorithms in the "may be useful" category include sampling solutions to linear equations [89], adiabatic optimization problems, and machine learning [187]. In these cases, the ultimate degree of quantum advantage will depend on several factors, including input-output complexity, such as the loading problem. We mentioned earlier and making a meaningful connection to applications. Besides, to offer a useful advantage, these algorithms must also outperform existing heuristics. Classical computing methods that, while not guaranteed to give an exact or optimal answer, do give very high-quality solutions. That is often enough for many applications. In summary, developing quantum algorithms that exhibit quantum advantage is at the heart of realizing the promise of a quantum computer. in the next section, we will look at the degree and type of quantum enhancement these algorithms provide. In the context of D-Wave quantum annealing versus gate model quantum computing [188, 189]. so, what is the potential of D-Wave's quantum annealing machine versus, IBM or others' quantum gate machines? And more specifically, which model will be a better choice for businesses in the immediate future? The universal gate model machines we know if we can build and engineer that system, there are problems that will have a quantum advantage when run on a quantum computer. But to get to that point requires fault tolerance and error correction. Now, in contrast to that, a quantum annealer addresses specifically classical optimization problems. it is not theoretically known whether a quantum annealer will afford quantum advantage or not over classical computers. There is no theory that says it will, nor is there a theory that says that it cannot. so, that is an open question. Now, it is true that there are many, many problems that can be broken down into optimization problems. Optimization problems are ubiquitous [190]. so, there is a huge application pull to develop computers, whatever they may be, that can solve optimization problems better than we can currently do. that is really the motivation for quantum annealing [191, 192]. we simply do not know if There is going to be quantum advantage there. Now, it could be that a quantum annealer is just a much better classical computer. if that is the case, it still has value. we think we still hope that There is a region of the parameter space where quantum annealers do show quantum enhancement. What we can say is that the quantum annealing in particular, the system that the D-Wave company has built, is really a fantastic engineering achievement. They very quickly, over a decade, went from just a few qubits up to systems with more than 2,000 qubits. They co-locate cryogenic electronics to control that machine. That is a fantastic achievement. But as a system engineer will know that to do that, they had to restrict the flexibility of that machine to a degree that it can scale quickly to those large sizes. so, the reason we bring that up is that there is some hope that, even though quantum enhancement has not been observed yet for a general class of problems the phase space that has been tested so, far is rather narrow. For example, only zz-type coupling interactions have been demonstrated. The degree of connectivity is limited to one specific type. the coherence times of the qubits that are used in

the D-Wave machine are rather low when compared to the coherence times that we have on gate model machines. so, by moving to different types of coupling for example, xx-or yy-, sometimes called non-stochastic type couplings or to moving towards much higher coherence qubits, there is a hope that this may actually be a region of parameter space where quantum enhancement can be found.

41 Quantum Advantage

In this section, we will compare the resource scaling for several algorithms when implemented on a classical computer and a quantum computer. An improvement (reduction) in the resource-scaling is a quantifiable metric for the degree of quantum advantage.

In the last section, we introduced several algorithms, including those like Shor's and Grover's algorithm, that are useful now if we can build a quantum computer that is large enough and robust enough to run the algorithm. We also looked at quantum simulation, a very promising set of algorithms which, when realized, has the potential for substantial economic impact, and may even find utility on today's smaller scale, non-error-corrected machines. we looked at a few potentially useful algorithms currently under development. For each of these examples, let us now look at the resource requirements for implementing an algorithm on both a classical and a quantum computer. It will then show the degree of quantum advantage one can obtain by using the quantum computer. We will also identify the leading limitations as They are currently known for realizing this advantage. Let us take them in order of their applicability as we see it today. At the top of the list is a quantum simulation [193] with application to quantum chemistry and material science [194,195]. The classical resources scale as 2^N-th power for simulation of N atoms, whereas the quantum resources scale as N to a constant power C, where C generally ranges from 2 to 6. Now, by resources, we mean both the time required to reach a solution, referred to as a temporal resource, and the number of logic or memory elements needed to implement the problem, often referred to as a spatial resource. To quantify the resource requirements, we quote a mathematical expression for the scaling law. We are looking at how the resource requirements grow as the problem gets larger, as parameterized by the size N, for example, the number of atoms being simulated. Presumably, as N gets larger, the problem gets harder. However, the question is, how does it get harder? Does it require exponentially more resources, such as 2^N-th power, or a polynomial scaling, like N to an integer power? For quantum simulation, we see that the classical resources scale exponentially with N, whereas the quantum resources only scale polynomials. It means that there is an exponential advantage to using a quantum computer. To simulate the system dynamics for a time t, both a classical and a quantum computer would require a similar number of time steps. For example, if we want to simulate the dynamics of a reaction for one second, we would divide it into a similar number of time slices. However, the quantum advantage is that there is an exponential reduction in the amount of memory needed to perform the simulation on a quantum computer. Thus, the quantum advantage is exponential. The main limitation is in determining the mapping of a physical problem onto the qubits, their couplings, and the gate operations needed to implement a quantum simulation. Next, factoring and related number-theoretic algorithms also have an exponential scaling in the number of classical resources, going as 2 to the n-th power and the number of digits being processed. In contrast, the quantum resource requirements scale polynomial, as N to the third power. Thus, the quantum advantage is again exponential. The main limitation, or perhaps uncertainty, is that the best-known classical algorithm has not been proven to be optimal. So, there may still be a more efficient classical algorithm yet to be discovered [196]. if so, then the degree of quantum advantage might also change. Sampling solutions to linear systems of equations [197] is the next algorithm. In this

case, we solve linear algebraic problems of the type $ax = b$, where a is matrix, b is a known vector, and x is an unknown vector. The classical resources scale exponentially as 2^N-th power, whereas the quantum resources scale only approximately as N. so, this algorithm also exhibits exponential advantage. The main limitation is related to a variety of restrictions on operating conditions. For example, a requirement for a sparse matrix A. The classical resources required for optimization problems also scale exponentially, as 2^N-th power, where N is again related to the size of the problem. However, in this case, the corresponding quantum resources, and the quantum advantage, are not well-defined. It is in part because one generally cannot determine if the resulting answer is indeed optimal. We can only tell if it is better than a solution We have had previously. Thus, we can only derive empirical evidence that a quantum optimization algorithm provides a quantum advantage. this is currently the main limitation of this algorithm. Finally, there is Grover's search algorithm for unsorted or unstructured data. Here, the scale of the classical resources with the number of data elements N, whereas the quantum resources go as the $\sqrt{N}$. Thus, the quantum advantage is the $\sqrt{N}$, a polynomial enhancement. The main limitation of this type of search algorithm is the data loading problem. Namely, how can we efficiently load in a large amount of unsorted or unstructured data that needs to be searched? In general, we can see that, for many, if not all these algorithms, there can be a substantial quantum advantage when the problem size N becomes large. for those with an exponential advantage, classical computers, and Moore's law-like scaling will never be able to catch a quantum computer's performance [198].

Developing applications that demonstrate a clear quantum advantage is challenging. However, there are several areas where the quantum advantage is known to exist (if we can build a large enough quantum computer to run the problem at scale).

Simulation of quantum systems: Quantum simulations of quantum systems afford an exponential advantage in the amount of space (memory) over classical simulations. Essentially, quantum systems generally comprise a very large number of variables to monitor during a simulation, and these can be handled more efficiently on a quantum computer.

Shor's factoring algorithm: has an exponential temporal advantage over the fastest known classical algorithm for factoring. While it has not been formally proven that an efficient classical algorithm does not exist, it is believed that it does not.

Linear systems of equations: An exponential quantum advantage has been proven for linear systems that satisfy certain properties. From a matrix A and a vector b, the quantum algorithm can find a vector x such that $Ax = b$ as long as A is sparse and has a low condition number.

Grover's search algorithm: has a proven polynomial speedup over the best possible classical search algorithms. This advantage is still subject to the data loading problem: the data to be searched must also be loaded into the quantum computer efficiently. Overcoming this loading problem is an active area of research.

It is challenging to prove that a quantum computer offers a quantum speedup for optimization problems. There are two main reasons for this. First, although one can generally identify a candidate solution and confirm that it is better than a previous solution, knowing (or proving) that the solution is optimal is generally not possible. Second, near-optimal solutions are often sufficient in practice for most problems. For example, in finding the shortest route home over a distance measured in kilometers, solutions that differ by a few meters are practically interchangeable with the true optimal solution. Proving speedup is thus confounded with the notion of "good enough," making such proofs challenging. There is currently intense research to find quantum algorithms that provide a quantum advantage for finding higher-quality solutions in less time than is possible on a classical computer.

The following table compares the resource scaling for several algorithms that feature a quantum advantage. The resource scaling with system size N is shown for both classical and quantum

versions of the algorithm when performed on a classical and quantum computing. The resulting degree of quantum advantage is shown along with current known limitations to the quantum algorithm implementation.

QUANTUM ALGORITHM SPEEDUPS

ALGORITHM	CLASSICAL RESOURCES	QUANTUM RESOURCES	QUANTUM ADVANTAGE	LIMITATION
Simulation (quantum chemistry)	2^N (for N atoms)	N^C	Exponential*	Mapping problem to qubits
Factoring (+ related number theoretic)	2^N (for N digits)	N^3	Exponential	Classical runtime limit unproven
Linear systems (Ax=b)	2^N (for N digits)	~N	Exponential	Strict conditions, e.g. sparse matrix
Optimization	2^N (for N items)	?	?	Empirical
Search (unsorted/unstructured data)	N (for N entries)	$\sqrt{N}$	Polynomial ($\sqrt{N}$)	Data loading

Figure 35: Quantum Speedups

42 The Promise of Quantum Communication

Quantum communication is qualitatively different from classical communication. In this section, we will discuss the promise of quantum communication and how to realize this potential by looking at what criteria must be met to realize the quantum advantage.

Just as quantum mechanics can dramatically enhance the way we process information, it also has the potential to enhance the way we communicate information. So, in this section, let us look at the promise of quantum communication. Communication is broadly defined as the conveyance of information, and it encompasses sending that information as well as receiving it. It certainly refers to verbal communication between people as well as nonverbal forms such as written text and signatures. It often comes in an encoded form. For example, the digital zeros and ones that are created by sampling the voices on a cell phone or representing the words of an email that are then routed from one computer to another. Even information on a chip must be shuttled around, communicated between a processor and its memory. Moreover, this concept can be extended to include several spatially distributed computing nodes, all of which are available to attack a problem but must be coordinated and require communication between these different nodes. All of these are examples of classical communication. However, quantum communication is different, and it represents a fundamentally different means to encode, convey, and authenticate information. These differences and the potential promise of quantum communication are reflected in three basic facts or theorems. The first is that classical information encoded in qubits can be transmitted two times faster than by classical means. This is achieved using a concept known as superdense coding, and it is a provable quantum enhancement. Two bits of classical information can be transmitted using a single qubit when that qubit is part of an entangled pair of qubits called a Bell state [**?**,199]. Now, two qubits, but We only needed to transmit one of them to send two classical bits. That is superdense coding. The second is that quantum bits cannot be copied or cloned. Moreover, it is, again, a provable consequence of quantum mechanics. Now, if We prepared a qubit in each state, we know that state and so We can prepare a second qubit in the same state. However, if We have a qubit in an unknown state, we cannot make an exact copy of it,

and that is called the no-cloning theorem. The third fact is that attempts to intercept or measure quantum bits can be detected. The no-cloning theorem cannot duplicate those bits. Thus, an eavesdropper must measure in order to glean any information. Moreover, it is the measurement process that always leaves a signature that is detectable in some way. What is needed to leverage these three facts and realize the promise of quantum communication? To realize the quantum advantage, we need to identify problems at the intersection of three key applications. The first is to find a problem that can leverage the enhanced capacity to encode information that quantum mechanics enabled. Second, there are problems related to authentication—the ability to confirm an individual's identity or the truth of an attribute. Furthermore, third are problems associated with collaboration, the collaboration between people or agents such as secret sharing or game theory [200]. For example, enhance means to auction or to vote without attribution. Alternatively, using quantum mechanics to verifiably share public goods in a way that avoids certain rational yet self-destructive tendencies such as the tragedy of the commons. Realizing the promise of quantum communication will require finding problems that exhibit quantum advantage. In the next section, we will look at several such problems.

Communication describes the conveyance of information between a sender and a receiver. Some familiar examples of classical communication include verbal or written language, encoded information transmitted from one location to another via smartphones or the internet, and fetch calls made by a classical processor to its memory unit. Classical communication beyond speaking or writing is generally mediated by classical states of light or electricity, incapable of assuming quantum superposition.

Alternatively, the field of quantum communication considers the advantages that can potentially be found by communicating via the transmission of quantum states. Such quantum communication is fundamentally different from classical communication with respect to encoding, conveying, and authenticating information. These differences, and the potential promise of quantum communication, are reflected in the following three basic concepts.

The first concept is called superdense coding, also referred to as quantum dense coding (QDC). QDC is a fundamental quantum communications protocol that conveys two bits of classical information through the transmission of a single qubit. This protocol makes use of quantum entanglement as a resource and requires that the two users [201], traditionally referred to as Alice and Bob, have access to pre-shared entangled qubits.

As an overview, the protocol presumes that Alice and Bob have already pre-shared entangled qubits, such that each has one qubit from an entangled pair. Alice wants to send information to Bob, and so she proceeds to encode her information by performing one of four operations to the qubit in her possession. She then sends this qubit to Bob, so that Bob now has both entangled qubits. Bob then performs a joint measurement on each of the qubits now in his possession. By doing this, he is able to determine which of the four operations Alice performed during the encoding phase, and hence reveal two bits of classical information (since $\log_2(4) = 2$). Thus, Alice manipulates and sends one qubit to Bob, and in doing so, effectively transmits two classical bits of information.

The important feature of quantum mechanics, which enables this protocol to work, is the presence of entanglement, which provides access to a larger space of possible states. More specifically, if Alice and Bob did not share entanglement between their qubits, then the most information Alice could encode on the qubit she transmits to Bob is $\log_2 = 1$ bit, since the state space is spanned by two orthogonal states (usually written as $|0\rangle$ and $|1\rangle$) it would just be a single, classical bit. However, due to the nonlocal nature of entanglement, Alice can hold just one of the two entangled qubits and still have access to the larger state space of the entangled qubits one that is spanned by the four orthogonal Bell states. Whichever Bell state Alice and Bob initially shared, Alice can change it to any of the other four through the application of one

of the four following unitary operations to her local qubit: Identity, X-gate, Y-gate, or Z-gate.

To see how this works, Let us look at the four distinct Bell states for qubits A and B and label each with a unique binary number (00, 01, 10, 11):

$$\begin{aligned}
|\Phi^+\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle_A|0\rangle_B + |1\rangle_A|1\rangle_B\right) \quad \rightarrow \quad 00 \\
|\Phi^-\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle_A|0\rangle_B - |1\rangle_A|1\rangle_B\right) \quad \rightarrow \quad 01 \\
|\Psi^+\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle_A|1\rangle_B + |1\rangle_A|0\rangle_B\right) \quad \rightarrow \quad 10 \\
|\Psi^-\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle_A|1\rangle_B - |1\rangle_A|0\rangle_B\right) \quad \rightarrow \quad 11
\end{aligned} \tag{37}$$

As an example, we assume that Alice and Bob initially share a $|\Phi^+\rangle$ state. If Alice wishes to send 00, then she need only send her qubit to Bob (the Identity operation). However, if Alice wishes to send the bits 10, then she must first perform an X-gate on her qubit, indicated with the subscript;
$A : (X \otimes I)|\Phi^+\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_A|1\rangle_B + |1\rangle_A|0\rangle_B\right) = |\Psi^+\rangle$, with similar relations for the $Y-$ and $Z-$ gates. Once the encoding is complete, Alice transmits her qubit to Bob. With possession of both qubits, Bob is able to perform a measurement known as a Bell-state measurement. This measurement allows him to determine which of the four Bell states the two qubits are in, so he can decode the message.

While QDC offers a simple, clear indication of how classical and quantum communications differ, there are several important challenges to its real-world implementation. The first is that distributing a Bell state between distant users, a requirement to begin QDC is not trivial in practice. One of the most promising methods currently for this is to use photons entangled in such degrees of freedom as polarization, frequency, or time of arrival. Unfortunately, a single photon is extremely fragile. Therefore, it is likely to be corrupted or lost during transmission through free-space or optical fiber over distances of tens or hundreds of kilometers. A second challenge is how to store those pre-shared entangled photons before they are needed for communication. A third challenge is that the most straightforward ways to perform Bell-state measurements as currently do not distinguish between all four Bell states. For example, a setup might not be able to distinguish between the $|\Psi^{\pm}\rangle$ states, and hence only $\log_2$ (3) bits are conveyed with a single qubit. For these reasons, the rate at which information can be conveyed using QDC is at present significantly less than the rate at which classical communication can be performed in practice, making QDC an important protocol to study from a theoretical perspective, but unlikely to be of practical use any time soon.

The second concept is the no-cloning theorem of quantum mechanics, which states that an arbitrary unknown quantum bit cannot be copied. This is a fundamental consequence of the linearity of quantum mechanics. It is important to stress that this theorem only applies to unknown quantum states, for if the state is known, it can be identically prepared over and over again. However, given a qubit in an unknown state, there is no means to copy it and end up with two qubits in the same unknown state. This concept has far-reaching consequences, one of the most direct of which is that quantum states used for quantum communication cannot be amplified, since an amplifier should essentially make many copies of a state and sends them along the transmission path to amplify the signal. Efforts to overcome this limitation have resulted in significant research into what are called quantum repeaters. These devices potentially enable longer distance transmission of quantum states, but at a heavy technological price.

The third concept is that attempts to intercept or measure a quantum state are detectable,

a pillar of secure quantum communication. The no-cloning theorem rules out duplication of an unknown qubit state. Therefore, a potential eavesdropper has no choice but to measure some aspect of a qubit to glean its information, and this measurement process always leaves a detectable signature. A sender and receiver, therefore, know when an eavesdropper is present.

Quantum communication holds the potential for quantum advantage. This promise is found at the intersection of three broad applications: source authentication, secure communication, and collaboration, each of which will be described in the next section.

43 Understanding Quantum Communication

The quantum advantage potential for quantum communication can be understood in the context of the number of participating parties, from two-user communication to multi-user distributed applications. In this section, we will discuss several such examples that illustrate the promise of quantum communication.

In the last section, we discussed the promise of quantum communication and its potential for quantum advantage. That advantage was based on three tenets of quantum mechanics, namely quantum-enhanced channel capacity, a concept known as superdense coding, the no-cloning theorem, which forbids copying an unknown quantum state, and a form of communication non-disturbance, by which any attempt to intercept or measure quantum bits can be detected. We can begin to understand how these tenets enable the promise of quantum communication by considering the number of participants as simultaneously participating in that quantum communication. The first example is the point to point secure communication using quantum key distribution. Secure communication relies on the use of a private key, essentially a string of bits used by a cryptographic algorithm. Quantum key distribution provides a means to transmit and share such a secret key securely. The secret key can then be used to encrypt and decrypt information. Any attempt to intercept that key can be detected, and when this happens, the compromised bits are simply discarded and replaced with additional secure bits. In this way, quantum mechanics enables two parties to communicate securely. Besides, We will discuss later, and quantum mechanics can also be used to generate true randomness, a resource for many cryptographic applications, and a step above existing classical pseudo-random number generators. Second are applications involving a few participants. One important example is quantum secret sharing amongst two or more people. Let us say Alice wishes to send a secret to both Bob and Charlie. She would like them to learn that information simultaneously so that neither Bob nor Charlie can have an advantage over one another by learning the secret first. Using quantum communication to distribute an entangled state shared between Bob and Charlie, Alice has created a situation where Bob and Charlie must coordinate to uncover what that secret is. They can coordinate on an open classical channel, but quantum mechanics will ensure that they receive the information simultaneously. Finally, the third set of applications involves multiple distributed participants or multiple distributed computing nodes that, in concert, perform a quantum algorithm. One example is quantum scheduling, which uses Grover's Algorithm to search for a time when n distributed people are available. Another example is related to distributed computing systems, such as the leader election problem. In this application, a unique leader or a master node must be chosen from amongst all the distributed computing nodes. In both cases, it is known that quantum communication and quantum computing together provide an advantage over classical approaches. Despite the promise and numerous examples of problems where quantum communication gives an advantage, there are only a few practical applications to date. One application that is available now, even commercialized [202], is quantum key distribution or QKD. In QKD schemes, Alice and Bob want to share a private key, and any attempts by

Eve to intercept it can be detected. There are demonstrations of QKD using photons over 100 kilometers of optical fiber and even a demonstration of transmitting signals between two points on earth using satellite quantum communication [26]. QKD even serves as a foil to quantum codebreaking by Shor's Algorithm. We have already discussed, Shor's Algorithm can efficiently break RSA public key encryption. It is also efficient against Diffie-Hellman key exchange and elliptic curve cryptographic schemes. However, messages encrypted using a one-time pad are, in principle, secure. One-time pad schemes use each bit in a key only once and then discard it. They, therefore, need to be continually renewed, and this can be done securely with QKD. The main issue with QKD security is that, as with most cryptographic schemes, there are many channels for the attack. For example, one must generally control access to the hardware used to implement a QKD scheme [203]. Another limitation is signal attenuation. Classical communication links use repeaters to regenerate and amplify a signal. However, due to the no-cloning theorem, this is not straightforward with quantum communication. Instead, We will discuss later in the section, a quantum version of repeaters has been proposed that can effectively extend communication lengths using quantum teleportation and distributed entanglement [204]. In conjunction with quantum memory, distributed entanglement would enable a quantum communication link to be established [205]. Other applications that may be useful include quantum secret sharing, auctioning or voting without attribution, verifiable quantum digital signatures, and message authentication. For all these examples, developing robust protocols and hardware systems that are immune to compromise will be necessary to realize the promise of quantum communication schemes fully.

44 Fundamentals of Quantum Communication

44.1 Non-Cloning Theorem

The no-cloning theorem states that it is impossible to replicate with certainty an arbitrary quantum bit. We can illustrate this statement by trying to come up with an operator $\hat{U}_{clone}$ that can clone a qubit. Let us assume we have found a unitary operator $\hat{U}_{clone}$ that can clone single-qubit states $|0\rangle$ and $|1\rangle$.

$$\hat{U}_{clone}|0\rangle \rightarrow |0\rangle|0\rangle \qquad \& \qquad \hat{U}_{clone}|1\rangle \rightarrow |1\rangle|1\rangle \tag{38}$$

However, if we apply this to an equal superposition state, following the rules of linear algebra, we find:

$$\begin{aligned} \hat{U}_{clone}\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) &\rightarrow \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|1\rangle) \\ &\neq \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) \end{aligned} \tag{39}$$

As this example suggests, it is impossible to define a $\hat{U}_{clone}$ that can clone any arbitrary quantum state. This is a consequence of linear algebra and the nature of quantum mechanics, and it is called the no-cloning theorem.

44.2 Non-Disturbance

Non-disturbance describes the subsequent alteration of a quantum state due to measurement. The measurement process projects the quantum state on a particular predefined basis and causes an identifiable change. This process can be illustrated using the Bloch sphere. Suppose our

predefined measurement basis is the z-axis. A measurement of a qubit in a superposition state will project it onto the z-axis. The resulting quantum state after projection (aligned with the z-axis) differs from the original superposition state before the measurement (it was not aligned with the z-axis). To conclude, attempted measurements of an arbitrary quantum state alter the quantum state and are, therefore, detectable.

44.3 Quantum Key Distribution

Quantum Key Distribution (QKD) is a communication method that uses unique features of quantum mechanics to exchange a secret key to encrypt and decrypt messages securely. In the underlying case, the keys are generated via a quantum channel between two parties, Alice and Bob. The principles of non-disturbance enable the detection of potential eavesdroppers on their quantum channels [206]. If the measured level of anomalies in their quantum measurement is below a certain threshold, the key is considered secure. Once a secure key is generated, it can, in turn, be used to encode and securely transmit information. The method of generating a secure key using quantum mechanics is known as quantum key distribution.

45 Industry Perspective: Introduction

In this section, we will discuss several leading industrial and start-up efforts targeting quantum computing. To get the discussion started, we asked several baseline questions. First, what technological approach are we employing to realize quantum computers, and why? Second, what is the business application we have in mind for the quantum computing systems we are developing? Third, what are the major technical hurdles we are facing? Fourth, how will our approach contribute to the advancement of scientific knowledge? Furthermore, fifth, what other industries are either vertically or horizontally integrated with respect to our efforts? We also add in other perspectives and thoughts not captured in these questions. The results are very interesting and informative. Thus, with that brief introduction, let us discuss the major commercial players in quantum computing today.

46 Industry Perspective: IBM

So, if we want to make a qubit, first we must decide what kind of qubit we want to make at IBM, we like to use superconducting circuit qubits. So, as the name already suggests, it is a circuit approach to qubits, and we use many superconductors. the favorite superconductors here are niobium and aluminum. we also make them on the silicon wafers. So, there is much expertise that we have locally and how to process and make them on the silicon platform. one ingredient that all the different superconducting circuits share is a Josephson junction, which is a little bit of an unusual circuit element. However, the way we make it is, we have two pieces of superconductors that are very weakly coupled. We just separate two superconducting electrodes by a very thin tunnel barrier. That is just an insulating layer. it is like 1/1000 of a hair thick. So very thin. This Josephson junction behaves like a non-linear inductor, but then to make a qubit, we need a second element, a capacitor. we make like of regular linear capacitor out of superconducting material. Then, tying these two together gives us an LC resonator that can behave as a qubit if we tune the parameters correctly. So now we have a qubit. In order to make a useful qubit processor, then we need to tie these different qubits together. We use microwave buses to couple them. then lastly, we provide input and output through a readout resonator to each of the qubits that then couples to the outside. Right over here is one of the

dilution refrigeration systems, which cool down the superconducting qubit devices. There are several different plates inside here, which all sit at different temperatures. That gets down to 15 millikelvins. That is colder than outer space itself. The sound we hear is a pulse 2 compressor, which essentially is pumping on a closed cycle of helium 4, which helps us get the system cold. We have much other equipment that we use in order to run the processors. So, there is a lot of microwave hardware, different passive components including filters and attenuators, and co-ax cables, which allow us to send the signals down to address the qubits and to read them out to us the controllability. This is a four-qubit package. We have a five-qubit device that is right now inside of the fridge, but the general idea for how we package up these devices and cool them down is the same. this is a printed circuit board to which we mount the qubit chips. we wire bond to them to connect to essentially these co-axial pins. These co-axial pins connect to cables inside the fridge, allowing us to send the signals down and take signals back and read them out. We are not going to have quantum computers in our pocket. We will have quantum systems in the cloud that we will be able to access. for many people, day-to-day basis, they may not even know that their information is coming from a quantum computer in the future. However, they would benefit from the value that quantum computers have created. By the time we have 50 qubits or so, that system no conventional classical computer, that We have ever built or could ever build, could emulate what that 50-qubit quantum system will have. we are going to see it in the years to come. A central question in general that one must answer here is considering the amount of effort and the challenge that it is to build a quantum computer, why do we want to do this? Moreover, the answer is that there are applications, or there are algorithms for which we know these algorithms vastly outperform their classical counterparts. So, we may be familiar with Shor's factoring algorithm or Grover's search and so forth, just to name a few. What is important to keep in mind here is that these algorithms require a universal fault-tolerant quantum computer [207–210]. At the current stage, we are still quite off from that. Several challenges must be overcome. The hardware must improve. We must develop better error-correcting codes so that the demands of the hardware become less. Nevertheless, at the same time, there has been steady and considerable progress in the experiments, with increased coherence rates and increased gate fidelities. We now enter a regime where it becomes challenging for classical computers to simulate these devices. Now, once we build increasingly large devices, we can ask ourselves, well, if this device is hard to simulate, is there already something meaningful that We can do with this? Furthermore, this is the question of trying to find near-term applications for quantum computers. Considering that we do not have a fault-tolerant computer, we must start making some concessions. Number one is we are going to have to look at algorithms or heuristics that do not come with a clean analysis, as they do for the cases as mentioned above. The other thing is that we must develop algorithms that are closer to the hardware that we are working on. lastly, we must think about errors constantly while developing those algorithms. We are currently focusing on three areas of applications: quantum simulation, quantum heuristics for optimization, and machine learning with quantum computers. One must keep this dichotomy to near-term and long-term in mind because many of the current algorithms that have been developed rely on having a fault-tolerant quantum computer. here, it is the task or goal basically to develop applications that are more closely tailored to the actual hardware that we are dealing with. Work is being done to develop error-correcting codes for use with quantum computers. Does the IBM Cube use an error-correcting code, or does it perform computations on a best-effort basis? If it does, what are the error-correcting codes it uses? And how does it implement that? So, many groups, including IBM, but groups worldwide, are focused on developing and demonstrating error-detection and error-correcting codes. today, there have been, we would say, very small-scale demonstrations of error detection by IBM and Google delved. there has been some demonstrations of error detection and correction, for example, at Yale, again,

on smaller scale systems. But if we are asking about the online quantum computer that IBM's using today, the answer is that it does not use error correction, active error correction, in this sense. As we will discuss there is a hierarchy to an architecture for building a quantum computer. at the bottom of that architecture, are the physical qubits. those qubits are they are good. But they are faulty. They have error rates of 10^{-3}, to 10^{-4}. to really run a computation at scale for a period of a day 24 hours, let us say we really need error rates at 10^{-15}, to 10^{-20}. So, to achieve that, we need to do active error correction. But in between the hardware and the active error correction, there is another layer that is been developed. this is used by groups worldwide. this is called passive error suppression or error mitigation [209]. what this is, essentially, dynamical decoupling. by applying the same types of gates, single and two-qubit gates, mostly single-qubit gates, that we use to control and run an algorithm, control the computer and run an algorithm, those same gates can be used to mitigate certain types of errors basically, coherent errors, errors where we do not lose information to the environment, but the qubit is dephasing in some way. those types of errors can be mitigated using dynamical decoupling. so, by applying these pulses, mitigating these coherent errors [211], we can make the fundamental physical qubits last even longer. They have even better error rates. that is important because error correction then, active error correction, is very expensive. the better the qubits are to begin with, the less overhead one needs to implement full-op error correction. so, what we would say is that we are not doing active error correction today, generally. But what we are doing, we would say, quite broadly is doing this passive error suppression. we want to call it error correction. So, we will call it passive error suppression or error mitigation [212]. Another part is, what is the status of error-correcting code development as it pertains to trapped ion qubits and semiconducting qubits? just to reiterate, we think in both cases, these technologies or the researchers who are pursuing these technologies are first working on error detection. So, to perform error correction, we first have to identify if and when an error occurred and on which qubits. the magic of quantum error correction, is that through specially designed syndrome measurements, we can make measurements which do not project the quantum information that we are trying to preserve. But they project onto a space where that information we gather is was there an error or not. So, we never discuss, for example, whether the qubit was in state 0 or state 1. We discussed that there was a bit flip error and that a 0 became a 1 or 1 became a 0. we discussed which qubit it happened on. so, with that information, we can go back and flip the qubit back to its correct value. But we never discussed what the correct value was. We never discuss whether it was supposed to be a 0 or supposed to be a 1. We only discussed that a bit flip error occurred, and we should flip it back. So, that intuitively the type of or the way that error correction works and to do that, we first have to detect if there was an error. We have to perform a syndrome measurement. So, many groups are working on doing those syndrome measurements for different types of error-correcting codes. Some of them are relatively simple codes, such as the bit flip code or the phase flip code. Some of them are more mature codes, like the surface code [213]. So, the surface code [214], for example, requires what is called a wait for parity check [215]. researchers, are trying to demonstrate that detection with high efficiency. then once their detection is mastered, then we can run error correction. We can go back in and correct for those errors once we discuss where they are.

47 Industry Perspective: Google

As with any information technology, quantum computing requires an entire stack. So, we go from application software to programming environments, all the way down to the hardware that realizes the actual quantum operations. for the hardware, we are currently banking on superconducting qubits since it is a technology suite that is complete. Thus, we know how to

initialize a qubit. We know how to manipulate a qubit, or several qubits, by gate operations. we know how to measure those qubits. we know to do all those operations with rather high fidelity. However, we also keep a keen eye on competing technologies, such as silicon qubits or topologically protected qubits. However, at this point, we feel good that with superconducting technology, We will be able to build a small error-corrected quantum computer. On the software side, the most natural way to offer users access to a quantum processor is through a cloud interface. For example, a superconducting qubit the processors, they sit in a refrigerator that keeps them at about 10 millikelvins. it is too bulky to put this under our desk. However, it is no problem to have those in a data center. we abstract this complexity away from the user by offering a cloud API through which we access these processors. the roadmap for developing applications looks as follows. We will first start by passing a waypoint, which is known as quantum supremacy. Quantum supremacy roughly means the point at which a quantum computer, or some quantum device, surpasses the equivalent available technology on classical computers or classical devices. We are particularly interested in computational quantum supremacy, which means that a quantum computer will surpass for a well-defined computational task what we can do with a state-of-the-art supercomputer and state of the art classical algorithms. We call it the way post because the quantum supremacy benchmark task is not necessarily tied to something useful. So, it is important to realize that once we have achieved quantum supremacy, we are not yet in the phase of error-corrected quantum computing. To do error correction, like in classical error correction, we need redundancy. Unfortunately, the ratio of physical qubits to logical qubits to have one well-protected logical qubit is very high. It is about 1,000 to one. So, in order to have 1,000 logical qubits, we would need about a million physical qubits. we are still several years away from that. So, about quantum supremacy. it says, it looks like quantum supremacy has not been proved yet. Why do we need to invest our resources in quantum technology-related projects now before quantum supremacy has been proven? we would say that the aspect of quantum computing that there is a quantum speed-up that would, if we could build that system, demonstrate quantum supremacy that has been proven mathematically, that there are problems for which a quantum computer, if and when we build it, will provide quantum speed-up. Now, this second part of that question related to quantum supremacy has not been demonstrated yet. we think that we are close. we will demonstrate on probably a computer on the order of 100 qubits a calculation of some type that cannot be efficiently simulated on a classical computer. we think that that will happen. that will be a concrete demonstration of a problem for which there is quantum supremacy [216, 217]. we think that will happen within a year or two. Many companies are focused on doing exactly that, which is making a demonstration of quantum supremacy. Now, will that be for a problem that is useful that we will then use in our company to solve some problem that we have? That remains to be seen. We do not know yet. It could be that quantum supremacy is first demonstrated on a problem which may be of importance to, fundamental physics. It may be a problem that demonstrates some aspect of physics, but still needs to be translated into a practical problem for businesses. But we think that the concept of quantum supremacy is much more important than that. The concept is that we have demonstrated, in a real physical system, this concept that there are tasks that a quantum computer can perform and come up with an answer, and we can check that It is the right answer, but that a classical computer just cannot do efficiently. it will be a concrete demonstration. Even though we know mathematically, we have proven that this should be the case and is the case, we think having that physical hardware and demonstrating it directly is important. Nevertheless, we are optimistic that there are several areas in which we can develop algorithms that can already do something useful before error correction. There are probably three applications that We would encourage investigation. One is the quantum simulation. So, this is simulating quantum systems that cannot be simulated by even the world's biggest supercomputers. The second area would be

on discrete optimization. So, this is optimizing very complicated optimization problems. the third is machine learning. In terms of verticals, the verticals that We have seen much interest in would be automotive, aerospace, chemistry, pharmaceuticals, and, more broadly, just defense. the reason why We would say that these are interesting, especially automotive and aerospace, is that both these kinds of industries have both complicated optimization problems, but they also have chemistry-related problems [218]. For example, accelerating the development of batteries for electric cars. Right now, there are many issues with the battery technology we use. For example, it uses a rare material, such as cobalt. So, the number of batteries with today's technologies that we can build. It is estimated to be 80 to 90 million batteries only, and then all cobalt reserves are exhausted. So, we need to find alternatives. there are just a lot of areas within the electrochemistry of a battery that, with quantum simulation, we possibly can better understand and devise better materials. Quantum processors are finicky beasts. They are cooled to below outer space temperatures. they need to be robust against noise and any interference that happens. so, one of the biggest challenges is just isolating these systems and keeping them in their quantum state. There are two main sources of noise for the systems. there is coherent noise and incoherent noise. we work to reduce both through excellent fabrication, also through the control electronics and the readout as well. Also, scalability is a big challenge for these systems. We are operating in a 20 to the 50-qubit range today. We are looking to scale up to 1,000, eventually a million, qubits. we must invent new technologies as we hit each of those certain milestones. What gotten we to one or two or 10 qubits is not going to get us to 1,000. so that scalability with the packaging, the electronics, the control is key for us to build a system that can scale up to a large enough size that we can run error correction on these chips, which we need. We all know how important computers are in our lives. We think about quantum computing. We have made a change into a realm where the basic rules are different. So, we think understanding the power of quantum computing is important for science, in general. In recent years, everyone has been very hyped, for a good reason, about machine learning and artificial intelligence. people have also been very excited about the possibility of using quantum computers to speed up computation. So, there is a natural question, can we put these things together? Furthermore, to that end, we have personally been thinking about specific setups for quantum computers to run machine learning tasks. We hope that we will discover some areas in classical machine learning where the quantum computers' additional power accelerates or speeds up the machine learning tasks or allows for more reliable predictions, or perhaps learns to learn with fewer data coming in. Another area, which We think is sort of an obvious approach to use quantum computers, is if our data itself is not classical. So, we all know we can store classical data, like images, on classical computers. However, if We gave us a quantum system and We just told we it was a quantum state, that cannot be represented by a long bit string. The quantum state is too complicated. But nonetheless, quantum states exist in nature. So, we could ask, if We presented a quantum state to a machine a quantum computer could it learn to distinguish some attribute of that state which a conventional computer could not do? Moreover, the conventional computer might not be able to do it because it cannot even accept the input. Alternatively, even if somehow or another, we can encode the input, the quantum computer is just more naturally inclined to be able to answer questions about quantum states. So, we think that is a whole other area for quantum machine learning [219], which needs to be explored. For more information about Google's Quantum AI research, [220–223].

48 Industry Perspective: Microsoft

Microsoft, are building a full quantum stack, and the approach we are taking harnessing topological qubits that perform longer or more complex computations at a lower cost and higher accuracy. What it allows us to do is store and process quantum information in a more protected way. It is stored in a topological state of matter, these Majorana quasiparticles. What remarkable is that it has built-in fault tolerance and error correction. The actual value of the qubit is stored in a joint parity state, say over a nanowire, where that state is stored across two wires to form a qubit. Moreover, the computation then occurs by moving these quasiparticles around one another. It becomes what we call a braid in space-time, and this braid in space-time is the actual computation. Now it turns out these braids can be implemented through measurement only. So, we do not ever move the quasiparticles, but we can do joint measurements of these quasiparticles, and through sequences of joint measurements, we can enact and implement quantum computation. So, while other quantum approaches have more qubits right now, by number, in the short term, right, we believe the topological approach will deliver orders of magnitude, better accuracy, and scalability in the long term. When We think about the first business application, we think about how We will bring quantum to Microsoft's cloud, Azure, and allow it to be integrated directly with people and organizations, apps, data, and infrastructure that is already existing. Moreover, we see users in Azure using quantum in concert with advanced classical conventional processing and storage. A quantum system, it will be unlike any other system built to date, and there are a handful of challenges that we must address as we seek to scale this system up. One challenge is the development of a robust, scalable qubit. We have been working on that advancement, with the topological qubit. Another challenge as we work up the stack is the development of the control and readout platform. We have been developing a state-of-the-art cryogenic compute and control system that cools matter to the coldest temperatures inside this dilution refrigerator, while also using an extremely small amount of energy. Throughout this full-stack, then, as we work up even further, we also need software to enable programming the quantum system and to enable running an actual application on the quantum system. We are building out not only a quantum development kit that includes a quantum focus programming language called $Q\#$, but we are also building the full runtime environment for running and controlling the quantum system. Within this runtime, there are several challenges to address, everything from optimizing compilation, layout, and scheduling of the algorithm on the actual device. We must be able to tune and calibrate and verify that the qubits and the operations and the compute. Then, at the highest level, we also face the challenge of developing quantum applications. Quantum computing and quantum algorithms present fundamentally new programming and algorithm design paradigms. How do we get speedups over classical approaches with these new paradigms? How do we fundamentally unlock new ideas in computing? We think we are only scratching the surface in terms of the potential of quantum computers. We already believe that what We have done in quantum computing has contributed to scientific knowledge. In 2012, the team member, Leo Kouwenhoven, and his team in Delft, they demonstrated evidence of the Majorana quasiparticle that underlies the topological qubit. This is not only fundamentally new physics, that really pushes the frontier of the scientific knowledge, but it is also a giant step forward toward a qubit that enables scaling more efficiently. In addition, we think other areas have pushed the scientific knowledge forward. It is in the area of studying quantum methods, quantum algorithms, and quantum programs. So, we have been able to bring new ideas from quantum computing over to classical methods and algorithms and programs. In turn, this has resulted in not only improvements in quantum algorithms, but We have seen improvements in classical algorithms from learning about quantum algorithms. So, for example, we have discovered remarkable connections between synthesizing gate sequences for a quantum computer, like single-qubit rotation opera-

tions, decomposing those into a sequence of universal discrete single-qubit operations. We have discovered that there is a connection there with algebraic number theory, which enables proofs of previously opened conjectures. It is truly amazing that quantum computing has the potential to really revolutionize the world to solve problems that we really cannot imagine solving with conventional classical computing. This is a fundamental, game-changing work. It is so exciting to be a part of that.

49 Industry Perspective: IonQ

We use individual atoms as the quantum bits. These are atomic ions, they are charged. We hold them in a vacuum chamber, and They are away from any surface. They are not part of the solid. So, they are nearly perfectly isolated from the outside environment, and this makes them perfect quantum testbeds to do quantum operations, quantum gates and to build up a large-scale quantum computer. There have been many results over the last few decades on manipulating these individual atoms and making small scale entangled states, small algorithms. They have largely been demonstration experiments, and they exploit the beautiful coherence properties of individual atoms and that each qubit is the same as its neighbor. So, when these qubits are idle, their quantum memories are as perfect as we can get because They are individual atomic clocks. There are challenges in scaling the system up to go beyond 50 or so qubits would require some sophisticated control engineering to not only trap these ions in a single device but also to think about moving entanglement not only between that device but to another device that has as many qubits. The challenge now in the implementation of quantum computing is to invent and perfect the controller system around these ions. This means that we must develop laser systems and optical control systems to move information around. While it is a challenge and this is unconventional computer hardware, the big benefit of having optical controllers from the outside is that the system is entirely reconfigurable. That means that the atoms themselves are not wired together explicitly; we wire them from the outside. So, we can change the pattern of wiring between qubits, and this leads to great opportunities in the programmability of the quantum computer. One key challenge facing the entire quantum computing industry is finding near-term commercially viable applications. There are very well-known applications for large scale quantum computers. However, today we are focused on developing flexible and fully re-programmable quantum systems to help drive this technology development. While we expect early applications to be those that map naturally to quantum processes, like quantum chemistry or material simulation, even if niche, these applications will help drive the technology development and will allow us to continue to attract top talent. One of the biggest hurdles for scaling trapped ion quantum computation to larger scales is the sophistication of the laser control that is needed to manipulate these qubits. Currently, we are using multiple channels of laser beams that are controlled by these acousto-optic modulators. Traditionally, these types of optics were constructed on a big optical table using individual optical components, as we would see in an atomic physics lab. As the system gets more sophisticated, the complexity of that optical system gets out of control and, therefore, leading to instabilities, meaning things drift over time, very difficult to service if something goes wrong or some parts of it get misaligned. Because it is so spread out over such a large area, it becomes very sensitive to environmental conditions, such as temperature, the humidity of the room. It turns out that there are significant advances in optical integration technologies that were mostly invented in other nearby industries, such as optical lithography, telecommunications where optics are used to send data signals very far away, and also imaging systems where people are building very large scale high-resolution images. So, we are starting to adopt the technological progress of these areas in these other fields to try to

design the control systems for the qubits that are much more scalable, has a lot of multiplexity, and also stable with the precision that we need to control the qubits. So, the ultimate solution, we believe, is to modularize the system where we take a large-scale quantum computer and break them up into a large number of smaller modules, each containing a manageable number, on the order of 100 qubits. We then try to make sure that those modules can be manufactured in vast numbers with a more reliable and compact geometry. Then assemble many, many of those modules and connect them together using a quantum interface in the form of optical communication or photonic interfaces, and that is how we are going to build a large-scale system. We find most exciting is not just the development and continued optimization of well-known algorithms today and applications today, but especially the discovery of new solutions, new capabilities. While we are in the very early days of quantum computing, if the development of classical computing is any indication, it is going to be very exciting next 5, 10, even 50 years.

50 Industry Perspective: Rigetti

Rigetti Computing is a full-stack quantum computing company building hybrid quantum-classical computing solutions. So, let unpack both of those. On the full stack side, this means we do everything from making the own design software to making the own chips in the case, superconducting qubit chips. However, we are packaging them in fridges, building control electronics, building software on top, and building that all into a cloud-deployed quantum programming environment, which, for us, is called Forest. So that is the first part. The reason we make this integrated approach is that a lot of the stuff at each of the layers has already been developed in academia, and now the big challenge is to integrate them and engineer them at scale and for a lower cost. The second bit is that we build hybrid quantum-classical computers, and this is to recognize that we are not going to have big, perfect quantum computers in the next few years. We are going to have small, noisy ones. So that means we should not treat quantum computers as standalone devices, standalone compute. They are co-processors that work in concert with classical computing resources. So, a classical computer can do things like tune around the noise in the quantum programs, or to assemble an ensemble of short quantum programs into the answer to a larger algorithm. So, from the ground-up, we have built the control racks, the instruction set, the programming environments, all to work in this hybrid model [224]. So, we are building full-stack hybrid superconducting qubit systems. There are three sets of areas that we are most interested in beginning with. Those are quantum simulation, optimization, and machine learning. So fundamentally, in quantum simulation, it is because we have these quantum systems that we can apply all the techniques We have discussed in classical approaches to solving computational quantum chemistry and map them into how to do quantum programming for quantum simulation. Then in optimization and ML, it is a little more nascent, but there we are using the fact that we have, basically, a large linear algebra resource and trying to figure out how to sample and exploit from it. So, we are excited about both those areas of development. We think the most important business challenge is not something that affects just Rigetti Computing. It affects the whole industry, and it is education. If we must have a Ph.D. in quantum computing to build the technology or use the technology, this is never going to change the world. So, we need to figure out a way to make a new kind of community. It is a quantum computer engineering community, and there needs to be a shared corpus. There needs to be shared the basic content. The basic textbooks are still to be written for this quantum computer engineering community. We need to both do that on a technical level, and then translate that into something on the business side that people come to understand. One way to think about quantum computers is as a way of exploring this new reality that is out there. We have known about quantum physics

for 110 years, and it is played its way subtly into how the technology works in a lot of ways, but we never really grasped it. A quantum computer is like a vessel that we can use to go into this new area of exploration to see what we can do with large, well-controlled quantum systems. That is a very, very fundamental question. On the one hand, if it turns out that we cannot build quantum computers, this would break the models of physics, which is almost as interesting as the stuff we can do by applying them in a powerful way. There is so much foundational stuff to do. There are so many foundational concepts to discover. There is just a lot of ways on how to create an impact on how the field will evolve and grow. There is just not a lot of historical opportunities to be at the forefront of something like that. It is like we are going back to the 50s or 60s and saying, this computing thing that might happen. How could we be involved in that? Furthermore, quantum computers can happen faster because we have all regular computing to help us.

51 Industry Perspective: QCI

We have been pioneering a couple of new approaches for how to make quantum computations more robust and more easily scalable. This is what we call hardware efficient error correction, which means we need fewer parts to build into our quantum computer. Something called the modular architecture, which is a way of assembling smaller pieces into a more powerful whole, entangles them. The machines we see in the lab, are used to cool the special superconducting quantum circuits to temperatures a few thousandths of a degree above absolute zero, where we can prolong their quantum behavior, and where all the elements inside become superconducting so that there is no electrical loss in the computer at all. There are several other players in this field, including Google and IBM. Moreover, they also use superconducting qubits and a circuit QED approach. They have decided to make their quantum processors out of a relatable technology, in the form of big 2D grids of these devices. QCI takes a different approach. We believe that just like supercomputers should be, and are, built out of networks of simple processors, quantum computers should not just be large grids of qubits. Thus, we are making the architecture modular, in the sense that the goal is to build a quantum computer out of reliable, easy to understand, and easily characterize units that are connected in a reconfigurable way. Thus, we feel we have a real head start, and a much easier path to rapidly scaling to large scale and useful quantum computers. What is special about the quantum system is that they offer possibilities that no classical system will ever offer. They operate on using, not the conventional logic we are used to, which is the Greek logic, but the logic of nature. There are aspects of it which we cannot predict as of now. We mean, there is an event and breakthrough that will make this new science available in fields that are not even thinkable as of now. So just a bit like asking the makers of the first classical vacuum tube computer to predict the internet.

52 Industry Perspective: D-Wave Systems

D-Wave uses superconductor integrated circuits to build annealing quantum computers. In terms of annealing quantum computing, this is kind of the most interesting part of what D-Wave has chosen to do because it is very different from what most of the quantum computing community has chosen to focus on, which is called gate model quantum computing. We are trying to harness quantum mechanical tunneling. So, the qubits have the possibility of tunneling back and forth between the zero and the one state. We can control how easy it is for them to tunnel back and forth between these two states. So, we start the algorithm, the quantum annealing algorithm, where all the qubits in the system are tuned so that it is easy to tunnel back and forth between

zero and one. Furthermore, the idea with quantum annealing is to start in that configuration and progressively make it more difficult to tunnel back and forth between the zero and the one state. So, at the beginning of the algorithm, the system, each one of the qubits is in a superposition of zero. One and the system is, in a sense, in a superposition of all possible answers that the final processor could encode. Some of those answers in that superposition are going to be the better or best answers for our set of constraints. We just do not know which ones yet. The algorithm works because we start with our dimmer switch turned down in this way, and we gradually turn up the strength of the constraints that we wanted to program on the processor. We are stating the problem more and more strongly as we go. Moreover, as we are doing this, we are making it more and more difficult for the qubits to tunnel back and forth between the zero and the one-state—each of them. The idea with this, and again, just at sort of a hand-wavy level, is that we progressively select out, preferentially select out, those combinations of zeroes and ones, which are going to end up being the better answers. The lower energy cost-function answers out of this large massive superposition. Annealing quantum computing is relevant in areas of optimization, sampling, and machine learning. It cannot run Grover's algorithm, which is a prescription for a gate model algorithm, but it naturally solves an optimization problem at its core. Everything in a superconducting integrated circuit is made with Josephson's junctions. The current generations of processors have about 135,000 of those junctions on them. So, scaling the technology to be able to yield at that level has been challenging. However, also, the actual architecture of the processor to function, to be programmed, measured, and annealing properly at that kind of scale has been a big challenge. D-Wave has had to tackle the problem of bringing classical control circuitry directly into the quantum processors' environment to control and manipulate them and read them out properly. However, we are not there yet. So, there is still is a little way to go to mature technology. There is a whole world in front of us. What we find are the major challenges still facing us are continuing to scale the processors and crossing into a territory where we can run real commercial applications. So, D-Wave was the first company to produce a commercial quantum computer. We are the first to be able to go through these rapid development cycles, get customer feedback. Then the physicists, engineers, and software developers can improve the architectures as sort of, as pointed by customers working on actual applications and making them better. People discuss, in the digital computing world, that Moore's Law is coming to an end. In the part of the quantum computing market, we have gone from 120 qubits to 500 to 1000 to 2000, and then machines with four of five thousand qubits are on the horizon in the next few years as well.

53 Limits of Classical Computing: A Brief Introduction to Computational Complexity

We have discussed the promise of quantum computers and their potential for quantum advantage. Some problems are very hard to compute on a classical computer, and for some of these problems, a quantum computer is much more efficient. However, what makes a problem hard in the first place? we have probably discussed problems being categorized as P or NP, $NP-$complete [225]. Nevertheless, what do these categories mean, and how do computer scientists classify a problem as an easy problem or a hard one? In this deep dive, We will discuss about the computational complexity of classical computing [84], how it is defined, and how problems are classified, including a discussion of several standard problems that are classified as P, and NP, and $NP-$complete. Let us describe a little bit about universality in the language of the circuit model of computation. That is what we naturally know best and is what we will build quantum computation upon as a language. So, in the circuit model, for example, we may have AND

gates and NOT gates. we want to claim that all Boolean circuits, circuits that compute Boolean functions, can be composed of AND and NOT. this is easy to see, but first, let us look at the family of Boolean functions. They will be functions that take, if there are n bits of input, say x_0 to x_{n-1}, and the result of this is going to be either 0 or 1, so for example, in the notation, We will say that we may have f of x_1x_2 be x_1x_2. this product represents an AND gate. Alternatively, for example, f of x is equal to $\bar{x}$. This is how we will represent a NOT gate? Now, the question of universality dovetails with another question. we have not finished the universality description yet. However, we want us to keep this in mind of complexity because it is not useful if We say something about universality without saying how much cost it takes to accomplish something. So, the key idea which we will not have time to go into much depth on, but We hope we already know something about, which is that some math problems are harder than others. we do not mean it is hard to multiply two million-digit numbers, and that is harder than multiplying two 10-digit numbers. No, that is not the point. The point is They are scaling with respect to the size of the problem. this leads to a statement which we hope that all of we already know of in some form, which is called the Strong Church-Turing Thesis. we will write this up for us because we think the words of this are instrumental in understanding the perspective. So, we say, any model of computation can be simulated on a specific kind of machine. the machine and model We would choose will be this one here, the Turing machine. However, we are going to choose a specific variant of it, the probabilistic Turing machine. we need to say something about the cost of this kind of simulation. the essence of that thesis is that this simulation cost comes with, at most, a polynomial increase in the number of elementary operations required. It is an excellent thesis. It defines equivalence between models, whether they be electrical and optical or electrical and DNA or quantum and DNA or quantum and classical. for even that to be possible, conceptually is remarkable. we have not defined a lot of the technical terms in this, like simulation and the overhead costs, but We hope we will appreciate that as we go along. So now, in order to highlight this statement of equivalence, let us make sure we are aware of one of the greatest motivating factors for quantum computing, which is the difference between two of the most important classes of mathematical problems. we will use this fact that many problems can be expressed as decision problems. So, for example, is the number M prime? Furthermore, the answer to this is yes or no. Is this a hard problem or an easy problem? This was not known for many years. It was then realized that we could answer this question with some randomness. we would not know it for certain. This is Rabin's primality testing algorithm. then some 15 years ago, somebody in India proved that we could do it deterministically. So, it moved from this probabilistic model, which people had to use previously for this question, to something which did not need probability. So today, there is a deterministic primality testing algorithm. This problem is called primality. That is one example. Here is another one. It is called factoring. Given a composite integer m, but not just the number that we are going to try to factor, also and a number l, an integer l, which is less than m, does m have a factor that is non-trivial which is less than l? So, we need to bound the sides of the range of numbers we are going to consider as being answers to the problem. again, this is a yes or no question. we have this distinction. If the time taken to answer this question, needed to answer this question, it is polynomial in the size of the question. Here, for example, for factoring, this is the number of bits of m, the number of digits in the number, not just the number itself, then we say that the problem is polynomial in complexity. We say it is in this class that We will call P. Now, let us break down the class on whether this is answered by a yes or by a no. If the yes instances of the problem are easily checked, and We will use the word verified as a technical term with the aid of a witness, which is a short description piece of information that enables somebody else who is not skilled at the arts but can be very reliable, to verify our claim. Sometimes, we discuss Merlin and Arthur [226]. We say that the Merlin is very clever and can come up with proofs, but we need the Arthur,

who is not clever but can be very, very reliable to take that proof and verify the claim. So, there are two parties to this, the verifier and the prover. the verifier takes this witness. If this is true, then the problem, even if it is challenging, is the fact that we can verify that the proof of it means we will say that the problem is in this class called NP. in some senses, we want to say this is non-polynomial, but the main point is the distinction between P and NP. for the sake of completeness, we have another parallel to this: the mirror image. If no instances, so this is the answer is yes, and the answer is no with the same language, then we say that the problem is in a different class called $co - NP$. we do this not because We want to say anything more about P versus $co - NP$ or NP versus $co - NP$, but just to share with we a trivial fact in a moment. So, the reason We show this is because so much of the motivation for computation today, and much of quantum computation comes from this question of NP versus P. We think that the P problems are easy. The NP problems are the hard ones and meaningful ones to do. There is this plot that we can make, which shows that, if this is the space of all mathematical problems, then P is a subset of something we might say is NP, but there is also this extra area over here, which is the hardest of the NP problems. we call these $NP-$complete problems. they are defined as such because, if we can solve any of the $NP-$complete problems, then it gives us a polynomial-time algorithm to solve any other problems in the NP regime. So, where is quantum computation in all this map? Well, we are not going to answer that for we today, but We hope that, through this, we will start to appreciate where quantum computation is relative to this landscape of the field of all problems and their complexities. Quantum computation sits a kind of difference in this landscape. It has a complexity class, typically of something we call BQP [227–229]. part of the reason is that the model is slightly different and does not fit directly into either of these classes, because sometimes the output is quantum mechanical. There are errors involved, and some things We will come to later. Good. We hope that many of us already knew about most of this, but We also hope that it starts to connect some things for us. An example of an $NP-$complete problem and now We going to go back to the concept of universality is, so an example is a problem called $3 - SAT$. this is about the satisfiability of Boolean functions, which looks something like this. So, suppose we have, again, a function of bits x_0 through x_{n-1}. The formulas that are involved in $3 - SAT$ look something like $x_1 + x_3 + x_9$, OR'd together, AND'd with there is a multiplication here another term, like $x_4 + \bar{x_7} + x_{11}$, and so forth and so on, where each one of these terms just has three bits. our goal is to say, does there exist an assignment of zeros and ones to the x inputs such that the output is equal to 1? It seems very simple. It is straightforward to write this problem on the board. However, if we can solve this problem fast in time. That is polynomial with n here, and then it turns out we can quickly solve all the rest of these problems. if we can solve this problem fast, we can solve problems like the optimum way to pack boxes into a FedEx truck, or the optimum way to route a packet in for information from San Francisco to Boston. we know, it is remarkable how powerful such a simple problem can be. However, we can show that, in practice, we cannot solve most of the instances of this problem as well as we would like to. Are there simple problems that are neither P nor NP? Are there simple problems that are neither P nor NP? Another interesting class that sits outside of this class is counting the number of solutions to a problem. That is the class called sharp$-P$. so forth, because we might count the number of things to count the number. However, we know this is how we get a career as a theoretical computer scientist. It is very effective.

54 Computational Complexity: Topics on Complexity Theory

So, what is complexity theory? Theoretical computer scientists love to come up with complexity classes [230]. So, P is polynomial-time computation. NP, things we can prove in polynomial time. So, who here does not know about NP? So let us explain NP a little bit. The most famous $NP-$complete problem may be the traveling salesman problem. Here we have a graph, distances between, let us call them, cities, which are the nodes of the graph. The question, is there a path visiting all cities less than length l? So, this might be 1, 2, 3, 5, or no 2, 3, 1, 1, 2, 3. So this would be a path of length 12. Moreover, there are distances We mean there are roads between the cities. We have not put on here at distances along those roads. If these were the only roads, it would be an easy problem. If there are only six vertices, it is an easy problem. So, it is easy to prove. So, if We want to prove that there is a path, we just give us the path, and we add up the distances and check it. Moreover, we can prove it. However, it is hard, or it may be hard to prove no. So how can We prove that there is no path shorter than distance 12 on that route? We could run through all paths and check all their distances, but that takes a very long time. If We know, and if We were trying to convince us that we have done this, would there be a better way to convince us other than just say, well, we ran through all the paths and checked them? Furthermore, there are better ways for the traveling salesman problem, but they all seem to be an exponential time in the worst case. Or the exponential time in the number of cities. Now, some problems are $NP-$Complete, which is as hard as any NP problem. In other words, if We could solve an $NP-$Complete the problem in polynomial time in the length of its input, we could solve any problem in NP in polynomial time the length of its input. We do that by reducing one problem to another. Furthermore, at some point in the 1990s, we think computer scientists came up with interactive proofs. There are three classes of interactive proofs. IP, no bound on the number of rounds, AM, and here "A" stands for Arthur, and "M" stands for Merlin. Merlin has infinite computing power, and Arthur only has a polynomial amount of computing power. So, A sends M message, M returns the message. So that is two rounds. Then there is also an MA. M gives proof. A verifies it with coin flips. So, NP, Merlin gives Arthur a proof, and Arthur verifies it deterministically. MA, Merlin gives Arthur a proof and Arthur verifies it, but probabilistically. Moreover, AM, Arthur sends Merlin a message, and Merlin returns a message. The classic example of AM is graph non-isomorphism. Are these two graphs isomorphic and NP. We are given two graphs, and We guess these have 5, 6, 1, 2, 3, 4, 5 6. So, the question is, is there a way to relabel the vertices so that they are the same? Moreover, the guessed answer is an NP, and an isomorphism is in NP. Merlin just gives Arthur a relabeling. So, Merlin says 1 in this graph corresponds to 5 in this graph. 3 in this graph corresponds to 4 in this graph. Non-isomorphism is an AM. Arthur takes one of the graphs. He re-numbers the vertices at random, and he sends it to Merlin. Furthermore, Merlin is supposed to Merlin says which graph it is. Now, if the two graphs are isomorphic, we know, Merlin has the scrambled graph, but a random scrambling of this graph looks exactly like a random scrambling of this graph. So, he cannot tell them apart. If Merlin has an infinitely powerful computer, he can try all possible permutations and tell them apart if the two graphs are not isomorphic. So that says that non-isomorphism is an Arthur-Merlin.

55 Classical Circuit Model

This section we will program and execute the Deutsch-Jozsa algorithm on a cloud-based quantum computer using the IBM Quantum Experience [231]. The Deutsch-Jozsa algorithm is pro-

grammed and implemented using a quantum circuit model of quantum computation. To get started, let us first review an example of a circuit model for classical computation.

In this section, we will run an actual quantum algorithm, the Deutsch Jozsa algorithm, on a real quantum computer. Of the section, we will first discuss in detail how that works. To do this, we will need to introduce the algorithm using a circuit model of quantum computation [1]. However, before we do that, let us first examine a circuit model in the context of classical computation. The circuit model for classical computation begins with input data prepared at the initialization stage. We can think of the computational bits in the input register being reset to all zeros and then set to the initial values that will be input to the computer. Now, this reset step may not be necessary for a classical computer. However, we will include it here for comparison with the quantum circuit model that We will discuss later. The initialized bits are then inputted to a computational stage. Here, a series of logical operations are implemented using classical Boolean logic gates to compute a series of functions. Furthermore, once this is complete, the output for this stage encodes the result of the computation. For example, let us consider a full adder circuit comprising a series of Boolean logic gates. The full adder takes as inputs the bits B1 through B3. Where B1 and B2 are the binary numbers we want to add, and B3 holds any carry in that we may have from a previous calculation. Here we do not have a carry forward, so its value is zero. An output of this circuit is the sum, B1 plus B2 modulo 2, where the modulo 2 arises because this is binary arithmetic. The second output is the arithmetic carry out that may result from the addition. These two values are then assigned to bits B4 and B5 and stored in the output register. In this case, we add 1 plus 1, which is 2 with decimal numbers. Furthermore, for the binary addition performed by this circuit, the addition is modulo 2, 1 plus 1 equal 0, and has a carry forward of 1. Finally, the results from this addition problem are obtained in a measurement stage, where the values of bits B4 and B5 are the binary representation of the answer, which may then be converted back to a decimal result. This type of initialized, compute, and measure process is the foundation for a universal classical machine. To access this machine, a user interface provides the input data and the program instruction set to calculate the desired function. This instruction set is then applied to the physical hardware using a controller layer. This layer takes the input data, sets the input bits, implements the physical logic gates according to the instruction set, measures the output, and then sends the resulting data back to the user interface. This computational model illustrates the basic building blocks that we need to implement an algorithm using a quantum circuit model. The main distinction will be the role of quantum mechanics and its impact on the initialization, compute, and measurement stages.

The blueprint for implementing an algorithm on a classical computer is the classical circuit mode. It describes how a set of input states is manipulated by a processing core and subsequently stored in an output register.

For example, a 2-bit$\times$2-bit binary multiplier uses a series of Boolean logic gates to multiply two input numbers, each taking an integer value between 0 and 3. The computation starts by first resetting the input register and then initializing it to the values of the two inputs represented as binary numbers. After that, the input states are then processed by a gate sequence that implements multiplication. Finally, the resulting output bit sequence, the computed result is placed in the output register where it can be readout.

All possible 2-bit inputs (a and b) and resulting outputs (c) are summarized in the following table:

Table 1: 2-bit inputs (a and b) and resulting outputs (c)

decimal	binary	decimal output	binary composition	binary output
$a \cdot b$	$[a_1 a_0] \cdot [b_1 b_0]$	c	$c_3 2^3 + c_2 2^2 + c_1 2^1 + c_0 2^0]$	$[c_3 c_2 c_1 c_0]$
$0 \cdot 0$	$(00) \cdot (00)$	0	$0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	0000
$0 \cdot 1$	$(00) \cdot (01)$	0	$0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	0000
$0 \cdot 2$	$(00) \cdot (10)$	0	$0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	0000
$0 \cdot 3$	$(00) \cdot (11)$	0	$0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	0000
$1 \cdot 0$	$(01) \cdot (00)$	0	$0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	0000
$2 \cdot 0$	$(10) \cdot (00)$	0	$0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	0000
$3 \cdot 0$	$(11) \cdot (00)$	0	$0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	0000
$1 \cdot 1$	$(01) \cdot (01)$	1	$0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$	0001
$1 \cdot 2$	$(01) \cdot (10)$	2	$0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$	0010
$2 \cdot 1$	$(10) \cdot (01)$	2	$0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$	0010
$1 \cdot 3$	$(01) \cdot (11)$	3	$0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$	0011
$3 \cdot 1$	$(11) \cdot (01)$	3	$0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$	0011
$2 \cdot 2$	$(10) \cdot (10)$	4	$0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	0100
$2 \cdot 3$	$(10) \cdot (11)$	6	$0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$	0110
$3 \cdot 2$	$(11) \cdot (10)$	6	$0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$	0110
$3 \cdot 3$	$(11) \cdot (11)$	9	$1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$	1001

A 2-bit×2-bit multiplier can be constructed using Boolean logic gates of the following type (note: this is not a unique construction):

AND gate: outputs logic-state 1 solely if both input bits are in logic state-1, and outputs 0 otherwise.

XOR gate: outputs logic-state 1 if the input bits differ, and outputs 0 otherwise.

Analyzing the output bits c_0 through c_3 yields the following conclusions:

- c_0 equals 1 only if both a and b are non-zero and odd. Therefore, an AND gate applied to the bits a_0 and b_0 is sufficient to compute c_0:
  $c_0 = a_0 \ AND \ b_0$

- c_1 will equal 1 if not all input bits and either a_0 and b_1 or b_0 and a_1 are in logic-state 1:
  $c_1 = (a_0 \ AND \ b_1) \ XOR \ (a_1 \ AND \ b_0)$

- c_2 will equal 1 only if both primary bits a_1 and b_1 but not all involved bits are in logic-state 1:
  $c_2 = ((a_0 \ AND \ b_0) \ AND \ (a_1 \ AND \ b_1)) \ XOR \ (a_1 \ AND \ b_1)$

- c_3 equals 1 only if all bits composing a and b are initialized in logic state 1:
  $c_3 = (a_0 \ AND \ b_0) \ AND \ (a_1 \ AND \ b_1)$

The result stored in the four-bit output register is obtained via a measurement of the register and, for the users convenience, can be converted back to a decimal result. The sequence of initialization, computation, and measurement summarizes the working principle of universal classical machines. Although universal quantum machines [232] follow the principles of quantum mechanics and therefore require a different approach for implementing individual computational steps, the concepts derived from the classical circuit models serve as the basis for designing quantum circuit models.

56 Quantum Circuit Model

In this section, we will discuss the quantum circuit model and its similarities and differences with the classical circuit model. The quantum circuit model applies to gate-model algorithms, such as the Deutsch-Jozsa algorithm that we will implement in the lab practicum.

In the last section, we introduced a circuit model for classical computation. This model included an initialization stage to set the input bits, and a compute stage that implemented classical logic gates to compute a function and a measurement stage of extracting the output. In this section, we will introduce an analogous circuit model for quantum computation. The basic structure is the same. Initialize, compute, and measure. We will start with an initialization stage where the qubits are first to reset to a state $|0\rangle$ and then initialized to their input values. We will assume here that the initial state is also $|0\rangle$, $|0\rangle$, $|0\rangle$. The initialized qubits are then inputted to the computation stage, where a series of quantum logic operations will be performed. In general, one of the first steps in this block is to create an equal superposition state. This is done by applying single qubit Hadamard gates to the input register. To see how this works, let us first consider just a single qubit in state $|0\rangle$. The Hadamard gate applied to this qubit takes state $|0\rangle$ and rotates it to an equal superposition state, $|0+1\rangle$, and the normalization constant is $\frac{1}{\sqrt{2}}$. Now, what happens when we have two qubits each initialized in state zero? Applying a Hadamard gate to each one independently rotates each into an equal superposition state $|0+1\rangle$. Moreover, when we take the tensor product effectively multiplying out the terms, we find an equal superposition state of 2 qubits, $|00\rangle + |01\rangle + |10\rangle + |11\rangle$. now, the normalization is the square of $\frac{1}{\sqrt{2}}$, which is $\frac{1}{2}$. Similarly, applying a Hadamard gate to three qubits, each initialized in state zero, results in an equal superposition state of three qubits. We have seen before, this state comprises eight terms, from state $|000\rangle$ to state $|111\rangle$. the normalization is now $\frac{1}{2\sqrt{2}}$. Each coefficient now has this value. Creating a large, equal superposition state sets the stage for quantum parallelism and quantum interference to occur during quantum logic operations. The logic operations themselves are the single-qubit and two-qubit gates we discussed earlier. For example, a single qubit $X-$ gate or another Hadamard gate or a two-qubit CNOT gate, chosen to implement a function or algorithm. As discussed previously, the quantum parallelism and quantum interference that occurred during these operations changed the coefficients' values in the superposition state. Finally, the qubits are then measured to determine the answer. As we also discuss earlier in the section, the measurement process projects the qubits onto the measurement basis. This effectively collapses the massive superposition state onto a single classical state with a probability that is the magnitude squared of its coefficient. In doing so, the measurement leads to a single classical binary result, either a 0 or a 1 for each qubit. Now, if we do not perform any logic operations and simply measure the equal superposition state from the input, each coefficient has the same value, and we have an equal probability, one eighth, of measuring any one of these states. Thus, as we discussed earlier in the section, to be successful, a quantum algorithm is designed such that after applying a designated sequence of quantum logic gates, ideally, all the probability amplitude resides in one of the coefficients. Moreover, this coefficient sits in front of the state that encodes the answer to the problem. Thus, when we make a projective measurement, the probability is a unity that we obtain this result. Using this basic quantum circuit model, we initialize, compute, and measure, and we can implement a universal quantum algorithm. In the next section, we will apply this model to a specific example, the Deutsch-Jozsa algorithm.

The structure of the classical circuit model initialization, computation, and measurement are directly applicable to quantum circuit models. The initialization stage starts with the qubits being set to their starting states, often the ground state $|0\rangle$ for each qubit. The initialized qubits are subsequently processed by a sequence of quantum logic operations in the computation stage.

In general, the first step of the computational stage is to create a massive superposition state to set the stage for quantum parallelism and quantum interference during the algorithm. For example, to create an equal superposition state from three qubits initialized in $|000\rangle$, we may apply Hadamard gates to each qubit individually, as shown below.

$$\begin{aligned} H \otimes H \otimes H|000\rangle &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right), (1) \\ &= \frac{1}{\sqrt{2}^3}\left(|000\rangle + |001\rangle + |010\rangle + |011\rangle\right. (2) \\ &+ \left.|100\rangle + |101\rangle + |110\rangle + |111\rangle\right) (3) \end{aligned} \tag{40}$$

After creating this equal superposition state, a sequence of single and two-qubit gates is applied to implement a particular function or algorithm. Quantum algorithms use quantum parallelism and quantum interference to ideally coalesce all of the probability amplitude into the coefficient of the state that encodes the answer; by bringing this coefficient's value to unity, the probability of measuring this state which goes as the magnitude squared of the coefficient is also unity. The values of the other coefficients and thus the probability of measuring states that do not contain the solution ideally decrease to zero. Sometimes, as in the Deutsch-Jozsa algorithm, the algorithm reaches a final answer in one step. In other algorithms, such as Shor's factoring algorithm or Grover's search algorithm, the algorithm repeats a cycle of operations that gradually modify the coefficients. The system converges to a solution after some number of iterations. In either case, at the end of the computation, the quantum system is in a state that comprises the solution state with coefficient near 1, and all other coefficients near 0.

Upon completion of the computation stage, the qubits at the output are measured to determine the answer. Since the solution state has a coefficient at or near 1, we are very likely to measure that state when we perform our measurement. Because of the probabilistic nature of this measurement, and the potential for imperfect coalescence to the solution state, the measurement may need to be performed several times to ensure that we did not project the wrong answer by chance. In practice, the entire process may need to be run multiple times even for algorithms that, in theory, have the correct coefficient converging to 1 because there is always noise in the system that can lead a small probability that the algorithm gives an incorrect result.

57 The Deutsch-Jozsa Algorithm

In this section, we will discuss how the Deutsch-Jozsa algorithm works in detail. The section will begin with a general discussion of the N-qubit algorithm. It then presents the key mathematical steps required to implement the algorithm for a single data qubit.

In this section, we will introduce the Deutsch-Jozsa Algorithm. Deutsch-Jozsa was one of the first quantum algorithms that exhibited a provable, exponential speedup over a classical algorithm. In this section, we will discuss how it works and walks through its key steps. by the end of this section, we will implement it on a real cloud-based quantum computer. To get started, imagine that we have an unknown function f. It takes N Bits as input, and it outputs a single result, either a 0 or a 1. Now, all we know about this function is that it has a unique property. It is either a constant function, or it is a balanced function. What do we mean by that? A constant function takes any input, and no matter what that input may be, the function always produces the same result. For example, independent of the input, the function always outputs a zero, or no matter the input, it always outputs a one. A balanced function, on the other hand, will take those inputs, and for half of them, we do not specify which half, but for half of them, it will output a zero, and for the other half, it will output a one. In this sense, the

output is balanced. The Deutsch-Jozsa problem is the following. Determine whether a function f is constant or balanced based on queries that we make of the function f. we can send any input state to the function, and we get a result back, and we can query the function as many times as we like. Now for N Bits, there are two to the $n-$th power different input states from which we can choose. we will need to run the function for at least half of them to determine with certainty if the function is constant or balanced. we will have to do it for half plus one. That is because even if we get all zeros for the first half of the states, we will need to try one more to see if the second half remains zeros, a constant function, or all one's balanced function. Thus, a deterministic classical algorithm will take $2^{\frac{N}{2}+1}$ step. it always works. Now, if we have N Qubits instead, we can create an equal superposition state of all N Qubits, and we will determine the answer in just one step. it always works. that is an exponential speed up. To see how this works in detail, let us simplify the problem to just one bit or one qubit. It is the same approach as for N Bits or N Qubits, but it will be much easier to follow if we just take n equals to one. In this case, for a constant function, we have f of 0 equals f of 1, and the output is either a 0 for both, or it is a 1 for both. for a balanced function, one of the outputs is 0, and the other is 1. In the truth table, we can make the following observation, that if we take the exclusive or f of 0 and f of 1, the result is 0 for a constant function, and 1 for a balanced function. In this case, classically, it takes two steps to implement the algorithm for n equal one. again, quantum-mechanically, it only takes one step. Now, that may not seem like a big speedup. It is only twice as fast, but, for N equal one, it is an exponential speed up. this generalizes to any number N that we may choose. Again, we are going with N equal one because it will be easier to see how the algorithm works. To implement the Deutsch-Jozsa Algorithm for N equal 1, we need two qubits. One is the data qubit, that is, the N equal one qubit, and the other is a helper qubit or an ancilla qubit. we will use the quantum circuit model to implement this algorithm. First, we initialize the qubits into their starting states. The data qubit is prepared in state zero and the helper qubit in state one. As we proceed through the algorithm, we will indicate the position at each stage of the algorithm and the state of these two qubits. For example, after initialization, we see that the data qubit is in state zero, and the helper qubit is in state one. To keep it all straight, we will use yellow to highlight the data qubit state and green to highlight the helper qubit state. We first create an equal superposition state by applying a Hadamard gate to both qubits. This results in a superposition state for each qubit. There is a plus sign for the data qubit since it started in state zero and a minus sign for the helper qubit since it started in state one. Next, we will just expand out the terms in the data qubit. The zero-state tensor product, the helper qubit superposition state plus one state tensor product, the helper qubit state. This state then inputs into the quantum circuit block. We will just call it U_f. The data qubit is x, and the helper qubit is y. U_f implements the set of logical operations. it does two things. First, it implements the unknown function f, and second, it outputs the exclusive or of bits y and f of x. We will show later how this can be implemented with single-qubit and two-qubit operations. For now, though, let us just figure out what happens. At the output of U_f, we have the helper qubit. it takes on the state y, x, or f of x. So, where the data qubit is zero, the helper qubit is x ord with f of 0. where the data qubit is one, the helper qubit state is x ord with f of 1. Now we make an observation. we should try this on scratch paper to convince ourselves that the expression can be written in this way with a minus 1 to the f of 0 power and a minus 1 to the f of 1 power. To check, there are four cases to consider. For example, if f of x equals 0 for any x, the x or expressions on the left maintain the superposition $0-1$. However, if f of x equals 1 for any x, then the x or expressions on the left result is $1-0$. to return this to the superposition state $0-1$, we need to multiply by a -1. It is achieved by taking -1 to the fx power. when fx equals 1, this is the minus one that we are looking for. The expression also works when f of 0 and f of 1 are not equal, but take on the values 0 and 1, the third case, or 1 and 0, the fourth case.

The fact that this works for all four cases simultaneously is an example of quantum parallelism. In the next step, we factor out the helper qubit term $0 - 1$ divided by root 2. we move the -1 to the f of 0 power and the -1 to the f of 1 power over to the data qubit. Next, we apply Hadamard gates to the data and helper qubits. On the data qubit, state zero rotates to $0 + 1$, and state 1 rotates to $0 - 1$. For the helper qubit, $0 - 1$ rotates to state 1. Next, we just note that -1 to the f of x power is the same as e to the power $-\pi$ f of x. then, we just rearrange terms to collect the coefficients of state zero and the coefficients of state one. this is an example of where quantum interference occurs, changing the coefficients values depending on whether the function is constant or balanced. For example, if f of 0 equals f of 1, a constant function, then b equals 0, and a is either plus or minus 1. Thus, a measurement will yield state 0 with unity probability. we observe that state zero is equivalent to state $f$of 0, x ord with f of 1. In contrast, if f of 0 does not equal f of 1, a balanced function, then an equal 0 and b is plus or minus 1. In this case, a measurement will yield state one with unity probability. we observe that state one can also be replaced with the state f of 0, x, or f of 1. Thus, if we measure a 0 on the qubit, the function is constant, but if we measure a 1 on the qubit, the function is balanced, and we will always measure a state one on the helper qubit. Thus, we were able to determine whether the function was constant or balanced with one evaluation of the quantum algorithm. the same is true if we had instead used N data qubits. One evaluation always works. In contrast, the classical case must evaluate half of the 2 to the N states plus one to get a deterministic answer. Thus, the quantum speedup is one step versus 2 to the N minus 1 plus 1 step. Lastly, we can look at ways to implement the logical operation U_f. We will not discuss this through all of them here. we can find them in the text unit following this section. However, four cases are corresponding to the two constant functions 0, 0, and 1, 1, and the two balanced functions 0, 1, and 1, 0. We encourage us to work through each of these cases. Now that we understand how the Deutsch-Jozsa Algorithm works within the quantum circuit model, we are ready to write a quantum computer program that will implement it on a quantum computer. We will see how that generally is done in the next section, and then more specifically, in the lab practicum.

To ensure that we understand each step in the computation stage of the Deutsch-Jozsa Algorithm, Let us a review of the section for the 1-qubit case.

Step 0: To start with, we have a data qubit initialized in the state $|0\rangle$, and a "helper" (or "ancilla") qubit initialized in $|1\rangle$.

$$|\Psi_0\rangle = |0\rangle|1_a\rangle \tag{41}$$

We will use the subscript "a" to designate the ancilla qubit. Also note that If we wanted to run the algorithm on an N-bit number, we would need N data qubits, but still only 1 ancilla qubit.

Step 1: The first step of the computation stage is to put both the data and ancilla qubits in a superposition state by applying a Hadamard gate to each.

$$|\Psi_1\rangle = H \otimes H|\Psi_0\rangle \tag{42}$$

$$|\Psi_1\rangle = H\rangle|0\rangle \otimes H|1_a\rangle \tag{43}$$

$$|\Psi_1\rangle = \Big(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\Big) \otimes \Big(\frac{|0_a\rangle - |1_a\rangle}{\sqrt{2}}\Big) \tag{44}$$

$$|\Psi_1\rangle = \frac{1}{\sqrt{2}}\Big[|0\rangle\Big(\frac{|0_a\rangle - |1_a\rangle}{\sqrt{2}}\Big) + |1\rangle\Big(\frac{|0_a\rangle - |1_a\rangle}{\sqrt{2}}\Big)\Big] \tag{45}$$

Step 2: Next, we apply the unitary operation U_f, which is implemented by a sequence of quantum logic gates (we will not worry about exactly which logic gates here). U_f has the property that if the data qubits are in-state $|x\rangle$, then it puts the ancilla qubit into the state

$1 \oplus f(x)$, where f is the function that is either balanced or constant, and $\oplus$ denotes addition modulo 2 (note that this is equivalent to XOR for a single bit).

$$|\Psi_2\rangle = U_f|\Psi_1\rangle \tag{46}$$

$$|\Psi_2\rangle = \frac{1}{\sqrt{2}}\Big[|0\rangle\Big(\frac{|0_a \oplus f(0)\rangle - |1_a \oplus f(0)\rangle}{\sqrt{2}}\Big) + |1\rangle\Big(\frac{|0_a \oplus f(1)\rangle - |1_a \oplus f(1)\rangle}{\sqrt{2}}\Big)\Big] \tag{47}$$

$$|\Psi_2\rangle = \frac{1}{2}\Big[|0\rangle|0_a \oplus f(0)\rangle - |0\rangle|1_a \oplus f(0)\rangle + |1\rangle|0_a \oplus f(1)\rangle - |1\rangle|1_a \oplus f(1)\rangle\Big] \tag{48}$$

We can rewrite this as:

$$|\Psi_2\rangle = \frac{1}{\sqrt{2}}\Big[|0\rangle(-1)^{f(0)}\Big(\frac{|0_a\rangle - |1_a\rangle}{\sqrt{2}}\Big) + |1\rangle(-1)^{f(1)}\Big(\frac{|0_a\rangle - |1_a\rangle}{\sqrt{2}}\Big)\Big] \tag{49}$$

So, we can see that the effect of applying U_f to our quantum state was the insertion of these $(-1)^{f(x)}$ factors. These factors will lead to constructive and destructive interference enhancing and suppressing terms in the output state.

Step 3: Once again, apply Hadamard gates to both qubits.

$$|\Psi_3\rangle = H \otimes H|\Psi_2\rangle \tag{50}$$

$$|\Psi_3\rangle = \frac{1}{\sqrt{2}}\Big[(-1)^{f(0)}\Big(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\Big)|1_a\rangle + (-1)^{f(1)}\Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big)|1_a\rangle\Big] \tag{51}$$

If $f(0) = f(1)$, the $|1\rangle$ terms on the data qubit will cancel out, leaving only$|0\rangle$ terms. But if $f(0) \neq f(1)$, then the $|0\rangle$ terms will cancel out (because one will get a plus sign and the other will get a negative sign) and we will be left with only $|1\rangle$ terms in the data qubit. So, we can now measure the data qubit, knowing that if we measure $|0\rangle$ then f(x) is constant, while if we measure $|1\rangle$, then f(x) is balanced.

If f(x) only operates on a 1-bit number (as in this example), then it is extremely easy to just measure f(x) on all the possible inputs classically (because there are only two possible inputs). However, imagine f(x) operates on a 50-bit number. We would then have to apply the function to $2^{50}/2 + 1 \approx 10^{14}$ inputs to know with 100% probability that f(x) is constant. In contrast, the quantum algorithm allows us to determine this is only one application of the function.

Even so, the problem is a bit contrived. Although not a deterministic answer, one can establish classically whether the function is balanced or constant with high likelihood after several trials. For example, if one repeatedly outputs a 0, then one gradually builds confidence that the function is likely constant. There are not any real-world problems in which we have a function that we know is either constant or balanced. To obtain the answer with certainty for any a priori unknown function, a classical computer will need to make $N/2+1$ queries, whereas the Deutsch-Jozsa algorithm only needs to make one. Thus, although the Deutsch-Jozsa algorithm does not address a practical problem, it is an easy-to-follow demonstration of how quantum computers can give exponential speedup over the best possible classical algorithms.

58 Introduction to Quantum Software Program General overview

Up to this point, we have focused primarily on quantum computing algorithms and hardware. However, how do we efficiently design a quantum processor? How do we validate that it is working properly? Moreover, how do we program it? As with classical computers, there is an

array of software that is used for these purposes. In this section, we will discuss the similarities and differences between the classical and quantum computing software stacks.

Like classical computers, quantum computers have a software stack used in the development of programs, analysis of designs, and testing of constructive programs and circuits. The function and examples of these quantum software tools are a topic of this section [31]. Let us start by contrasting quantum software tools to classical ones to perform similar types of functions. To create a classical circuit design, one may use a schematic capture CAD tool. A quantum analog to this is the circuit composer from IBM's quantum experience. For program-based designs, for example, HDL circuit designs are written in VHDL or Verilog, or high-level computer programs are written in a language like c++, one uses a compiler to translate the program to a lower level format. Numerous quantum compilers exist that perform the same function. Some examples include Quipper developed by Dalhousie University and Q sharp from Microsoft. These quantum compilers typically produce an intermediate format known as QASM [233] or quantum assembly language. Finally, to translate a program or circuit to a format that a machine can execute, one uses an assembler. A similar function is required for quantum circuits. Each hardware platform will have specific gates that it can execute, and the connectivity that it provides is specific to the technology's topology constraints. Mapping from QASM programs to hardware is something that IBM's QISKit software performs [233], in this case, for superconductor-based technology. For classical circuit designs, a circuit simulation tool like Spice is commonly used to determine circuit properties like power consumption, timing and to verify the correctness of the circuit. One of the main concerns of quantum circuits is how noise or errors impact the operation of the circuit. Quantum simulation tools exist at multiple levels of abstraction that model error in the operation of quantum circuits. Quantum assimilation tools are also used to calculate the fidelity of gates operating under the control waveforms and to verify circuits' correctness. The last category of tools as those used for testing of fabricated chips, PCBs. For classical chips, one may use hardware and software tools, such as JTAG and boundary-scan, to verify the operation of a chip. For quantum circuits, quantum characterization, validation, and verification, or QCVV tools provide a similar function. These tools evaluate results obtained from experimental runs to calculate the Fidelity devices and to verify their correctness. Let us now discuss the main steps one would use to program a quantum computer. Program generation, hardware-specific circuit mapping, and hardware control and execution. The program generation phase is hardware agnostic, whereas the other two phases deal with aspects specific to the architecture of the hardware platform [234]. Another difference between quantum computing systems and classical ones is that the control and processing systems are typically separate. There is a classical system for control and a quantum device for processing. In many cases, these two systems are physically separated. For example, a superconducting quantum computer may use room temperature instruments for control, whereas the quantum circuits require millikelvin temperatures, and must be located within a dilution refrigerator. Let us now look at the three main methods for program generation, the first being schematic capture. One of the front ends of IBM's quantum experience, the composer tool, is an example of a schematic capture tool, where one uses a GUI to construct a circuit consisting of one and two-qubit gates and measurements. The tool knows of underlying hardware constraints and prevents the user from violating these. In many cases, it is convenient to specify the quantum circuit as a program, which allows for the specification of larger-scale circuits. This leads to the motivation behind the second method for program generation, high-level quantum languages. Examples of this are tools like Quipper, Q Sharp, and Scaffold. High-level programs written in these languages are compiled into QASM circuits or can be displayed as circuit diagrams. The last method for program generation that we will discuss is problem specific generation. Google's Open Fermion software tool is an example of this method. This package is designed specifically for formulating simulations of quantum

chemistry. An advantage of this approach is that it incorporates the steps and the known methods used for the class of problems, without requiring the user to know detailed domain knowledge or the methods used to solve these problems. A user specifies a problem at a high level, and the package generates a quantum circuit that can be mapped to a hardware-specific platform or simulation systems. Thus, we have seen the type of software tools commonly used to program a quantum computer and how these tools are like tools used for classical computers. In the next section, we will look at other types of software used in the control, design, and testing of quantum computers.

59 Introduction to Quantum Software Analysis Testing

In this second section on software, we discuss of instrument control then transitions to software used for circuit validation, verification, and benchmarking, and software tools used to model qubit performance in the presence of noise at various levels of complexity.

As discussed previously, a quantum computer requires a classical control system. For small to medium-sized quantum computers, the control can be realized using commercially available instruments, such as arbitrary waveform generators, microwave sources, and laser systems. Software is required to program and synchronize the various instruments required. Labor is one example of this type of software. This software also provides a result visualization and logging functions. Quantum computer control systems that are custom designs require firmware and software control software. IBM's QISKit for superconducting circuits and ARTIQ for ion traps are examples of these types of software control systems. Additionally, and especially for small scale experiments, many experimental groups use homegrown software control systems. We now discuss hardware testing and validation of quantum systems, which is typically performed with the aid of analysis techniques and software known as QCVV, Quantum Characterization, Verification, and Validation. These techniques take the results of a sequence of experiments and produce descriptive metrics, like gate fidelity or operator descriptions of the underlying gates. Two of the most popular techniques in use today are randomized benchmarking and gate set tomography. In randomized benchmarking, an experiment consists of a long sequence of a repeated target gate interspersed with random other gates. The random gates essentially randomize the error seen in the target gate and allow us to calculate the average fidelity of this target gate. This procedure also mitigates the impact that imperfect state preparation and measurement have on the fidelity. One of the disadvantages of randomized benchmarking is that it only provides a single metric for the gate, namely, the fidelity, which may not be enough to help diagnose the cause of error in the system. Gate set tomography goes beyond randomized benchmarking by providing process map descriptions of the quantum gates. These process maps are operator descriptions of both the gate and the error seen in the experiment. Both randomized benchmarking and gate set tomography require a large amount of data and are, therefore, limited to the analysis of small quantum systems. Developing scalable QCVV techniques is an ongoing area of research, which will be increasingly important as larger quantum systems emerge. As mentioned earlier, noise and error are a major concern in today's quantum devices and computers. One of the main uses of quantum simulation software is to understand the impact of this noise. One can apply modeling and simulation at many different levels of abstraction, ranging from finite element models of materials to devices up to quantum circuit models. QuTiP is a Python-based package that provides libraries useful for modeling and simulating open quantum systems, and systems interacting with unwanted degrees of freedom in the environment. Static modeling can also be applied to devices and circuits to obtain important metrics affecting their performance as qubits. These metrics include the coherence time of the qubits and the sensitivity of the qubit

to specific types of noise. Simulating the performance of quantum error correction circuits is another important use of modeling and simulation. Here, one uses simulation to determine the logical error rate of the circuit and determine the scaling of this logical error rate as a function of the individual physical devices. One final example of simulation does not involve error at all. Simulation is also used to understand the computational advantage that quantum computers have over classical ones. Quantum supremacy circuits have been proposed as circuits that are difficult to simulate classically but are easy for a quantum computer to execute. Several groups have developed parallel simulators that run on high-performance computing systems and use these simulators to determine how large a circuit is required to demonstrate a quantum advantage [235]. Thus, in summary, quantum computers have a software stack that serves the same purpose as the software stack used for classical computers [236]. These quantum software tools are important for programming, design, and testing of today's quantum computers. It will be even more important to the goal of realizing large scale quantum computers.

There exists an array of software tools used in the design, programming, and validation of classical and quantum computers. On the one hand, circuits are designed using CAD software, including circuit layout tools and conceptual circuit schematic-capture. On the other hand, programs are written and compiled through several layers of abstraction, from the high-level program codes we use to "write programs" to the compilation of instruction sets and low-level hardware-specific implementations [237]. Besides, the software is used to make predictions of performance for differing hardware architectures. Once assembled, hardware needs to be benchmarked with the assistance of software. All together, these various software tools comprise the software stack. There are similarities and differences between the classical and quantum versions of a software stack.

The path from the underlying quantum-physical operations (unitary evolutions) introduced in the previous section to actual physical quantum systems is highly dependent on the hardware [238]. Aside from high-level descriptions of quantum gate sequences, there are currently no set standards and, just as many qubit modalities continue to compete, the development of the quantum computing software stack today is a very diverse undertaking. It is linked to the efforts of several academic groups, larger corporations, and small start-up companies, many of which have only formed in the last 2 years. This page of a volunteer-run wiki provides a comprehensive list of a large number of available software platforms, ordered by type, and the (classical) programming language by which they are realized.

Programming a quantum computer comprises three primary types of software:

1. Program generation software
2. Circuit mapping software
3. Control and execution software

Program generation is generally technology agnostic, whereas circuit mapping, control, and execution all require knowledge of the hardware architecture. Program generation can be further divided into three categories:

1. schematic capture (generally related to the quantum circuit model)
2. high-level programming languages (abstracting away subroutines such as phase estimation or period-finding via the Quantum Fourier Transform) and lower-level assembly languages (abstracting away from specific hardware controls and subroutines)
3. problem-specific platforms (e.g., to generate auxiliary inputs to quantum chemistry calculations)

In general, computer code may be abstracted multiple times before the high-level language used by human beings to "code" a quantum computer is compiled and assembled into the instructions that are ultimately used at the hardware level. One example of this reduction is the quantum assembly language (QASM) that we will use to program the Deutsch-Jozsa algorithm

directly. QASM is very generic and close to the quantum circuit model.

As this reduction occurs, the hardware-agnostic codes must eventually be implemented on physical circuits using hardware-specific control and execution. Whereas the control and execution systems are typically co-located (and, in many cases, the same technology base) for classical computers, quantum processors are generally controlled by classical hardware. These are different technologies and often reside in different locations. For example, a superconducting quantum computer is controlled with classical electronic instruments at room temperature, while the quantum information is processed inside the dilution refrigerator at cryogenic temperatures [1].

60 IBMQ Experience and QASM Programming

In this section, We introduces the IBM Quantum Experience quantum computing system, a graphical user interface called the composer, and a programming language called QASM.

We think one of the most exciting developments in quantum computing in recent years has been the ubiquitous access to real quantum devices. Just a couple of years ago, students would discuss concepts and quantum information and quantum computation without any real means for hands-on experimentation. However, in 2016 with the launch of the IBM Q Experience, the first generation of quantum computers came online. They have generated much excitement, with nearly 80,000 unique users running more than three million experiments. this section will introduce us to some of the resources that will help us get started with quantum programming. Let us start by looking at the composer, which is a simple graphical user interface for building and executing quantum programs. This space has two parts. In the top part, we see information about the devices that are available for experiments. On the left, we see a schematic of the chip where each white square is one of the qubits. right next to that is a diagram of how the qubits are connected to each other. This connectivity diagram influences the operations we can do on this device, as We will see in a moment. One important thing that we must get used to when working with quantum computers is that They are always noisy and imperfect [239]. So, on the right, we see a lot of information about the noise level on each of these qubits and each of the gates. We do not have to believe these numbers. we can design and run certain experiments that will let us measure the exact errors. There is a lot of more interesting data about each of these chips. For example, we see that this one is sitting in a dilution refrigerator with a temperature of around 21 millikelvin, which is colder than the surface of Pluto. At the bottom of the page is the composer itself. we can see there are five wires, each representing one of the qubits in the quantum computer. on the panel on the right, we have the gates available for building the quantum circuit. As we know, the circuit model is a simple yet powerful model for quantum computation. A quantum circuit is a recipe for how to transform the state of several qubits by applying various gates. It can be shown that any quantum computation can be done by just operating on one or two qubits at a time. the very small set of gates is enough for universal quantum computation. So as simple as this interface seems, we can do quite complex computations with it. So, let us do a simple entanglement experiment. All the qubits initially start as being in the zero states. To create an entangled pair of qubits, we first put one of the qubits in a superposition state by applying a Hadamard gate to it. Next, we toggle the second qubit conditioned on the first qubit by applying a CNOT gate. This leaves the qubits to go in a state of 0, 0 plus 1, 1, a state that cannot be described in terms of the state of each qubit individually. It is an entangled state. However, how can we know what state in the qubits are? The qubit is state-space inaccessible to us. the only way we can get any information is to measure the qubits. So, let us add two final measurement operations to this circuit. However, each time we measure the qubits, we read a normal classical bitstream, such as 0, 0, or 1, 1. So, to infer

the qubits' state right before the measurement, we have to repeat this experiment many times and approximate the probability distribution that existed with the entangled states. So now that we are done building the circuit let us execute it. we see that we have the option of either sending our circuit to a simulator or real hardware [122]. First, let us do a simulation to make sure the circuit is working as expected. here is what we get, a histogram of all the measurement outcomes. By default, in the IBM Q Experience, each circuit will be executed about 1,000 times. As we expected, roughly 50% of the time, we measured both qubits in state 0. the other 50% of the time, we measured both qubits in state 1. This is an indication of the state correlations that arise as a result of quantum entanglement. Now let us submit the circuit to a real quantum computer. By clicking run, our circuit will be sent to the IBM Research Labs in New York, where they will be translated to the language of qubit manipulations, namely, control pulses. After we receive the measurement results, we see that, indeed, the same outcome is achieved. However, there are some imperfections here in the result. this is exactly due to the noisy qubits and gates that We discussed earlier. A composer is a great tool for just playing around and visualizing our circuits. However, we may prefer to build or alter our circuits more quickly. we can do this by switching to the QASM Editor, which gives us a textual representation of the circuit. we can also import QASM from the file. QASM, or quantum assembly, is a circuit description language. The graphical interface we were using previously was being translated to QASM code under the hood. QASM is a hardware-agnostic language, meaning it can be translated to any physical chip or even a different quantum computing technology altogether. It expresses data dependencies without explicit timings for the instructions, which will be decided at a later stage. Here we see the same circuit we just built written in QASM. The first line imports a standard library of gates for us to use. This is exactly akin to the menu of gates we had previously. The next two lines define the qubits as a quantum register of size 5 and the classical bits, which will be used to hold the final measurement results. Finally, we have the Hadamard, CNOT, and measurement operations. Let us suppose that we want to repeat the entangling experiment, but this time on two different qubits, let us qubit 2 and 3. This seems trivial. We just change the indices on Q and C. However. We get an error message saying that a CNOT is not allowed from Q2 to Q3. That is right. This is due to the connectivity graph of the qubits in this chip. We see that in order to do a CNOT operation between these two qubits, we need to designate Q3 as control and Q2 as a target. Luckily, there is an easy transformation that can be used to flip a CNOT. This circuit identity is achieved by sandwiching the CNOT that we have between two layers of Hadamard's, giving us the flipped CNOT. Now we repeat the experiment on the real chip again. We see the same result as we expected. If we look closely, however, we see that the accuracy of the results is slightly degraded compared to the previous experiment. This is because the new circuit uses more gates, leading to a higher chance of errors accumulating in the circuit. To conclude, it is easy to use the IBM Q Experience to program a quantum computer. All the many layers of technology that goes into building a working quantum computing stack are conveniently abstracted. Quantum information science is no longer only done in the lab [240]. Now we can control a real quantum computer remotely. While these devices are still small and noisy, in a way, they force us to be cleverer in using them. For example, we must consider the connectivity of the qubits and try to use fewer gates in order to preserve information fidelity. We will see many more interesting examples of quantum information science if we visit the IBM Q Experience user guide, and each one accompanied it with its composer circuit and QASM code. we also encourage us to get involved in the Community Forum, which is a great place to discuss anything related to quantum computing and discussing it from each other.

In the following sections, we will discuss Open QASM [233] and how this programming language can be used to demonstrate several simple quantum circuits and, ultimately, the Deutsch-Jozsa quantum algorithm on the IBM Quantum Experience quantum computer.

The IBM Q Experience platform allows anyone with access to the internet to write and run quantum algorithms on a real quantum computer. In this section, we will receive the necessary resources to implement algorithms on the IBM platform. However, we encourage to register on the IBM page to access the composer's graphical interface (alternatively, if we are comfortable writing text-based programs, the interface here may be just fine).

The IBM Q composer allows one to express quantum circuits and quantum algorithms in a simple graphical way, and on the IBM QE site, we will be able to save the results of our programs and access them later. To introduce us to the use of the platform, IBM has a user beginner's guide. If we want to discuss how to implement more complicated algorithms, we recommend we visit their full user guide. For researchers, it is now possible to request exclusive access.

Once in the composer, we can create a new experiment and choose if we want to run our program on a real quantum computer or to simulate it using a classical computer. When we decide to run our code on the quantum computer, we will have to choose the number of times that we want to run the code and which "backend" we will use. There are four possible backends on which we can run our codes. We will only use either the classical numerical simulator or one of the five-qubit backends, ibmqx2 [241] (may be unavailable due to maintenance) and ibmqx4 (when available) [241]. We will discuss more backend options in the following sections.

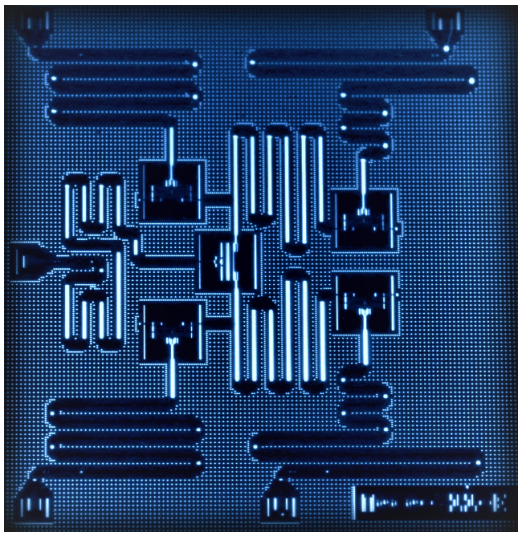


Figure 36: IBM QEX2

The IBM Q Experience platform not only allows us to use the composer, but it also allows us to write our code using Open QASM (Quantum Assembly Language). This is a simple text language that has elements of C and assembly languages. By using QASM, we can describe any quantum circuit, in principle, since it has built-in a universal set of quantum gates. In this section, we will discuss how to express quantum circuits and write quantum algorithms using QASM. If we want to discuss the theory behind QASM, we recommend that we read the following paper written by IBM researchers: Open Quantum Assembly Language [233].

Here, we will receive detailed instructions on how to use QASM, we will guide us step by step on how to create our first programs, and we will teach us how to read our results. In this first set of exercises, we will discuss syntax and common mistakes, measurements, single-qubit gates, two-qubit gates, and backends. We will have the opportunity to practice our knowledge and to prove our skills in the end-of-section test. Besides the results that the IBM platform gives us,

we will also make a theoretical analysis of the quantum state evolution, so we will easily see the correspondence between the analytical and experimental results.

The numerical simulation employed here is based on the IBM QISKit engine, an open-source package that allows the inclusion of gate and measurement noise, for added realism. This noise model is based on recent calibration data from the 5-qubit IBM QE machines. The output histograms are statistically similar to those that we would obtain from real quantum computers.

By default, each exercise begins with numerical simulations to give us the most rapid feedback for our check-our-understanding answer submission. With select exercises, once our submission is deemed correct, our QASM program will automatically be submitted for execution on a real quantum computer. Once it is run, we should see a plot comparing the simulated and real quantum computer's results, side-by-side, like this:

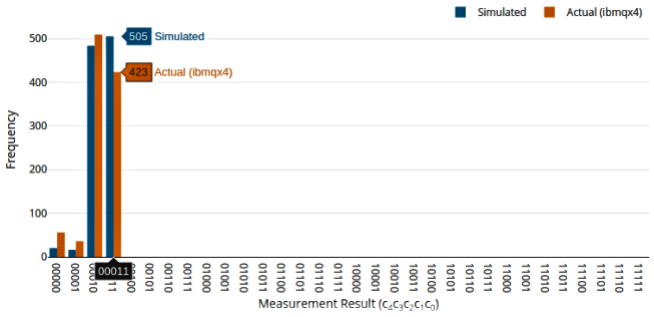


Figure 37: Real QC Results Histogram

We will need to be patient, however, because these run requests must be queued for execution since hardware availability is limited. Each program must be run multiple times to generate the output statistics needed for a pedagogical explanation.

61 Syntax of QASM

In the figure below, each horizontal line represents the evolution of a qubit with time proceeding from left to right. The five qubits are labeled in order as: q[0], q[1], q[2], q[3] and q[4]. The qubits are always initialized in the quantum state:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} (1) \tag{52}$$

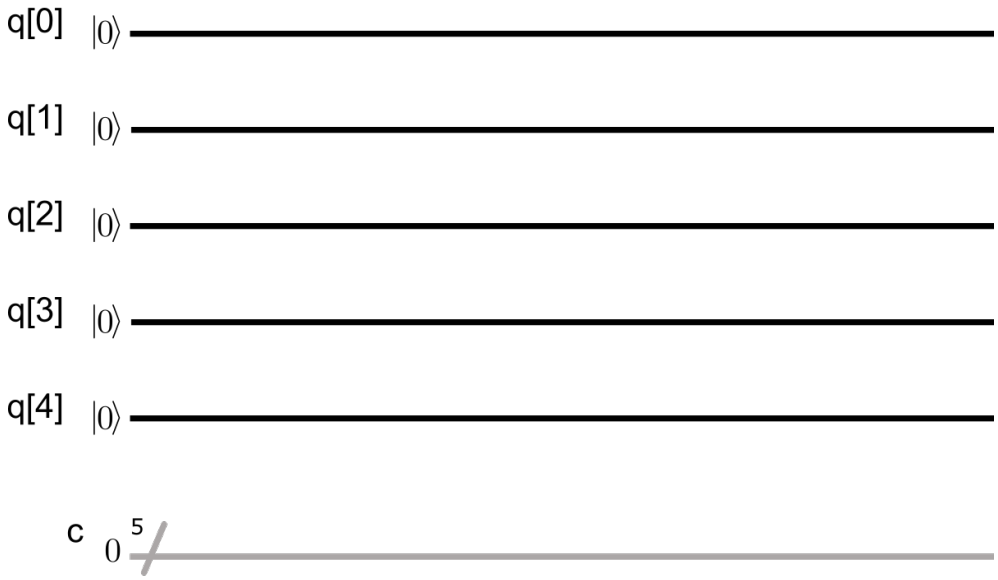


Figure 38: Circuit

Classical bits used to store measurement results are indicated by the letter "c" and the number "5" indicates that there are five classical bits in the register being represented with the grey line.

The figure below shows a graphical representation of adding a measurement block to the first qubit. The vertical grey line indicates that the measurement result will be stored in the classical bit register. The number below the grey arrowhead indicates into which classical bit the result will be stored. In this case, the result is stored in bit c[0].

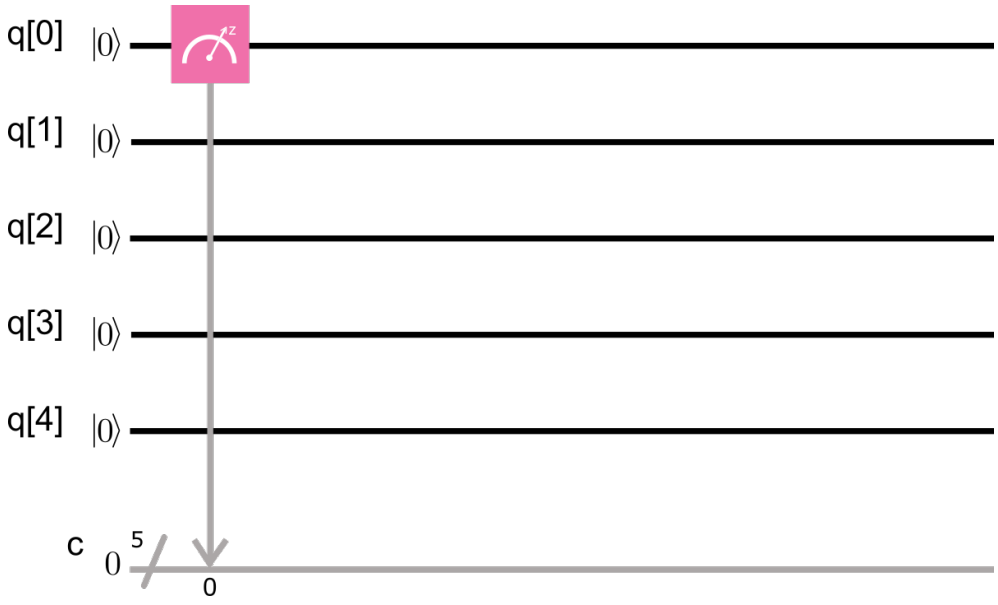


Figure 39: Circuit

```
The following QASM code implements the quantum circuit above:
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];

// This is a comment
measure q[0] -> c[0];
```

Let us take a look at what each line of the previous code does.

Line 1: Includes all gates in the quantum gate library. By adding this, we can then use the X-gate, Y-gate, Z-gate, CNOT gate. Notice that each line has to end in a semicolon ";" (without quotes).

Line 2: Defines a quantum register of five qubits. In this example, we are using five qubits, but we can elect to use any number of qubits from one to five. Note that even if we define a two-qubit register, the quantum circuits depicted here and at the IBM quantum experience site will always graphically illustrate all five qubits by default. This is because the physical quantum computer being used has 5 qubits. Nonetheless, we can choose to work with only a subset of those five qubits.

Line 3: Defines a classical register of five bits. This register is used to store the quantum measurement results. As with the quantum register, even if we define a two-bit classical register, the quantum circuits will indicate all five bits by the grayline. Nonetheless, we can choose to work with only a subset of those five bits.

Line 4: we can insert blank lines to help us organize our sections.

Line 5: we can also insert comments using the "//" (two back-slashes, without quotes). This helps other readers understand our intention in each line. Text that appears after the "//" (without quotes) is not evaluated.

Line 6: Measures the first qubit q[0] and stores the measurement result in bit c[0]. We could alternatively store the information on c[1], but, for organizational purposes, we recommend a direct numerical mapping q[0] to c[0].

1. Syntax Error I, Identify on which line there is an error:

```
include‘‘qelib1.inc’’;
qreg q[5];
creg c[5];
measure q[0] -> c[0];
```

Solution:
A space is needed between include and "qelib1.inc".

2. Syntax Error II, Identify on which line there is an error:

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5]
measure q[0] -> c[0];
```

Solution:
There is a missing ; at the end of the line.

3. Syntax Error III, Identify on which line there is an error:

```
1 include ‘‘qelib1.inc’’;
2 qreg q{5};
3 creg c[5];
4 measure q[0] -> c[0];
```

Solution:
Parentesis should be square [] not .

4. Syntax Error IV, Identify on which line there is an error:

```
1 include ‘‘qelib1.inc’’;
2 qreg q[5];\\ this is a comment
3 creg c[5];
4 measure q[0] -> c[0];
```

Solution:
Comments sould be added after // not
.

5. Syntax Error V, Identify in which line there is an error:

```
1 include ‘‘qelib1.inc’’;
2 qreg q[2];
3 creg c[2];
4 measure q[0] -> c[2];
```

Solution:
The classical register index should be less than 2. In line 3, creg $c[2]$ defined a classical register with two indexes $c[0]$ and $c[1]$.

62 Measurements

As we discussed in the previous section, the quantum measurement is probabilistic. When we perform a measurement, we project the state of the qubit onto either $|0\rangle$ or $|1\rangle$ with a probability that is the magnitude squared of their respective coefficients (their probability amplitudes). To understand how QASM deals with measurements, Let us go back to the first example,

```
1 include ‘‘qelib1.inc’’;
2 qreg q[5];
3 creg c[5];
4
5 // This is a comment
6 measure q[0] -> c[0];
```

In this example, the first qubit q[0] is measured, and the result of the measurement is stored in classical bit c[0]. The analytical probabilities of projecting q[0] onto states $|0\rangle$ and $|1\rangle$ are given by $p(q[0], |0\rangle) = |\langle 0|0\rangle|^2 = 1$ and $p(q[0], |1\rangle) = |\langle 0|1\rangle|^2 = 0$.

The IBM QE platform either simulates QASM code or runs it on a real quantum computer, producing numerical results that can be presented as a table. The following figure shows the tabular result of simulating the previous code over 10 "shots" identically prepared and executed experiments. In this example, the qubit was projected 10 times onto state $|0\rangle$. Note that because the qubit is only projected on to one state in this example, the only label is 0.

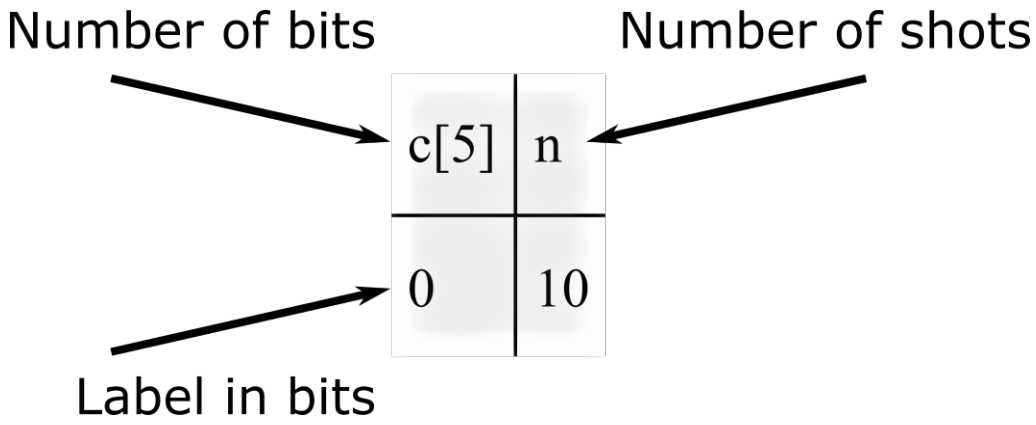
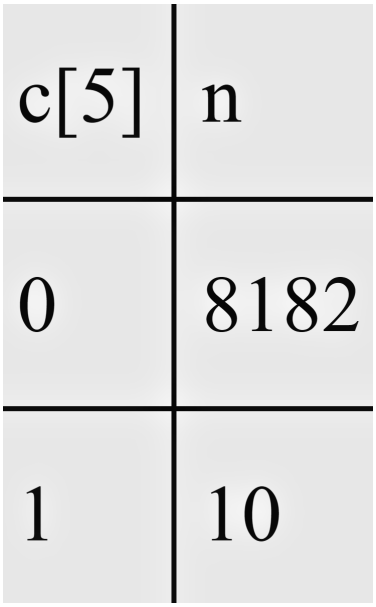


Figure 40: Distribution

The results are different when the code is run on a real (imperfect) quantum computer. The figure below shows the result of executing the code 8192 times. In this case, there are two different labels because qubit q[0] was projected to state $|0\rangle$ but was sometimes projected to state $|1\rangle$. Specifically, it was projected on to $|0\rangle$ 8182 times and onto $|1\rangle$ 10 times.



c[5]	n
0	8182
1	10

Figure 41: Distribution

6. Measuring I, Which option creates a classical register with 3 indexes?
creg c[0];
creg c[1]
creg c[2]
creg c[3]
Solution:
For the classical register to have 3 bits, it must be defined as "creg c[3]", this gives we indices c[0], c[1] and c[2]. The first qubit is labeled q[0], the second is labeled q[1], and so on. To store the second qubit in c[1] we must assign "$q[1] -> c[1]$"

7. Measuring II, Which quantum circuit corresponds to the graphical representation of the following code

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];
measure q[1] -> c[3];
measure q[3] -> c[1];
```

Figure 42: MC 1c

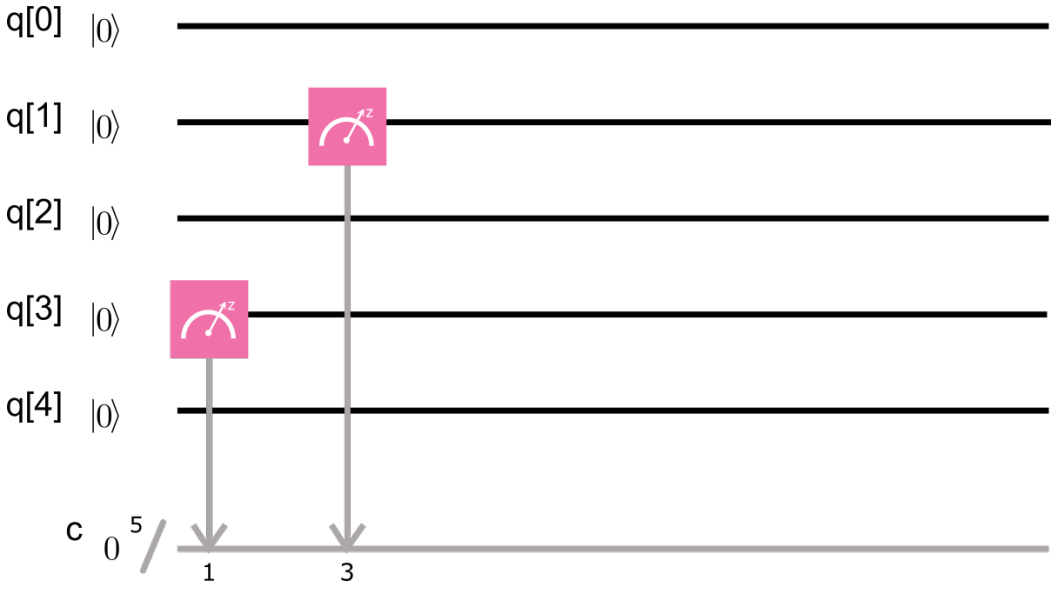


Figure 43: MC 1b

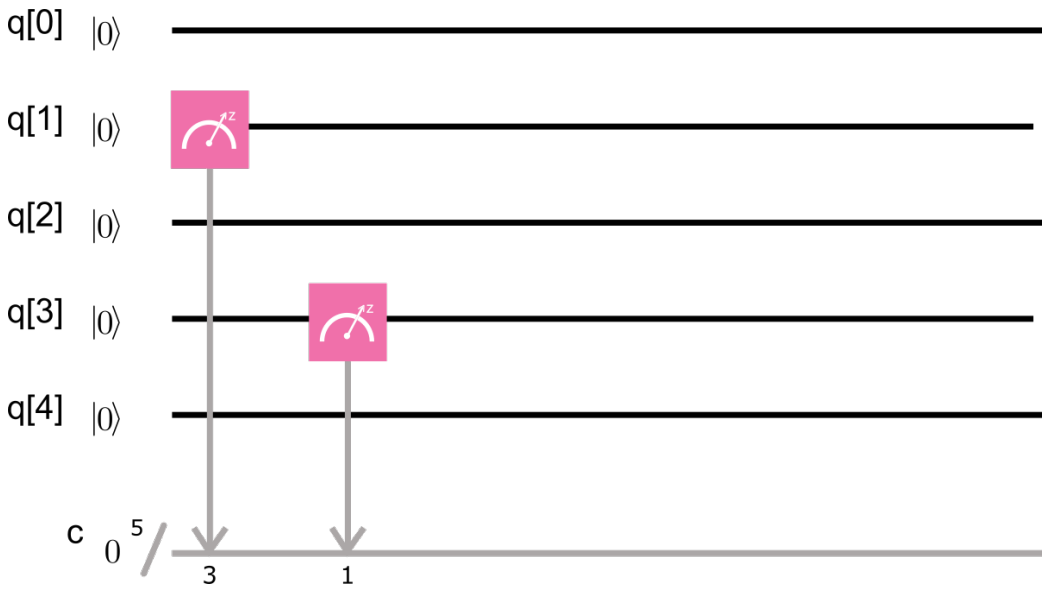


Figure 44: MC 1a

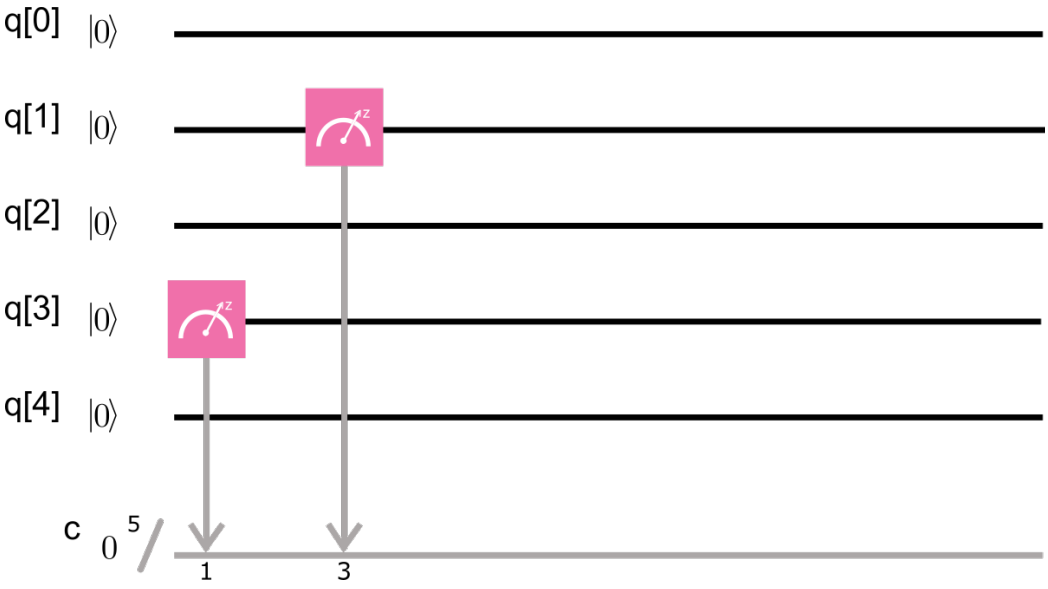


Figure 45: MC 1d

To better understand how the measurement results are stored in the classical register, Let us consider the quantum circuit below.

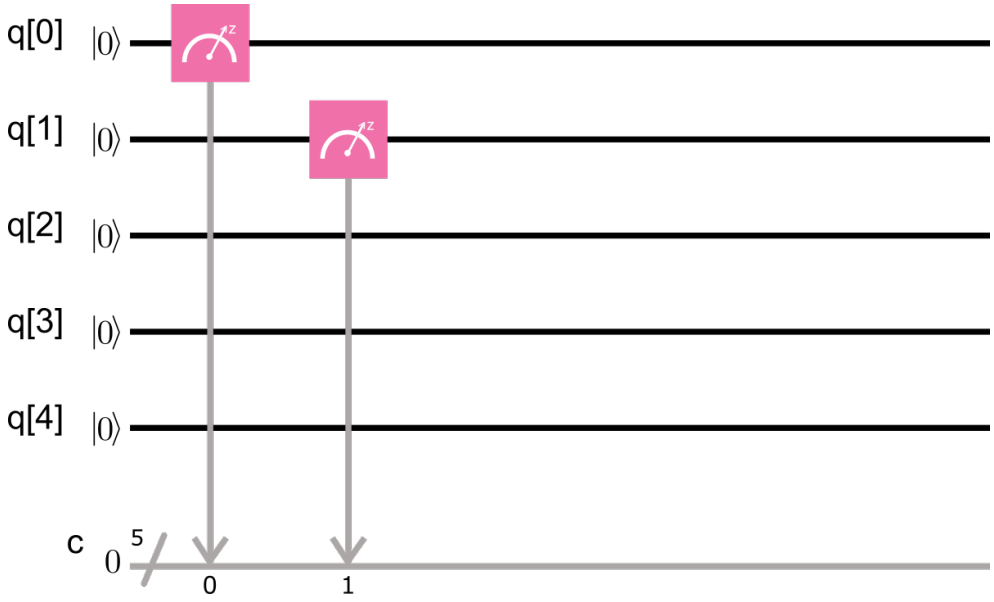


Figure 46: circuit

The QASM code that generates this circuit is

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

Similar to the previous example, the analytical probabilities of projecting qubits q[0] and q[1] onto states $|0\rangle$ and $|1\rangle$ are respectively given by $p(q[0],|0\rangle) = |\langle 0|0\rangle|^2 = 1, p(q[0],|1\rangle) = |\langle 0|1\rangle|^2 = 0, and p(q[1],|0\rangle) = |\langle 0|0\rangle|^2 = 1, p(q[1],|1\rangle) = |\langle 0|1\rangle|^2 = 0$.

The following figure shows the result of simulating the previous code with 10 shots. Qubits q[0] and q[1] were projected 10 times into state $|0\rangle|0\rangle$. Note that the label in the left column is “00” because two qubits are being measured.

c[5]	n
00	10

Figure 47: Distribution Updated

The figure below shows the result of running the previous code in a real quantum computer 1024 times. Since the qubit q[0]measurement was stored in c[0], and q[1] in c[1], the labels are given in the order c[1]c[0]. The number n=1014 at the right of label "00" indicates that qubits q[0] and q[1] were projected onto state $|00\rangle$ 1014 times. The number n=1 at the right of label "01" indicates that qubits q[0] and q[1] were projected onto state $|01\rangle$ 1 time. The number n=9 at the right of label "10" indicates that qubits q[0] and q[1] were projected onto state $|10\rangle$ 9 times. Note that there is no label "11"; this is because the qubits were never projected on to state $|11\rangle$.

c[5]	n
00	1014
01	1
10	9

Figure 48: Distribution Updated

The following table shows a summary of the previous results. We can see that the measurement results are in general but not perfect agreement with the analytical probabilities.

Table 2: Results

Analytical Probabilities	Quantum state $\|q[0]q[1]\rangle$	Result Label c[1]c[0]	Projection Frequency n
1	$\|00\rangle$	00	1014
0	$\|01\rangle$	10	9
0	$\|10\rangle$	01	1
0	$\|11\rangle$	11	0

To understand how the labeling of the results works can be complicated, so let us analyze the following example.

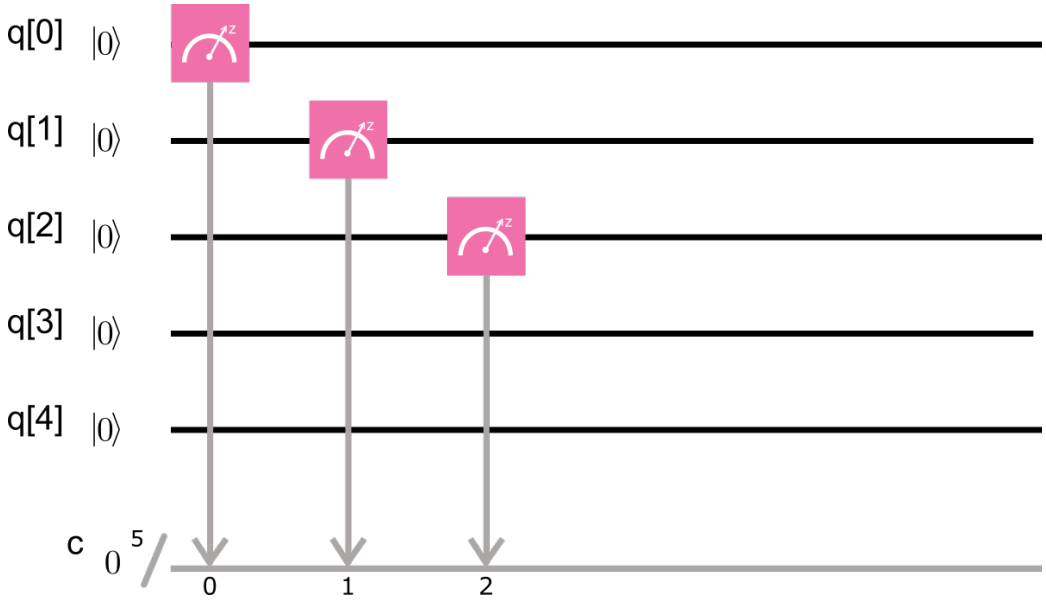


Figure 49: Circuit

The code that generates this quantum circuit is

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

Since the q[0], q[1] and q[2] measurements are stored in c[0], c[1] and c[2] respectively, the projection results will be stored as in Table 2. If instead the measurements are stored in the following way,

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];
```

```
measure q[0] -> c[2];
measure q[1] -> c[1];
measure q[2] -> c[0];
```

then the projection results will be stored as shown in Table 3

Table 3: q[0] − > c[0]; q[1] − > c[1]; q[2] − > c[2]

Label c[2]c[1]c[0]	Quantum state $\|q[0]q[1]q[2]\rangle$
000	$\|000\rangle$
001	$\|100\rangle$
010	$\|010\rangle$
011	$\|110\rangle$
100	$\|001\rangle$
101	$\|101\rangle$
110	$\|011\rangle$
111	$\|111\rangle$

Table 4: q[0] − > c[2]; q[1] − > c[1]; q[2] − > c[0]

Label c[2]c[1]c[0]	Quantum state $\|q[0]q[1]q[2]\rangle$
000	$\|000\rangle$
100	$\|100\rangle$
010	$\|010\rangle$
110	$\|110\rangle$
001	$\|001\rangle$
101	$\|101\rangle$
011	$\|011\rangle$
111	$\|111\rangle$

Consider the quantum circuit and measurement results below and respond to the following questions.

Table 5: q[0] − > c[2]; q[1] − > c[1]; q[2] − > c[0]

c[5]	Quantum state n
000	7939
001	113
010	135
011	3
100	11

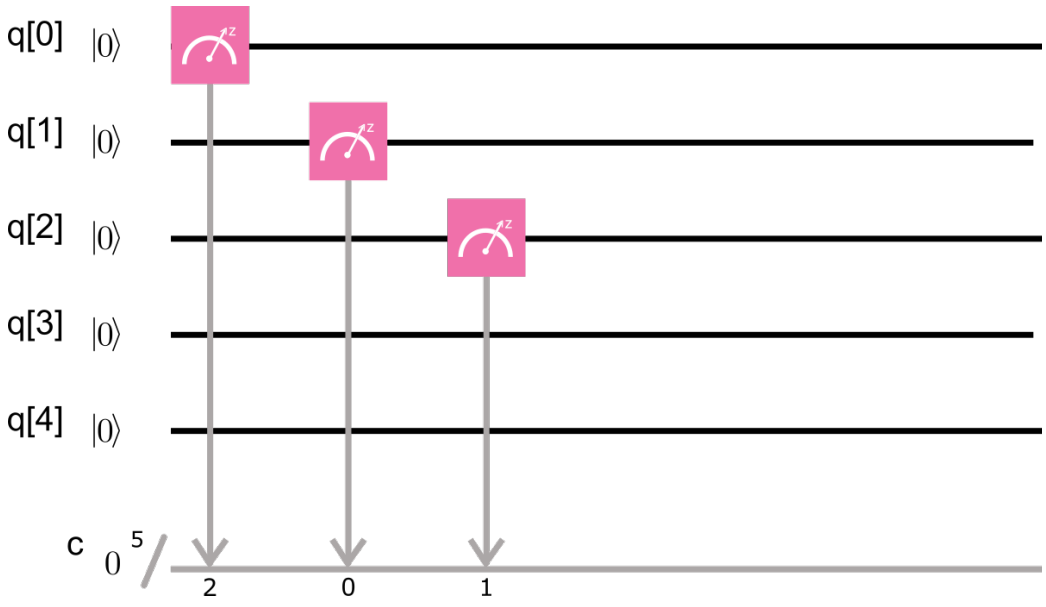


Figure 50: Assessment

8. Measuring III, Which QASM code generates this quantum circuit?

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];
measure q[0] -> c[2];
measure q[1] -> c[0];
measure q[2] -> c[1];
```

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];
measure q[0] -> c[2];
measure q[1] -> c[1];
measure q[2] -> c[0];
```

9. Measuring IV: How many times was the code run? Solution:
The sum of all the measurements gives the number of times the experiment is run; 7939+113+135+3+11=8201

10. Measuring V, How many times were the qubits projected to state $|001\rangle$ Solution:
The corresponding label to state $|001\rangle$ is 010. This label shows n=135.

11. Measuring VI, How many times were the qubits projected to state $|010\rangle$? Solution: The corresponding label to state $|001\rangle$ is 001. This label shows n=113.

12. Measuring VII, How many times were the qubits projected to state $|011\rangle$? Solution:
The corresponding label to state $|011\rangle$ is 011. This label shows n=3.

13. Measuring VIII, How many times were the qubits projected to state $|100\rangle$? Solution:
The corresponding label to state $|100\rangle$ is 100. This label shows n=11.

63 Single-Qubit Gates

Up until this point, we have discussed how to measure qubits using QASM and the IBM quantum computer. The IBM platform also enables us to perform single-qubit and two-qubit operations. Table 1 shows the predefined single-qubit gates and the QASM line to apply them on qubit q[0]. We can extrapolate this concept to other qubits.

Table 6: single-qubit gates

Gate	QASM line
$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	x q[0];
$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	y q[0];
$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	z q[0];
$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	h q[0];
$S = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{pmatrix}$	s q[0];
$S^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & -e^{i\frac{\pi}{2}} \end{pmatrix}$	sdg q[0];
$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$	t q[0];
$T^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & -e^{i\frac{\pi}{4}} \end{pmatrix}$	tdg q[0];

In the figure below, an X-gate is applied to the first qubit q[0], and then q[0] and q[1] are measured. The final state of the two-qubit system is given by $|10\rangle$.

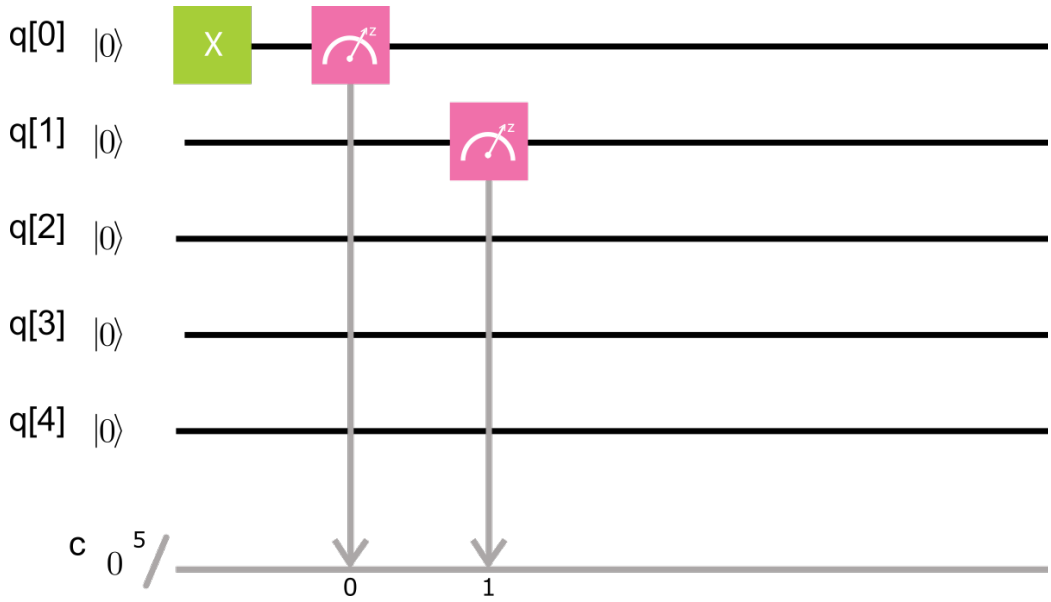


Figure 51: Circuit

The code below generates the above quantum circuit. Note that there are two blank lines, 4 and 6, this is just to make the code look cleaner. In this way, we will have a first section in which we define our registers, a second section in which we apply the quantum gates, and a third section in which we measure.

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];

x q[0];

measure q[0] -> c[0];
measure q[1] -> c[1];
```

Table 2 shows the results of running the code 1024 times on the perfect quantum simulator. Notice that the two qubits were projected onto state $|10\rangle$ every time.

Table 7: Results of running the code 1024 times on the perfect quantum simulator

c[5]	n
01	1024

Table 3 shows the results of running the code 1024 times on a real IBM quantum computer. In this case, notice that the two qubits were projected onto state $|00\rangle$ 70 times, onto state $|01\rangle$2 times, onto state $|10\rangle$ 941 times, and onto state, $|11\rangle$ 11 times. The difference between the results in tables 2 and 3 are not only given by the errors in the measurements, but also by the error in the X-gate.

Table 8: Results of running the code 1024 times on a real IBM quantum computer

c[5]	n
00	70
01	941
10	2
11	11

14. Single-qubit Gates I) Which code corresponds to the following quantum circuit

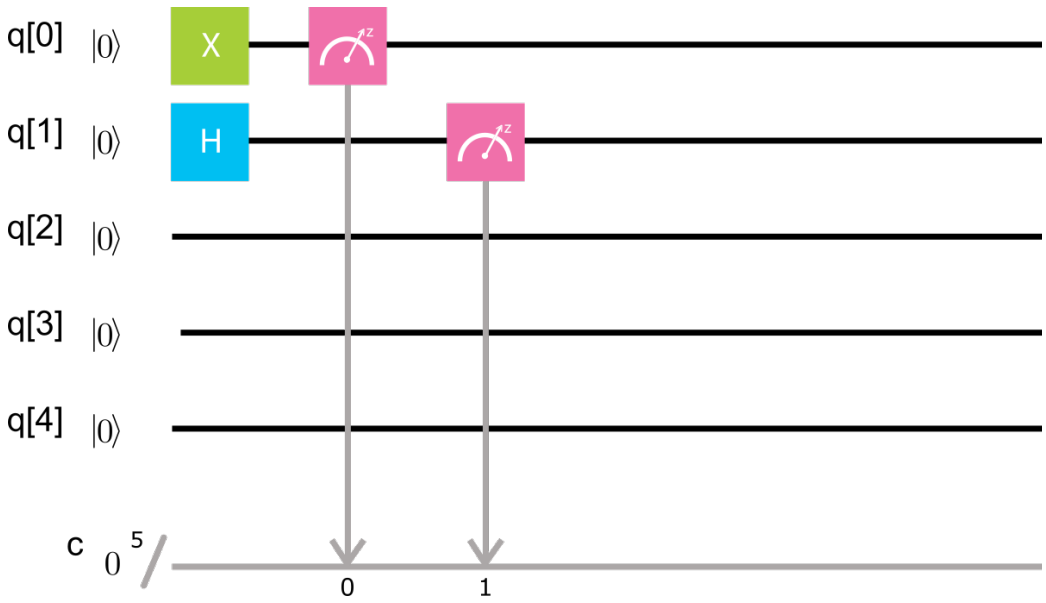


Figure 52: Circuit

```
include ‘‘qelib1.inc’’;

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

x q[0];
h q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

```
include ‘‘qelib1.inc’’;

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

x q[1];
h q[2];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

```
include ‘‘qelib1.inc’’;

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

x q[0];
h q[1];
measure q[0] -> c[1];
```

```
measure q[1] -> c[0];
```

```
include ‘‘qelib1.inc’’;

qreg q[5];
creg c[5];
x q[1];
h q[2];
measure q[1] -> c[2];
measure q[2] -> c[1];
```

Solution:
Since an X-gate is applied on qubit q[0], line 6 should be “x q[0]”. Similarly, line 7 should be “h q[1]”, as this is applying a Hadamard gate on qubit q[1].

15. Single-qubit Gates II

    Which the following code generates a quantum circuit

```
include ‘‘qelib1.inc’’;

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

x q[0];
x q[2];
measure q[1] -> c[1];
measure q[3] -> c[3];
```

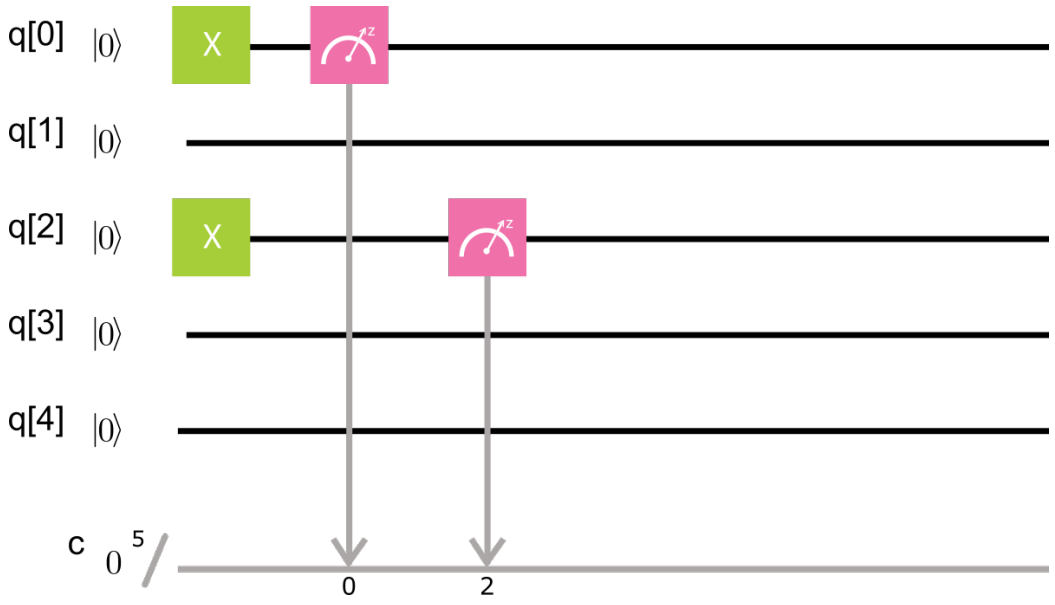


Figure 53: IBM C15a

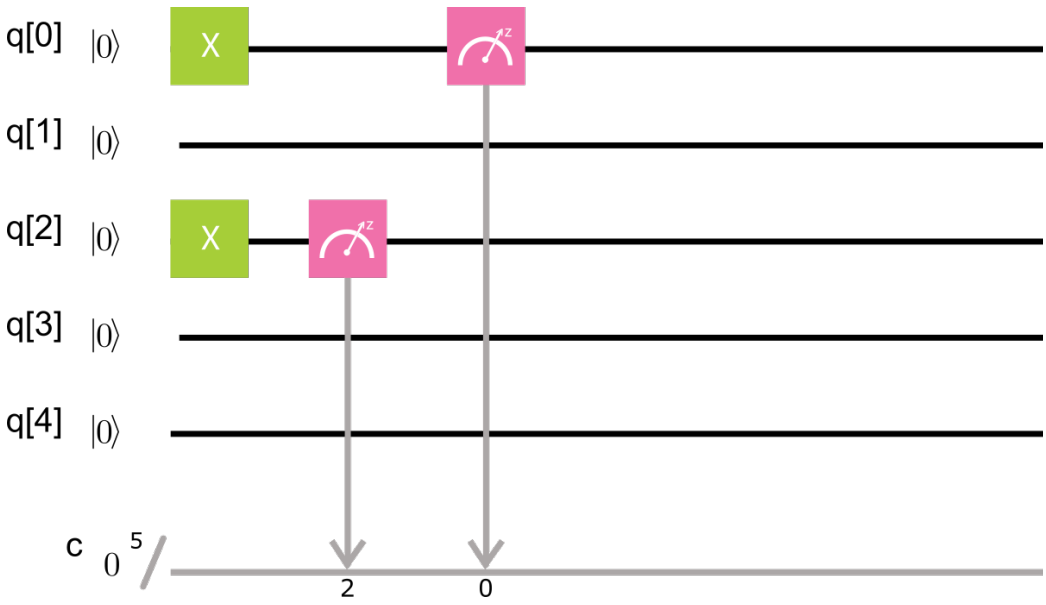


Figure 54: IBM C15b

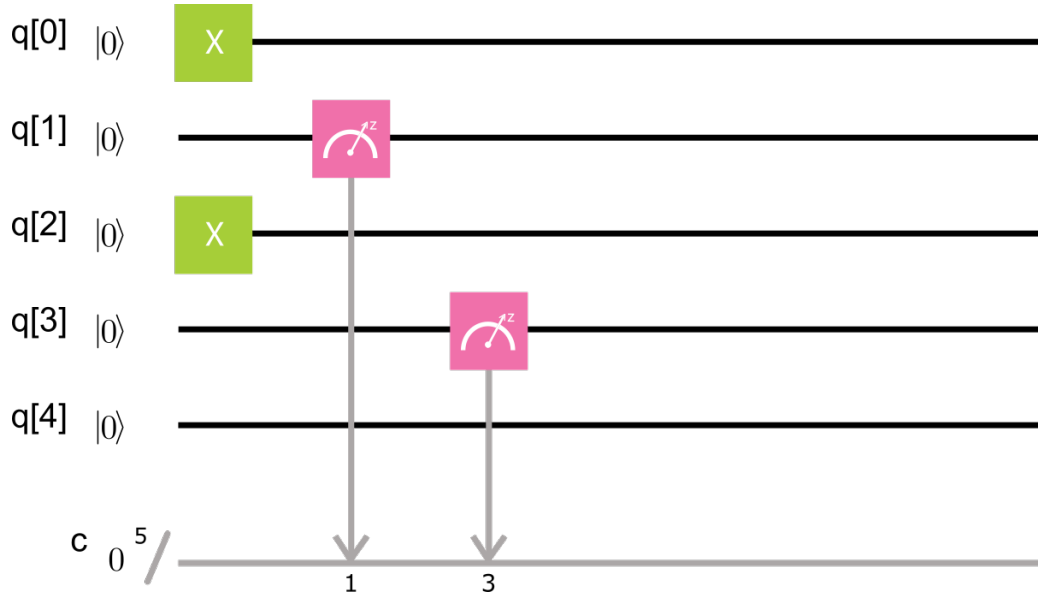


Figure 55: IBM C15c

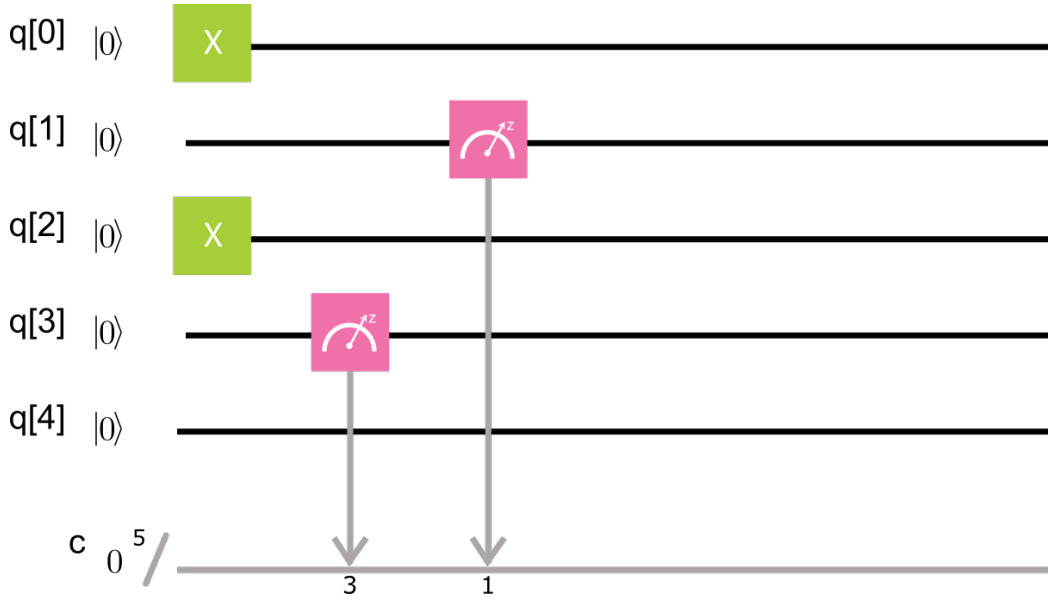


Figure 56: IBM C15d

16. First QASM Code, In the following activity we will run QASM code. Notice that the measurement results will not be given in a tabular format presented as previously, but instead we will see a histogram. Write in the console the following QASM lines:

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];
h q[1];
h q[2];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

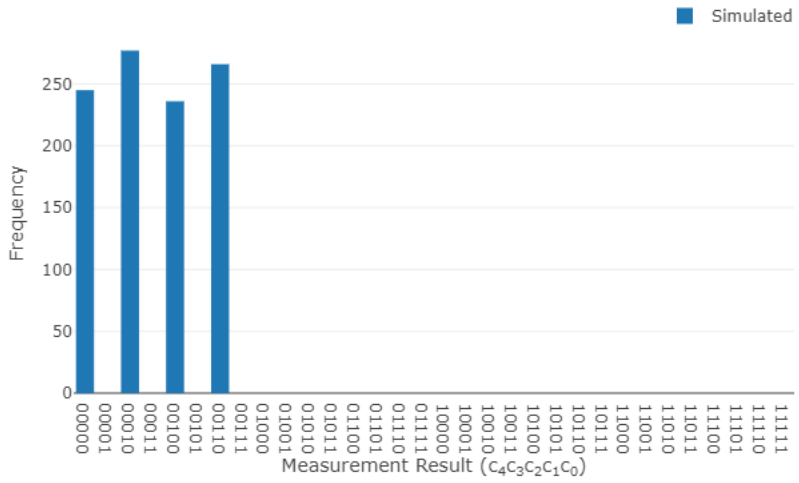


Figure 57: Plot

17. Two Single-Qubit Gates, Write the code that generates the following quantum circuit

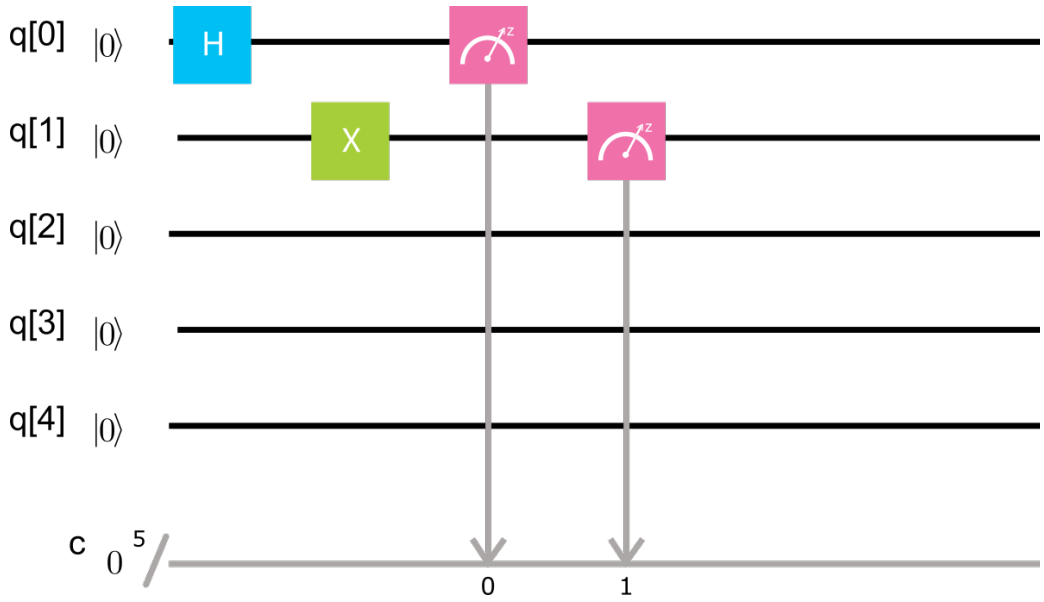


Figure 58: IBM QE2

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];
h q[0];
```

```
5 x q[1];
6 measure q[0] -> c[0];
7 measure q[1] -> c[1];
```

Solution:
Initially, qubits q[0] and q[1] are in the ground state $|00\rangle$. A Hadamard gate is applied on q[0], and an X-gate on q[1]. This leaves both qubits in the final state $\frac{|0\rangle+|1\rangle}{\sqrt{2}}|0\rangle$. In a perfect simulation, the two-qubit system would project on states $|01\rangle$ and $|11\rangle$ with probability 0.5 each.

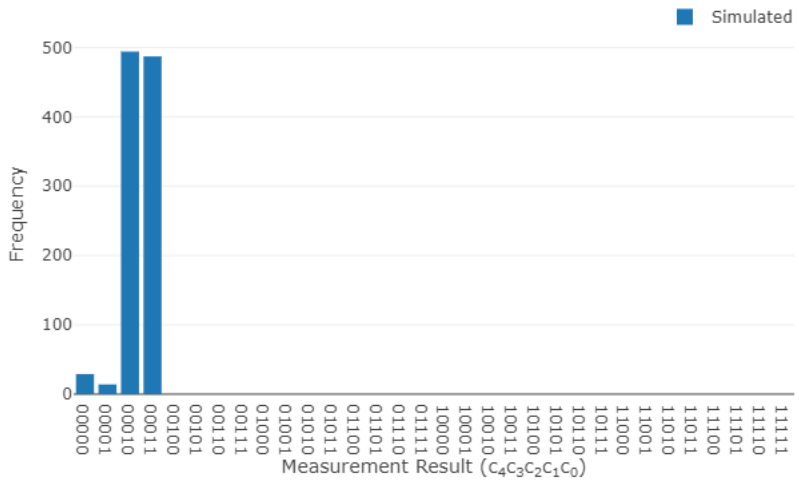


Figure 59: Plot

64 Two-Qubit Gates and Backends

Controlled NOT Gate

As we discussed, a universal gate set includes two-qubit gates. A commonly used two-qubit gate is the controlled-NOT operation or CNOT gate. Independent of which qubit is used as the control qubit and which as the target qubit, the CNOT gate will always apply an X-gate onto the target qubit if the control qubit is in the $|1\rangle$state. When the first qubit is defined as control and the second as a target, the matrix form of the CNOT is given by

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{53}$$

Notice that not all combinations of control and target qubits are allowed in the IBM QE. The combinations that we can use are determined by the topology of the IBM Q experience backend.

64.1 ibmqx2 Backend

The first backend that we will show is called ibmqx2. The figure below shows its topology.

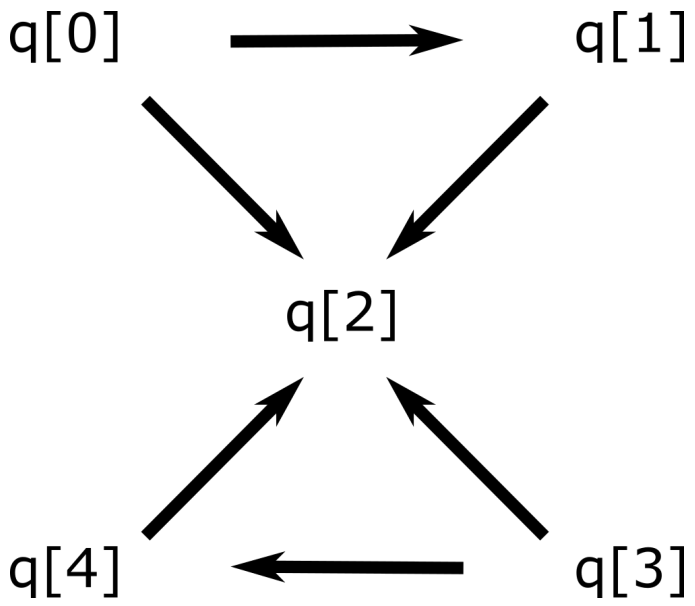


Figure 60: Topology

The following table shows all the allowed combinations for applying a CNOT-gate in QASM using the ibmqx2 backend.

Table 9: CNOT-gate in QASM

Control qubit	Target qubit	QASM line
q[0]	q[1]	cx q[0],q[1]
q[0]	q[2]	cx q[0],q[2]
q[1]	q[2]	cx q[1],q[2]
q[3]	q[2]	cx q[3],q[2]
q[3]	q[4]	cx q[3],q[4]
q[4]	q[2]	cx q[4],q[2]

18. CNOT Gate. The following code generates the quantum circuit below.

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];
x q[0];
cx q[0],q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

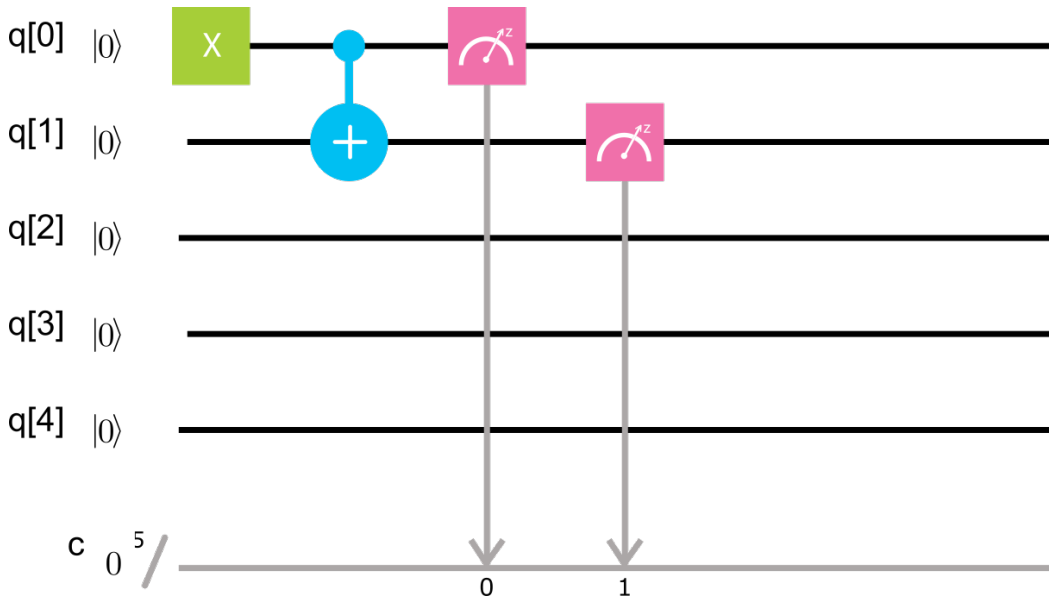


Figure 61: CNIT2

In the console, write the code that generates the following circuit.

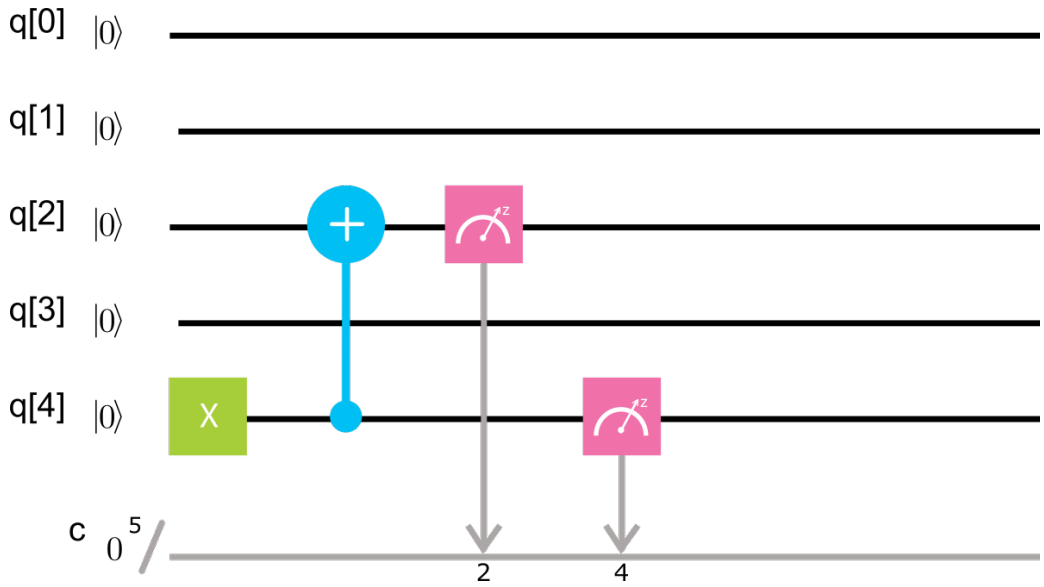


Figure 62: CNOT3

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];
x q[4];
cx q[4],q[2];
```

```
measure q[2] -> c[2];
measure q[4] -> c[4];
```

Solution:
An X-gate is applied on qubit q[4], followed by a CNOT gate with q[4] as control qubit and q[2] as target. The final state of the five qubit system is given by $|00101\rangle$. The only non-zero analytical probability is $p(q[2]q[4], |01\rangle) = 1$. This is, if the code is run in an perfect quantum computer, the system should always project onto $|00101\rangle$

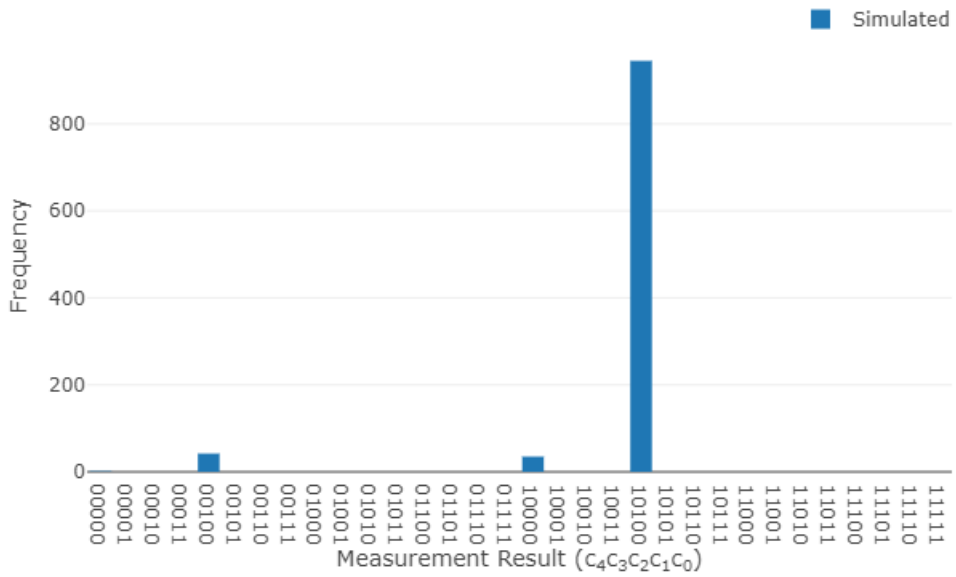


Figure 63: Plot

19. Bell State Creation, In the console, write the code that generates the following circuit

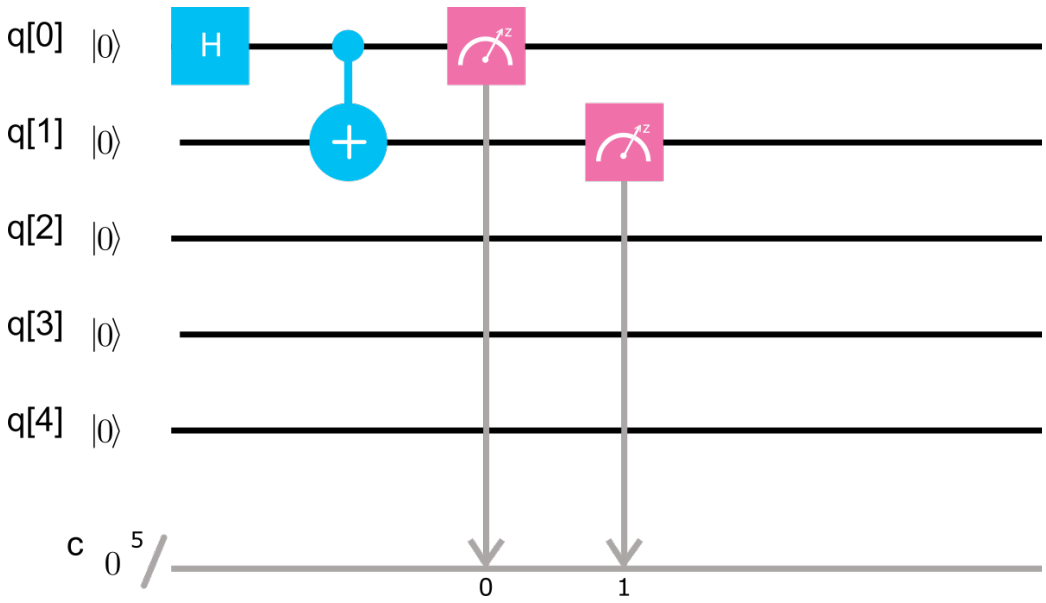


Figure 64: IBM QE4

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];
h q[0];
cx q[0],q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

Solution:
A Hadamard gate is applied on qubit q[0], followed by a CNOT gate with q[0] as control qubit and q[1] as target. The final state of the five qubit system is given by $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$. The non-zero analytical probability are $p(q[0]q[1], |00\rangle) = 0.5$, and $p(q[0]q[1], |11\rangle) = 0.5$.

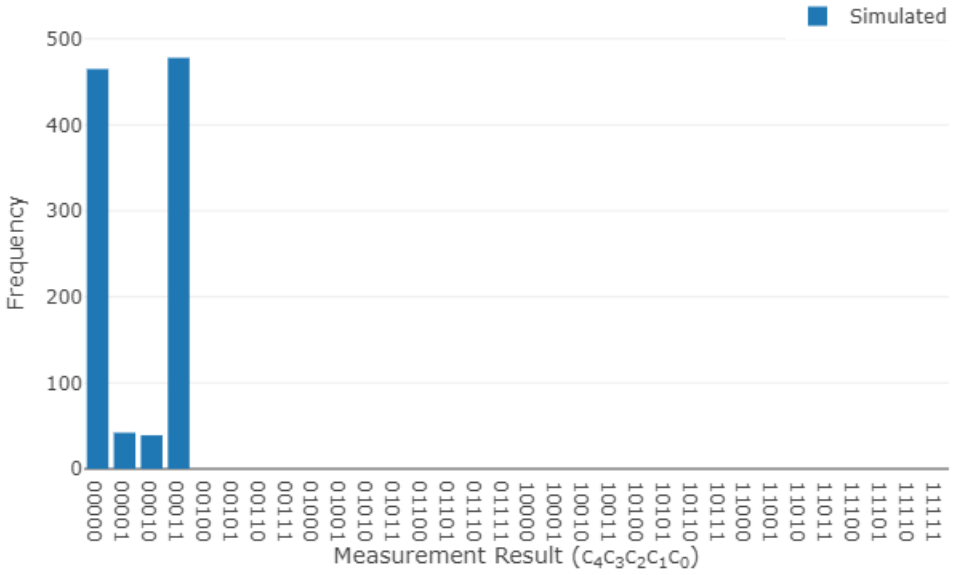


Figure 65: Plot

64.2 ibmqx4 Backend

The second backend that we will show is called ibmqx4. The figure below shows its topology.

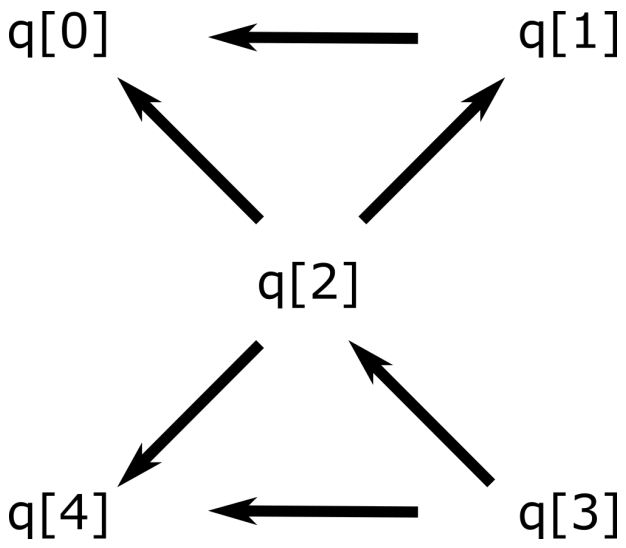


Figure 66: Topology 2

The following table shows all the allowed combinations for applying a CNOT-gate in QASM using the ibmqx4 backend.

Table 10: CNOT-gate in QASM

Control qubit	Target qubit	QASM line
q[1]	q[0]	cx q[1],q[0]
q[2]	q[1]	cx q[2],q[1]
q[2]	q[0]	cx q[2],q[0]
q[2]	q[4]	cx q[2],q[4]
q[3]	q[2]	cx q[3],q[2]
q[3]	q[4]	cx q[3],q[4]

20. Which of the following QASM programs will work properly with the ibmqx4 backend, meaning that the gates use allowed configurations of control and target qubits? (select all that are valid)

```
include ‘‘qelib1.inc’’;

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

x q[0];
h q[1];
cx q[0],q[1];

measure q[0] -> c[0];
measure q[1] -> c[1];
```

```
include ‘‘qelib1.inc’’;

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

x q[1];
h q[2];
cx q[2],q[1];

measure q[1] -> c[1];
measure q[2] -> c[2];
```

```
include ‘‘qelib1.inc’’;

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

x q[2];
h q[3];
```

```
cx q[2],q[3];

measure q[2] -> c[2];
measure q[3] -> c[3];
```

```
include ‘‘qelib1.inc’’;

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

x q[3];
h q[4];
cx q[3],q[4];

measure q[3] -> c[3];
measure q[4] -> c[4];
```

65 N=1 Data Qubit + 1 Ancilla Qubit

The following quantum circuit is an example of the Deutsch-Jozsa (DJ) algorithm implemented on a single (N=1) data qubit. In this case, the function being evaluated is $f(x) = x_0$, and it is balanced. In addition to the data qubit, the DJ algorithm generally needs a second qubit, an ancilla qubit, to facilitate quantum interference during the algorithm evolution.

In the input register, qubit q[0] serves as the data qubit, and q[1] the ancilla qubit. The Oracle circuit in this case is a CNOT gate, as seen in the composer image:

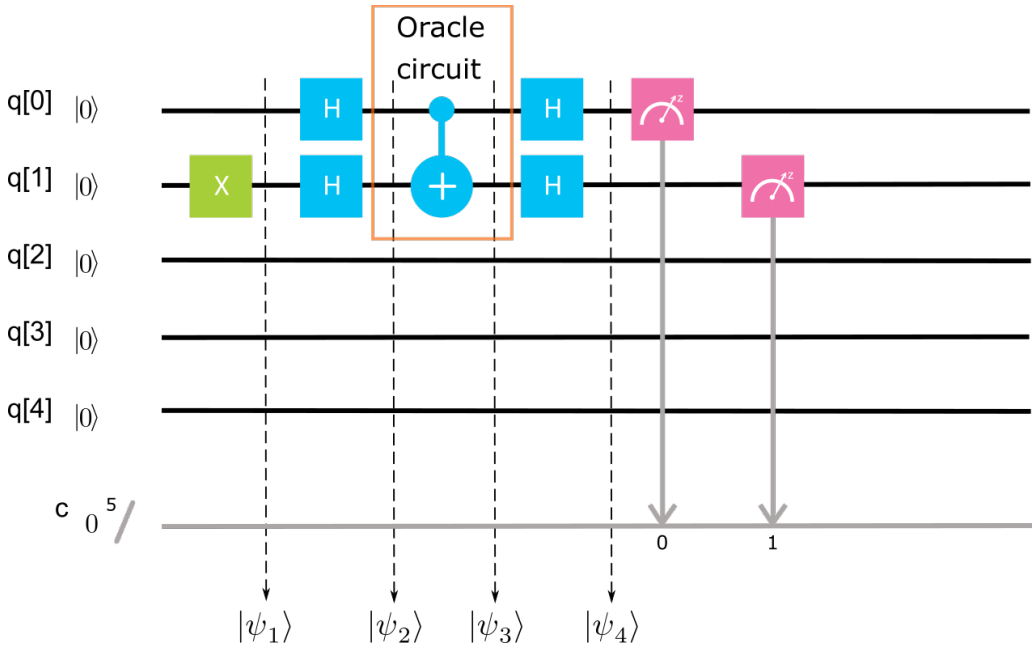


Figure 67: N2 balanced circuit

Let us now walk through the algorithm step-by-step. All qubits in the IBM Quantum Experience are initialized to state $|0\rangle$,

$$|\psi_0\rangle = |0\rangle\,|0\rangle\,. \tag{54}$$

Since the ancilla qubit needs to be prepared in state $|1\rangle$ as prescribed by the DJ algorithm, an X gate is applied to q[1],

$$\begin{aligned} |\psi_1\rangle &= |0\rangle\,X\,|0\rangle\,, \\ &= |0\rangle\,|1\rangle\,. \end{aligned} \tag{55}$$

Next, a hadamard gate is applied to each qubit to create an equal superposition state of the two qubits.

$$\begin{aligned} |\psi_2\rangle &= H\,|0\rangle\,H\,|1\rangle\,, \\ &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right), \\ &= \frac{1}{2}\left(|0\rangle\,(|0\rangle - |1\rangle) + |1\rangle\,(|0\rangle - |1\rangle)\right). \end{aligned} \tag{56}$$

The Oracle then performs a CNOT gate, with q[0] the control qubit and q[1] the target qubit.

$$\begin{aligned} |\psi_3\rangle &= \frac{1}{2}CNOT_{0,1}\left(|0\rangle\,(|0\rangle - |1\rangle) + |1\rangle\,(|0\rangle - |1\rangle)\right) \\ &= \frac{1}{2}\left(|0\rangle\,(|0\rangle - |1\rangle) + |1\rangle\,(|1\rangle - |0\rangle)\right) \\ &= \frac{1}{\sqrt{2}}\left(|0\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) - |1\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\right) \\ &= \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right). \end{aligned} \tag{57}$$

Two additional Hadamard gates are then applied to the output state of the oracle, facilitating quantum interference and leading to the output state $|f(0) \oplus f(1)\rangle$ for q[0] and state $|1\rangle$ for q[1].

$$\begin{aligned} |\psi_4\rangle &= H\frac{|0\rangle - |1\rangle}{\sqrt{2}}H\frac{|0\rangle - |1\rangle}{\sqrt{2}} \\ &= |1\rangle\,|1\rangle \end{aligned} \tag{58}$$

In a DJ algorithm, the function is constant if the values of the data qubits in the output register are all 0's. Here, they are not, as$f(0) \oplus f(1) = 1$, and so the function is confirmed to be balanced.

Although the DJ algorithm is generally prescribed as N data qubits plus one ancilla qubit, certain specific functions can design a "simplified" or "compiled" version of the oracle that reduces the overhead in implementing the algorithm. In the above graphic, the oracle shown also performs the DJ algorithm on N=2 data qubits for the function $f(x) = x_0 \oplus x_1$. It works, in part, because the function being balanced is determined by x_0 independent of x_1 and visa versa. In this "compiled" version of the code, both q[0] and q[1] are data qubits, and q[1] is effectively playing the role of both data qubit and ancilla qubit. At the output, both data qubits are measured and, since they are not both 0's, the function is confirmed balanced.

21. N=1 Data Qubit, Balanced Function, In the console below write the QASM code that generates the following circuit:

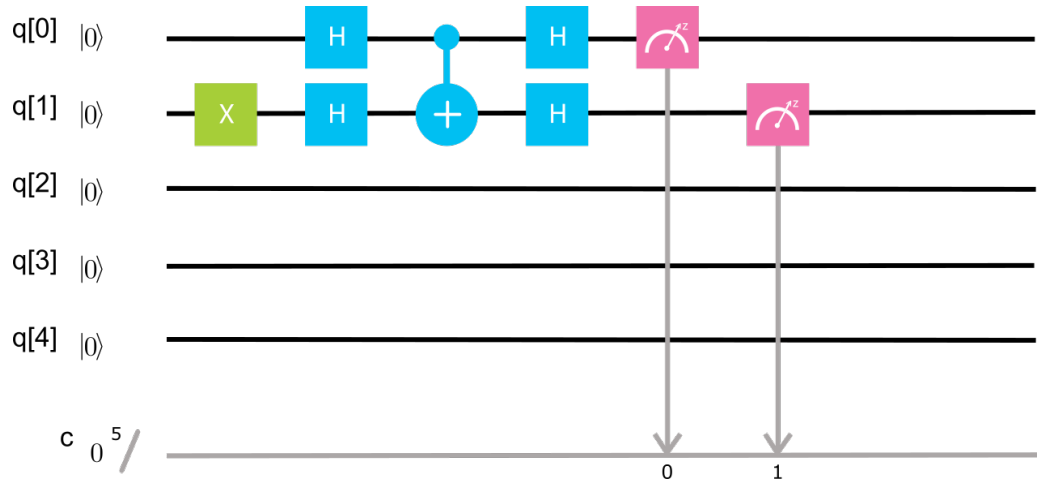


Figure 68: N2 balanced circuit NE

Submit our response, which will be evaluated with the grader, employing a numerically simulated quantum computer. Once we have a correct submission, our QASM code will automatically be queued to run on a real quantum computer at IBM.

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];

x q[1];
h q[0];
h q[1];
cx q[0],q[1];
h q[0];
h q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

Solution:
we just tested the DJ algorithm for N=2 qubits. This circuit implements in above section. Where the top qubit gives $f(0) \oplus f(1)$, and the bottom qubit remains unchanged (so it is 1). Thus, since the function is balanced, $f(0) \oplus f(1) = 1$, and we expect the top quit to be 1, and thus both qubits are measured to be 1. The result should thus be a histogram showing $|11\rangle$ with high probability, e.g.:

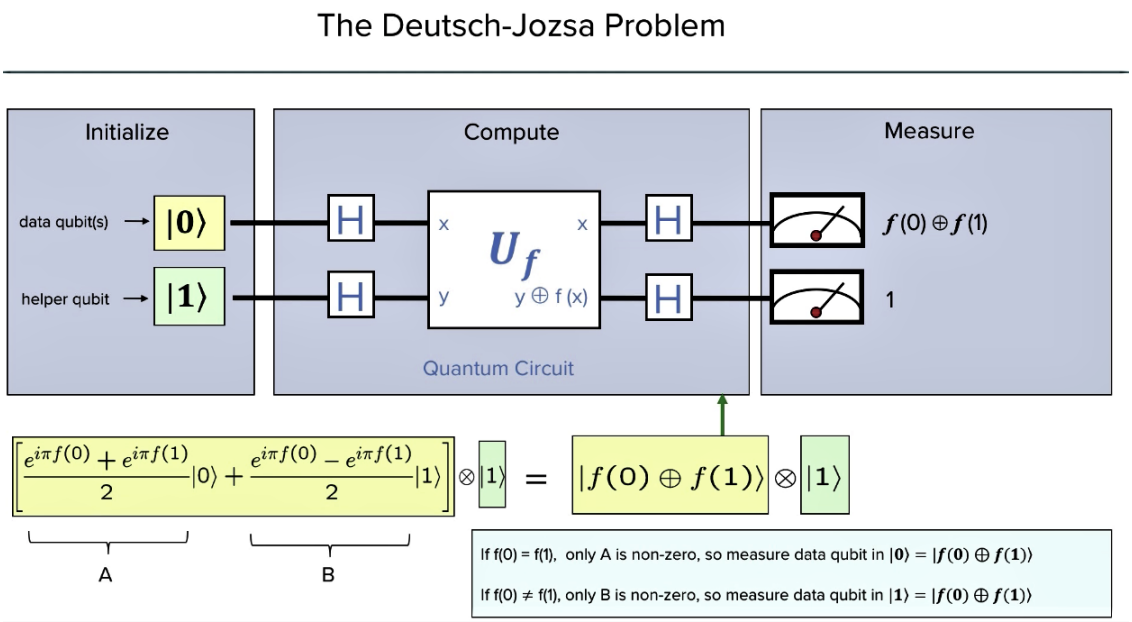


Figure 69: DJ circuit-output

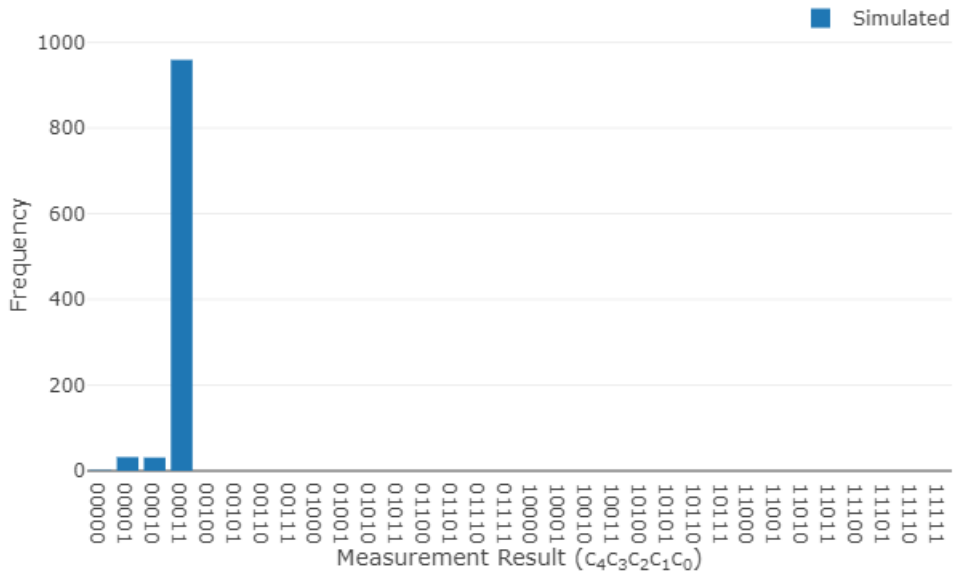


Figure 70: Plot

22. N=1 Data Qubit, Constant Function, In the console below write the QASM code that generates the following circuit:

Figure 71: N2 constant circuit

Submit our response, which will be evaluated with the grader, employing a numerically simulated quantum computer. Once we have a correct submission, our QASM code will automatically be queued to run on a real quantum computer at IBM.

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];

x q[1];
h q[0];
h q[1];
h q[0];
h q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

Solution:
we just tested the DJ algorithm for N=2 qubits for a constant function. Where the top qubit gives $f(0) \oplus f(1)$, and the bottom qubit remains unchanged (so it is 1). Thus, since the function is constant, $f(0) \oplus f(1) = 0$, and we expect the top quit to be 0, and thus the output state should be $|01\rangle$. The result should thus be a histogram showing $|11\rangle$ with high probability, e.g.:

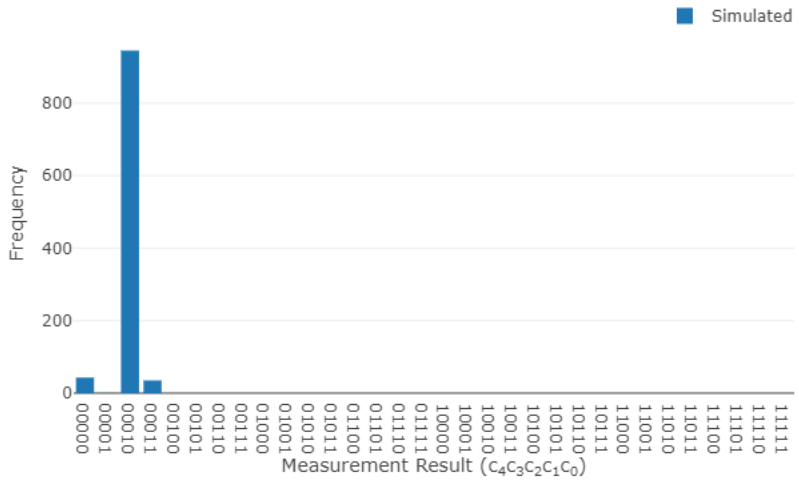


Figure 72: Plot

66 N=3 Data Qubits

The following quantum circuit is an N=3 data qubit example of the Deustch-Jozsa (DJ) algorithm for a balanced function $f(x) = x_0 \oplus x_1 x_2$. This function can be understood to be balanced, as toggling the first bit $f(x) = x_0$ will toggle the value of f(x) independent of the values of bits x_1 and x_2.

Generally, the DJ algorithm prescribes three data qubits plus one ancilla qubit for this function. However, the oracle circuit shown below is an example of a "compiled" version of the DJ, that is, an implementation that is simplified by design by taking advantage of a particular (but fairly generic) structure of the problem (note the compilation can be done without knowing anything about the answer). In this case, the input register needs only three qubits, all of which are data qubits to test this particular three-bit function f(x). The quantum interference facilitated by the ancilla qubit in the "general" algorithm is built into the oracle circuit in this design.

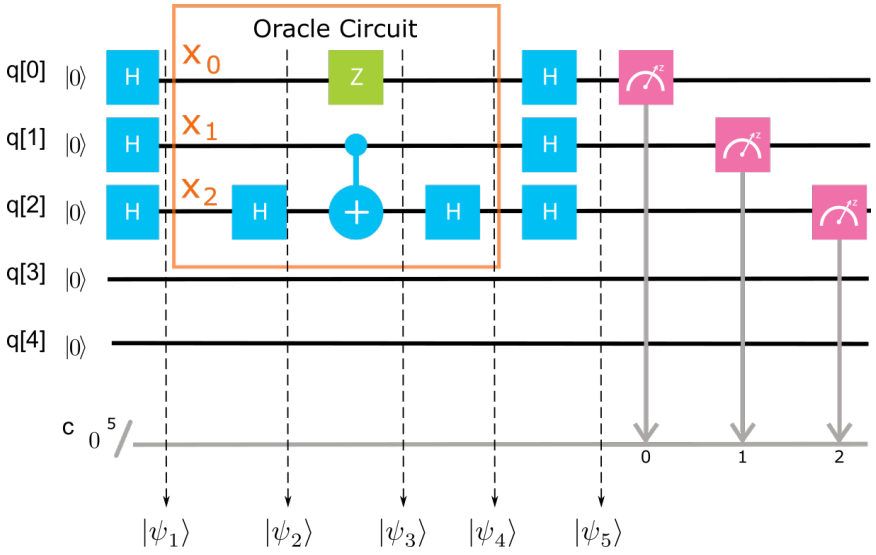


Figure 73: Circuit N3 Balanced

The three data qubits q[0], q[1], and q[2] are all initialized in state $|0\rangle$. Note that none of the qubits are initialized in state $|1\rangle$ as is generally prescribed for the ancilla qubit by the general version of the DJ algorithm.

$$|\psi_0\rangle = |0\rangle\,|0\rangle\,|0\rangle\,(1) \tag{59}$$

A three-qubit equal superposition state is then created by applying a Hadamard gate to each qubit.

$$\begin{aligned} |\psi_1\rangle &= H\,|0\rangle\,H\,|0\rangle\,H\,|0\rangle \\ &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \end{aligned} \tag{60}$$

The oracle circuit U_f consists a Z-gate on q[0], and a controlled-Z gate (CZ gate), realized by a CNOT gate with q[1] the control qubit and q[2] the target qubit between two Hadamards H applied to q[2]. Starting with the first Hadamard,

$$\begin{aligned} |\psi_2\rangle &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(H\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \\ &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)|0\rangle \end{aligned} \tag{61}$$

Next, the Z-gate is applied on q[0], and the CNOT gate to q[2] (target) and q[1] (control).

$$\begin{aligned} |\psi_3\rangle &= \left(Z\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) CNOT_{1,2}\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)|0\rangle \\ &= \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle\,|0\rangle + |1\rangle\,|1\rangle}{\sqrt{2}}\right) \end{aligned} \tag{62}$$

And, then the second Hadamard gate is applied to q[2].

$$\begin{aligned} |\psi_4\rangle &= \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle H|0\rangle + |1\rangle H|1\rangle}{\sqrt{2}}\right) \\ &= \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle|0\rangle + |0\rangle|1\rangle + |1\rangle|0\rangle - |1\rangle|1\rangle}{2}\right) \end{aligned} \tag{63}$$

After the oracle circuit, each qubit is again operated on by a Hadamard gate, leaving the final state of the 3 qubits as

$$\begin{aligned} |\psi_5\rangle &= H\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\left(\frac{H|0\rangle H|0\rangle + H|0\rangle H|1\rangle + H|1\rangle H|0\rangle - H|1\rangle H|1\rangle}{2}\right) \\ &= |1\rangle\left(\frac{H|0\rangle H|0\rangle + H|0\rangle H|1\rangle + H|1\rangle H|0\rangle - H|1\rangle H|1\rangle}{2}\right) \\ &= \frac{|1\rangle|0\rangle|0\rangle + |1\rangle|1\rangle|0\rangle + |1\rangle|0\rangle|1\rangle - |1\rangle|1\rangle|1\rangle}{2} \end{aligned} \tag{64}$$

In each Deutsch-Jozsa algorithm realization, if the data qubits at the output are all 0's, then the function is constant. Otherwise, it is balanced. In this case, there are no states $|0\rangle|0\rangle|0\rangle$ in the output superposition state with non-zero probability amplitude, and thus any measurement will have at least one of the qubits in-state $|1\rangle$. Consequently, the function is confirmed to be balanced in just one query.

23. Balanced DJ Algorithm, In the console below write the QASM code that generates the following circuit:

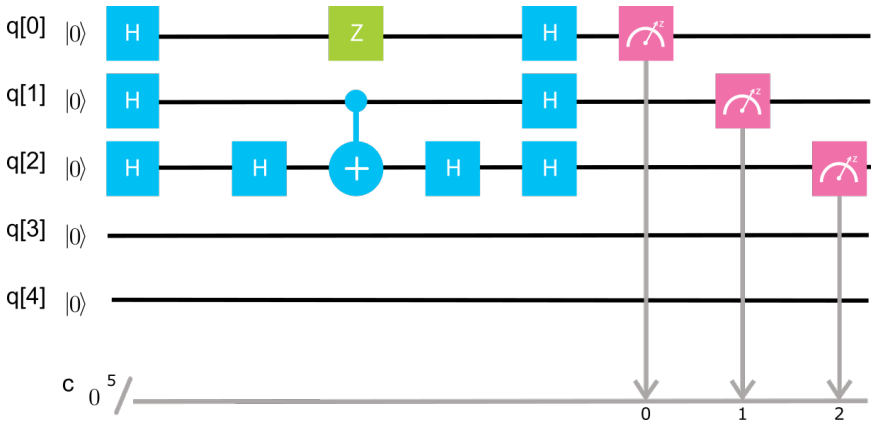


Figure 74: Circuit N3 Balanced NE

Submit our response, which will be evaluated with the grader, employing a numerically simulated quantum computer. Once we have a correct submission, our QASM code will automatically be queued to run on a real quantum computer at IBM.

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];
```

```
4
5  h q[0];
6  h q[1];
7  h q[2];
8  h q[2];
9  z q[0];
10 cx q[1],q[2];
11 h q[2];
12 h q[0];
13 h q[1];
14 h q[2];
15 measure q[0] -> c[0];
16 measure q[1] -> c[1];
17 measure q[2] -> c[2];
```

Solution: we just tested the DJ algorithm for N=3 qubits. The code was run 1024 times. The expected histogram is something like this: four states with nonzero probability all have the first qubit being equal to 1 this is the signature expected for the function being balanced.

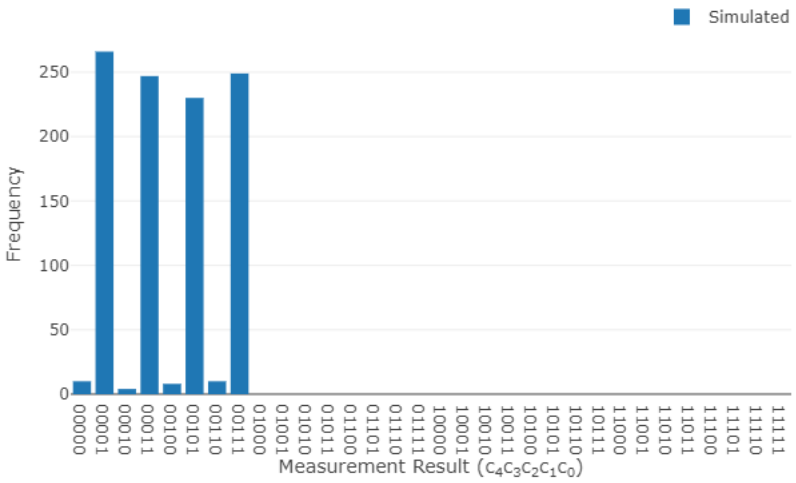


Figure 75: Plot

24. Constant DJ Algorithm, In the console below write the QASM code that generates the following circuit:

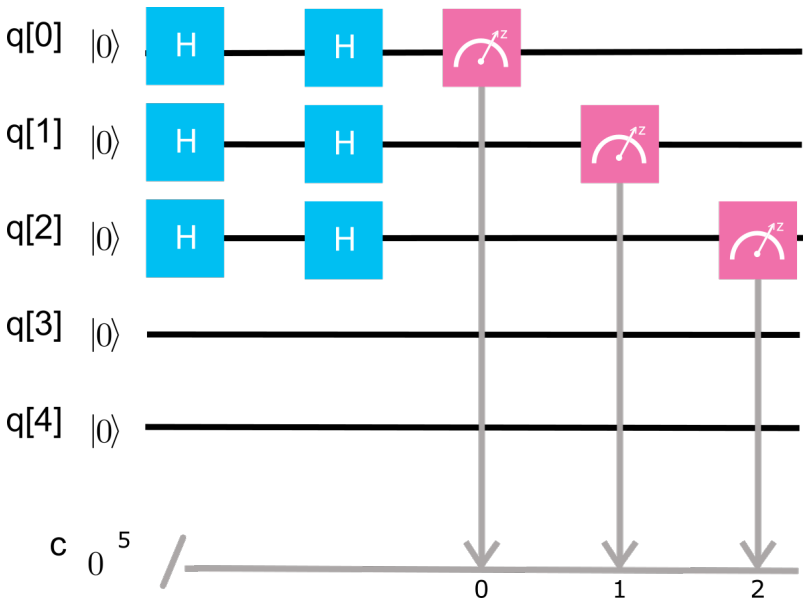


Figure 76: Circuit N3 Constant

Submit our response, which will be evaluated with the grader, employing a numerically simulated quantum computer. Once we have a correct submission, our QASM code will automatically be queued to run on a real quantum computer at IBM.

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];

h q[0];
h q[1];
h q[2];
h q[0];
h q[1];
h q[2];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

Solution: we just tested the DJ algorithm for N=3 qubits. The code was run 1024 times. The expected histogram of measurement results is something like this: fact that the output state is all zeros is expected, because all the Hadamard gates pair up and simply cancel each other.

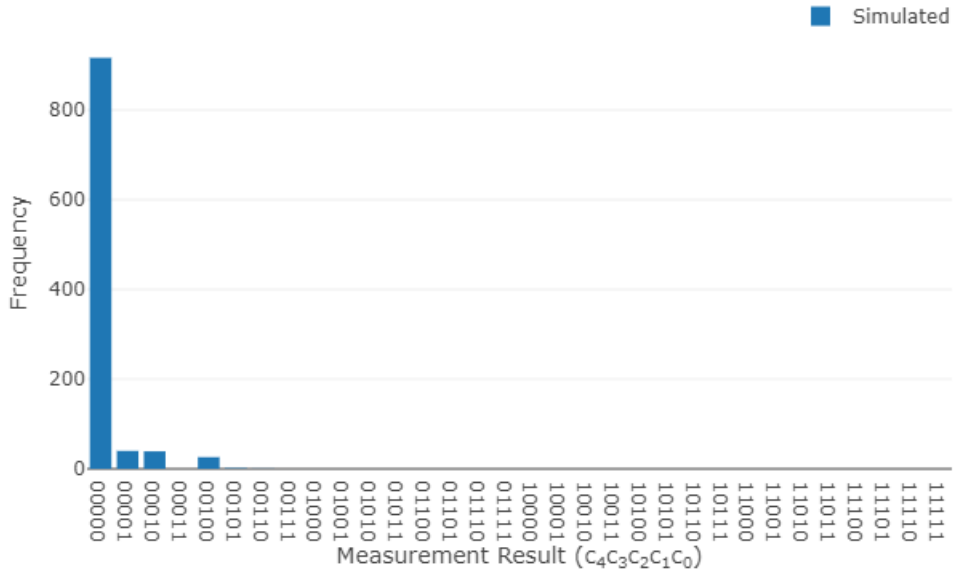


Figure 77: Plot

25. QASM Code I In the console below write the QASM code that generates the following circuit

Figure 78: GA1

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];

h q[3];
cx q[3],q[4];
x q[4];
measure q[3] -> c[3];
measure q[4] -> c[4];
```

Solution:
The circuit should act on qubits q[3] and q[4], and produce an output histogram like this one: The output is a superposition of $|00010\rangle and |00001\rangle$; by inspection, this is because the output after the Hadamard and CNOT is (up to normalization) $|00000\rangle + |00011\rangle$; the last X gate then flips the bottom qubit.

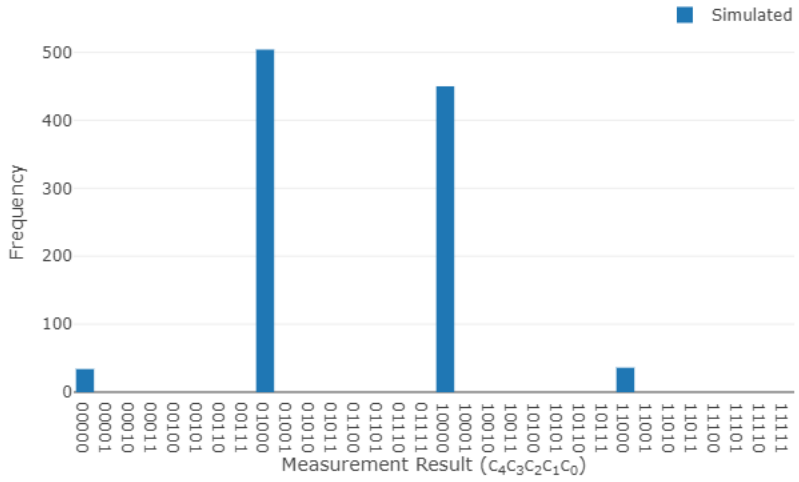


Figure 79: Plot

26. QASM Code II In the console below, write the QASM code that generates the following circuit.

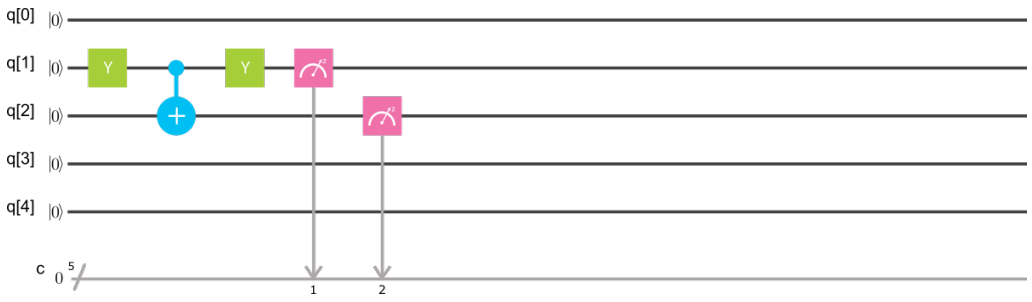


Figure 80: GA2

```
include ‘‘qelib1.inc’’;
qreg q[5];
creg c[5];

y q[1];
cx q[1],q[2];
```

```
y q[1];
measure q[1] -> c[1];
measure q[2] -> c[2];
```

Solution:
The circuit should act on qubits q[1] and q[2], and produce an output histogram like this one: The output is a single state, $|00100\rangle$; by inspection, this is because the first Y gate flips the control qubit from 0 to 1, causing the CNOT to flip its target qubit; the second Y gate flips the control qubit back to 0.

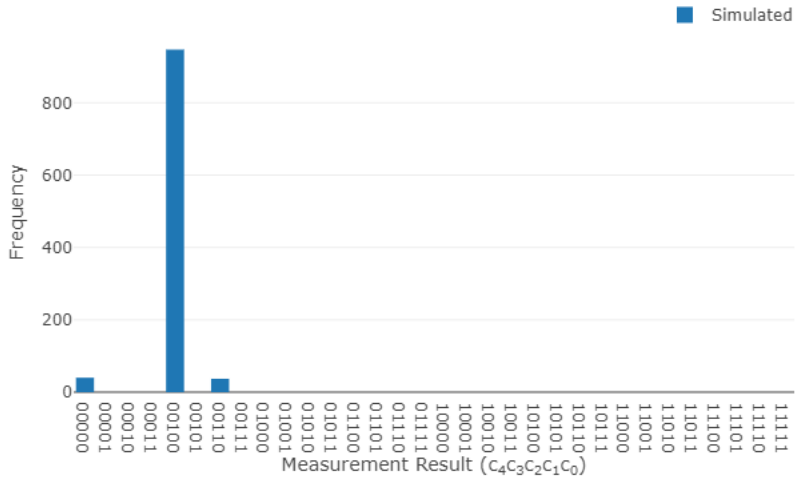


Figure 81: Plot

27. QASM Code III In the console below, write the QASM code that generates the following circuit.

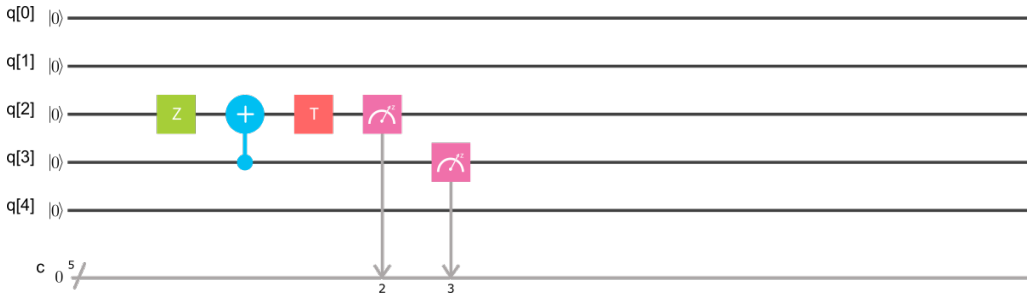


Figure 82: GA3

```
include ‘‘qelib1.inc’’;
qreg q[5];
```

```
creg c[5];

z q[2];
cx q[3],q[2];
t q[2];
measure q[2] -> c[2];
measure q[3] -> c[3];
```

Solution:
The circuit should act on qubits q[2] and q[3], and produce an output histogram like this one: The output is a single state, $|00000\rangle$; by inspection, this is because the CNOT has a control qubit in the 0 states, so the CNOT does nothing. The Z and T gates also do nothing on their input, the $|0\rangle$ state, so the output is all zeros.

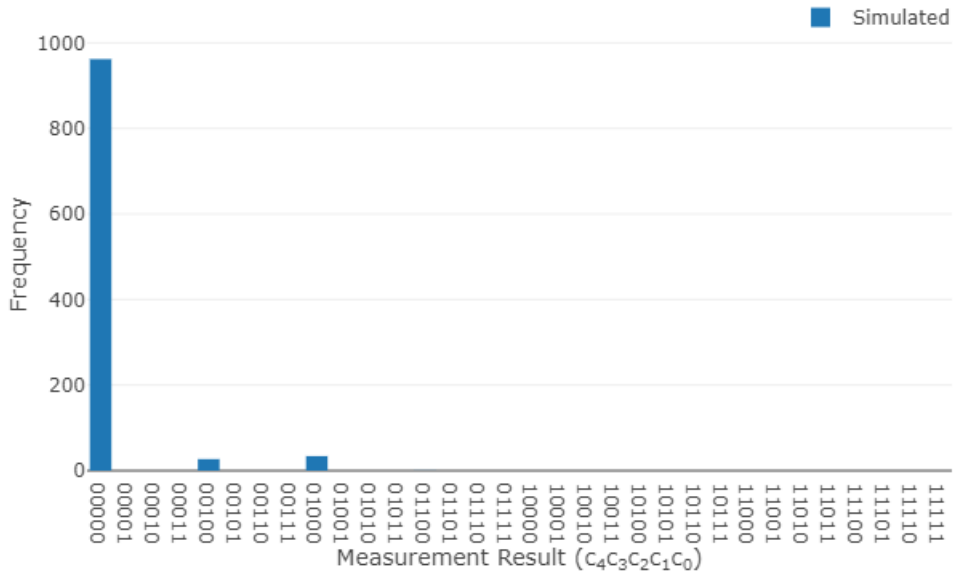


Figure 83: Plot

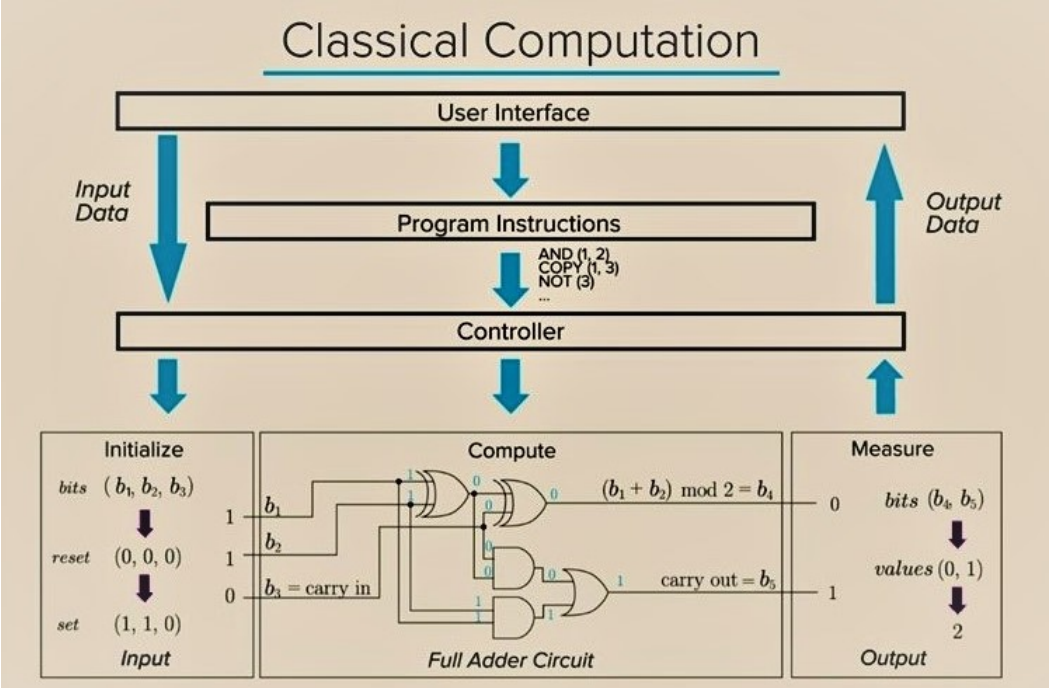


Figure 84: Classical Circuit Model

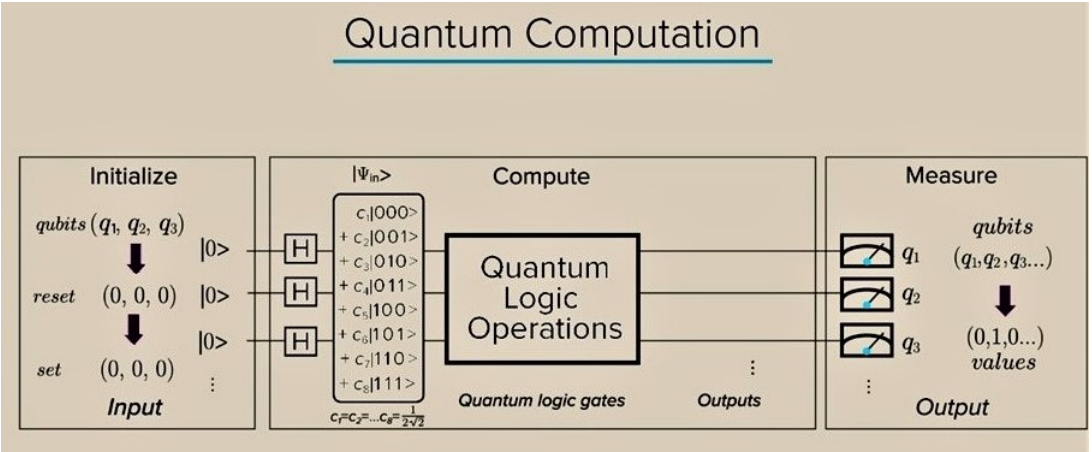


Figure 85: Quantum Circuit Model

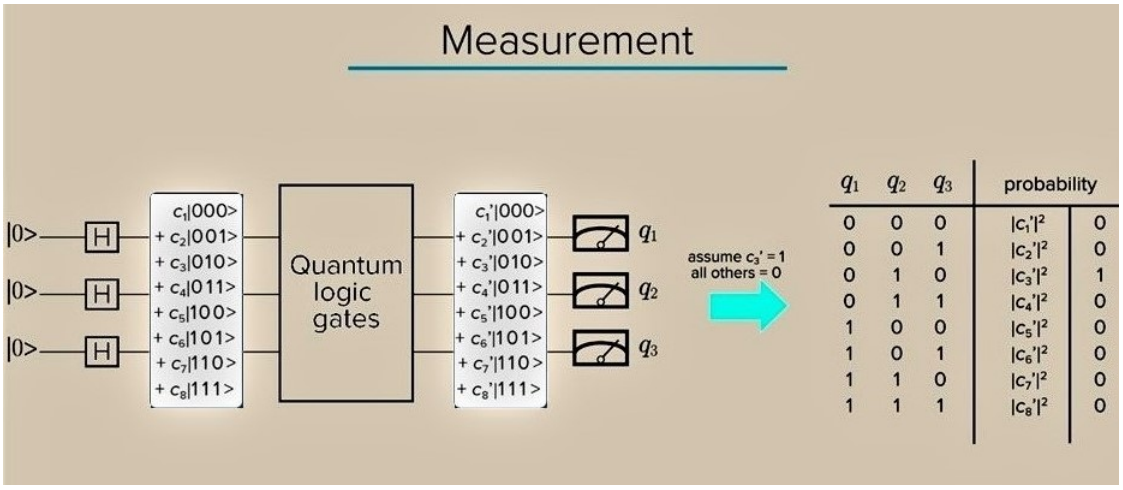


Figure 86: Quantum Circuit Model

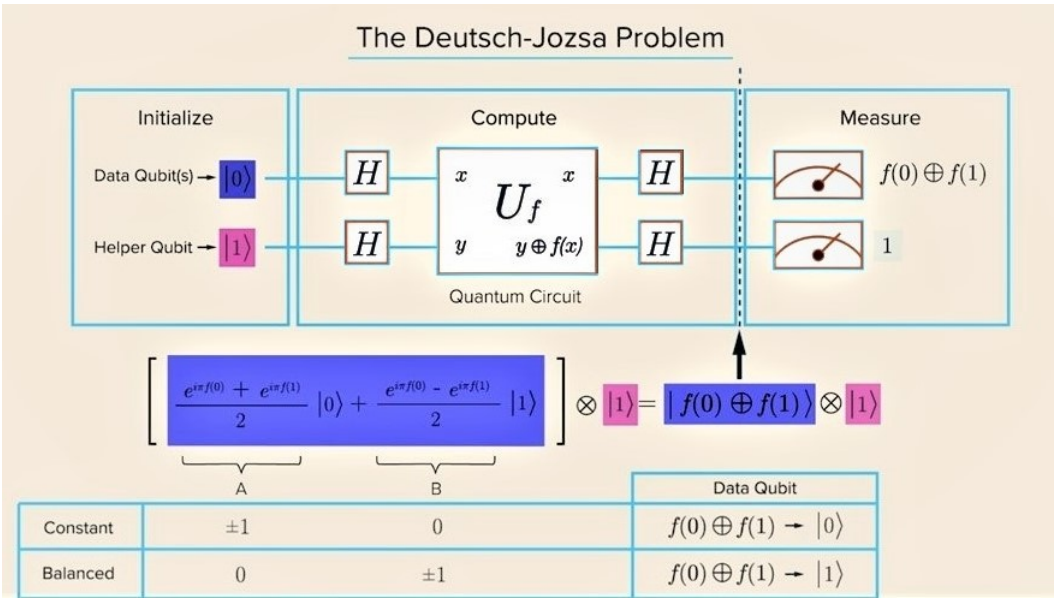


Figure 87: The Deutsch-Jozsa Algorithm

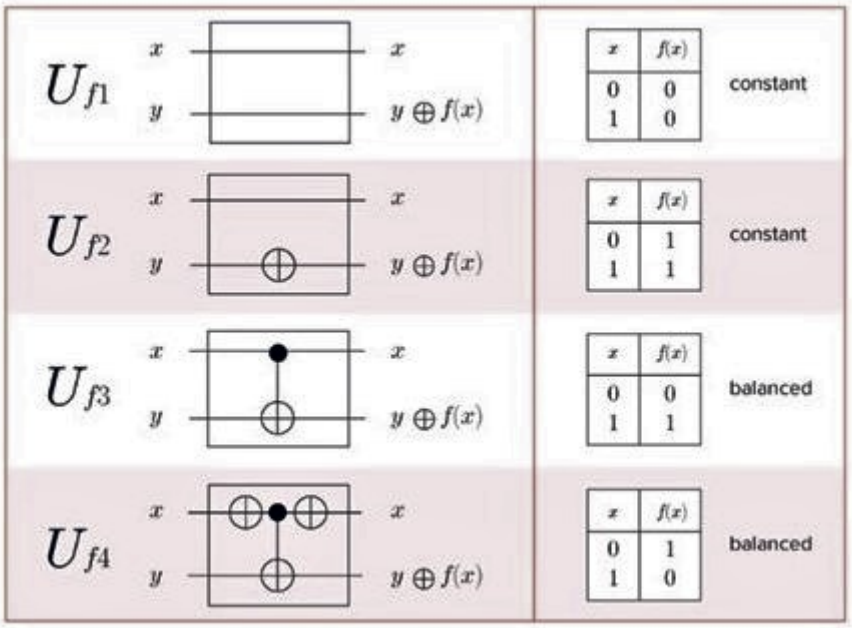


Figure 88: The Deutsch-Jozsa Algorithm

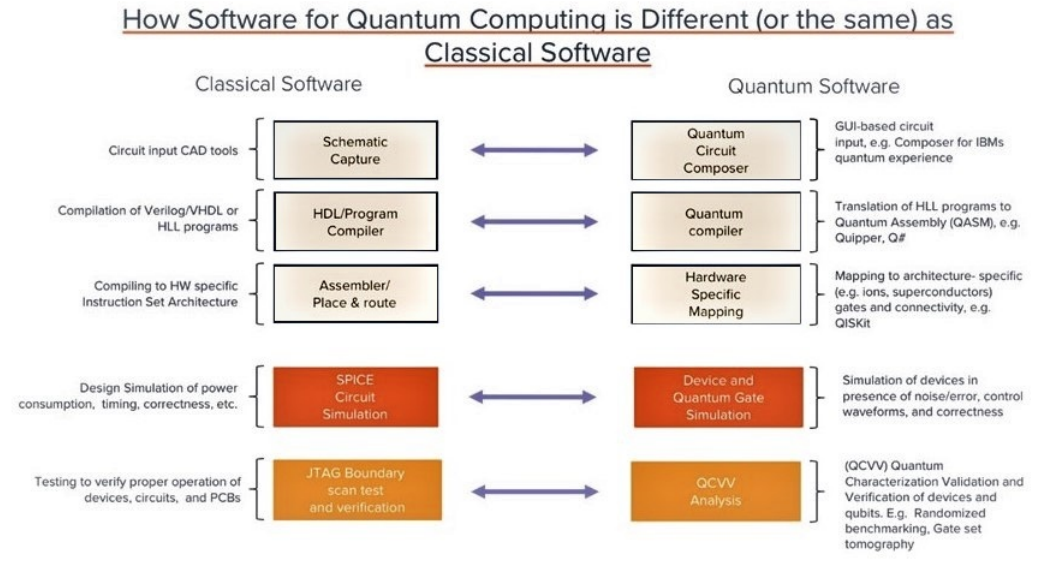


Figure 89: Quantum Software

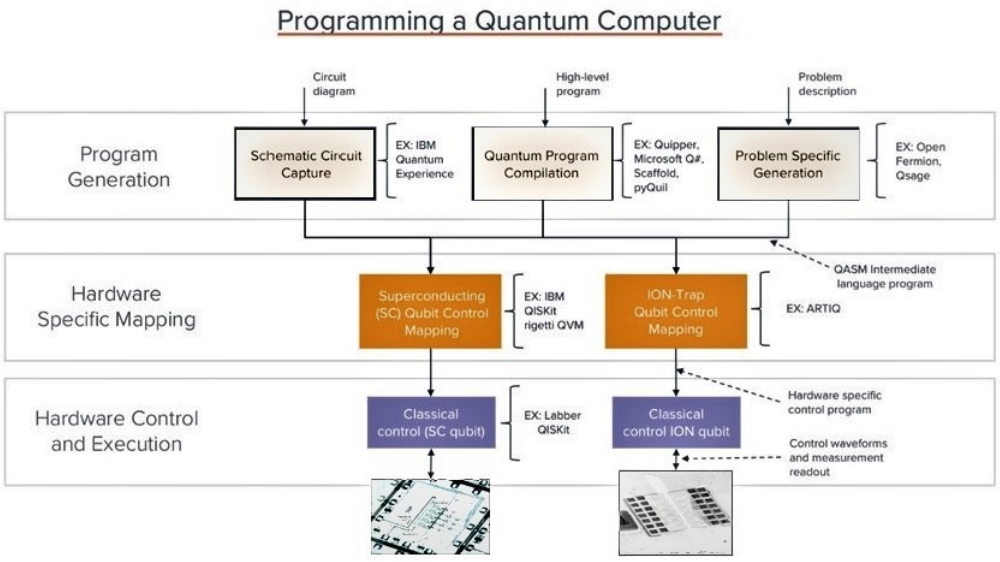


Figure 90: Quantum Software

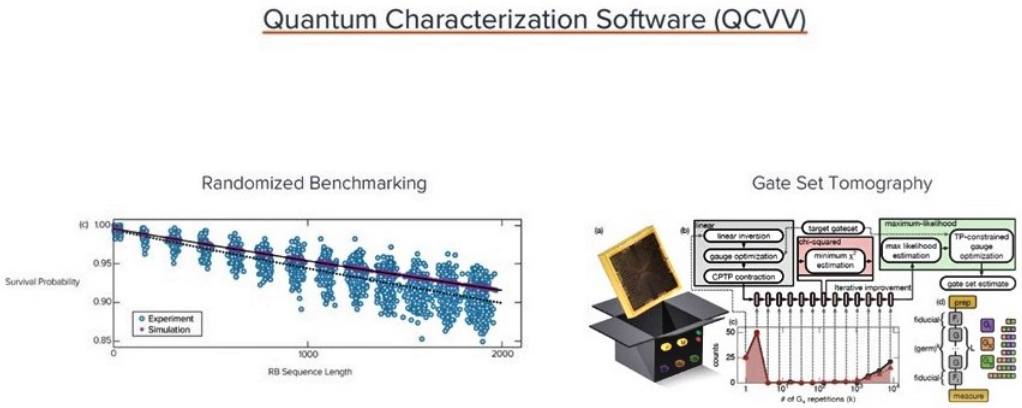


Figure 91: Programming Tools

References

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. USA: Cambridge University Press, 2011.

[2] I. Chuang, "Quantum Information Science II MIT EECS 6.443J / Physics 8.371J / 18.409," 2014. [Online]. Available: http://cua.scripts.mit.edu/8.371/S2014/index.php

[3] J. Preskill, "Quantum Computation Physics 219/Computer Science 219 (Formerly Physics 229)," 2019. [Online]. Available: http://www.theory.caltech.edu/~preskill/ph219/index.html

[4] U. Vazirani, "Quantum Computation CS294-2," 1997. [Online]. Available: https://people.eecs.berkeley.edu/~vazirani/qc.html

[5] J. Watrous, "Theory of Quantum Information John Watrous's Lecture Notes," 2011. [Online]. Available: https://cs.uwaterloo.ca/~watrous/LectureNotes.html

[6] U. Vazirani, "Quantum Computation CS294-2," 2007. [Online]. Available: https://people.eecs.berkeley.edu/~vazirani/quantum.html

[7] S. Aaronson, "Quantum Computing Since Democritus PHYS771," 2006. [Online]. Available: https://www.scottaaronson.com/democritus/

[8] P. W. Shor, "Quantum Computation," 2003, library Catalog: ocw.mit.edu. [Online]. Available: https://ocw.mit.edu/courses/mathematics/18-435j-quantum-computation-fall-2003/

[9] I. Chuang and P. Shor, "Quantum Information Science," 2006, library Catalog: ocw.mit.edu. [Online]. Available: https://ocw.mit.edu/courses/media-arts-and-sciences/mas-865j-quantum-information-science-spring-2006/

[10] S. Aaronson, "Quantum Complexity Theory," 2010, library Catalog: ocw.mit.edu. [Online]. Available: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-845-quantum-complexity-theory-fall-2010/

[11] A. W. Harrow, "Quantum Information Science 6.443/8.371/18.436," 2018. [Online]. Available: https://web.mit.edu/8.371/www/index.html

[12] A. Childs, "Quantum algorithms CO 781," 2008. [Online]. Available: http://www.math.uwaterloo.ca/~amchilds/teaching/w08/co781.html

[13] R. Cleve, "Introduction to Quantum Information Processing CS 667, C&O 681, PHYS 767," 2007. [Online]. Available: https://cs.uwaterloo.ca/~cleve/courses/F07CS667/index.html

[14] M. Russo, A. Thaker, and S. Adam, *The Coming Quantum Leap in Computing*, May 2018. [Online]. Available: https://www.bcg.com/en-in/publications/2018/coming-quantum-leap-computing.aspx

[15] *The Next Decade in Quantum Computing—and How to Play.* [Online]. Available: https://www.bcg.com/en-in/publications/2018/next-decade-quantum-computing-how-play.aspx

[16] "Here, there and everywhere," *The Economist.* [Online]. Available: https://www.economist.com/news/essays/21717782-quantum-technology-beginning-come-its-own

[17] A. W. Harrow, "Why now is the right time to study quantum computing," *XRDS: Crossroads, The ACM Magazine for Students*, vol. 18, no. 3, pp. 32–37, Mar. 2012. [Online]. Available: http://arxiv.org/abs/1501.00011

[18] M. Sisodia, A. Shukla, A. A. A. de Almeida, G. W. Dueck, and A. Pathak, "Circuit optimization for IBM processors: A way to get higher fidelity and higher values of nonclassicality witnesses," *arXiv:1812.11602 [quant-ph]*, Dec. 2018. [Online]. Available: http://arxiv.org/abs/1812.11602

[19] A. Cervera-Lierta, "Exact Ising model simulation on a quantum computer," *Quantum*, vol. 2, p. 114, Dec. 2018. [Online]. Available: http://arxiv.org/abs/1807.07112

[20] J. R. McClean, K. J. Sung, I. D. Kivlichan, Y. Cao, C. Dai, E. S. Fried, C. Gidney, B. Gimby, P. Gokhale, T. Häner, T. Hardikar, V. Havlíček, O. Higgott, C. Huang, J. Izaac, Z. Jiang, X. Liu, S. McArdle, M. Neeley, T. O'Brien, B. O'Gorman, I. Ozfidan, M. D. Radin, J. Romero, N. Rubin, N. P. D. Sawaya, K. Setia, S. Sim, D. S. Steiger, M. Steudtner, Q. Sun, W. Sun, D. Wang, F. Zhang, and R. Babbush, "OpenFermion: The Electronic Structure Package for Quantum Computers," *arXiv:1710.07629 [physics, physics:quant-ph]*, Feb. 2019. [Online]. Available: http://arxiv.org/abs/1710.07629

[21] E. Farhi, J. Goldstone, and S. Gutmann, "A Quantum Approximate Optimization Algorithm," *arXiv:1411.4028 [quant-ph]*, Nov. 2014. [Online]. Available: http://arxiv.org/abs/1411.4028

[22] S. E. Smart, D. I. Schuster, and D. A. Mazziotti, "Experimental data from a quantum computer verifies the generalized Pauli exclusion principle," *Communications Physics*, vol. 2, no. 1, pp. 1–6, Jan. 2019. [Online]. Available: https://www.nature.com/articles/s42005-019-0110-3

[23] N. N. Hegade, B. K. Behera, and P. K. Panigrahi, "Experimental Demonstration of Quantum Tunneling in IBM Quantum Computer," *arXiv:1712.07326 [quant-ph]*, Feb. 2019. [Online]. Available: http://arxiv.org/abs/1712.07326

[24] M. J. Bremner, R. Jozsa, and D. J. Shepherd, "Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 467, no. 2126, pp. 459–472, Feb. 2011. [Online]. Available: http://arxiv.org/abs/1005.1407

[25] T. Shang, K. Li, R. Chen, and J. Liu, "Continuous-Variable Quantum Network Coding Against Pollution Attacks," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 196–206.

[26] S.-K. Liao, W.-Q. Cai, J. Handsteiner, B. Liu, J. Yin, L. Zhang, D. Rauch, M. Fink, J.-G. Ren, W.-Y. Liu, Y. Li, Q. Shen, Y. Cao, F.-Z. Li, J.-F. Wang, Y.-M. Huang, L. Deng, T. Xi, L. Ma, T. Hu, L. Li, N.-L. Liu, F. Koidl, P. Wang, Y.-A. Chen, X.-B. Wang, M. Steindorfer, G. Kirchner, C.-Y. Lu, R. Shu, R. Ursin, T. Scheidl, C.-Z. Peng, J.-Y. Wang, A. Zeilinger, and J.-W. Pan, "Satellite-Relayed Intercontinental Quantum Network," *Physical Review Letters*, vol. 120, no. 3, p. 030501, Jan. 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.120.030501

[27] C. L. Edmunds, C. Hempel, R. Harris, H. Ball, V. Frey, T. M. Stace, and M. J. Biercuk, "Measuring and Suppressing Error Correlations in Quantum Circuits," *arXiv:1712.04954 [quant-ph]*, Dec. 2017. [Online]. Available: http://arxiv.org/abs/1712.04954

[28] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018. [Online]. Available: http://arxiv.org/abs/1801.00862

[29] M. Delehaye and C. Lacroûte, "Single-ion, transportable optical atomic clocks," *Journal of Modern Optics*, vol. 65, no. 5-6, pp. 622–639, Mar. 2018. [Online]. Available: http://arxiv.org/abs/1804.02168

[30] N. Huntemann, C. Sanner, B. Lipphardt, C. Tamm, and E. Peik, "Single-ion atomic clock with 3×10^-18 systematic uncertainty," *Physical Review Letters*, vol. 116, no. 6, p. 063001, Feb. 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.116.063001

[31] R. LaRose, "Overview and Comparison of Gate Level Quantum Software Platforms," *Quantum*, vol. 3, p. 130, Mar. 2019. [Online]. Available: http://arxiv.org/abs/1807.02500

[32] N. Killoran, J. Izaac, N. Quesada, V. Bergholm, M. Amy, and C. Weedbrook, "Strawberry Fields: A Software Platform for Photonic Quantum Computing," *Quantum*, vol. 3, p. 129, Mar. 2019. [Online]. Available: http://arxiv.org/abs/1804.03159

[33] A. D. King, J. Carrasquilla, J. Raymond, I. Ozfidan, E. Andriyash, A. Berkley, M. Reis, T. Lanting, R. Harris, F. Altomare, K. Boothby, P. I. Bunyk, C. Enderud, A. Fréchette, E. Hoskinson, N. Ladizinsky, T. Oh, G. Poulin-Lamarre, C. Rich, Y. Sato, A. Y. Smirnov, L. J. Swenson, M. H. Volkmann, J. Whittaker, J. Yao, E. Ladizinsky, M. W. Johnson, J. Hilton, and M. H. Amin, "Observation of topological phenomena in a programmable lattice of 1,800 qubits," *Nature*, vol. 560, no. 7719, pp. 456–460, Aug. 2018. [Online]. Available: https://www.nature.com/articles/s41586-018-0410-x

[34] T. Gabor, S. Zielinski, S. Feld, C. Roch, C. Seidel, F. Neukart, I. Galter, W. Mauerer, and C. Linnhoff-Popien, "Assessing Solution Quality of 3SAT on a Quantum Annealing Platform," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 23–35.

[35] C. F. Abenojar, Eric, *IBM \textbar Quantum Computing*. [Online]. Available: https://www.ibm.com/quantum-computing/

[36] *IBM Quantum Experience - Dashboard*, publication Title: IBM Quantum Experience. [Online]. Available: https://quantum-computing.ibm.com/

[37] *Qiskit*, publication Title: IBM Developer. [Online]. Available: https://developer.ibm.com/technologies/quantum-computing/projects/qiskit/

[38] *Qiskit/openqasm*. Qiskit, May 2020. [Online]. Available: https://github.com/Qiskit/openqasm

[39] *quantumlib*, publication Title: GitHub. [Online]. Available: https://github.com/quantumlib

[40] *quantumlib/OpenFermion*. quantumlib, May 2020. [Online]. Available: https://github.com/quantumlib/OpenFermion

[41] K. M. Svore, A. Geller, M. Troyer, J. Azariah, C. Granade, B. Heim, V. Kliuchnikov, M. Mykhailova, A. Paz, and M. Roetteler, "Q#: Enabling scalable quantum computing and development with a high-level domain-specific language," *Proceedings of the Real World Domain Specific Languages Workshop 2018 on - RWDSL2018*, pp. 1–10, 2018. [Online]. Available: http://arxiv.org/abs/1803.00652

[42] M. Soeken, T. Häner, and M. Roetteler, "Programming Quantum Computers Using Design Automation," *arXiv:1803.01022 [quant-ph]*, Mar. 2018. [Online]. Available: http://arxiv.org/abs/1803.01022

[43] *Quantum computing \textbar Microsoft*. [Online]. Available: https://www.microsoft.com/en-us/quantum

[44] *IonQ*, publication Title: IonQ. [Online]. Available: https://ionq.com/

[45] *Home*, publication Title: Rigetti Computing. [Online]. Available: https://rigetti.com/

[46] *Rigetti QCS*. [Online]. Available: https://qcs.rigetti.com/request-access

[47] Q. Circuits and I. . S. P. S. . N. Haven, *Quantum Circuits, Inc.*, publication Title: Quantum Circuits, Inc. [Online]. Available: https://www.quantumcircuits.com/

[48] *D-Wave Systems*. [Online]. Available: https://www.dwavesys.com/home

[49] K. Shiba, K. Sakamoto, K. Yamaguchi, D. B. Malla, and T. Sogabe, "Convolution filter embedded quantum gate autoencoder," *arXiv:1906.01196 [quant-ph]*, Jun. 2019. [Online]. Available: http://arxiv.org/abs/1906.01196

[50] M. Schuld, "Machine learning in quantum spaces," *Nature*, vol. 567, no. 7747, pp. 179–181, Mar. 2019. [Online]. Available: https://www.nature.com/articles/d41586-019-00771-0

[51] L. Bassman, K. Liu, A. Krishnamoorthy, T. Linker, Y. Geng, D. Shebib, S. Fukushima, F. Shimojo, R. K. Kalia, A. Nakano, and P. Vashishta, "Towards simulation of the dynamics of materials on quantum computers," *Physical Review B*, vol. 101, no. 18, p. 184305, May 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.101.184305

[52] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, Mar. 2019. [Online]. Available: https://www.nature.com/articles/s41586-019-0980-2

[53] J. Stuart, R. Panock, C. Bruzewicz, J. Sedlacek, R. McConnell, I. Chuang, J. Sage, and J. Chiaverini, "Chip-Integrated Voltage Sources for Control of Trapped Ions," *Physical Review Applied*, vol. 11, no. 2, p. 024010, Feb. 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.11.024010

[54] N. M. Linke, D. Maslov, M. Roetteler, S. Debnath, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, "Experimental Comparison of Two Quantum Computing Architectures," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3305–3310, Mar. 2017. [Online]. Available: http://arxiv.org/abs/1702.01852

[55] G. Verdon, M. Broughton, and J. Biamonte, "A quantum algorithm to train neural networks using low-depth circuits," *arXiv:1712.05304 [cond-mat, physics:quant-ph]*, Aug. 2019. [Online]. Available: http://arxiv.org/abs/1712.05304

[56] M. Schuld, A. Bocharov, K. Svore, and N. Wiebe, "Circuit-centric quantum classifiers," *Physical Review A*, vol. 101, no. 3, p. 032308, Mar. 2020. [Online]. Available: http://arxiv.org/abs/1804.00633

[57] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, no. 6, pp. 467–488, Jun. 1982. [Online]. Available: https://doi.org/10.1007/BF02650179

[58] A. Hey, *Feynman And Computation*. CRC Press, Mar. 2018.

[59] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Nov. 1994, pp. 124–134.

[60] A. M. Steane, "Error Correcting Codes in Quantum Theory," *Physical Review Letters*, vol. 77, no. 5, pp. 793–797, Jul. 1996, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.77.793

[61] E. F. Dumitrescu, A. J. McCaskey, G. Hagen, G. R. Jansen, T. D. Morris, T. Papenbrock, R. C. Pooser, D. J. Dean, and P. Lougovski, "Cloud Quantum Computing of an Atomic Nucleus," *Physical Review Letters*, vol. 120, no. 21, p. 210501, May 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.120.210501

[62] P. A. M. Dirac and R. H. Fowler, "The fundamental equations of quantum mechanics," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 109, no. 752, pp. 642–653, Dec. 1925. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1925.0150

[63] P. a. M. Dirac, "On Quantum Algebra," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 23, no. 4, pp. 412–418, Oct. 1926. [Online]. Available: https://www.cambridge.org/core/journals/mathematical-proceedings-of-the-cambridge-philosophical-society/article/on-quantum-algebra/EDEC0A7C8B907C6720C0AAF51BBF68A5

[64] P. A. M. Dirac, "THE MATHEMATICAL FOUNDATIONS OF QUANTUM THEORY," in *Mathematical Foundations of Quantum Theory*, A. R. Marlow, Ed. Academic Press, Jan. 1978, pp. 1–8. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780124732506500054

[65] P. a. M. Dirac, "The basis of statistical quantum mechanics," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 25, no. 1, pp. 62–66, Jan. 1929. [Online]. Available: https://www.cambridge.org/core/journals/mathematical-proceedings-of-the-cambridge-philosophical-society/article/basis-of-statistical-quantum-mechanics/91CE79926CED070AE10A45EF224CC2BB

[66] P. A. M. Dirac and R. H. Fowler, "Quantum mechanics of many-electron systems," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 123, no. 792, pp. 714–733, Apr. 1929. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1929.0094

[67] ——, "On the theory of quantum mechanics," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 112, no. 762, pp. 661–677, Oct. 1926. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1926.0133

[68] K. Setia, S. Bravyi, A. Mezzacapo, and J. D. Whitfield, "Superfast encodings for fermionic quantum simulation," *Physical Review Research*, vol. 1, no. 3, p. 033033, Oct. 2019. [Online]. Available: http://arxiv.org/abs/1810.05274

[69] R. T. Thew, K. Nemoto, A. G. White, and W. J. Munro, "Qudit quantum-state tomography," *Physical Review A*, vol. 66, no. 1, p. 012303, Jul. 2002. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.66.012303

[70] J. Rué and S. Xambó, *MATHEMATICAL ESSENTIALS OF QUANTUM COMPUTING*. [Online]. Available: https://mat-web.upc.edu/people/sebastia.xambo/QC/qc.pdf

[71] M. Laforest, *the mathematics of quantum mechanics.* [Online]. Available: https://uwaterloo.ca/institute-for-quantum-computing/sites/ca.institute-for-quantum-computing/files/uploads/files/mathematics_qm_v21.pdf

[72] N. D. Mermin, *Quantum Computer Science: An Introduction*, Aug. 2007. [Online]. Available: /core/books/quantum-computer-science/66462590D10C8010017CF1D7C45708D7

[73] E. National Academies of Sciences, *Quantum Computing: Progress and Prospects*, Dec. 2018. [Online]. Available: https://www.nap.edu/catalog/25196/quantum-computing-progress-and-prospects

[74] P. R. Giri and V. E. Korepin, "A Review on Quantum Search Algorithms," *Quantum Information Processing*, vol. 16, no. 12, p. 315, Dec. 2017. [Online]. Available: http://arxiv.org/abs/1602.02730

[75] A. Y. Kitaev, A. Shen, M. N. Vyalyi, and M. N. Vyalyi, *Classical and Quantum Computation.* American Mathematical Society, 2002.

[76] F. de Ridder, N. Neumann, T. Veugen, and R. Kooij, "A Quantum Algorithm for Minimising the Effective Graph Resistance upon Edge Addition," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 63–73.

[77] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, T. Magerlein, E. Solomonik, E. W. Draeger, E. T. Holland, and R. Wisnieff, "Breaking the 49-Qubit Barrier in the Simulation of Quantum Circuits," *arXiv:1710.05867 [quant-ph]*, Dec. 2018. [Online]. Available: http://arxiv.org/abs/1710.05867

[78] R. Orus, "A Practical Introduction to Tensor Networks: Matrix Product States and Projected Entangled Pair States," *Annals of Physics*, vol. 349, pp. 117–158, Oct. 2014. [Online]. Available: http://arxiv.org/abs/1306.2164

[79] P. W. Shor, "Capacities of quantum channels and how to find them," *Mathematical Programming*, vol. 97, no. 1, pp. 311–335, Jul. 2003. [Online]. Available: http://link.springer.com/10.1007/s10107-003-0446-y

[80] A. Paler, "On the Influence of Initial Qubit Placement During NISQ Circuit Compilation," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 207–217.

[81] I. D. Kivlichan, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, W. Sun, Z. Jiang, N. Rubin, A. Fowler, A. Aspuru-Guzik, H. Neven, and R. Babbush, "Improved Fault-Tolerant Quantum Simulation of Condensed-Phase Correlated Electrons via Trotterization," *arXiv:1902.10673 [physics, physics:quant-ph]*, Aug. 2019. [Online]. Available: http://arxiv.org/abs/1902.10673

[82] J. Jonsson, K. Moriarty, B. Kaliski, and A. Rusch, *PKCS #1: RSA Cryptography Specifications Version 2.2.* [Online]. Available: https://tools.ietf.org/html/rfc8017

[83] L. K. Grover, "A fast quantum mechanical algorithm for database search," *arXiv:quant-ph/9605043*, Nov. 1996. [Online]. Available: http://arxiv.org/abs/quant-ph/9605043

[84] S. Aaronson and A. Arkhipov, "The Computational Complexity of Linear Optics," *arXiv:1011.3245 [quant-ph]*, Nov. 2010. [Online]. Available: http://arxiv.org/abs/1011.3245

[85] R. Orús, "Tensor networks for complex quantum systems," *Nature Reviews Physics*, vol. 1, no. 9, pp. 538–550, Sep. 2019. [Online]. Available: https://www.nature.com/articles/s42254-019-0086-7

[86] M. Barbeau, "Secure Quantum Data Communications Using Classical Keying Material," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 183–195.

[87] I. Bloch, J. Dalibard, and S. Nascimbène, "Quantum simulations with ultracold quantum gases," *Nature Physics*, vol. 8, no. 4, pp. 267–276, Apr. 2012. [Online]. Available: https://www.nature.com/articles/nphys2259

[88] J. Zhang, G. Pagano, P. W. Hess, A. Kyprianidis, P. Becker, H. Kaplan, A. V. Gorshkov, Z.-X. Gong, and C. Monroe, "Observation of a Many-Body Dynamical Phase Transition with a 53-Qubit Quantum Simulator," *Nature*, vol. 551, no. 7682, pp. 601–604, Nov. 2017. [Online]. Available: http://arxiv.org/abs/1708.01044

[89] C.-C. Chen, S.-Y. Shiau, M.-F. Wu, and Y.-R. Wu, "Hybrid classical-quantum linear solver using Noisy Intermediate-Scale Quantum machines," *Scientific Reports*, vol. 9, no. 1, p. 16251, Dec. 2019. [Online]. Available: http://arxiv.org/abs/1903.10949

[90] N. P. Bauman, E. J. Bylaska, S. Krishnamoorthy, G. H. Low, N. Wiebe, C. E. Granade, M. Roetteler, M. Troyer, and K. Kowalski, "Downfolding of many-body Hamiltonians using active-space models: Extension of the sub-system embedding sub-algebras approach to unitary coupled cluster formalisms," *The Journal of Chemical Physics*, vol. 151, no. 1, p. 014107, Jul. 2019. [Online]. Available: https://aip.scitation.org/doi/10.1063/1.5094643

[91] C. Hempel, C. Maier, J. Romero, J. McClean, T. Monz, H. Shen, P. Jurcevic, B. Lanyon, P. Love, R. Babbush, A. Aspuru-Guzik, R. Blatt, and C. Roos, "Quantum chemistry calculations on a trapped-ion quantum simulator," *Physical Review X*, vol. 8, no. 3, p. 031022, Jul. 2018. [Online]. Available: http://arxiv.org/abs/1803.10238

[92] A. J. McCaskey, Z. P. Parks, J. Jakowski, S. V. Moore, T. Morris, T. S. Humble, and R. C. Pooser, "Quantum Chemistry as a Benchmark for Near-Term Quantum Computers," *arXiv:1905.01534 [physics, physics:quant-ph]*, May 2019. [Online]. Available: http://arxiv.org/abs/1905.01534

[93] B. P. Lanyon, J. D. Whitfield, G. G. Gillett, M. E. Goggin, M. P. Almeida, I. Kassal, J. D. Biamonte, M. Mohseni, B. J. Powell, M. Barbieri, A. Aspuru-Guzik, and A. G. White, "Towards quantum chemistry on a quantum computer," *Nature Chemistry*, vol. 2, no. 2, pp. 106–111, Feb. 2010. [Online]. Available: https://www.nature.com/articles/nchem.483

[94] M. Cerezo, A. Poremba, L. Cincio, and P. J. Coles, "Variational Quantum Fidelity Estimation," *Quantum*, vol. 4, p. 248, Mar. 2020. [Online]. Available: https://quantum-journal.org/papers/q-2020-03-26-248/

[95] E. Anschuetz, J. Olson, A. Aspuru-Guzik, and Y. Cao, "Variational Quantum Factoring," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science,

S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 74–85.

[96] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, Sep. 2017. [Online]. Available: https://www.nature.com/articles/nature23879

[97] M. A. Thornton and D. L. MacFarlane, "Quantum Photonic TRNG with Dual Extractor," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 171–182.

[98] S. Endo, T. Jones, S. McArdle, X. Yuan, and S. Benjamin, "Variational quantum algorithms for discovering Hamiltonian spectra," *Physical Review A*, vol. 99, no. 6, p. 062304, Jun. 2019. [Online]. Available: http://arxiv.org/abs/1806.05707

[99] S. McArdle, T. Jones, S. Endo, Y. Li, S. Benjamin, and X. Yuan, "Variational ansatz-based quantum simulation of imaginary time evolution," *npj Quantum Information*, vol. 5, no. 1, p. 75, Dec. 2019. [Online]. Available: http://arxiv.org/abs/1804.03023

[100] X. Yuan, S. Endo, Q. Zhao, Y. Li, and S. C. Benjamin, "Theory of variational quantum simulation," *Quantum*, vol. 3, p. 191, Oct. 2019. [Online]. Available: https://quantum-journal.org/papers/q-2019-10-07-191/

[101] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023023, Feb. 2016. [Online]. Available: https://doi.org/10.1088%2F1367-2630%2F18%2F2%2F023023

[102] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, no. 1, p. 4213, Jul. 2014. [Online]. Available: https://www.nature.com/articles/ncomms5213

[103] Y. Li and S. C. Benjamin, "Efficient Variational Quantum Simulator Incorporating Active Error Minimization," *Physical Review X*, vol. 7, no. 2, p. 021050, Jun. 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.7.021050

[104] P. J. J. O'Malley, R. Babbush, I. D. Kivlichan, J. Romero, J. R. McClean, R. Barends, J. Kelly, P. Roushan, A. Tranter, N. Ding, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, E. Jeffrey, E. Lucero, A. Megrant, J. Y. Mutus, M. Neeley, C. Neill, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, P. V. Coveney, P. J. Love, H. Neven, A. Aspuru-Guzik, and J. M. Martinis, "Scalable Quantum Simulation of Molecular Energies," *Physical Review X*, vol. 6, no. 3, p. 031007, Jul. 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.6.031007

[105] J. M. Gambetta, J. M. Chow, and M. Steffen, "Building logical qubits in a superconducting quantum computing system," *npj Quantum Information*, vol. 3, no. 1, pp. 1–7, Jan. 2017. [Online]. Available: https://www.nature.com/articles/s41534-016-0004-0

[106] S. Bravyi, J. M. Gambetta, A. Mezzacapo, and K. Temme, "Tapering off qubits to simulate fermionic Hamiltonians," *arXiv:1701.08213 [quant-ph]*, Jan. 2017. [Online]. Available: http://arxiv.org/abs/1701.08213

[107] F. Leymann, "Towards a Pattern Language for Quantum Algorithms," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 218–230.

[108] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum Computation by Adiabatic Evolution," *arXiv:quant-ph/0001106*, Jan. 2000. [Online]. Available: http://arxiv.org/abs/quant-ph/0001106

[109] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, "Adiabatic Quantum Computation is Equivalent to Standard Quantum Computation," *arXiv:quant-ph/0405098*, Mar. 2005. [Online]. Available: http://arxiv.org/abs/quant-ph/0405098

[110] J. C. Bardin, E. Jeffrey, E. Lucero, T. Huang, O. Naaman, R. Barends, T. White, M. Giustina, D. Sank, P. Roushan, K. Arya, B. Chiaro, J. Kelly, J. Chen, B. Burkett, Y. Chen, A. Dunsworth, A. Fowler, B. Foxen, C. Gidney, R. Graff, P. Klimov, J. Mutus, M. McEwen, A. Megrant, M. Neeley, C. Neill, C. Quintana, A. Vainsencher, H. Neven, and J. Martinis, "A 28nm Bulk-CMOS 4-to-8GHz \textbackslashtextless2mW Cryogenic Pulse Modulator for Scalable Quantum Computing," *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, pp. 456–458, Feb. 2019. [Online]. Available: http://arxiv.org/abs/1902.10864

[111] P. Rebentrost and S. Lloyd, "Quantum computational finance: quantum algorithm for portfolio optimization," *arXiv:1811.03975 [quant-ph]*, Nov. 2018. [Online]. Available: http://arxiv.org/abs/1811.03975

[112] H. Irie, G. Wongpaisarnsin, M. Terabe, A. Miki, and S. Taguchi, "Quantum Annealing of Vehicle Routing Problem with Time, State and Capacity," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 145–156.

[113] M. Paini and A. Kalev, "An approximate description of quantum states," *arXiv:1910.10543 [quant-ph]*, Nov. 2019. [Online]. Available: http://arxiv.org/abs/1910.10543

[114] D. P. Divincenzo, "Two-Bit Gates are Universal for Quantum Computation," *Physical Review A*, vol. 51, no. 2, pp. 1015–1022, Feb. 1995. [Online]. Available: http://arxiv.org/abs/cond-mat/9407022

[115] D. P. DiVincenzo and IBM, "The Physical Implementation of Quantum Computation," *arXiv:quant-ph/0002077*, Apr. 2000. [Online]. Available: http://arxiv.org/abs/quant-ph/0002077

[116] J. Yoneda, K. Takeda, A. Noiri, T. Nakajima, S. Li, J. Kamioka, T. Kodera, and S. Tarucha, "Quantum non-demolition readout of an electron spin in silicon," *Nature Communications*, vol. 11, no. 1, pp. 1–7, Mar. 2020. [Online]. Available: https://www.nature.com/articles/s41467-020-14818-8

[117] E. D. Herbschleb, H. Kato, Y. Maruyama, T. Danjo, T. Makino, S. Yamasaki, I. Ohki, K. Hayashi, H. Morishita, M. Fujiwara, and N. Mizuochi, "Ultra-long coherence times amongst room-temperature solid-state spins," *Nature Communications*, vol. 10, no. 1, pp. 1–6, Aug. 2019. [Online]. Available: https://www.nature.com/articles/s41467-019-11776-8

[118] P. V. Klimov, J. Kelly, Z. Chen, M. Neeley, A. Megrant, B. Burkett, R. Barends, K. Arya, B. Chiaro, Y. Chen, A. Dunsworth, A. Fowler, B. Foxen, C. Gidney, M. Giustina,

R. Graff, T. Huang, E. Jeffrey, E. Lucero, J. Y. Mutus, O. Naaman, C. Neill, C. Quintana, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. C. White, S. Boixo, R. Babbush, V. N. Smelyanskiy, H. Neven, and J. M. Martinis, "Fluctuations of Energy-Relaxation Times in Superconducting Qubits," *Physical Review Letters*, vol. 121, no. 9, p. 090502, Aug. 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.121.090502

[119] R. H. (Bo) Ewald, "An Introduction to Quantum Computing and Its Application," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 3–8.

[120] D. Maslov, Y. Nam, and J. Kim, "An Outlook for Quantum Computing [Point of View]," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 5–10, Jan. 2019.

[121] M. Mohseni, A. T. Rezakhani, and D. A. Lidar, "Quantum-process tomography: Resource analysis of different strategies," *Physical Review A*, vol. 77, no. 3, p. 032322, Mar. 2008. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.77.032322

[122] B. Villalonga, S. Boixo, B. Nelson, C. Henze, E. Rieffel, R. Biswas, and S. Mandrà, "A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware," *npj Quantum Information*, vol. 5, no. 1, p. 86, Dec. 2019. [Online]. Available: http://arxiv.org/abs/1811.09599

[123] A. D. Córcoles, J. M. Gambetta, J. M. Chow, J. A. Smolin, M. Ware, J. Strand, B. L. T. Plourde, and M. Steffen, "Process verification of two-qubit quantum gates by randomized benchmarking," *Physical Review A*, vol. 87, no. 3, p. 030301, Mar. 2013. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.87.030301

[124] S. Sheldon, L. S. Bishop, E. Magesan, S. Filipp, J. M. Chow, and J. M. Gambetta, "Characterizing errors on qubit operations via iterative randomized benchmarking," *Physical Review A*, vol. 93, no. 1, p. 012301, Jan. 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.93.012301

[125] E. Magesan, J. M. Gambetta, and J. Emerson, "Characterizing quantum gates via randomized benchmarking," *Physical Review A*, vol. 85, no. 4, p. 042311, Apr. 2012. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.85.042311

[126] S. Mavadia, C. L. Edmunds, C. Hempel, H. Ball, F. Roy, T. M. Stace, and M. J. Biercuk, "Experimental quantum verification in the presence of temporally correlated noise," *npj Quantum Information*, vol. 4, no. 1, pp. 1–9, Feb. 2018. [Online]. Available: https://www.nature.com/articles/s41534-017-0052-0

[127] C. C. McGeoch, "Principles and Guidelines for Quantum Performance Analysis," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 36–48.

[128] C. C. McGeoch, R. Harris, S. P. Reinhardt, and P. I. Bunyk, "Practical Annealing-Based Quantum Computing," *Computer*, vol. 52, no. 6, pp. 38–46, Jun. 2019.

[129] W. J. Hardy, C. T. Harris, Y.-H. Su, Y. Chuang, J. Moussa, L. N. Maurer, J.-Y. Li, T.-M. Lu, and D. R. Luhman, "Single and double hole quantum dots in strained Ge/SiGe quantum wells," *Nanotechnology*, vol. 30, no. 21, p. 215202, Mar. 2019. [Online]. Available: https://doi.org/10.1088%2F1361-6528%2Fab061e

[130] L. M. K. Vandersypen and M. A. Eriksson, "Quantum computing with semiconductor spins," *Physics Today*, vol. 72, no. 8, pp. 38–45, Aug. 2019. [Online]. Available: https://physicstoday.scitation.org/doi/10.1063/PT.3.4270

[131] L. Childress and R. Hanson, "Diamond NV centers for quantum computing and quantum networks," *MRS Bulletin*, vol. 38, no. 2, pp. 134–138, Feb. 2013. [Online]. Available: https://www.cambridge.org/core/journals/mrs-bulletin/article/diamond-nv-centers-for-quantum-computing-and-quantum-networks/978A4B94242CF28F9C60F0D9E95E9CBD

[132] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-Ion Quantum Computing: Progress and Challenges," *Applied Physics Reviews*, vol. 6, no. 2, p. 021314, Jun. 2019. [Online]. Available: http://arxiv.org/abs/1904.04178

[133] V. M. Schäfer, C. J. Ballance, K. Thirumalai, L. J. Stephenson, T. G. Ballance, A. M. Steane, and D. M. Lucas, "Fast quantum logic gates with trapped-ion qubits," *Nature*, vol. 555, no. 7694, pp. 75–78, Mar. 2018. [Online]. Available: https://www.nature.com/articles/nature25737

[134] C. Roos, "Viewpoint: Moving Traps Offer Fast Delivery of Cold Ions," *Physics*, vol. 5, Aug. 2012. [Online]. Available: https://physics.aps.org/articles/v5/94

[135] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, C. Ryan-Anderson, and B. Neyenhuis, "Demonstration of the QCCD trapped-ion quantum computer architecture," *arXiv:2003.01293 [quant-ph]*, Mar. 2020. [Online]. Available: http://arxiv.org/abs/2003.01293

[136] J. Chen, J. B. Altepeter, M. Medic, K. F. Lee, B. Gokden, R. H. Hadfield, S. W. Nam, and P. Kumar, "Demonstration of a Quantum Controlled-NOT Gate in the Telecommunications Band," *Physical Review Letters*, vol. 100, no. 13, p. 133603, Apr. 2008. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.100.133603

[137] T. Monz, P. Schindler, J. T. Barreiro, M. Chwalla, D. Nigg, W. A. Coish, M. Harlander, W. Hänsel, M. Hennrich, and R. Blatt, "14-Qubit Entanglement: Creation and Coherence," *Physical Review Letters*, vol. 106, no. 13, p. 130506, Mar. 2011. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.106.130506

[138] H. Goto, "Bifurcation-based adiabatic quantum computation with a nonlinear oscillator network," *Scientific Reports*, vol. 6, no. 1, pp. 1–8, Feb. 2016. [Online]. Available: https://www.nature.com/articles/srep21686

[139] ——, "Quantum Computation Based on Quantum Adiabatic Bifurcations of Kerr-Nonlinear Parametric Oscillators," *Journal of the Physical Society of Japan*, vol. 88, no. 6, p. 061015, Mar. 2019. [Online]. Available: https://journals.jps.jp/doi/10.7566/JPSJ.88.061015

[140] B. Abdo, N. T. Bronn, O. Jinka, S. Olivadese, A. D. Corcoles, V. P. Adiga, M. Brink, R. E. Lake, X. Wu, D. P. Pappas, and J. M. Chow, "Active protection of a superconducting qubit with an interferometric Josephson isolator," *Nature Communications*, vol. 10, no. 1, p. 3154, Dec. 2019. [Online]. Available: http://arxiv.org/abs/1810.07234

[141] N. Bergeal, R. Vijay, V. E. Manucharyan, I. Siddiqi, R. J. Schoelkopf, S. M. Girvin, and M. H. Devoret, "Analog information processing at the quantum limit with a Josephson

ring modulator," *Nature Physics*, vol. 6, no. 4, pp. 296–302, Apr. 2010. [Online]. Available: https://www.nature.com/articles/nphys1516

[142] B. Abdo, F. Schackert, M. Hatridge, C. Rigetti, and M. Devoret, "Josephson amplifier for qubit readout," *Applied Physics Letters*, vol. 99, no. 16, p. 162506, Oct. 2011. [Online]. Available: https://aip.scitation.org/doi/10.1063/1.3653473

[143] A. Bouland and S. Aaronson, "Generation of universal linear optics by any beam splitter," *Physical Review A*, vol. 89, no. 6, p. 062316, Jun. 2014. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.89.062316

[144] P. T. Greenland, S. A. Lynch, A. F. G. van der Meer, B. N. Murdin, C. R. Pidgeon, B. Redlich, N. Q. Vinh, and G. Aeppli, "Coherent control of Rydberg states in silicon," *Nature*, vol. 465, no. 7301, pp. 1057–1061, Jun. 2010. [Online]. Available: https://www.nature.com/articles/nature09112

[145] A. Kshetrimayum, H. Weimer, and R. Orús, "A simple tensor network algorithm for two-dimensional steady states," *Nature Communications*, vol. 8, no. 1, pp. 1–7, Nov. 2017. [Online]. Available: https://www.nature.com/articles/s41467-017-01511-6

[146] Q. L. He, L. Pan, A. L. Stern, E. C. Burks, X. Che, G. Yin, J. Wang, B. Lian, Q. Zhou, E. S. Choi, K. Murata, X. Kou, Z. Chen, T. Nie, Q. Shao, Y. Fan, S.-C. Zhang, K. Liu, J. Xia, and K. L. Wang, "Chiral Majorana fermion modes in a quantum anomalous Hall insulator–superconductor structure," *Science*, vol. 357, no. 6348, pp. 294–299, Jul. 2017. [Online]. Available: https://science.sciencemag.org/content/357/6348/294

[147] H. Zhang, C.-X. Liu, S. Gazibegovic, D. Xu, J. A. Logan, G. Wang, N. van Loo, J. D. S. Bommer, M. W. A. de Moor, D. Car, R. L. M. Op het Veld, P. J. van Veldhoven, S. Koelling, M. A. Verheijen, M. Pendharkar, D. J. Pennachio, B. Shojaei, J. S. Lee, C. J. Palmstrøm, E. P. A. M. Bakkers, S. D. Sarma, and L. P. Kouwenhoven, "Quantized Majorana conductance," *Nature*, vol. 556, no. 7699, pp. 74–79, Apr. 2018. [Online]. Available: https://www.nature.com/articles/nature26142

[148] H. K. Xu, C. Song, W. Y. Liu, G. M. Xue, F. F. Su, H. Deng, Y. Tian, D. N. Zheng, S. Han, Y. P. Zhong, H. Wang, Y.-x. Liu, and S. P. Zhao, "Coherent population transfer between uncoupled or weakly coupled states in ladder-type superconducting qutrits," *Nature Communications*, vol. 7, no. 1, pp. 1–6, Mar. 2016. [Online]. Available: https://www.nature.com/articles/ncomms11018

[149] A. E. Antipov, A. Bargerbos, G. W. Winkler, B. Bauer, E. Rossi, and R. M. Lutchyn, "Effects of Gate-Induced Electric Fields on Semiconductor Majorana Nanowires," *Physical Review X*, vol. 8, no. 3, p. 031041, Aug. 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.8.031041

[150] J. J. Wallman and J. Emerson, "Noise tailoring for scalable quantum computation via randomized compiling," *Physical Review A*, vol. 94, no. 5, p. 052325, Nov. 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.94.052325

[151] M. A. Broome, S. K. Gorman, M. G. House, S. J. Hile, J. G. Keizer, D. Keith, C. D. Hill, T. F. Watson, W. J. Baker, L. C. L. Hollenberg, and M. Y. Simmons, "Two-electron spin correlations in precision placed donors in silicon," *Nature Communications*, vol. 9, no. 1, pp. 1–7, Mar. 2018. [Online]. Available: https://www.nature.com/articles/s41467-018-02982-x

[152] J. A. Sedlacek, J. Stuart, D. H. Slichter, C. D. Bruzewicz, R. McConnell, J. M. Sage, and J. Chiaverini, "Evidence for multiple mechanisms underlying surface electric-field noise in ion traps," *Physical Review A*, vol. 98, no. 6, p. 063430, Dec. 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.98.063430

[153] R. J. Niffenegger, J. Stuart, C. Sorace-Agaskar, D. Kharas, S. Bramhavar, C. D. Bruzewicz, W. Loh, R. McConnell, D. Reens, G. N. West, J. M. Sage, and J. Chiaverini, "Integrated optical control and enhanced coherence of ion qubits via multi-wavelength photonics," *arXiv:2001.05052 [physics, physics:quant-ph]*, Jan. 2020. [Online]. Available: http://arxiv.org/abs/2001.05052

[154] R. Behnia, M. O. Ozmen, A. A. Yavuz, and M. Rosulek, "TACHYON: Fast Signatures from Compact Knapsack," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. Toronto, Canada: Association for Computing Machinery, Jan. 2018, pp. 1855–1867. [Online]. Available: https://doi.org/10.1145/3243734.3243819

[155] T. P. Harty, D. T. C. Allcock, C. J. Ballance, L. Guidoni, H. A. Janacek, N. M. Linke, D. N. Stacey, and D. M. Lucas, "High-Fidelity Preparation, Gates, Memory, and Readout of a Trapped-Ion Quantum Bit," *Physical Review Letters*, vol. 113, no. 22, p. 220501, Nov. 2014. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.113.220501

[156] C. J. Ballance, T. P. Harty, N. M. Linke, M. A. Sepiol, and D. M. Lucas, "High-Fidelity Quantum Logic Gates Using Trapped-Ion Hyperfine Qubits," *Physical Review Letters*, vol. 117, no. 6, p. 060504, Aug. 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.117.060504

[157] Y. Wang, S. Crain, C. Fang, B. Zhang, S. Huang, Q. Liang, P. H. Leung, K. R. Brown, and J. Kim, "High-fidelity Two-qubit Gates Using a MEMS-based Beam Steering System for Individual Qubit Addressing," *arXiv:2003.12430 [physics, physics:quant-ph]*, Apr. 2020. [Online]. Available: http://arxiv.org/abs/2003.12430

[158] J. P. Gaebler, T. R. Tan, Y. Lin, Y. Wan, R. Bowler, A. C. Keith, S. Glancy, K. Coakley, E. Knill, D. Leibfried, and D. J. Wineland, "High-Fidelity Universal Gate Set for ${ˆ{9}\textbackslashtextbackslashmathrm{Be}}ˆ{+}$ Ion Qubits," *Physical Review Letters*, vol. 117, no. 6, p. 060505, Aug. 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.117.060505

[159] C. D. Bruzewicz, R. McConnell, J. Stuart, J. M. Sage, and J. Chiaverini, "Dual-species, multi-qubit logic primitives for Ca + /Sr + trapped-ion crystals," *npj Quantum Information*, vol. 5, no. 1, pp. 1–10, Nov. 2019. [Online]. Available: https://www.nature.com/articles/s41534-019-0218-z

[160] W. Loh, J. Stuart, D. Reens, C. D. Bruzewicz, D. Braje, J. Chiaverini, P. W. Juodawlkis, J. M. Sage, and R. McConnell, "A Brillouin Laser Optical Atomic Clock," *arXiv:2001.06429 [physics, physics:quant-ph]*, Jan. 2020. [Online]. Available: http://arxiv.org/abs/2001.06429

[161] M. Mueller, K. Hammerer, Y. L. Zhou, C. F. Roos, and P. Zoller, "Simulating open quantum systems: from many-body interactions to stabilizer pumping," *New Journal of Physics*, vol. 13, no. 8, p. 085007, Aug. 2011. [Online]. Available: http://arxiv.org/abs/1104.2507

[162] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019. [Online]. Available: https://www.nature.com/articles/s41586-019-1666-5

[163] I. L. Markov, A. Fatima, S. V. Isakov, and S. Boixo, "Quantum Supremacy Is Both Closer and Farther than It Appears," *arXiv:1807.10749 [quant-ph]*, Sep. 2018. [Online]. Available: http://arxiv.org/abs/1807.10749

[164] J. Randall, S. Weidt, E. D. Standing, K. Lake, S. C. Webster, D. F. Murgia, T. Navickas, K. Roth, and W. K. Hensinger, "Efficient preparation and detection of microwave dressed-state qubits and qutrits with trapped ions," *Physical Review A*, vol. 91, no. 1, p. 012322, Jan. 2015. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.91.012322

[165] G. S. Paraoanu, "Microwave-induced coupling of superconducting qubits," *Physical Review B*, vol. 74, no. 14, p. 140504, Oct. 2006. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.74.140504

[166] D. Rosenberg, D. Kim, R. Das, D. Yost, S. Gustavsson, D. Hover, P. Krantz, A. Melville, L. Racz, G. O. Samach, S. J. Weber, F. Yan, J. L. Yoder, A. J. Kerman, and W. D. Oliver, "3D integrated superconducting qubits," *npj Quantum Information*, vol. 3, no. 1, pp. 1–5, Oct. 2017. [Online]. Available: https://www.nature.com/articles/s41534-017-0044-0

[167] D. Rosenberg, S. Weber, D. Conway, D. Yost, J. Mallek, G. Calusine, R. Das, D. Kim, M. Schwartz, W. Woods, J. L. Yoder, and W. D. Oliver, "3D integration and packaging for solid-state qubits," *arXiv:1906.11146 [cond-mat, physics:quant-ph]*, Jul. 2019. [Online]. Available: http://arxiv.org/abs/1906.11146

[168] C. Song, K. Xu, W. Liu, C.-p. Yang, S.-B. Zheng, H. Deng, Q. Xie, K. Huang, Q. Guo, L. Zhang, P. Zhang, D. Xu, D. Zheng, X. Zhu, H. Wang, Y.-A. Chen, C.-Y. Lu, S. Han, and J.-W. Pan, "10-Qubit Entanglement and Parallel Logic Operations with a Superconducting Circuit," *Physical Review Letters*, vol. 119, no. 18, p. 180511, Nov. 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.119.180511

[169] M. H. Devoret and R. J. Schoelkopf, "Superconducting Circuits for Quantum Information: An Outlook," *Science*, vol. 339, no. 6124, pp. 1169–1174, Mar. 2013. [Online]. Available: https://science.sciencemag.org/content/339/6124/1169

[170] J. I.-J. Wang, D. Rodan-Legrain, L. Bretheau, D. L. Campbell, B. Kannan, D. Kim, M. Kjaergaard, P. Krantz, G. O. Samach, F. Yan, J. L. Yoder, K. Watanabe, T. Taniguchi, T. P. Orlando, S. Gustavsson, P. Jarillo-Herrero, and W. D. Oliver, "Coherent control of a hybrid superconducting circuit made with graphene-based van der

Waals heterostructures," *Nature Nanotechnology*, vol. 14, no. 2, pp. 120–125, Feb. 2019. [Online]. Available: https://www.nature.com/articles/s41565-018-0329-2

[171] Y. Sung, F. Beaudoin, L. M. Norris, F. Yan, D. K. Kim, J. Y. Qiu, U. von Lüpke, J. L. Yoder, T. P. Orlando, S. Gustavsson, L. Viola, and W. D. Oliver, "Non-Gaussian noise spectroscopy with a superconducting qubit sensor," *Nature Communications*, vol. 10, no. 1, p. 3715, Sep. 2019. [Online]. Available: https://www.nature.com/articles/s41467-019-11699-4

[172] F. Yan, S. Gustavsson, A. Kamal, J. Birenbaum, A. P. Sears, D. Hover, T. J. Gudmundsen, D. Rosenberg, G. Samach, S. Weber, J. L. Yoder, T. P. Orlando, J. Clarke, A. J. Kerman, and W. D. Oliver, "The flux qubit revisited to enhance coherence and reproducibility," *Nature Communications*, vol. 7, no. 1, pp. 1–9, Nov. 2016. [Online]. Available: https://www.nature.com/articles/ncomms12964

[173] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm, "Simple Pulses for Elimination of Leakage in Weakly Nonlinear Qubits," *Physical Review Letters*, vol. 103, no. 11, p. 110501, Sep. 2009. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.103.110501

[174] C. Müller, J. Lisenfeld, A. Shnirman, and S. Poletto, "Interacting two-level defects as sources of fluctuating high-frequency noise in superconducting circuits," *Physical Review B*, vol. 92, no. 3, p. 035442, Jul. 2015. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.92.035442

[175] S. Yarkoni, H. Wang, A. Plaat, and T. Bäck, "Boosting Quantum Annealing Performance Using Evolution Strategies for Annealing Offsets Tuning," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 157–168.

[176] A. Das and B. K. Chakrabarti, "Colloquium: Quantum annealing and analog quantum computation," *Reviews of Modern Physics*, vol. 80, no. 3, pp. 1061–1081, Sep. 2008. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.80.1061

[177] Y. Susa, Y. Yamashiro, M. Yamamoto, I. Hen, D. A. Lidar, and H. Nishimori, "Quantum annealing of the p-spin model under inhomogeneous transverse field driving," *Physical Review A*, vol. 98, no. 4, p. 042326, Oct. 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.98.042326

[178] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model," *Physical Review E*, vol. 58, no. 5, pp. 5355–5363, Nov. 1998. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.58.5355

[179] M. Pino and J. J. García-Ripoll, "Quantum annealing in spin-boson model: from a perturbative to an ultrastrong mediated coupling," *New Journal of Physics*, vol. 20, no. 11, p. 113027, Nov. 2018. [Online]. Available: https://doi.org/10.1088%2F1367-2630%2Faaeeea

[180] J. A. Smolin, G. Smith, and A. Vargo, "Pretending to factor large numbers on a quantum computer," *Nature*, vol. 499, no. 7457, pp. 163–165, Jul. 2013. [Online]. Available: http://arxiv.org/abs/1301.7007

[181] J. Chen, F. Zhang, C. Huang, M. Newman, and Y. Shi, "Classical Simulation of Intermediate-Size Quantum Circuits," *arXiv:1805.01450 [quant-ph]*, May 2018. [Online]. Available: http://arxiv.org/abs/1805.01450

[182] T. Stollenwerk, E. Lobe, and M. Jung, "Flight Gate Assignment with a Quantum Annealer," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 99–110.

[183] A. Mehta, M. Muradi, and S. Woldetsadick, "Quantum Annealing Based Optimization of Robotic Movement in Manufacturing," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 136–144.

[184] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997. [Online]. Available: http://arxiv.org/abs/quant-ph/9508027

[185] E. Kapit, "A Very Small Logical Qubit," *Physical Review Letters*, vol. 116, no. 15, p. 150501, Apr. 2016. [Online]. Available: http://arxiv.org/abs/1510.06117

[186] A. Kirke, "Application of Grover's Algorithm on the ibmqx4 Quantum Computer to Rule-based Algorithmic Music Composition," *ArXiv*, 2019.

[187] D. Ristè, M. P. da Silva, C. A. Ryan, A. W. Cross, A. D. Córcoles, J. A. Smolin, J. M. Gambetta, J. M. Chow, and B. R. Johnson, "Demonstration of quantum advantage in machine learning," *npj Quantum Information*, vol. 3, no. 1, pp. 1–5, Apr. 2017. [Online]. Available: https://www.nature.com/articles/s41534-017-0017-3

[188] M. Streif, F. Neukart, and M. Leib, "Solving Quantum Chemistry Problems with a D-Wave Quantum Annealer," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 111–122.

[189] E. Pelofske, G. Hahn, and H. Djidjev, "Solving Large Maximum Clique Problems on a Quantum Annealer," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 123–135.

[190] T. Inagaki, Y. Haribara, K. Igarashi, T. Sonobe, S. Tamate, T. Honjo, A. Marandi, P. L. McMahon, T. Umeki, K. Enbutsu, O. Tadanaga, H. Takenouchi, K. Aihara, K.-i. Kawarabayashi, K. Inoue, S. Utsunomiya, and H. Takesue, "A coherent Ising machine for 2000-node optimization problems," *Science*, vol. 354, no. 6312, pp. 603–606, Nov. 2016. [Online]. Available: https://science.sciencemag.org/content/354/6312/603

[191] T. Vyskočil, S. Pakin, and H. N. Djidjev, "Embedding Inequality Constraints for Quantum Annealing Optimization," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 11–22.

[192] Y. Susa, Y. Yamashiro, M. Yamamoto, and H. Nishimori, "Exponential Speedup of Quantum Annealing by Inhomogeneous Driving of the Transverse Field," *Journal of the Physical Society of Japan*, vol. 87, no. 2, p. 023002, Jan. 2018. [Online]. Available: https://journals.jps.jp/doi/10.7566/JPSJ.87.023002

[193] N. P. Bauman, G. H. Low, and K. Kowalski, "Quantum simulations of excited states with active-space downfolded Hamiltonians," *The Journal of Chemical Physics*, vol. 151, no. 23, p. 234114, Dec. 2019. [Online]. Available: https://aip.scitation.org/doi/10.1063/1.5128103

[194] M. Motta, E. Ye, J. R. McClean, Z. Li, A. J. Minnich, R. Babbush, and G. K.-L. Chan, "Low rank representations for quantum simulation of electronic structure," *arXiv:1808.02625 [physics, physics:quant-ph]*, Aug. 2018. [Online]. Available: http://arxiv.org/abs/1808.02625

[195] R. Babbush, N. Wiebe, J. McClean, J. McClain, H. Neven, and G. K.-L. Chan, "Low-Depth Quantum Simulation of Materials," *Physical Review X*, vol. 8, no. 1, p. 011044, Mar. 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.8.011044

[196] E. Tang, "A quantum-inspired classical algorithm for recommendation systems," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2019. Phoenix, AZ, USA: Association for Computing Machinery, Jun. 2019, pp. 217–228. [Online]. Available: https://doi.org/10.1145/3313276.3316310

[197] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for solving linear systems of equations," *Physical Review Letters*, vol. 103, no. 15, p. 150502, Oct. 2009. [Online]. Available: http://arxiv.org/abs/0811.3171

[198] I. I. Soloviev, N. V. Klenov, S. V. Bakurskiy, M. Y. Kupriyanov, A. L. Gudkov, and A. S. Sidorenko, "Beyond Moore's technologies: operation principles of a superconductor alternative," *Beilstein Journal of Nanotechnology*, vol. 8, no. 1, pp. 2689–2710, Dec. 2017. [Online]. Available: https://www.beilstein-journals.org/bjnano/articles/8/269

[199] *The BIG Bell Test*, publication Title: The BIG Bell Test. [Online]. Available: https://thebigbelltest.org/

[200] F. S. Khan and T. S. Humble, "Nash Embedding and Equilibrium in Pure Quantum States," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science, S. Feld and C. Linnhoff-Popien, Eds. Cham: Springer International Publishing, 2019, pp. 51–62.

[201] J. D. Wong-Campos, S. A. Moses, K. G. Johnson, and C. Monroe, "Demonstration of Two-Atom Entanglement with Ultrafast Optical Pulses," *Physical Review Letters*, vol. 119, no. 23, p. 230501, Dec. 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.119.230501

[202] M. Mohseni, P. Read, H. Neven, S. Boixo, V. Denchev, R. Babbush, A. Fowler, V. Smelyanskiy, and J. Martinis, "Commercialize quantum technologies in five years," *Nature News*, vol. 543, no. 7644, p. 171, Mar. 2017. [Online]. Available: http://www.nature.com/news/commercialize-quantum-technologies-in-five-years-1.21583

[203] Q. Zhuang, Z. Zhang, J. Dove, F. N. C. Wong, and J. H. Shapiro, "Floodlight Quantum Key Distribution: A Practical Route to Gbps Secret-Key Rates," *Physical Review A*, vol. 94, no. 1, p. 012322, Jul. 2016. [Online]. Available: http://arxiv.org/abs/1510.08737

[204] S. A. Podoshvedov, "Efficient Quantum Teleportation of Unknown Qubit Based on DV-CV Interaction Mechanism," *Entropy*, vol. 21, no. 2, p. 150, Feb. 2019. [Online]. Available: https://www.mdpi.com/1099-4300/21/2/150

[205] N. Friis, O. Marty, C. Maier, C. Hempel, M. Holzäpfel, P. Jurcevic, M. B. Plenio, M. Huber, C. Roos, R. Blatt, and B. Lanyon, "Observation of Entangled States of a Fully Controlled 20-Qubit System," *Physical Review X*, vol. 8, no. 2, p. 021012, Apr. 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.8.021012

[206] L. Gyongyosi, S. Imre, and H. V. Nguyen, "A Survey on Quantum Channel Capacities," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1149–1205, 2018. [Online]. Available: http://arxiv.org/abs/1801.02019

[207] J. M. Chow, J. M. Gambetta, E. Magesan, D. W. Abraham, A. W. Cross, B. R. Johnson, N. A. Masluk, C. A. Ryan, J. A. Smolin, S. J. Srinivasan, and M. Steffen, "Implementing a strand of a scalable fault-tolerant quantum computing fabric," *Nature Communications*, vol. 5, no. 1, p. 4015, Jun. 2014. [Online]. Available: https://www.nature.com/articles/ncomms5015

[208] A. D. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow, "Demonstration of a quantum error detection code using a square lattice of four superconducting qubits," *Nature Communications*, vol. 6, no. 1, p. 6979, Apr. 2015. [Online]. Available: https://www.nature.com/articles/ncomms7979

[209] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, "Error mitigation extends the computational reach of a noisy quantum processor," *Nature*, vol. 567, no. 7749, pp. 491–495, Mar. 2019. [Online]. Available: https://www.nature.com/articles/s41586-019-1040-7

[210] J. I. Colless, V. V. Ramasesh, D. Dahlen, M. S. Blok, M. E. Kimchi-Schwartz, J. R. McClean, J. Carter, W. A. de Jong, and I. Siddiqi, "Computation of Molecular Spectra on a Quantum Processor with an Error-Resilient Algorithm," *Physical Review X*, vol. 8, no. 1, p. 011021, Feb. 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.8.011021

[211] S. Endo, Q. Zhao, Y. Li, S. Benjamin, and X. Yuan, "Mitigating algorithmic errors in Hamiltonian simulation," *Physical Review A*, vol. 99, no. 1, p. 012334, Jan. 2019. [Online]. Available: http://arxiv.org/abs/1808.03623

[212] S. Endo, S. C. Benjamin, and Y. Li, "Practical Quantum Error Mitigation for Near-Future Applications," *Physical Review X*, vol. 8, no. 3, p. 031027, Jul. 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.8.031027

[213] D. Litinski, "A Game of Surface Codes: Large-Scale Quantum Computing with Lattice Surgery," *Quantum*, vol. 3, p. 128, Mar. 2019. [Online]. Available: https://quantum-journal.org/papers/q-2019-03-05-128/

[214] N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto, "Layered Architecture for Quantum Computing," *Physical Review X*, vol. 2, no. 3, p. 031007, Jul. 2012. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.2.031007

[215] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "A Limit on the Speed of Quantum Computation in Determining Parity," *Physical Review Letters*, vol. 81, no. 24, pp. 5442–5444, Dec. 1998. [Online]. Available: http://arxiv.org/abs/quant-ph/9802045

[216] A. W. Harrow and A. Montanaro, "Quantum computational supremacy," *Nature*, vol. 549, no. 7671, pp. 203–209, Sep. 2017. [Online]. Available: https://www.nature.com/articles/nature23458

[217] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, "Characterizing Quantum Supremacy in Near-Term Devices," *Nature Physics*, vol. 14, no. 6, pp. 595–600, Jun. 2018. [Online]. Available: http://arxiv.org/abs/1608.00263

[218] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, "Elucidating reaction mechanisms on quantum computers," *Proceedings of the National Academy of Sciences*, vol. 114, no. 29, pp. 7555–7560, Jul. 2017. [Online]. Available: https://www.pnas.org/content/114/29/7555

[219] I. Kerenidis and A. Prakash, "Quantum Recommendation Systems," in *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), C. H. Papadimitriou, Ed., vol. 67. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, pp. 49:1–49:21. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2017/8154

[220] B. Foxen, C. Neill, A. Dunsworth, P. Roushan, B. Chiaro, A. Megrant, J. Kelly, Z. Chen, K. Satzinger, R. Barends, F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, S. Boixo, D. Buell, B. Burkett, Y. Chen, R. Collins, E. Farhi, A. Fowler, C. Gidney, M. Giustina, R. Graff, M. Harrigan, T. Huang, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, P. Klimov, A. Korotkov, F. Kostritsa, D. Landhuis, E. Lucero, J. McClean, M. McEwen, X. Mi, M. Mohseni, J. Y. Mutus, O. Naaman, M. Neeley, M. Niu, A. Petukhov, C. Quintana, N. Rubin, D. Sank, V. Smelyanskiy, A. Vainsencher, T. C. White, Z. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, "Demonstrating a Continuous Set of Two-qubit Gates for Near-term Quantum Algorithms," *arXiv:2001.08343 [quant-ph]*, Feb. 2020. [Online]. Available: http://arxiv.org/abs/2001.08343

[221] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, S. Boixo, M. Broughton, B. B. Buckley, D. A. Buell, B. Burkett, N. Bushnell, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, S. Demura, A. Dunsworth, E. Farhi, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Graff, S. Habegger, M. P. Harrigan, A. Ho, S. Hong, T. Huang, W. J. Huggins, L. Ioffe, S. V. Isakov, E. Jeffrey, Z. Jiang, C. Jones, D. Kafri, K. Kechedzhi, J. Kelly, S. Kim, P. V. Klimov, A. Korotkov, F. Kostritsa, D. Landhuis, P. Laptev, M. Lindmark, E. Lucero, O. Martin, J. M. Martinis, J. R. McClean, M. McEwen, A. Megrant, X. Mi, M. Mohseni, W. Mruczkiewicz, J. Mutus, O. Naaman, M. Neeley, C. Neill, H. Neven, M. Y. Niu, T. E. O'Brien, E. Ostby, A. Petukhov, H. Putterman, C. Quintana, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, D. Strain, K. J. Sung, M. Szalay, T. Y. Takeshita, A. Vainsencher, T. White, N. Wiebe, Z. J. Yao, P. Yeh, and A. Zalcman, "Hartree-Fock on a superconducting qubit quantum computer," *arXiv:2004.04174 [physics, physics:quant-ph]*, Apr. 2020. [Online]. Available: http://arxiv.org/abs/2004.04174

[222] J. R. McClean, Z. Jiang, N. C. Rubin, R. Babbush, and H. Neven, "Decoding quantum errors with subspace expansions," *Nature Communications*, vol. 11, no. 1, p. 636, Dec. 2020. [Online]. Available: http://arxiv.org/abs/1903.05786

[223] T. Takeshita, N. C. Rubin, Z. Jiang, E. Lee, R. Babbush, and J. R. McClean, "Increasing the Representation Accuracy of Quantum Simulations of Chemistry without Extra Quantum Resources," *Physical Review X*, vol. 10, no. 1, p. 011004, Jan. 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.10.011004

[224] R. Shaydulin, H. Ushijima-Mwesigwa, C. F. A. Negre, I. Safro, S. M. Mniszewski, and Y. Alexeev, "A Hybrid Approach for Solving Optimization Problems on Small Quantum Computers," *Computer*, vol. 52, no. 6, pp. 18–26, Jun. 2019.

[225] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, 2014. [Online]. Available: https://www.readcube.com/articles/10.3389%2Ffphy.2014.00005

[226] B. M. Terhal and D. P. DiVincenzo, "Adaptive Quantum Computation, Constant Depth Quantum Circuits and Arthur-Merlin Games," *arXiv:quant-ph/0205133*, Mar. 2004. [Online]. Available: http://arxiv.org/abs/quant-ph/0205133

[227] S. Aaronson, "BQP and the Polynomial Hierarchy," *arXiv:0910.4698 [quant-ph]*, Oct. 2009. [Online]. Available: http://arxiv.org/abs/0910.4698

[228] S. Aaronson and A. Ambainis, "Forrelation: A Problem that Optimally Separates Quantum from Classical Computing," *arXiv:1411.5729 [quant-ph]*, Nov. 2014. [Online]. Available: http://arxiv.org/abs/1411.5729

[229] R. Raz and A. Tal, "Oracle separation of BQP and PH," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2019. Phoenix, AZ, USA: Association for Computing Machinery, Jun. 2019, pp. 13–23. [Online]. Available: https://doi.org/10.1145/3313276.3316315

[230] E. Bernstein and U. Vazirani, "Quantum complexity theory," in *in Proc. 25th Annual ACM Symposium on Theory of Computing, ACM*, 1993, pp. 11–20.

[231] G. B. Lesovik, I. A. Sadovskyy, M. V. Suslov, A. V. Lebedev, and V. M. Vinokur, "Arrow of time and its reversal on the IBM quantum computer," *Scientific Reports*, vol. 9, no. 1, pp. 1–8, Mar. 2019. [Online]. Available: https://www.nature.com/articles/s41598-019-40765-6

[232] H. Goto, "Universal quantum computation with a nonlinear oscillator network," *Physical Review A*, vol. 93, no. 5, p. 050301, May 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.93.050301

[233] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, "Open Quantum Assembly Language," *arXiv:1707.03429 [quant-ph]*, Jul. 2017. [Online]. Available: http://arxiv.org/abs/1707.03429

[234] R. S. Smith, M. J. Curtis, and W. J. Zeng, "A Practical Quantum Instruction Set Architecture," *arXiv:1608.03355 [quant-ph]*, Feb. 2017. [Online]. Available: http://arxiv.org/abs/1608.03355

[235] JMiszczak, *List of QC simulators*, Oct. 2015, publication Title: Quantiki Type: Text. [Online]. Available: https://quantiki.org/wiki/list-qc-simulators

[236] C. S. Calude, "De-quantizing the solution of Deutsch's problem," *International Journal of Quantum Information*, vol. 5, no. 03, pp. 409–415, 2007.

[237] K. E. C. Booth, M. Do, J. C. Beck, E. Rieffel, D. Venturelli, and J. Frank, "Comparing and Integrating Constraint Programming and Temporal Planning for Quantum Circuit Compilation," *arXiv:1803.06775 [quant-ph]*, Mar. 2018. [Online]. Available: http://arxiv.org/abs/1803.06775

[238] J. D. Franson and R. A. Brewster, "Limitations on the use of the Heisenberg picture," *arXiv:1811.06517 [quant-ph]*, Dec. 2018. [Online]. Available: http://arxiv.org/abs/1811.06517

[239] M. J. Bremner, A. Montanaro, and D. J. Shepherd, "Achieving quantum supremacy with sparse and noisy commuting quantum computations," *Quantum*, vol. 1, p. 8, Apr. 2017. [Online]. Available: http://arxiv.org/abs/1610.01808

[240] J. Preskill, "Quantum computing and the entanglement frontier," *arXiv:1203.5813 [cond-mat, physics:quant-ph]*, Nov. 2012. [Online]. Available: http://arxiv.org/abs/1203.5813

[241] A. Shukla, M. Sisodia, and A. Pathak, "Complete characterization of the directly implementable quantum gates used in the IBM quantum processors," *arXiv:1805.07185 [quant-ph]*, Jul. 2018. [Online]. Available: http://arxiv.org/abs/1805.07185