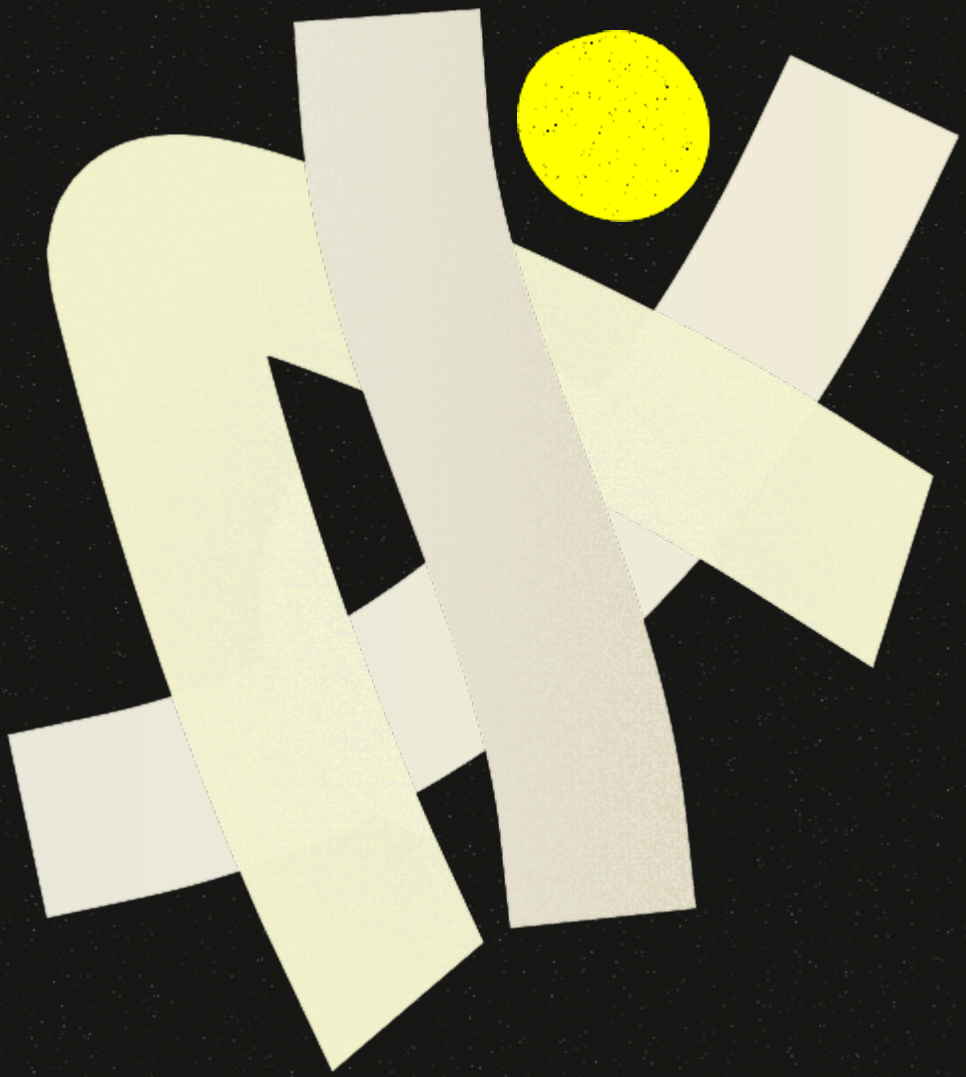
Quantum Computing
A Guide for the Curious

Michael Solomentsev



Quantum Computing: A Guide for the Curious

Michael Solomentsev

Copyright © 2026 Michael Solomentsev

This work is licensed under Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0). To view a copy of this license, visit https://creativecommons.org/licenses/by-nc-nd/4.0/.

"If you can't explain something to a first-year student, then you haven't really understood."

- Richard Feynman

Contents

Chapter 1

Introduction

1.1 Why Quantum Computing?

We live in a world of increasingly complex systems, growing far too complicated and interconnected for any single person to understand. Perhaps we establish some small fiefdom in a field where we 'get' what's going on, but cede control of other subjects to the so-called experts. We use mental models and heuristics to feel the contours of murky superstructures: Falling interest rates cause investment in the economy; growing transistor counts mean more powerful computers; increased renewable energy production means a greater need for grid-connected energy storage; et cetera.

As is so often true today, the most salient example of this is artificial intelligence. Most of us don't quite know how large language models work. If forced to, I personally could say some things about linear algebra, tokens, and transformers; but my explanation would be incomplete at best and misleading at worst. Nevertheless, I know enough about large language models to navigate a world increasingly shaped by them. And it's clear that this step – this ability to update our understanding of specific technologies and systems – is increasingly important. Imagine opening your internet browser in 2026 without knowing that text, images, and video could all be convincingly realistic but totally AI generated!

Enough about AI. As I've just noted, it's more or less legible. I want to discuss a different technology whose fundamental hallmark is its illegibility. Of course, I'm talking about quantum computing (the title of this work does a poor job at building suspense!). If you're reading this, you are probably familiar with the concept abstractly: a machine that utilizes quantum mechanics to perform computations – a quantum computer. As we'll discuss, this kind of machine has been long-theorized, but has proven extremely hard to build. The field began with the ideas of physicists and mathematicians, acquired footholds in academic research institutions, and has since broken out into the world of commercial, for-profit ventures. The profile of the industry has risen dramatically in recent years, perhaps coupled with the trend of quantum computing startups going

public.

Industry insiders and outside boosters regularly hail the revolutionary promise of quantum computation. The pace of PR from quantum startups and big players like Google is increasing; companies like IonQ are raising huge sums of money in the public markets; and there's no shortage of quantum evangelists on LinkedIn or at conferences. And yet, people who understand the field are in short supply, and seem uniquely unwilling or unable to help the rest of us construct the correct mental model of the field. I'm serious! If you try to learn about LLMs, there is an endless supply of folks trying to teach you how transformers work from the ground up. There are dozens of highly technical people writing publicly about the semiconductor industry for generalist audiences. And from the quantum computing people? Crickets! Trust me, I've looked, and I'm being mostly fair in my characterization!

And so, here we are. Consider me your somewhat reticent guide to the world of quantum computing, dragged into this out of a sense of moral necessity. If the industry won't explain itself, it falls to me to do so.

My personal interest in quantum computing began with reading various press releases posted on Hacker News (a generalist tech forum). I'm an electrical engineer with a PhD in power electronics. I'm perfectly used to having the typically software-focused posts on HN be incomprehensible to me, but these posts were more illegible than the norm. Moreover, the comments sections on these posts – usually relatively helpful – were instead marred by inter-commentariat bickering about the significance (or lack thereof) of the results in question. Consider my curiosity piqued: I wanted to know more about this whole quantum computing thing.

I also happen to be a deep-tech entrepreneur, and at an event in the summer of 2025, I happened to listen to a remarkably lucid pitch from a guy with a quantum computing startup. He seemed to be an exceptionally straight shooter who spoke specifically about avoiding the hype in the field. I approached him at the next break and introduced myself as a technically literate outsider who wanted to become 'conversationally fluent' in quantum computing. I asked him if he had any book recommendations for someone like me. He looked at me askance, paused a beat, and told me I could get a PhD in physics. Not quite the answer I was hoping for.

We live in an era dominated by hype cycles. It is par for the course that evangelists will tell us that some new technology will revolutionize the world. It is unfortunate but true that fortunes are built regardless of the veracity of those promises. We don't have to go far back to remember the craze around NFTs – a technology that was on face absurd, but carried enough technical gravitas to legitimize investment (put charitably).

Quantum computing advocates have put out their fair share of weighty claims – purporting that the technology may be used for cryptography[1], material discovery[2], defense[3], handwriting analysis[4], finance[5], and much more. But more so than in other hype cycles, the actual technology here is so difficult to understand. To grasp what's going on in the field, one must have a reasonable fluency in computational complexity theory, condensed matter physics, semiconductor

engineering, and a half dozen other subjects. You have to wonder – will this industry, backed by billions in public and private capital, end up looking more like NFTs or like LLMs?

There are some helpful resources for learning about quantum computing as a layperson (Thomas Wong's Introduction to Classical and Quantum Computing, Scott Aaronson's writings, Michael Nielsen's work), but they tend to focus on primarily the theory of computation. As an engineer, I'm interested in not only the theory, but the practical details of implementation. As an entrepreneur, I'm interested in the landscape of companies and where value might be created. As a member of society, I'm concerned billions are getting incinerated in a tech-washed financial scam. Finding no suitable explanation for the motivated layman, I'm writing one.

I hope for this to not compete with those other excellent texts, but serve as a complement. While I will discuss the foundations of computation, I hope to differentiate this work in a few ways: This text will aim to A) not harp on the details of the math and physics too much; B) provide a clear picture of the hardware landscape at this moment in time; C) explain the potential real-world applications of this technology; and perhaps most importantly, D) do all of this with a certain removed cynicism. I am not a quantum physicist; I am not employed by the quantum industry. I have no dog in the fight but admittedly, am a natural skeptic. This industry seems to have an infinite capacity for absorbing funding but struggles to communicate the ultimate value of their technology.

This book is written to answer the questions I have and to explain my conclusions to you, dear reader. Without further ado, let's begin.

1.2 What is Quantum Computing?

To illustrate the difficulties in answering this question, let me try to pose an adjacent one: What is classical computing? Well – it's obviously what happens when you organize billions of transistors into a complex system of interconnected memory and processing units. It's just as obviously a branch of applied mathematics stemming from questions around computability and Turing machines. It's software engineers and semiconductor companies and mathematicians and materials scientists and the billions of people who use their computers on a daily basis. Quantum computing is just the same – an amorphous blob of interrelated and competing stakeholders clustered around a particular 'field'.

What makes quantum even more difficult to pin down is how much of the field is yet to be determined. To build a classical computer, we use semiconductor manufacturing processes that, while constantly evolving, all build on a multi-decade long lineage of linear improvements to photolithography. I can explain that to a motivated high school student. In quantum, there are at least a half dozen competing modalities for physical implementation of these devices, all with radically different advantages. Similarly, there are several wildly varied fields where quantum computing might be practically used, many of which aren't very engaged with the quantum community itself.

Here's my high-level overview of the different niches of quantum computing, all of which we will visit in greater detail later:

- Physicists: The genesis of quantum computing as a field came from Nobel prize winning physicist Richard Feynman, who speculated that solving the almost intractable problems of simulating quantum behavior might be possible with a computer that leveraged that same quantum nature. Many cite simulation as the primary use case of future quantum computers, with applications as varied as drug discovery and superconducting material development. Physicists are involved in every single sub-field of the industry, from theory to engineering.

- Computer science theorists: These mathematicians and computer scientists are fundamentally interested in what quantum computation may be able to do. The industry owes a great debt to these folks – if it wasn't for Peter Shor's development of an efficient quantum algorithm for factoring, the field may have languished in academic obscurity.

- Quantum computer developers: This includes academic groups, startups, and tech giants all working to develop real, physical quantum computers. Many of these groups are pursuing competing technical approaches (superconducting vs trapped ion vs neutral atom vs etc.). Individual members of these organizations might be physicists, electrical engineers, software engineers, and technical folks of all kinds.

- Quantum 'ecosystem' companies: These are mostly startups, not concerned with building quantum computers but orchestrating the system around them. Some are building hardware (interconnects, controls), while others are dealing with the software and programming of these computers. Likely some of these capabilities are being built in house as well by those organizations in the prior bullet point.

- Cryptography & Security People: Those focused on the security risks presented by successful quantum computer deployment. These folks are typically concerned with developing and deploying 'post-quantum cryptography' – i.e. encryption schemes that are expected to be resilient to quantum-based attacks.

- Industry users: Ostensibly businesses will pay to use quantum computers. Some businesses are actively working with QC hardware developers for potential use cases. Some are releasing vacuous press releases declaring quantum advantage for tasks like hedge fund management[6].

- Quantum evangelists & investors: This is an over-loaded category that includes both technically sophisticated venture capitalists and retail traders who are making high risk bets on $IONQ. Like it or not, these folks are part of the quantum ecosystem, and many press releases are tailored to catch their interest (and increase stock prices).

But admittedly, I'm dodging the original question. Here's my simplistic answer: Quantum computing is a field that is trying to build, control, and use a new kind of computer that utilizes quantum mechanics.

1.3 History of Quantum Computing

Quantum computing, as its name suggests, straddles the divide between two very disparate fields.

Physicists have known about some of the strange manifestations of quantum mechanics on the macroscopic world for a long time – the famous double slit experiment, showing the wave-particle duality of light, was conducted in 1801[7]! Over a period of about 150 years, quantum mechanics was shaped into a consistent theory by a murderer's row of all-timer physicists like de Broglie, Heisenberg, Dirac, and von Neumann. During World War II, the field found a real-world application in the development of nuclear weapons as part of the Manhattan project.

WWII also served as a catalyst for the formalization and application of many theories related to computation, most notably in the field of cryptography (making and breaking codes). Alan Turing, famous for breaking the German enigma cipher, proposed the 'Turing machine' – a theoretical device with a strip of 0's and 1's, a printer, and a defined set of rules that could perform (or attempt to perform) computations. Today's computers look nothing like Turing machines, but this thought experiment served as an important foundation for thinking about how efficiently machines could solve problems (more on this idea, called computational complexity later).

In 1980, several academics made the first steps to combine the two fields, seemingly independently[8]. Paul Benioff described a 'quantum Turing machine' that could model Hamiltonians (governing equations of quantum systems)[9]. Soon after, Richard Feynman (Nobel prize winning physicist, bongo player, and safecracker, among other things) proposed that to efficiently simulate quantum systems, you could utilize a computer with quantum properties itself[10]. Over in the USSR, mathematician Yuri Manin discussed using quantum mechanical phenomena to perform computations[11]. People generally point to these proposals as the start of the discipline.

From here, it feels like the field is a snowball gradually picking up momentum as it rolls down a mountain. The 1980s experience gradual theoretical advancement, with the development of simple quantum algorithms (which we will discuss in greater detail later). David Deutsch publishes a seminal paper that concretizes several foundational principles in the field in 1985[12]. In 1994, Peter Shor proposes an algorithm for the factoring of large numbers[13], which gives the field a 'killer app' (startup speak for financially valuable use case) – breaking encryption schemes! From here, government funding begins to pour into the field.

Now the experimentalists begin making some progress. In the mid-90s to 2010s, academics and national labs (in particular, the National Institute for Standards and Technology and Sandia National Labs) make significant gains in

demonstrating various qubit modalities and operations on those qubits.

In 1999, the first (of many) quantum startups is created: D-Wave Systems, which is still alive today. D-Wave attempted to commercialize a very specific and limited kind of quantum computer, known as a quantum annealer, but has recently pivoted to being a more 'normal' quantum company (to the extent that any quantum company could be considered normal). Since then, quantum computing has drifted slowly into the mainstream. IBM has long been doing foundational quantum research, but other big companies have jumped into the ring, including Honeywell, Google, Microsoft, and Amazon. These days, quantum startups are a dime a dozen, with several having gotten large enough to go public: D-Wave, Rigetti, IonQ, Infleqtion, Quantinuum and others.

My antennas are finely tuned to the public consciousness, and I think the quantum computing industry is on the cusp of being ubiquitous. All the more reason, I figure, to properly understand it.

1.4 This Work's Purpose

I am not a quantum physicist, nor a quantum computing expert. Reading this will not turn you into either. I'm convinced there is a sizable population who are intrigued by this beguiling topic who find the current mainstream media coverage facile and publications in the field incomprehensible. I hope this work to be a middle ground. I am aiming to approach the topic with intellectual rigor, but with the knowledge that I cannot possibly explain (or understand) the field at its complete depths. To that end, this work is limited in scope to four sections:

1. COMPUTATION: What is quantum computation, and how does it differ from classical computation? What can quantum computers do in theory?

2. IMPLEMENTATION: How does one build a quantum computer? What are the engineering challenges and constraints that make this difficult?

3. APPLICATION: What would successfully developed quantum computers actually do?

4. STATE OF PLAY: What does the field look like today?

My goal is to give you an idea of the contours of this field: a mental model for what quantum computing is today and could be in the future. Keep in mind that this is an extremely dense and technical subject, and you should not feel discouraged if you do not understand all the presented material at first glance. I invite you to focus on the sections that you are most excited or interested by, while encouraging you to at least skim the rest - theory, hardware, software, and applications are all very interlinked here. No matter what, I hope this text proves useful.

1.5 Acknowledgments and Resources

Let me re-emphasize: I am not a quantum physicist! In many ways, this text is a re-hashing of work done by others far smarter than I. My contribution, as I see it, is contextualization, organization, and an ability to translate heavy material to a more casual level. There's obviously a huge amount of work that has educated me about quantum computing, much of it that I will cite in the following pages. However, I want to call out certain works that I think A) I've leaned on particularly heavily; B) serve as excellent introductions into their respective sub-fields:

- Section 1: Computation
  - "Introduction to Classical and Quantum Computing" by Thomas Wong is an wonderful introduction to the theory of both classical and quantum computation. It's so good!
  - "Quantum Computing Since Democritus" by Scott Aaronson is a wide-ranging survey of all things computational complexity theory, with its center aimed at quantum computation.
  - "Quantum Computation and Quantum Information" by Michael Nielsen and Isaac Chuang (often called 'Mike & Ike') is the textbook for the field. It's also a big-boy book with serious math and should be read after the above two.
- Section 2: Implementation
  - Austin Fowler's Coursera course "Hands-on quantum error correction with Google Quantum AI" is a great interactive walkthrough (albeit a difficult one) into the world of error correction and surface codes.
  - Similarly, Joschka Roffe's paper, "Quantum Error Correction: An Introductory Guide" does what its title suggests, very well.
  - The Qolab team's recent roadmap paper: "How to Build a Quantum Supercomputer: Scaling from Hundreds to Millions of Qubits" doesn't exactly fulfill its titular promise, but is a good overview of scaling problems and potential solutions for the superconducting field.
  - For many of the qubit hardware implementations, the best learning resources are actually Youtube videos!
    * Daniel Sank of Google is an incredible presenter. These three talks (1, 2, 3) are all very nice introductions to superconducting qubits. In fact, those latter two videos are from a UCLA event with several talks worth watching.
    * This video from Janka Biznárová covers superconducting qubit manufacturing, which I don't really discuss at all!
    * See this talk from Duke Professor Crystal Noel on trapped ion systems.

- Section 3: Applications:
  - Information is distributed here. If you're interested, I would recommend you look through the citations for this section.

Huge thanks to the creators of the above resources, and all the great people who've taken their personal time to educate me about the field. Any errors are solely my own.

Chapter 2

Computation

2.1 Computation? How hard could it be?

Before we understand anything about quantum computation, let's motivate why it might be useful. But what does useful even mean? And so, we begin our journey in the world of theoretical computer science.

We are all intimately familiar with 'computers', but perhaps less universal is the concept of 'computation'. The average person uses their computer to browse the internet, send emails, and look at TikTok. Quantum computers will not displace classical computers for any of those tasks[†]. We will instead scope our discussion to solving specifically defined mathematical problems. While some of these problems are totally abstract, others are extremely practical. Many of the applications of conventional computers we find so useful are undergirded by these numerical problems: Cryptography protects our data; machine learning algorithms shape the content we see; linear algebra is the key to performing simulations of physical phenomena.

Understandably, we may want to know how 'difficult' a problem is for a computer (classical or quantum) to solve. Computer scientists have come up with a concept, known as computational complexity, which categorizes certain problems based on how many resources (whether they be time, operations, or memory) they take to solve.

Let's take addition as an example. Say I have two numbers, each represented with n bits[‡]. Using a classical computer, I can add those numbers with approximately 9n operations (technically we would call them gates, but let's just think of them as steps in a recipe)[§]. As anyone who's taken a computer science class will know, we can represent the computational intensity of this task with what's

†. Although they will, of course, run DOOM: https://github.com/Lumorti/Quandoom

‡. We will do a brief review of bits shortly, but here's the ultra-Cliff notes, if you aren't familiar: Bits are how data is stored on classical computers; they can be 0 or 1; more bits means we can store bigger numbers.

§. Oh no, we really do seem to be putting the cart before the horse. Gates are operations between bits, like AND, OR, or NOT.

called big-O notation: addition has linear complexity and thus is in $O(n)$. Essentially, this metric codifies that as n increases, the computational overhead required to perform this task (add two n-bit numbers) scales linearly with n. However, this need not be the case.

There are problems with constant complexity – $O(1)$. Say we want to know if a n-bit number is even or odd. A computer can check this by simply looking at the value of the last bit of the number: If it's a one, the number is odd; if it's a zero, the number is even. This holds if n is equal to one or equal to ten million. Thus, we can solve this problem in constant time.

There are problems with higher than linear complexity. Some problems have polynomial complexity, such as $O(n^2)$ or $O(n^3)$. For instance, multiplying two matrices requires polynomial time*. This is much less desirable than a linear complexity, because adding more bits to the problem can substantially increase how long it takes to run. However, many problems are even worse than this and have exponential complexity, like $O(2^n)$. The quintessential example of this is factoring. Given a large integer, it is very time-consuming to break it up into its constituent factors. This forms the basis for modern encryption, which uses the difficult nature of factoring to provide security for encoded data.

Generally, we make a distinction between tasks with polynomial complexity and those with exponential complexity. Anything with polynomial or below complexity is deemed efficient, i.e., tractable for classical computers to solve. Problems with exponential complexity are considered to scale so poorly that they cannot easily be solved using classical computers. This tends to be a useful practical distinction, but it's worth examining it a bit deeper. Depending on the value of n, an algorithm with $O(n^8)$ (nominally an efficient polynomial algorithm) may be significantly slower than an inefficient exponential $O(1.25^n)$ approach. However, as n tends toward infinity, the polynomial algorithm will eventually win out. Thus, it's important to consider what the specific complexity is for a given algorithm and use case. Nevertheless, it tends to be the case that in most real applications, we desire polynomial-time, efficient algorithms.

Theoretical computer scientists have a hobby of using complexity to sort problems into different classes. If a problem can be solved in polynomial time, they say it is in class **P**. Perhaps the most famous problem in computer science relates to class **P** and another class, **NP**. **NP** is the class of problems that can be verified in polynomial time. A problem like factoring is very much in NP – if an algorithm spits out a factor, we can easily check if it's right by dividing the input by the output. The famous problem asks: is **P** equal to **NP**? That is, if I can verify a problem efficiently, can I also compute it efficiently?

Most people think that **P** is not equal to **NP**. This intuitively makes sense. If it were possible to efficiently factor large primes, modern encryption schemes wouldn't be very secure (and lots of resources have been spent trying to figure out how to factor large primes efficiently).

A logical next step, especially given the context of this work, is asking if a quantum computer can solve problems efficiently that a classical computer

∗. Funnily enough, the bound on matrix multiplication is the strange $O(n^{2.137\ldots})$.

cannot. We will discuss this further below, but let's assume for a moment that quantum computers can do the same operations classical computers can (i.e., we can use them like a classical computer, and not access their 'quantumness'). Because of this, quantum computers can solve problems in the same number of steps as their classical brethren. Thus, the class of problems quantum computers can solve efficiently at least is equal to **P**.

It turns out that quantum computers can also do some problems efficiently that classical computers can't. As such, the theorists have defined a separate class from **P** called '**BQP**' which includes all problems that can be solved by quantum computers in polynomial time with bounded error. Why bounded error? We will learn that quantum computers are, at their core, probabilistic devices. We will soon examine some algorithms where there is a probability of outputting the wrong result. Luckily, these algorithms typically can achieve higher accuracy via running in longer but still polynomial time. Practically, **BQP** is the direct quantum analogue to **P**.

There are several kinds of interesting and potentially practically useful algorithms that are in **BQP** but not in **P**[*]. Notably, factoring can be done in polynomial time on a quantum computer, as can certain kinds of algorithms that relate to chemical simulation. This is really exciting, and is the fundamental reason why the world is interested in developing quantum computers. We will explore this efficient factoring algorithm later in this chapter, and consider other real-world applications in Chapter 5.

But this is no free lunch! As we will examine in Chapter 3, quantum computers are likely to operate slower and with less usable (qu)bits than their classical brethren. Algorithms with equivalent computational complexity are likely to run longer (and require more resources) on a quantum computer than on a classical one. After all, I can readily access billions of transistors operating at multi-GHz speeds on my laptop – that's pretty good! For a theoretical quantum speedup to result in a practically realizable speedup, it's likely that the quantum algorithm must reduce an exponential-time problem into a polynomial-time one.

We are now armed with a core motivating principle: quantum computers may be able to solve certain problems faster than classical computers. This is the singular driving force behind the billions of dollars of investment and hundreds of years of person-hours spent on quantum computer development. But we are still hopelessly confused as to what quantum computation is.

2.2 A Qubit

Claude Shannon invented the term 'bit' in 1948 to refer to units of information[†] The term has become a foundational concept for computing. A singular bit can be 0 or 1, and we can encode 2^N values in N bits (3 bits gives us $2^3 = 8$

∗. It is more accurate to say that it is hypothesized that they are in BQP and not in P. It hasn't yet been proven that BQP is not equal to P.

†. Claude Shannon's master's thesis probably had more impact than 99% of people's life-work. The man was a legend!

Input	NOT(Input)
0	1
1	0

Table 2.1: The truth table of a NOT gate.

Input 1	Input 2	AND(Input 1, Input 2)
0	0	0
0	1	0
1	0	0
1	1	1

Table 2.2: The truth table of an AND gate.

codewords: 000; 001; 010; 011; 100; 101; 110; 111). We can do math on these bits, and use groups of them to encode integers, decimal numbers, sounds, pictures, or video.

Functionally, we implement bits as being voltage levels stored on semiconductor chips. Think of this as banks of light switches, which can either be turned on or off. A row of 5 alternating switches could be in the state 11010, encoding the number 26 in binary $(1*(2^4)+1*(2^3)+0*(2^2)+1*(2^1)+0*(2^0) = 16+8+2 = 26)$.

We can also do computation on those bits – the simplest operation we can do is flipping an individual bit. Say we flip the last switch in the previous example, transforming its state to 11011. Now, the switches represent a binary 27*. The act of flipping a bit is called a NOT gate. The NOT gate is an example of a 1-bit gate (with one input and output). This logic is codified in its truth table, shown in Table 2.1. We can also implement a 2-bit gate, such as the AND gate, which takes two bits as inputs and outputs a single bit. That output is 1 if and only if both inputs are also 1 (if A is true AND B is true, then output triue), as shown in Table 2.2. There are several 2-bit gates, such as AND, OR, XOR (exclusive or), and NAND (not and).

It's now time to begin our first foray into the world of quantum computing. To those most interested in physical implementations, this text's singular focus on theory for this first section may be disappointing, but the tight interrelation between theory, software, and hardware in the quantum world makes this extremely critical material to understand. So don't skip it (at least skim it)!

The quantum computing analogue to a bit is a quantum bit, or *qubit* (incredible creativity in nomenclature). The qubit has several interesting and unique properties, the first and foremost that it can exist in a superposition of states. Pop science articles will typically describe this by saying that the qubit can be

∗. Note that I'm making an explicit demarcation between the physical state of the switches and the number they represent. I could easily say that the leftmost bit actually represents the sign of the number I am trying to represent, such that 11011 means negative 11 (1011 being binary 11), and 01011 means positive 11. I want to emphasize this – there are the physical bits in a system, and there is what they represent in our computations, but these are not the same thing.

both 0 and 1 at the same time. Somehow, this both over- and under-sells the unique properties of the qubit.

It is more accurate to say that a single qubit exists in a probability distribution of states. We are going to use a helpful visual aid known as the Bloch sphere, shown in Fig. 2.1. This is a 3D unit sphere with an arrow (a vector) pointing to somewhere on the surface of the sphere, shown below. The vertical axis corresponds to states totally analogous to the conventional bit – if the arrow points straight up, we call that the $|0\rangle$ state, and straight down, the $|1\rangle$ state. If we start in $|0\rangle$ and rotate the arrow 180 degrees, we end up in $|1\rangle$ – that's equivalent to a NOT gate. This operation is shown in Fig. 1B. So far, this is actually exactly the same as conventional bits! Give me 5 qubits, and I can encode 26 in binary via the state $|1\rangle \otimes |1\rangle \otimes |0\rangle \otimes |1\rangle \otimes |0\rangle$. Of course, this would be a tremendous waste of the engineering effort required to give me those qubits, but it does reveal a powerful result: A quantum computer can at least do everything a classical computer can do.

Quantum people use 'bra ket notation' to denote quantum states. We're not going to worry too much about the details here. Just know that if it's inside $|$these funny lines$\rangle$, it's a qubit state. $|\psi\rangle$ is a generic variable used to denote an arbitrary qubit state. If we have multiple qubits, we can put them together like so: $|A\rangle \otimes |B\rangle$ or $|AB\rangle$ are equivalent.

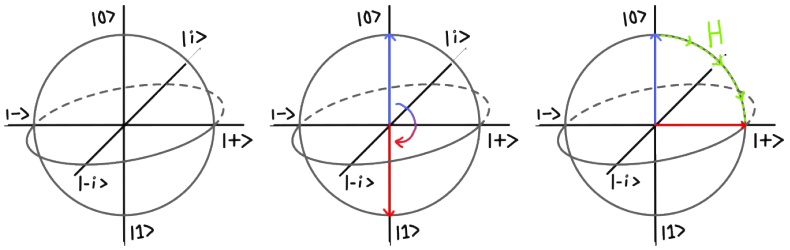


Figure 2.1: (A) Empty Bloch sphere; (B) Bloch sphere showing a qubit beginning in $|0\rangle$ (in blue) that gets rotated 180 degrees to $|1\rangle$ (in red); (C) Bloch sphere showing a qubit be rotated 90 degrees from $|0\rangle$ to $|+\rangle$.

OK, let's go one step further. What if instead of rotating the arrow by 180 degrees, we rotate it by 90 degrees, towards the horizontal axis – the state labeled $|+\rangle$? This is shown in Fig. 1C. Well, we can now describe the state of the qubit, which we will call $|\psi\rangle$, as being a linear combination of $|0\rangle$ and $|1\rangle$. We will refer to this combination as a superposition. In fact, it is exactly 50% $|0\rangle$ and 50% $|1\rangle$. This should immediately seem strange to you. After all, we would never describe a conventional bit as being half 0 and half 1. The unintuitive and unique property here is that when we measure the qubit, it collapses to either $|0\rangle$ or $|1\rangle$, each with 50% probability*. Crucially, until we measure it, the qubit is still in

*. Note that we are measuring along what's called the 'computational basis' – i.e. the possible outputs are either $|0\rangle$ or $|1\rangle$. We can also measure along any arbitrary basis, which just looks like two points opposite from each other on the Bloch sphere. $|+\rangle$ and $|-\rangle$ form an

that superposition: the act of measurement collapses its state. We can express this mathematically like so:

$$|\psi\rangle = |+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

The coefficients in front of $|0\rangle$ and $|1\rangle$ are not the probabilities of measuring each state but are instead 'amplitudes'. We can translate this amplitude into a probability of measuring each state by squaring it ($1/\sqrt{2}^2 = 1/2$, or 50%). Amplitudes, unlike probabilities, can be negative and complex (have imaginary numbers in them*).

Thus, the state of a single qubit can be described using four continuous parameters (the real and imaginary components of each the two amplitudes). When we draw a vector in a Bloch sphere, we are implicitly projecting this four dimensional vector space down into three dimensions. We can do so without loss of information because the qubit occupies a limited space in those four dimensions: we know that the probabilities of measuring each state must sum to 1! This then creates a restriction on what values the amplitudes can take on. As we describe multiple qubits, we will need even more parameters to characterize the system, expanding into what is known as a multi-dimensional 'Hilbert space'.

We could also rotate the arrow to a new point on the Bloch sphere called $|\psi'\rangle$, shown in Fig. 2.2. Note that $|\psi'\rangle$ is much closer to $|0\rangle$ than $|1\rangle$. Correspondingly, if we were to measure $|\psi'\rangle$, it is much more likely to 'collapse' to the $|0\rangle$ state than the $|1\rangle$ state. This is the core of the concept of superposition.

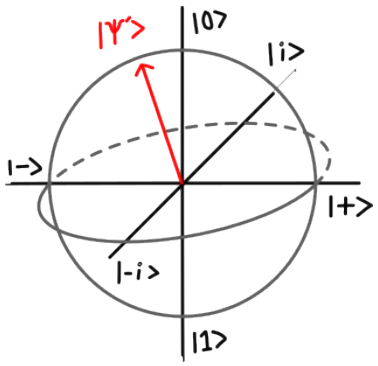


Figure 2.2: A qubit in the state $|\psi'\rangle$.

There's a complexity here that I'm glossing over. $|+\rangle$ is a state where the probabilities of measuring $|0\rangle$ and $|1\rangle$ are equal; but it is not the only state where

alternative basis. Mostly we will measure in the computational basis, although some quantum protocols perform intermediate measurements in alternative bases. For the most part, it's not worth worrying about.

∗. An imaginary number has a coefficient of $\sqrt{-1}$, typically denoted with an i. A complex number has a real and imaginary component, such as $10 - 2i$.

this is true. All points on the circle highlighted in green in Fig. 2.3, including the states $|-\rangle$, $|i\rangle$, and $|-i\rangle$, have equal probabilities of measuring $|0\rangle$ and $|1\rangle$.

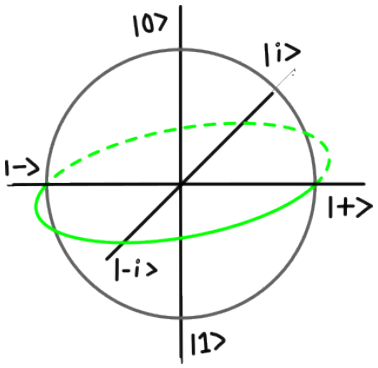


Figure 2.3: A Bloch sphere with points equidistant from $|0\rangle$ and $|1\rangle$ highlighted in green.

However, these states are not equivalent. We say they have different phases. We can represent these states as linear combinations of the $|0\rangle$ and $|1\rangle$ states:

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

$$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

It's easy to see that the amplitudes of the $|-\rangle$ state have the same magnitudes, but different signs than the amplitudes of $|+\rangle$. While they are impossible to differentiate by a simple measurement in the $|0\rangle/|1\rangle$ (computational) basis*, these qubits will behave differently in multi-qubit interactions. As we will see, a critical part of quantum algorithms is how these amplitudes will either constructively or destructively interfere to get the desired results.

*. But we could distinguish them easily if we measured both in the $|+\rangle/|-\rangle$ basis.

Quantum Amplitudes & Probabilities

Quantum amplitudes are essentially what would happen if probabilities could be negative[a]. In a 2-qubit quantum state like:

$$|\psi\rangle = a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle$$

Each amplitude corresponds to the probability of measuring that particular outcome. Hence:

$$P(|00\rangle) = |a|^2$$

$$P(|01\rangle) = |b|^2$$

$$P(|10\rangle) = |c|^2$$

$$P(|11\rangle) = |d|^2$$

Importantly, $|a|^2 + |b|^2 + |c|^2 + |d|^2$ must equal 1, because those are all the possible options post-measurement. Notice that even if a is negative or complex, the probability of measuring $|00\rangle$ is still positive. This is what we refer to as 'phase'. As noted above, two qubits could be in different states, but still have the same measurement probabilities, due to a phase difference. Those negative signs are incredibly important to implementing quantum computation, because they allow operations to create constructive and destructive interference. We will see examples of this shortly.

a. Note that I'm stealing this interpretation directly from Scott Aaronson, who wrote the very nice primer to quantum computational complexity theory "Quantum Computing Since Democritus."

Let's also revisit the concept of gates. I noted before that with a bit, the only 1-bit gate we can perform is flipping it (AKA the NOT gate). In the qubit example, that NOT gate is equivalent to performing a 180 degree rotation around the sphere. But note that we can also make any arbitrary rotation around the unit sphere! I can go from $|0\rangle$ to $|\psi\rangle$ (that's a gate) or from $|\psi\rangle$ to $|\psi'\rangle$ (that's another gate). We can string gates together, and these will also be gates. As we will learn, we can assemble any arbitrary logical operation we want from a more limited set of gates. This is advantageous because certain physical implementations of quantum computing only allow the use of certain gates (or, by their nature, make it easy to use some gates but not others).

Single Qubit Gates are 2x2 Unitary Matrices

All quantum math can be expressed in terms of linear algebra and matrices. We can express qubit states as vectors and 1 qubit gates as 2x2 unitary matrices. A unitary matrix U has the following property: $UU^* = I$, that is, when multiplied by its conjugate transpose, gives the identity matrix. Let's give an example, using the Hadamard gate, H:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1*1+1*0 \\ 1*1+0*-1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle$$

You'll often see circuit diagrams of computations like the below. In these diagrams, each horizontal line represents a single qubit, and each box represents a gate. Gates must always have the same number of inputs and outputs (unlike classical computing, where something like a AND gate has two inputs and one output). The qubit moves left to right in time, experiencing gates along the way. See Fig. 2.4, where a Hadamard gate (H) transforms an input from the $|0\rangle$ state to the $|+\rangle$ state.

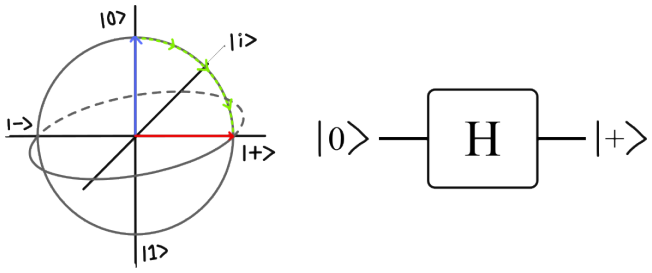
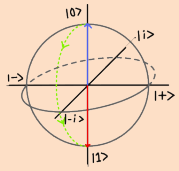


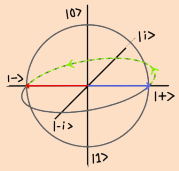
Figure 2.4: Bloch Sphere with a $|0\rangle$ initialized qubit experiencing a Hadamard gate, and corresponding circuit diagram.

Common 1-Qubit Gates

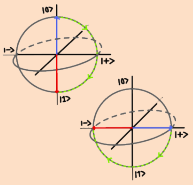
Here are several of the most common 1 qubit gates. X, Y, and Z all rotate the state 180 degrees around their respective axes. X functions as a bit flip, Z as a phase flip, and Y as a combination of the two. These three are known as Pauli gates. The S and T gates are essentially fractional phase rotations, functioning as 90 and 45 degree rotations respectively. The T gate is particularly important because it is necessary to achieve universal quantum computing in many implementations, but difficult to practically implement. More on that later.



$|0\rangle$ — X — $|1\rangle$ $\quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

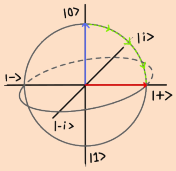


$|+\rangle$ — Z — $|-\rangle$ $\quad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

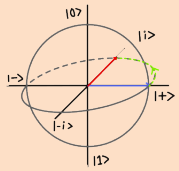


$|0\rangle$ — Y — $|1\rangle$

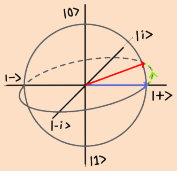
$|+\rangle$ — Y — $|-\rangle$ $\quad \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$



$|0\rangle$ — H — $|+\rangle$ $\quad \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$



$|+\rangle$ — S — $|i\rangle$ $\quad \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$



— T — $\quad \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$

2.3 A Diversion into Analog Computing

It's tempting to read the above and think that a qubit on its own is more powerful than a bit. After all, a bit can be either 0 or 1, but the qubit can encode something in this continuous multi-dimensional vector space! There are a few complications to this. The first is that while we can put the qubit in an arbitrary state, we can only measure it once, which collapses it to either $|0\rangle$ or $|1\rangle$. If given a qubit in a special state, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, we cannot extract the values of α and β. After all, as soon as you measure the qubit, it will collapse to one of two states and some information will be lost. As a result, it is not possible to encode arbitrarily large amounts of information in a single qubit*. This is dictated by Holevo's theorem[15].

Putting that issue aside, there are also problems with the idea of simply using the qubit as an outright analog computer. Analog computing is a very old idea that aims to use continuous values in computation, rather than the discrete values that traditional bits provide. We can visualize this by imagining a special analog water computer. This special computer would be programmed to pump water between several tanks before depositing a specific amount into an output vessel. The volume of the output would correspond to the answer to the question posed to the computer. This is a very attractive idea, and proponents of this kind of computation often argue that it could be substantially more efficient than traditional computing†.

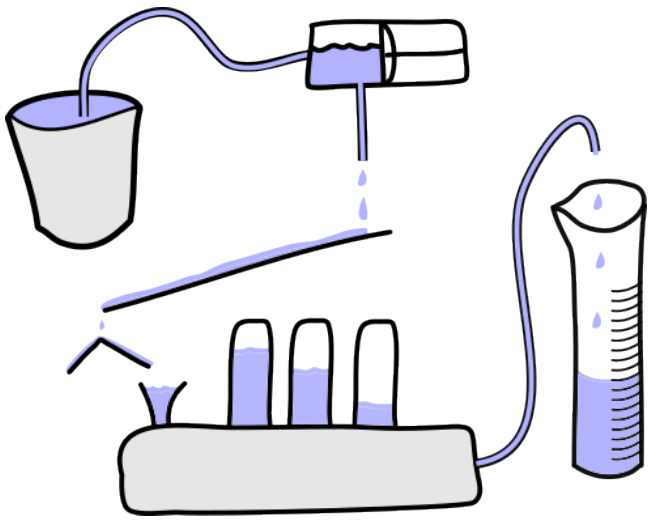


Figure 2.5: A doodle of an analog water computer.

∗. Although interesting novel results suggest that can perform computations that encode information more densely than classical bits given a very specific set of criteria.[14]

†. Several startups are attempting to commercialize various analog computing schemes, albeit using semiconductors and not water!

However, analog computing suffers from several difficulties, the primary one being errors in the system. In a traditional computer, every computation can be executed with virtually 100% accuracy. We can do thousands or millions of operations in a row, and get the right answer in the end, because binary operations on our bits are extremely reliable. However, in an analog computer, individual components or gates may not be as precise as desired, and errors may compound over the course of a calculation. Say one of my pumping operations (essentially equivalent to a gate) dispenses 1% more water than it needs to during every individual operation. If that 'gate' gets used hundreds of times over a long calculation, the result could be extremely wrong. Moreover, it may be unverifiably wrong! This is a simple but potentially fatal problem for the idea.

As a result, when we talk about quantum computing, we are most often not talking about using it in a 'analog fashion'. That is – in most algorithms we expect to encode inputs in binary, and get back answers in binary. However, quantum algorithms will put qubits in many intermediate, superimposed states before they return their final answer. Along the way, they will use algorithms that leverage quantum properties to perform active error correction on those qubits, mitigating the build-up of these errors. Quantum computing is thus distinct from both classical binary computing and classical analog computing.

It is worth noting that some in the field are working on 'quantum simulators', which is in fact akin to an analog quantum computer[16]. These simulators aim to recreate some sort of quantum mechanical system or property using more macro-scale particles (atoms are commonly used). This kind of simulator is actually what Feynman discussed in his seminal proposal of quantum computation. This approach is distinct from the gate-based quantum computers we've been talking about, since it never deals with any sort of digital values. Instead, analog measurements are taken within the simulator, which then correspond to some physical property of the simulated problem. This approach is being actively pursued by many academic groups for problems such as superconductivity. We will further discuss using quantum computers for simulation in sections below, but note that this analog approach is an alternative – one that may be easier and more practical for certain types of problems.

2.4 Qubits, plural

So far, we've learned that qubits have the property of superposition, wherein they obey probabilistic laws that determine the outcome of any measurement. Let us now examine another unique property of qubits: Entanglement. We are going to talk about entanglement right now as a purely mathematical concept, as something that changes the mathematical representation of the state of the qubit system. It is also a physical phenomenon, one that can be generated using clever engineering on real quantum computing hardware, but let's put that aside. For now, entanglement is simply a math operation that will enable very unique (and potentially useful) methods of computation.

Put simply, two qubits are entangled when one's state is dependent on the

other's. Measurement of one qubit affects the state of its entangled counterparts. The prototypical example of this is the two-qubit state $|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. If the first qubit is measured to be $|0\rangle$, the state of the second qubit must also be $|0\rangle$. Correspondingly, if we measure one qubit to be $|1\rangle$, the other must also be in state $|1\rangle$. This particular state is known as one of the Bell states*.

Entangled vs. Product States

Some multi-qubit states are entangled, and others are not – these are called product states. Product states can be factored into products of component qubits, while entangled states cannot. For example:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

If we can extricate two different single qubit states from this state, then it is a product state:

$$|\psi\rangle = (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle) = |A\rangle \otimes |B\rangle = |A\rangle|B\rangle$$

However, some states cannot be factored like this, such as the example given in the text above:

$$|\psi\rangle = 1/\sqrt{2}|00\rangle + 1/\sqrt{2}|11\rangle$$

It's impossible to factor this into two independent qubit states! These two qubits are maximally entangled. It's also now impossible to draw them as two independent Bloch spheres.

This should not appear intuitive to the reader. This is not a process implemented with logic gates; it's not the same as "if qubit A is $|0\rangle$, then qubit B becomes $|0\rangle$." Instead, it is a quantum mechanical phenomenon. Two qubits can become entangled, then separated, and when one is measured the other's state will collapse instantaneously. Einstein famously called this 'spooky action at a distance.' This property freaked Einstein out a lot, because he understood physics at a very deep level. As someone who doesn't understand physics that well, I don't get that worked up about it.

Unfortunately it's not easy to visualize entanglement, and we have to abandon our trusty friend the Bloch sphere. Originally, each qubit existed in what's called $\mathbb{C}^2$, where its state was defined by two complex numbers (its amplitudes). We could graph on our 3-dimensional Bloch sphere. A set of N qubits will be in $\mathbb{C}^{2^N}$, and be defined by 2^N amplitudes (each a complex number). If the qubits are independent, we could draw the system using N Bloch spheres. However, if these qubits become entangled, their states become dependent on one another, and we no longer can graph the entire system state in any reasonable way. We can and

∗. The Bell states are a particular set of 2-qubit entangled states which are maximally entangled. There are four of them.

will project this multidimensional vector space down to two or three dimensions, but that's not helpful for the moment. The crucial thing to understand is that entanglement turns individual qubits into a complex, interdependent system.

Entanglement is a very important enabling property for quantum computation, as we will see. But how does one actually entangle two qubits? Well, we need a way for two qubits to interact. As you might guess, this is done via 'two-qubit gates'.

There are a variety of two-qubit gates, but one of the most common is CNOT. CNOT, or 'controlled not,' is a gate that flips the second qubit (the target) if the first qubit (the control) is $|1\rangle$. Like so:

$$CNOT\,|11\rangle = |10\rangle$$

$$CNOT\,|10\rangle = |11\rangle$$

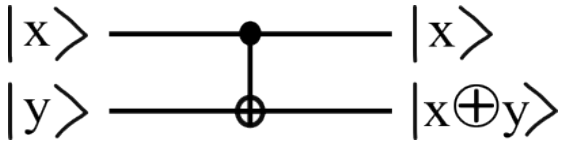
$$CNOT\,|01\rangle = |01\rangle$$

$$CNOT\,|00\rangle = |00\rangle$$

Crucially, we can apply CNOT to superpositions! Recall that $|+\rangle$ is equal to $1/\sqrt{2}\,|0\rangle + 1/\sqrt{2}\,|1\rangle$.

$$CNOT\,|+\rangle\,|0\rangle = CNOT(1/\sqrt{2}\,|00\rangle + 1/\sqrt{2}\,|10\rangle) = 1/\sqrt{2}\,|00\rangle + 1/\sqrt{2}\,|11\rangle$$

And now we've produced an entangled state! Hooray! We can also draw CNOT gates in our circuit diagram like so:



Here's an important note: Even though $|x\rangle$ is the 'control' qubit and $|y\rangle$ is the 'target' qubit, $|x\rangle$ is changed by the CNOT gate. This is known as 'phase kickback' and we will see more examples of it in the future.

Phase Kickback

Phase kickback is not intuitive at all, so here's a basic example of the phenomenon:

$$CNOT(|+\rangle|-\rangle) = CNOT(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle))$$

$$= CNOT(\frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle)$$

$$= \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|11\rangle - \frac{1}{2}|10\rangle$$

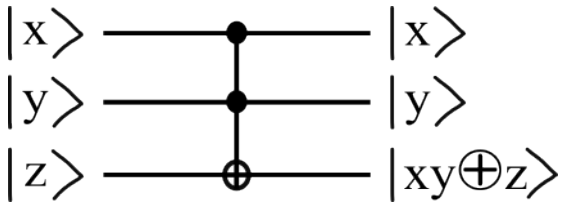
$$= \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle - \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$$

$$= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$= |-\rangle|-\rangle$$

So our CNOT ended up changing our control qubit, but left the target qubit unchanged. How very strange things are in quantum land!

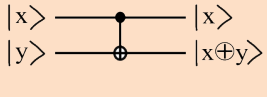
There are also other multi-qubit gates we may want to implement. One oft discussed one is the Toffoli gate, which uses three qubits. People often use Toffolis as short hand for how complex a circuit is – as in "Yeah we can implement that with a thousand qubits and a million Toffolis". A Toffoli gate is closely related to CNOT and can in fact just be thought of as a CCNOT, or controlled CNOT: $CCNOT|110\rangle = |111\rangle$. A Toffoli has two control qubits and one target qubit. The target qubit's state is only flipped if both control qubits are $|1\rangle$ (or, as we discussed before, if they are superpositions of $|1\rangle$). The circuit diagram for a Toffoli is below.



Common Multi-Qubit Gates

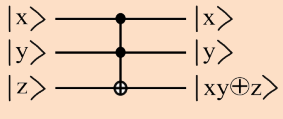
Here's a table of common multi-qubit gates and their associated matrices. Note that any gate can be turned into a controlled gate, here denoted by the 'CU' gate, where U represents any arbitrary 1-qubit gate.

CNOT



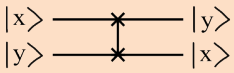
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Toffoli



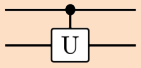
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

SWAP



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

CU



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix}$$

As previously mentioned, depending on the physical implementation of the qubits, some of these gates may be easier or harder to achieve in practice. As a result, it is necessary to ask which of these gates are necessary for quantum computation. A set of gates which can perform all quantum computations is known as a 'universal gate set'. In classical computing, the NAND gate is a universal gate set all by itself, hence why digital logic is implemented with NAND gates. In quantum computing, we are not so lucky to have a single universal gate. A universal gate set must be able to A) create superposition; B) create entanglement; C) reach complex amplitudes*. One such gate set is CNOT, H, T. We shall return to these concepts later in Chapter 2.

It may not be immediately obvious what all this entanglement and qubit business actually enables. One simple and close-enough-to-true way to look at it is by thinking about how much information is embedded inside a multi-bit versus multi-qubit system. In a three-bit system, we can encode $2^3 = 8$ different values. We counted those states out above. In a three-qubit system, we describe the state of the system with 8 complex amplitudes, like so:

∗. Note that these are necessary but not sufficient conditions. The Clifford group CNOT, H, S meets these criteria but is not universal. This actually becomes very critical when one tries to implement surface code-based error correction, which has a very easy time implementing Clifford gates, but has to rely on complex 'magic state distillation' tactics to create T gates to accomplish universal quantum computing. More on this later.

$$|\psi\rangle = a\,|000\rangle + b\,|001\rangle + c\,|010\rangle + d\,|011\rangle + e\,|100\rangle + f\,|101\rangle + g\,|110\rangle + h\,|111\rangle$$

Let me emphasize this: each of these amplitudes (a through h) are complex numbers that would require several bytes to properly represent (potentially many bytes depending on how precise our quantum computation needs to be). The number of these amplitudes scales like 2^N with N qubits. That becomes very large, very fast. Note that we cannot arbitrarily access the values of those amplitudes, we can only selectively measure the qubits and collapse the system into a particular state. Nevertheless, we can do intermediate operations that take advantage of the large number of these amplitudes, as well as their ability to constructively and destructively interfere, to perform computations. This is the powerful and unique thing about qubits and quantum computation.

Pure vs. Mixed States

Here's a very sophisticated piece of quantum terminology (using it will surely give you lots of quantum street cred). All the quantum states we've discussed so far are known as 'pure states'. A pure state is one where we are certain of the quantum state the system is in. It might be a superposition, and measurement outcomes are probabilistic, but the state is known. However, sometimes systems are in unknown or 'less known' quantum states. Perhaps we think there's a 50% chance of a system being in $|\psi\rangle$ and a 50% chance of being in $|\chi\rangle$. We'd characterize this as being in a mixed state.

Pure states are represented by points on the Bloch sphere, as we've seen. Mixed states are represented by points inside of the Bloch sphere. The closer a point is to the surface, the more certain we are of the state – i.e. the 'purer' it is.

Importantly, mixed states obey the laws of classical probability, not quantum laws. We can accurately say that the system has X% probability of being in a state, without worrying about complex numbers or amplitudes.

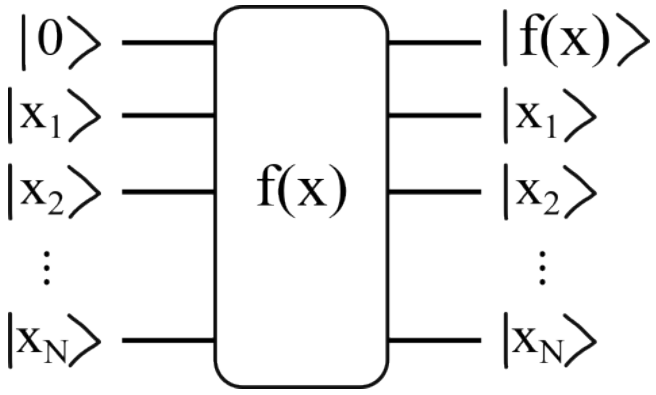
We're now going to look at three different quantum algorithms, in increasing order of complexity. Unfortunately, even the first and simplest quantum algorithm is still non-trivial to understand (so please don't feel disheartened if you are confused!). The point of looking at these sections is not to beat an understanding of quantum computer science theory into you, but to help you understand the general approaches and unique characteristics of quantum algorithms. So while our examples will grow more complicated as we progress, our analysis will also begin to focus more on high-level takeaways than quantum circuits. Depending on your level of interest, you may want to skim or skip this section, although I do think that understanding generally what a quantum algorithm is can be helpful

for thinking about a quantum computer more broadly. On the other hand, if you find yourself actively wishing for more rigor and math, I suggest looking at Thomas Wong's excellent "Introduction to Classical and Quantum Computing."

2.5 Quantum Advantage: Deutsch-Jozsa Algorithm

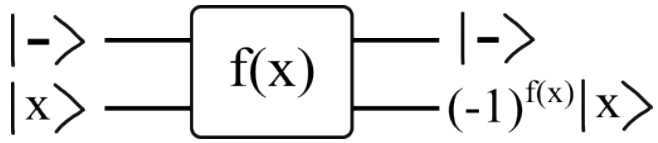
Let us examine a simple algorithm that provably improves on what a classical computer can do. But before we do that, we need to get one last piece of preamble out of the way. Those pesky theoretical computer scientists, never happy to make things too simple, often use 'oracles' in their problems and algorithms. An oracle is a fancy way of saying a black boxed function that the algorithm in question has access to. The algorithm can call on the oracle as needed and will get back the oracle's answer. We never worry about the implementation of the oracle, and simply count how many times we need to call upon it. This oracle may not be possible to implement in the real world – or it may take longer computational time than the rest of the algorithm! Nevertheless, the oracle is a helpful tool in the toolbox for theorists to begin their work. Shor's factoring algorithm began as an oracle implementation, then was fully implemented sans-oracle. In the next two examples, we will use oracles. Both the classical and quantum algorithms will have access to the oracle, but as we will see, the quantum algorithm can successfully solve the problem with fewer calls to that function. This is the most rudimentary kind of distinction between computational classes, known as an oracle separation.

In quantum land, we can both query and receive qubits in superposition. We can represent the oracle in our quantum circuit notation as a multi-qubit gate, like below:



Note that the oracle has n input qubits, and an answer qubit. In standard form, the answer qubit is changed by the oracle's internal logic, and the input qubit is not. However, for the algorithm we are about to look at, we need to add yet another complication – a phase oracle. This is closely related to the concept

of phase kickback, which we discussed above in the context of the CNOT gate. The idea here is that by putting the answer qubit in a particular state ($|-\rangle$) prior to feeding it to the oracle, the oracle operates differently: it instead changes the phase of the input qubits (leaving the answer qubit in that $|-\rangle$ state). There are some mathematics that make this work , but for the purposes of this text, it's simply another way of defining the black box function, so we will gloss over it. See the below diagram for an example of a phase oracle.



Now, after much deliberation, we can begin looking at the Deutsch-Jozsa algorithm[17] – baby's first quantum advantage. The problem is contrived and the speedup is not particularly impressive, but it exists nonetheless. The problem is this: You are given an oracle (we will call it $f(x)$), that takes in a bitstring of length n, and outputs either 1 or 0. We know that $f(x)$ has a special property; it is either 'constant' or 'balanced'. If it is constant, no matter what the input is, it will always output the same value (but we don't know if that value will be always 0 or always 1). If it is balanced, it will output 0 for exactly half of all input bitstrings, and 1 for the other half. Given an unknown $f(x)$ via an oracle, we would like to find out if it is constant or balanced.

Let's begin by figuring out where classical computers stand on this problem. Let's say we're working with $n = 4$, i.e. bitstrings of length 4 or $2^4 = 16$ possible inputs. We can start querying our oracle, input by input. Say $f(0000)$ returns 0. Then we query $f(0001)$. If it equals 1, we know immediately that the function is balanced (since we've gotten different results on our two queries, it can't be constant). But if $f(0001) = 0$, then we're not sure! Maybe it's constant, or maybe it's balanced and we got unlucky. We can keep querying the oracle with incrementally higher bitstrings until we either get a 1, or we've queried more than half of the input bitstrings. If we've gotten all the way up to 1001 (binary 9) and every result is 0, we know that the function cannot be balanced (since at least half of the outputs of a balanced function must be 1).

So in the classical case, we will need to query the oracle at most $(n/2 + 1)$ times, although we can sometimes do much better than that. If we're willing to accept some error, we can query the oracle fewer times, and expect to be right often enough with some likelihood. If I ask the oracle 500 times and it comes out 0 every time, I'm pretty sure the function is constant, even if n is big!

What can a quantum computer do with this oracle? It turns out that we can get the answer in exactly one query of the oracle. The circuit to do so is shown in Fig. 2.6

Three preliminary comments:

1. As noted above, this is a phase oracle, so the input qubits will get a phase rotation depending on what the oracle's response is. Generally, for an input

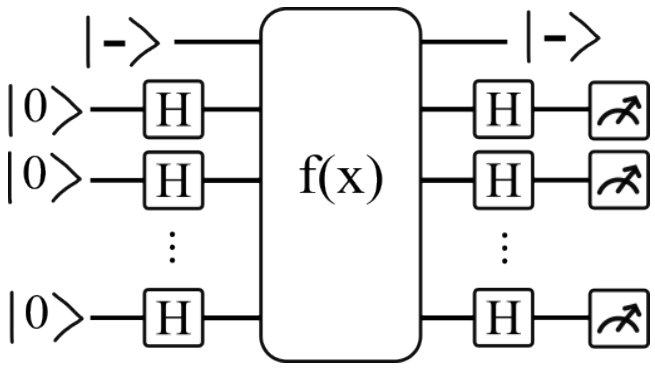


Figure 2.6: The quantum circuit for an implementation of the Deutch-Jozsa algorithm.

bitstring in state $|x\rangle$, the output will look like $(-1)^{f(x)} * |0\rangle$. That is, a 180 degree phase will be applied to x if the oracle returns 1.

2. There's a new and unfamiliar block in the diagram, at the far right hand side of most of the qubits. That's the symbol for measurement, and simply means we will measure each qubit (in the computational basis) and get $|0\rangle$ or $|1\rangle$ as possible results.

3. You'll notice the input qubits all go through a Hadamard gate before being fed into the oracle. Recall that a Hadamard gate moves qubits from the $|0\rangle$ to the $|+\rangle$ state. Recall further that a $|+\rangle$ state is an equal superposition between the $|0\rangle$ and $|1\rangle$ states. One way to think about this operation is that we are putting the set of n input qubits into an equal superposition of all possible bitstrings. If we were to measure the qubits after the first Hadamards but before the oracle, we have an equal chance of measuring every bitstring from $|00..00\rangle$ to $|11..11\rangle$. So we are thus applying the oracle to this state which represents the uniform superposition of all the input bitstrings.

I'm going to abstract away a lot of the details of the math in service of clarity, but we can express the state of the system before measurement as:

$$\frac{1}{\sqrt{2^N}} \sum_{x \in \{0,1\}^N} (-1)^{f(x)} H^{\otimes n} |x\rangle$$

Where x is an individual N-length bitstring and $H^{\otimes n}$ is the Hadamard gate being applied to each of the n qubits. Expanding that second term over all the n turns out to be non-trivial and produces an unwieldy expression. However, we can express the amplitude of the $|00..00\rangle$ state fairly simply:

$$\alpha |00..00\rangle = \frac{1}{2^N} \sum_{x \in \{0,1\}^N} (-1)^{f(x)} |00..00\rangle$$

Remember that α, the amplitude of this state, represents the square root of the probability of measuring this state. Let's evaluate this if $f(x)$ is constant: The argument of the summation always evaluates to 1 or -1 and there are 2^n different terms to evaluate, so the summation ends up being 2^n or -2^n. α is thus 1 or -1, and the probability of measuring the all-zero state is 1! This is an example of constructive interference, where amplitudes end up summing to increase the probability of a specific result.

If $f(x)$ is balanced, $f(x)$ will output 1 half the time and -1 the other half of the time, so the summation will evaluate to 0. The probability of measuring the all-zero state is thus 0. This is an example of destructive interference, where amplitudes end up canceling out to decrease the resulting probability of a state.

Operationally, the measurement occurs, and we can make an evaluation: If $|00..00\rangle$ is measured, we know that $f(x)$ is constant. If any other state is measured, $f(x)$ must be balanced. We've solved the problem in exactly 1 query to the oracle, substantially better than the classical case.

A few comments here:

- This is a very instructive demonstration of how quantum computing actually works. The popular media explanation of the field is: "Quantum computers will evaluate all possible inputs at once, then somehow report back the answer for a particular input". This is not at all how this algorithm works. We put all possible inputs into the oracle via superposition, but rely on constructive and destructive interference to get the answer to the defined problem.

- Deutsch-Jozsa relies on an extremely specific problem statement with quite severe restrictions. This is what enables us to get the wonderful constructive/destructive interference results, and our $O(1)$ query complexity. Quantum computing theorists talk about the concept of 'structure,' which Deutsch-Jozsa is artificially designed to have. Put differently, certain problems simply have characteristics that make them amenable to being solved quantumly. Most people in the field think that for there to be a useful quantum speedup for a problem, there has to be at least some of this 'structure'.

- Our quantum speedup is from a linear ($O(n)$) to a constant ($O(1)$) query complexity. This isn't that good, and practically not enough to result in realistic quantum advantage for this contrived problem. Moreover, while the worst case bound on the classical case is $(n/2 + 1)$ queries, if one is willing to accept a risk of error, one can get very high probability chance of success with even relatively small ($\sim$10 queries) amounts of classical queries, even for large values of n. Quantum algorithm designers are 'at war' with the classical algorithms – and the classical algorithms can fight back!

- We haven't talked about how to implement the oracle. While the oracle is relatively simple in this case, it need not be. More on this when we discuss our next algorithm.

Models of Quantum Computation

Everything so far in this work has described computation through a lens of gate-based operations. The story I'm telling you is simple: start your qubits in an all $|0\rangle$ state, subject them to a bunch of gates, then measure at the end. This is often called the circuit-based model for quantum computing. This is not the only way to think about or implement quantum computation!

One well-known, but markedly different, model is adiabatic computation. Note first that the adiabatic model and the circuit (gate-based) model are proven to be mathematically identical[18]. That is to say, anything you can do with gates, you can do adiabatically. Adiabatic computation is sort of an analog, more 'physics-flavored' way of implementing computations. The idea is that the qubits are prepared in a sort of initial state, and we are going to subject them to a time varying set of external fields (the physicists would call this a time-varying Hamiltonian). The ground state, or lowest energy state, of the initial Hamiltonian is equivalent to our initial prepared state. The ground state of the final Hamiltonian should be equivalent to the answer of the problem we want to solve. In the middle, the computer has to somehow interpolate between those two, perhaps linearly, although not necessarily.

This method might be a more natural and easier way of implementing certain algorithms or performing certain quantum physics simulations. In terms of commercial and academic interest, adiabatic methods seem to lag behind circuit-based computation. A notable exception is D-Wave Systems, which attempted to commercialize a technology that implemented 'quantum annealing'. Quantum annealing was supposed to be a useful sub-set of a full adiabatic method, but appears to have produced no real effective results. D-Wave has recently acquired a superconducting qubit company, signaling that they, like most everyone else, are probably more thinking about gates and circuits than adiabatic computation.

Another notable alternative is 'measurement-based' quantum computation. Instead of subjecting qubits to a series of sequential operations, measurement-based computation generates what are called 'cluster' or 'resource' states, which are a large system of entangled qubits. The qubits are then progressively measured in specific ways that implement computation. The outcomes of initial measurements determine which qubits will be measured next, and how. This is also referred to as 'one-way computation', since the measurements cannot be reversed. This sort of approach is actually a very natural way of implementing computation for hardware that uses photons, for reasons we will discuss later.

Models of Quantum Computation (continued)

In any case, it is helpful pedagogically to think about things in the circuit/gate-based model. It simply is easier to understand. Even the strange measurement based implementations typically start by taking an algorithm in circuit form, then compiling it to their measurement scheme.

2.6 Useful (?) Quantum Advantage: Grover's Algorithm

We've now learned about a quantum algorithm that improves on any classical approach to the same problem. However, Deutsch-Jozsa's algorithm applies to a totally trivial problem. We are now going to look at an algorithm with potentially broad applicability to a variety of tasks: Grover's algorithm.

Grover's algorithm was developed in 1996 by Lov Grover[19]. Interestingly, Grover proposed this an algorithm for 'database search' problems. Unfortunately, the physical constraints of quantum computing make this a strange and impractical application of the approach. Grover's is practically better suited for problems where the answer can be represented as the solution to an equation (this includes optimization problems), that is then made into the oracle.

Grover's algorithm is a good deal more complex than Deutsch-Jozsa, and accordingly we will operate at a higher level of abstraction – thinking about operations rather than individual gates and states.

Let's define the problem: We have a function $f(x)$, that takes in an n-bit value, and returns either 0 or 1. The function (or oracle) returns 1 for a particular value of x, and 0 otherwise. You can (as Grover did) conceive of this as a database search program – e.g. I have a phone number and I want to find the single owner of that phone number; or instead as some kind of math problem with a single optimal answer or condition that can be assessed by $f(x)$. In this way, Grover's can be applied to extremely general problems if formulated correctly.

What's the classical solution? Well, if we take the phone number example, we have to search through the phone book, line-by-line, until we find the match. This is called brute force searching, and is quite computationally expensive. For phone numbers of n-bits, this requires searching through on the order of 2^n entries. Thus, we would say it scales exponentially with n. To make notation simpler, we will call the total number of possible entries N (which is equal to 2^n). Note that for many problems, you can do better than this brute force search approach!

The quantum approach can do better than that. Before getting into the details, I'd like to introduce a very helpful mental model for thinking about this problem. Let's say we have n qubits. At the end of our computation, we will measure those qubits and they will be in one of $N = 2^n$ states, just like how n bits can represent 2^n different values. In the middle of computation, the qubits will

be in some intermediate state, which can be expressed as a linear combination of several or all of those N states. We can represent this intermediate state via a N dimensional vector, as shown below for a 4-qubit example:

$$\begin{bmatrix} \sqrt{P(|0000\rangle)} \\ \sqrt{P(|0010\rangle)} \\ \sqrt{P(|0011\rangle)} \\ \vdots \\ \sqrt{P(|1111\rangle)} \end{bmatrix}$$

Unfortunately, it's difficult to visualize that vector. If it were 2- or 3-dimensions, we could graph it, but as a species living in 3-dimensions, it's pretty tricky for us humans to imagine $2^4 = 16$-dimensional vector spaces in our mind. What we can do though, is compress this higher-dimensional vector space into 2 dimensions. Remember that every single dimension, or row, of this vector simply corresponds to a different output state of the set of qubits. When we look at the value in the vector in a particular dimension, we are just evaluating the probability of the qubits being measured in that state. Thus, the two effective dimensions we will pick (for the purpose of understanding Grover's algorithm) are: 1) the dimension that corresponds to the correct solution to our search problem; 2) all the other ones combined – i.e. the wrong answers. This will hopefully become clearer in a moment.

We're going to begin in the same way we did in Deutsch-Jozsa: Feeding n $|0\rangle$ state initialized qubits into a set of Hadamard gates. As we noted before, the qubits are now in an equal superposition between the $N = 2^n$ possible output bitstrings. How should this look in our 2-D representation? Well, if we were to measure the qubits now, our probability of getting the correct answer would be 1/N – it's totally random chance. Thus, there will only be some small component of the vector space that points toward the correct answer, and the dimension that contains every other answer will be much larger. We draw this in Fig. 2.7. $|t\rangle$ represents the 'noT right' answers; $|r\rangle$ the 'Right' answer; and $|e\rangle$ the 'Equal everything' vector.

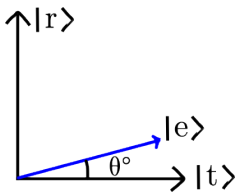


Figure 2.7: The first step of Grover's algorithm, where the system is initialized to be an even superposition of all possible output bitstrings.

Note that theta – the angle that $|e\rangle$ deviates from $|t\rangle$ - will be smaller the larger n is. The next step is to feed our qubits into our oracle. Just like in Deutsch-Jozsa, we're going to use a phase oracle. The sign of the correct answer

will be inverted, but otherwise the qubits will stay in the same state. This is equivalent to reflecting our qubits around the 'not-right' axis. This is easy to visualize, and shown in Fig. 2.8.

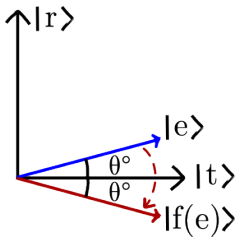


Figure 2.8: The second step of Grover's algorithm, where the qubits are fed into the oracle, and reflected around the 'not-right' axis.

Here's the most conceptually tricky step: We need to reflect our qubits around that initial, equal superposition state. I'm not going to explain how to do this. You really don't care do you? Just know: We can do this. If you really care, read a textbook*. Now what state are our qubits in? Look at Fig. 2.9.

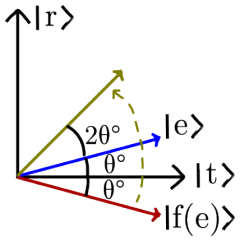


Figure 2.9: The third step of Grover's algorithm, wherein gates are applied to reflect the system over the initial, equally superimposed state vector.

Notice that we are now closer to the 'right answer' axis! If we measured the qubits now, we would be more likely to measure the correct answer than when we started. Wow! Unfortunately, that probability still could be small, depending on how big n is (and thus, how small θ is). Our solution is to simply repeat this process again and again: First we use the oracle and reflect across 'not-the-answer', then reflect across the 'equal-superposition' state to get closer to pointing toward 'the-answer'. To get the vector pointing towards 'the-answer' we repeat this process $\sqrt{N}$ times, as shown in Fig. 2.10. We can then measure the qubits and get the correct answer with very high probability! We've beaten the classical approach by a factor of $\sqrt{N}$!

This sqrt(N) speedup is often called a 'quadratic' or 'Grover' speedup and is commonly found in many quantum algorithms. In some ways, this is excellent:

*. Once again, see a very lucid explanation in Wong's "Introduction to Classical and Quantum Computing."

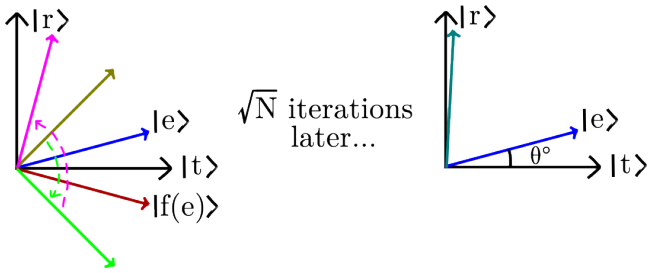


Figure 2.10: The ensuing process of Grover's algorithm, wherein steps 2 and 3 are repeated $\sqrt{(N)}$ times to bring the system closer to the correct output bitstring.

For large values of n, this might be a huge reduction in oracle queries necessary to find the solution. It's also just sort of magical that via superposition and quantum-ness, we can somehow parse through 2^n possible entries in way fewer queries. But in other ways, it is not so exciting: Problems that take exponential time ('NP Hard' problems) still take exponential time, because $O(2^n/\sqrt{2^n})$ is still exponential complexity. As discussed previously, physical implementations of quantum computers are expected to run at slower clock frequencies than classical computers, so a quadratic quantum speedup may still be slower than a classical approach. And importantly, it might be really hard to implement that oracle! This is the fatal flaw with using Grover's for database search. We need an oracle that somehow 'knows' which answer is correct, despite that being the problem we are trying to solve! In classical computing, that oracle would be implemented via querying some sort of structured memory where our database lives. Unfortunately, implementing a 'quantum memory' (often referred to as a quantum RAM, for random access memory), where we can query lots of data in superposition, is extremely difficult, and perhaps impossible,* making Grover's even more difficult to implement.

Where does this leave us? Grover's is extremely cool and a useful tool in the proverbial toolbox, but unlikely to be the foundation of quantum's 'killer app'. Onwards we go...

∗. Why? Well, we're sort of starting from zero every time we run a quantum algorithm. As we'll learn, qubits are constantly fighting against 'decoherence' and it's very hard to keep them in arbitrary states during the course of running a single algorithm, let alone for a longer period of time. So it's generally not possible to assume access to stored, quantum-accessible data. The time it takes us to load all that data into the quantum computer may end up destroying our hard-earned algorithmic advantage.

No Cloning

I've mentioned above that it's difficult, perhaps impossible, to make a quantum memory. One reason for this is the 'no-cloning theorem'[20]. We'd like to make a copy of some data to put into our theoretical memory, like so:

$$|\psi\rangle |0\rangle \rightarrow |\psi\rangle |\psi\rangle$$

It turns out, that for unknown $|\psi\rangle$, this is not possible! There is no generic quantum gate that enables us to duplicate our qubits. This makes certain algorithms more difficult to implement, and makes error correction MUCH harder (more on that later). More generally, it challenges our ability to think of quantum computers as simply quantum versions of regular computers. It is so critical for a normal CPU's operation to reach into memory to pull out certain values, or to put results into memory. This is not a trivial operation in the quantum world.

2.7 Useful Quantum Advantage: Shor's Algorithm

Now we turn our attention to Shor's algorithm – the most famous (and potentially most useful) quantum algorithm. Invented by Peter Shor in 1993[21], Shor's algorithm is a method of factoring large numbers. Factoring is a very important problem, since most modern encryption schemes rely on factoring being hard in order to guarantee security. RSA (named after its inventors, Rivest, Shamir, & Adleman), one of the most widely used encryption schemes, uses two large prime numbers to generate both a public and private key. If one could factorize the public key, it would be trivial to generate the private key and thus be able to decrypt the secret data.

Fortunately, factoring is classically difficult. The best-known classical algorithm, called the general number field sieve, scales with approximately $O(e^{n^{1/3}})$, which is sub-exponential but not efficient. Thus, our cryptographic schemes are currently secure! But quantum computers should be able to do better, utilizing Shor's algorithm. We will again examine this algorithm with a bit more abstraction, looking at the steps taken rather than the individual gates. Shor's algorithm takes a number N which is the product of two large primes, p and q. N is known, but we would like to find p and q.

1. We pick a number a in between 1 and N. We check if it is a factor or a multiple of a factor of N (i.e. did we luck into picking the right answer). If it is, we succeeded! If it isn't, we move onto step 2. Note that this is a classical computation, not a quantum one.

2. This is the fantastic quantum-accelerated step. We perform an operation called 'period finding' using a and N. The operation a^x modulo N has a property where as values of x are stepped through, the modulus will

end up cycling through a set of values. We can find out the length of the cycle, known as a period, with a period finding algorithm. This operation, performed quantumly, has complexity $O(n^3)$, which is polynomial and thus efficient! It produces the period, r.

3. We now have a and r. It turns out that $a^{(r/2)} + 1$ and $a^{(r/2)} - 1$ have a wonderful property, which is that they each share a common factor with N. Thus, we just figure out the greatest common denominator between those terms and N, and we have p and q respectively. Again, note that this is a classical step.

Shor's thus can factor large numbers in polynomial time, significantly better than any known classical approach. This obviously has large potential impacts on digital information security, but we will reserve discussing those implications for Section III: Applications.

I do want to note a few things specifically about Shor's as an algorithm here:

- Shor's is, to our knowledge, the only quantum algorithm that is practically useful and has a provable exponential speedup over known classical approaches. It is, without exaggeration, the bedrock which the field's current level of funding and interest lies upon. As such, there is considerable interest in using Shor's as a sort of benchmark – i.e. how large of a number can you factor on your quantum computer using Shor's? Unfortunately, factoring even small numbers requires a large amount of quantum computational resources: This excellent Craig Gidney (a prolific researcher at Google Quantum AI*) article estimates that factoring the number 21 requires approximately 2,400 multi-qubit gates. That's a lot! As such, there's a small cottage industry of researchers and institutions using numerical tricks to 'factor' large numbers 'quantumly'. The only upside of this is the delightful tiny cottage industry of researchers debunking or skewering these factorizations, which reached its apex with: "Replication of Quantum Factorisation Records with an 8-bit Home Computer, an Abacus, and a Dog."[22]

- It is explicitly a classical/quantum hybrid algorithm. It specifically uses the quantum computer to perform a subroutine which is (relatively) easy quantumly, but hard classically. In general, we want to do as little as possible on the quantum side, leveraging our classical infrastructure as much as possible. This will be true for many of the potential applications of quantum computing we will discuss later in this work.

- Remember how we discussed that the 'structure' of the problem poised in Deutsch-Jozsa was what enabled the quantum speedup? In the same way, Shor's exploits the 'structure' of the problem. Factoring large numbers (and specifically period finding) has some specific relation to the 'hidden

∗. I've always wondered whether the researchers in Google's quantum division resent the ridiculous name of their group.

subgroup problem' for finite Abelian groups* It's plausible that other problems may relate to this hidden subgroup concept in some way, and thus a Shor's-level speedup may be accessible in such cases. Unfortunately, those problems haven't yet been found.

2.8 Commentary

If you've followed thus far, you now understand how quantum computation differs from classical computation and how it can speed up the calculation of certain problems. This is roughly what you would get from an elementary quantum information and computation textbook, albeit with much less rigor. Perhaps it answers a base curiosity about what quantum computation is. Perhaps it sparks even more questions, about how these mathematical objects can be implemented in real life, and if those quantum algorithms can perform useful calculations outside of academic proposals. If so, please read on.

∗. An Abelian group is simply a set of numbers and an operation such that the operation is associative, commutative, and inversible. The hidden subgroup problem is a bit more esoteric and beyond my mathematical intuition.

Chapter 3

Implementation

In this section, we will discuss the gory details that make quantum computing difficult to implement physically. We will then examine some of the many different architectures that are being developed for future quantum computers and compare their relative strengths and weaknesses.

A warning: Moreso than the previous section, the information contained here may go out of date. It's possible that some of these technical approaches will become dominant, and others will lead to intellectual dead-ends. Progress in these fields is actively occurring, and in some cases, moving rather fast. As such, while I will describe the current state of the field, I will focus on the basics – e.g. how does a superconducting transmon qubit work, rather than the details of Google's Willow superconducting quantum processor.

3.1 Qubits in Practice, Not Theory

Recall our discussion from chapter 2 regarding classical bits: We currently implement these bits as voltage levels, stored in memory and processed by logic gates implemented by transistors in silicon. This is a choice informed by decades of semiconductor engineering, but it is equally possible (however much less scalable) to implement classical bit-based computation using light switches, tanks of water, or cans of soup. Anything where we can encode two different levels (logical 1 and logical 0) can be used as a bit. The trouble of course, is interacting with those bits in a scalable way: we've settled on transistors because we can create billions of them and easily control them.

Qubits are a bit more... finicky. Our qubits need several unique properties: superposition & entanglement. Unfortunately for quantum computer engineers, most things around us don't seem to obey quantum properties on the macroscale. I can't put my light switch in a superposition! So we have to look for specific qubit 'platforms' which have a quantum nature. This naturally leads us to several of the most promising qubit technologies: photons, atoms, & ions. The last of the big four implementations is superconducting qubits, which exist on a

larger, non-atomic scale, but utilize the properties of supercooled semiconductors to access quantum effects.

In each of these cases, we have to encode the state of the qubit ($|0\rangle$ or $|1\rangle$) in some specific physical property of the qubit. This could be the spin of the photon, the frequency of some vibration, or the energy of the system. This varies in each of the platforms, as does the mechanism of interacting with the system (implementing gates and measurement).

3.2 Fragile Qubits

Real-world qubits are inherently fragile. While I can turn my laptop off for several months and expect to come back to the system in the same state with my data uncorrupted, it's a tall order for a qubit to last seconds in the same state. We know definitionally that quantum states can be affected (collapsed) by outside measurement. It's not a very far leap to say that those quantum states can be similarly affected by any interaction with the outside world*. We can spend a lot of time precisely engineering the qubit system and still have our qubits end up in different, unknown states. This is called 'decoherence', and a critical metric for any quantum computer is how long a qubit can last before decoherence occurs. You'll often actually see two relevant metrics for decoherence times, known as T_1 and T_2. T_1 is the time a qubit can stay in the $|1\rangle$ state, before it drops back down into the lower-energy $|0\rangle$ state. T_2 relates to the phase coherence of a state – i.e. how long can it last in a specific phase before that information is lost. Clearly, engineers would like to maximize both values to create stable, long-lived qubits.

This also explains why lots of quantum computers need to be supercooled or placed into a vacuum: They want to avoid heat and interference from their environment! There's a popular misconception that the massive, upside-down steel cylinders shown in press photographs and Fig. 3.1 are computers themselves, when they are just dilution refrigerators, specifically used for cooling superconducting quantum computers to tens of milli-Kelvin. These fridges have concentric inner partitions that progressively cool the system down to the working temperature. In other qubit hardware implementations, that temperature varies, but it is typical for the system to require some level of cooling, vacuum, or isolation from the environment.

So even with careful engineering, noise will occur. Our wonderful little qubits will begin to decohere, and the state that they we think they are in may not be the state they are actually in. And here, we get to potentially the most vexing, complicated, and misunderstood topic in all of quantum computation: error correction.

∗. Note that I'm speaking in generalities here, and things can actually be quite different depending on the implementation of the qubit. In general, qubits are much more sensitive to decoherence and noise than standard bits, but it is a spectrum, with superconducting qubits on the 'most-sensitive' end.

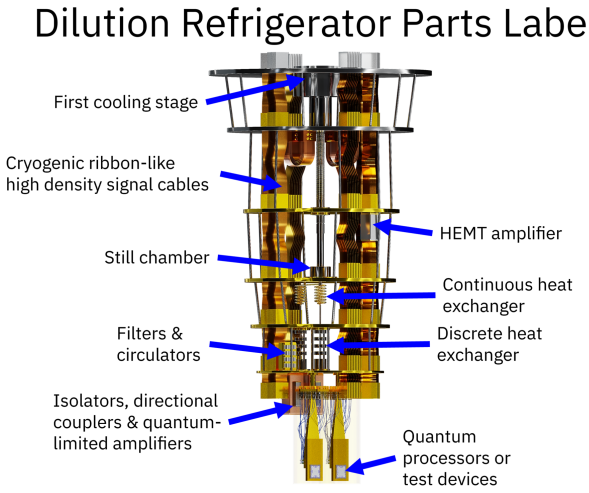


Figure 3.1: A labeled diagram of a dilution refrigerator. The first cooling stage at the top would exist somewhere around 30-50K, while the superconducting qubits would sit at the bottom at 10s of millikelvin. Reproduced from Onri Jay Benally on Wikicommons.

Qudits

We're about to start talking about real world systems that we use to implement the theoretical concepts we call qubits. In each case, there is some property of the system that exhibits quantum behavior as it transitions between some set of states. In some cases, it may be that the quantum system can exist in more than two states. In those cases, one state will be assigned to $|0\rangle$, and another to $|1\rangle$. Those other states are typically considered undesirable leakage states. However, there is some interest in actually using those other states for computation.

The other states are labeled in bra-ket notation, and increment up from $|1\rangle$: $|2\rangle$, $|3\rangle$, etc. These systems are called 'qudits' (notice the D), after systems with 'D' levels. For this work, we won't consider any qudit-based computation schemes, and all states greater than $|1\rangle$ are unwanted errors!

3.3 Error Correction Cliff Notes

I apologize. I've billed this section as being about the practical concerns of quantum computer implementation, and you've probably thought we would be talking about engineering problems. Real fun hardware implementation questions, with cool photos of semiconductors and atoms and lasers and the like. We will get to

that, but first we need to do some more theory and math. You're welcome to skip this section, but just know that error correction knowledge is what gives you real credibility on the mean streets out there. All the kids will laugh at you when you admit you don't know what a qLDPC code is.

To save you embarrassment, I'm going to summarize all the important takeaways of error correction up front. In the next sub-section, we'll do some math and familiarize ourselves with a surface code, but these are the practical concepts you should have in your mental model of error correction.

- To stop errors from occurring and propagating, we need to implement error correction. Error correction exists at the intersection of physical engineering and software protocol. It consists of rapidly repeating gates then measurements that are supposed to both detect and suppress errors.

- Errors may be introduced from the decoherence of individual qubits, noise from the environment, or from gate operations. As we'll see, superconducting qubits are very sensitive, and require very aggressive error correction. Other qubit modalities, for example some implementations of trapped ions, have long coherence times and are more resistant to noise. Error correction is still necessary for those systems, because it's very difficult to guarantee the accuracy of any particular gate operation. Certain companies are betting they can get away with doing less error correction than competing approaches[23][24].

- Error correction is really tricky. In classical computing, we can just duplicate data. We can't do that in a quantum computer, because of the 'no-cloning' rule. So instead, we encode a single qubit's information across many qubits. This is why you'll sometimes hear people talk about 'physical qubits' versus 'logical qubits'. A logical qubit is comprised of many physical qubits, and we fundamentally care about logical qubits, because that's what we do operations on. The physical/logical ratio varies based on the technological regime we are working in, but use something like 1000:1 as a rough guideline. Considering that current physical qubit counts are in the low hundreds, we have a long way to go.

- There are lots of different implementations of error correction. You'll commonly see 'surface codes' and 'color codes'. These are advantageous because they can be natively implemented on a grid of qubits, which is what superconducting and neutral atom systems naturally look like. However, there's a lot of interest in 'low density parity check' (LDPC) codes. These codes can improve the physical/logical ratio, but can't be implemented directly on a grid. Rather, they require being able to make qubits that aren't immediately adjacent interact with each other. That's easier for certain implementations, like ions & neutral atoms, where qubits can be physically moved around.

- Remember that a logical qubit doesn't just look like X copies of the same physical qubit (because we literally could not do that with our current laws

of physics). Instead, the data is sort of 'broken apart' across the physical qubits. That means that interacting with that data is not necessarily trivial. Ideally, to apply a gate to a logical qubit, we would just apply that gate to all of its constituent physical qubits – this is called a 'transversal' gate. Unfortunately, for any error correcting code, there will always be some gate that the system needs to be able to perform (i.e. part of the universal gate set) that *cannot* be done transversally[25]. So no matter what, some gates will always be hard to implement on a logical qubit.

- For surface codes in particular, it's easy to perform Clifford gates (CNOT, H, S) on logical qubits, but full computation requires more than just Clifford gates. As a result, there's a lot of concern and literature on how to generate 'magic states', which implement T gates (which let you do non-Clifford gates, enabling universal quantum computation). Thus you'll see papers discussing how to do magic state 'distillation'[26] and 'cultivation'[27]. This is one of the big rate limiting steps for practically implementing lots of algorithms. Different code implementations have different strengths: For example, color codes may be denser and faster to perform computations on[28], and LDPC codes may require fewer physical qubits.

- Error correction leans on some very important theoretical results that say that if physical qubit error rates can be kept low enough, then continuous error correction should be able to guarantee successful computation. Perhaps the most important experimental result of the last decade is Google's paper that showed they were able to successfully decrease the logical qubit error rate below the physical qubit rate, and that increasing code distance (essentially size) further decreased that rate[29]. That single result alone validated an enormous amount of research on quantum error correction.

- Those theoretical results about error correction rely on some important assumptions that may not be practically true. It assumes:

  - That errors are limited to phase and bit flip errors. Think of these as essentially just errors that move the state of the qubits around on the Bloch sphere – just like single qubit gates. Unfortunately, many physical qubits struggle with something called 'leakage'. We discussed the principle of 'qudits' briefly above: in real-world qubit implementations, a system may have more than two states (e.g. it may have $|2\rangle$, $|3\rangle$, and so on). Leakage occurs when one of our qubits accidentally slips out of the bottom two energy levels into some higher state. Error correction can't handle these kinds of faults, so systems must be carefully engineered to prevent leakage.
  - Errors are independent and uncorrelated. This is troublesome. Lots of plausible error-causing mechanisms, like radiation coming in from the outside world[30], might create correlated errors in many qubits at a time. For quantum computation to last hours to days, these issues will have to be mitigated.

- When running these error correcting codes, there are data qubits (which store the actual data) and ancilla qubits (which give us information about the state of the data qubits). The code requires lots of repeated measurements of the ancilla qubits, which we then need to translate into a guess about the occurrence of errors in the system. There is not an obvious and easy way to 'decode' the measured information into error information. Even in the best case, there is uncertainty about what kind of error has (or hasn't occurred), and implementing this process at high speed may require heuristic or machine learning inspired methods. NVIDIA has just thrown their hat into this ring with their announcement of their 'Ising models' for decoding[31]. This is a huge problem and an active field of research.

- You may have noticed that most of my citations and examples here come from Google's quantum program. This is because they seem to be furthest along when it comes to practical implementation of error correction (and are certainly one of the teams most open about their results here). This isn't to say that superconducting qubits (Google's primary chosen modality) are the definitive best choice for future development, but rather that they are one of the farthest along in current development.

Phew. That was a lot of information! Let's take a breather and learn some error correction theory.

3.4 Error Correction 101

Practically speaking, in classical computation, bits don't fail that often. The most common rationale for a bit being corrupted (a 0 turning into a 1, or vice versa) is due to radiation. This is generally not a concern for most applications but it is an issue for electronics that operate in the upper atmosphere or space. Thus, we often see error correction & mitigation used in aerospace applications.

The simplest way of doing this is redundancy. If we want to store a value, we will instead store three copies of the same value. This is the simplest possible error correcting 'code'. A '1' becomes mapped to '111', and '0' becomes mapped to '000'. Note that in typical binary notation, 111 would mean 7, but we've changed its meaning here, because we're representing our data in a different way. It's very easy to see the original data we mean to store in this encoding, but that won't always be the case.

If an error occurs, what happens? Let's say we have a logical 1, which is encoded via the actual bit values '111'. A beam of radiation comes in and flips the middle bit. Now our bits look like '101', which doesn't have a clear logical value in our encoding. We use something called a 'syndrome' to figure out what logical value our computer should use. In this case, the syndrome essentially performs a majority vote – two of our bits say 1, and one says 0, so we should use 1. This was correct in this instance, but let's say a beam of radiation hits '111' and flips two bits, and we are left with '100'. Now our syndrome would report logical 0, the incorrect value. Any error correcting code is only resilient

to a certain amount of error. The strength of a code is expressed via its 'code distance'. A repetition code like this has distance 3, and can correct 1 error. If we encoded each logical bit using 5 normal bits ($0 \rightarrow 00000$; $1 \rightarrow 11111$), the code could correct two errors at a time.

Note that I've described this process as a 'majority vote' but this would actually be implemented with what's called a 'parity check'. A parity check simply checks if two adjacent bits are the same value. For our three bit code, we'd do two parity checks, between the first and second bit and between the second and third bit. If both parity checks are clean, there is no error detected. If either or both parity checks reports a difference, we can flip a bit to make them all the same parity. Something important to see is that in this scheme, the checks don't care about the actual value of the bits, just about their relative parity. This is important because in quantum-land, if we measure a qubit directly, we collapse its state. However, if we can implement our syndrome using a parity check function that indirectly compares the relative states of two qubits, we may be able to avoid direct measurement of those qubits.

Unfortunately, the errors that qubits experience are significantly harder to deal with than those of classical bits. A classical bit can flip from 0 to 1, or vice versa. A qubit can flip from $|0\rangle$ to $|1\rangle$, or to some intermediate value – we call these bit flip errors. Qubits can also have errors in their phase: $|+\rangle$ could flip to $|-\rangle$, or again to some intermediate degree. That would be a phase flip error. Thankfully, we can treat potentially intermediate analog errors (i.e. partial rotations on the Bloch sphere) as linear combinations of full phase or bit flip errors. As a result, we really can think just in terms of preventing discrete errors.

How might we begin to do this? Well, we could attempt to implement a repetition code for qubits. But there's a problem. Remember the no-cloning theorem? We can't just duplicate an arbitrary qubit state, i.e. turn $|\psi\rangle$ into $|\psi\rangle |\psi\rangle$. What we can do, is entangle a qubit with an ancilla qubit, and create some combined quantum state.

$$CNOT(|\psi\rangle |0\rangle) = CNOT((\alpha |0\rangle + \beta |1\rangle) |0\rangle) = \alpha |00\rangle + \beta |11\rangle = |\psi_L\rangle$$

Note that this is not just $|\psi\rangle |\psi\rangle$! This is an entangled state, not two independent qubits. This new system is our logical qubit, which we will denote with subscript L. We say this is the 'encoded' state.

Now let's say the qubits are at risk of a bit flip error (we can call this an X error, since it looks like applying an X gate). How do we figure out (and try to correct) that error? Well, we need to measure the qubits. But remember, measuring either of the constituent physical qubits directly would collapse the system's state, losing information. So what we do is akin to the parity check described above in the context of classical error correction. We take another qubit (an ancilla) and perform two conditional Z measurements on it, one for each of the two physical qubits in our logical qubit. These measurements are called the stabilizer. Why? Well, as we'll see, performing the stabilizer measurements actually reduces the probability of an error propagating through the system.

We then measure the ancilla, and the system (which consists of three entangled qubits) collapses to one of two states. That measured value is our syndrome, which we use to decode the error. If the error has a $p_x = a_x^2$ chance of occurring per qubit (and a $(1 - p_x) = k_1^2$ chance of not occurring, we can express the state of the system as:

$$(k_1^2 + k_x^2 X_1 X_2) |\psi_L\rangle |0\rangle + k_1 k_x (X_1 + X_2) |\psi_L\rangle |1\rangle$$

If the ancilla ends up in $|1\rangle$, that indicates that the qubits are not the same parity, and a bit flip error has occurred. If the ancilla is measured to be $|0\rangle$, then the two data qubits will collapse to being the same parity – this either means no error has occurred, or both qubits have experienced a bit flip (2 errors!). Notice that even in this very simple example, we still have uncertainty about the state of the system, and cannot cleanly map our syndrome to an error (or lack thereof). We can do the math to figure out the probability of the system being in that double bit flipped state. It turns out that conditional on measuring $|0\rangle$, there's only a p_x^2 chance of error remaining in the system. That means that this code, specifically the application of the stabilizer, does suppress errors in the system.

Let's identify two other practical issues with using this code: First, if we do detect an error, we can't really correct it! We don't know which of the two qubits was the one that experienced the bit flip error. This makes this more of an error detecting code than an error correcting code. Second, we can only check for bit flip errors, not phase flip errors. We can rectify both these issues by, you guessed it, adding more qubits.

There are many families of error correcting codes (we discussed color codes and LDPC codes in our quick highlights above), but the most mature are the surface codes. Surface codes have natural advantages in that they naturally can be implemented on a grid of qubits, requiring only nearest-neighbor interactions on a square grid. I'm not going to cover surface codes in depth, but we can look at a very high level example of how they are implemented.

Figure 3.2 shows a grid of qubits, alternating between data and ancilla in a checkerboard pattern. The logical qubit's data is broken apart across the states of all the data qubits, and the error code is implemented via stabilizer measurements (conditional Z and X's) on the ancilla qubits. Each ancilla checks against either bit or phase flip errors, and only gets one kind of measurement projected on it. The surface code is implemented in cycles, where each ancilla gets a conditional gate from each of its neighboring nodes, then a measurement occurs. There are a lot of funky implementation details (such as, what order to do these gates in), but this is the general schema. Once again, note that the measurements of the ancilla do not generally translate directly into concrete information about what errors have occurred, and there is computational effort that must be expended trying to 'decode' the errors.

We've now understood a few fundamental principles about quantum error correction. I've done a big summary above, but I want to emphasize a few points as we begin to discuss physical qubit implementations. Error is an inherent part of quantum computation, and successful quantum computers will design qubits

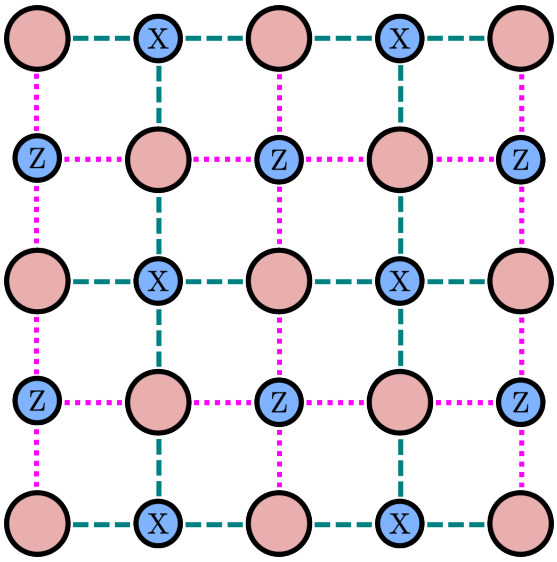


Figure 3.2: A small surface code patch, with data qubits in red and ancilla measurement qubits in blue. Ancilla qubits alternate between those that conduct X and Z measurements respectively.

to incur as few errors as possible in the first place, while implementing active error correction schemes which are continuously fixing the errors that do occur. This is a tremendously difficult problem, and informs nearly every aspect of qubit engineering and system design.

3.5 Quantum Skepticism

This feels like an appropriate time to acknowledge that some people simply don't think building a fault tolerant quantum computer is possible. I think there is a spectrum to the validity of these doubts, so I'd like to present a few different common critiques and explain their perspectives.

The most common and perhaps least sound skepticism is the common refrain 'Why haven't quantum computers factored 35 yet?', often dropped below any quantum computing-related internet post. The idea being, if quantum computers are so powerful, why haven't we seen evidence of their ability to do tasks that would be trivial for typical computers? We've already discussed the difficulty of implementing Shor's algorithm above; so it's no surprise that today's small scale and limited quantum computers are unable to factor large numbers. I find this argument incredibly unpersuasive: Everyone in the field knows that today's devices are simply not performant enough! If we had thousands of qubits and working error correction and still couldn't factor 35, that would be a different story. Craig Gidney argues that we might expect error correction performance to

follow an exponential trendline[32] - this would similarly imply nonlinear growth in actual computational capabilities. So I think this is a lousy and intellectually lazy tack!

An alternative critique says: "These quantum computers are simply too hard/impossible to build!" This paper is a very representative example[33]. On some level, it's difficult to engage with this line of argument, since there are any number of characteristics of a quantum computer which are incredibly difficult to implement (qubit count, control, gate fidelity, interconnections, error correction schemes, syndrome decoding, etc. etc.). Certainly there are those who thought the systems of today (100+ qubits implemented across several different kinds of physical platforms) would have been impossible. At the same time, the remaining challenges in scaling up to thousands and eventually millions of qubits seem daunting and perhaps insurmountable. There is no guarantee that these problems can be solved, but nor is there a guarantee that they cannot. So, I think this critique is potentially, but not necessarily, true. It most definitely is not a reason to abandon development!

Lastly, I want to highlight the perspective of Israeli mathematician Gil Kalai, who's often cited as one of the most prominent and vocal quantum computing skeptics. Gil's point of view[34] is essentially that there will be some level of correlated noise in quantum systems that error correction schemes will not be able to mitigate. Once again, this is difficult to prove or disprove. Existing error correction experiments have not yet hit this limit, but it is entirely possible, but perhaps not plausible, that they may one day, neutering efforts to reach full fault tolerant operation.

Personally, I find elements of these arguments compelling, and as a natural skeptic I am sympathetic to their claims. In particular, I think scaleup is an extremely difficult problem for just about every hardware platform being produced, one that may cull a great number of quantum hardware companies in the near future. I can't quite sit in the quantum computing isn't possible camp, though. As we are about to see, the field has come a long way in physically implementing qubits, and the influx of interest and funding could lead to even more rapid progress.

3.6 Physical Qubit Implementations

And now we arrive at my favorite section: Understanding how qubits actually get built. In the following sections, we will look at several leading architectures for qubit implementations. In each, we will attempt to answer a few important questions:

- How is a qubit encoded in this physical system?
- How is the state of a qubit (or qubits) changed – i.e. how do we perform 1 and 2 qubit gate operations?
- How do we measure our qubits?

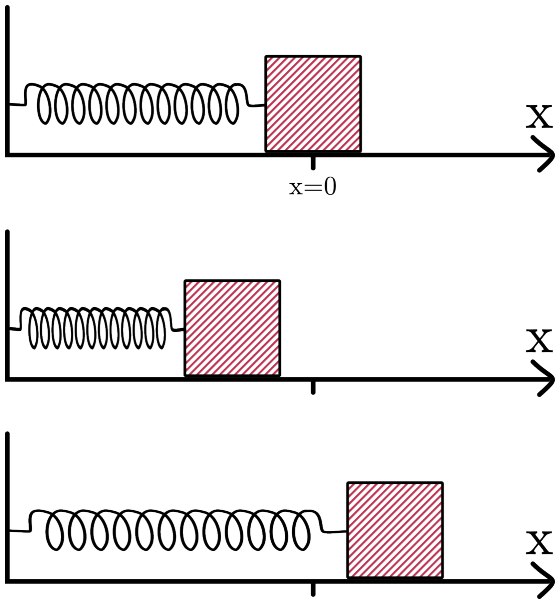


Figure 3.3: Diagram of mass (m) mounted to spring with position x and momentum p.

- What does the full system look like?
- What are the relative strengths or weaknesses of this approach relative to the others?

3.6.1 Superconducting

We begin with superconducting qubits. This approach is probably the most mature, with a relatively long lineage of academic work bolstered by significant investment from tech giants like Google and IBM. As previously mentioned, Google's Quantum AI team has some of the most impressive announced results in both hardware (qubit counts & fidelity) and software (algorithms with realized quantum advantage, implemented on their hardware).

Unfortunately, in many regards, superconducting qubits are also the modality that is most difficult to understand. Of my listed 'Big 4' (superconducting, atoms, ions, photons), superconducting qubits are the only one not using a basic physical particle – something you might learn about in high school chemistry class. The layman knows what an atom is, but not what a 'transmon' is. Thankfully, we can build up a physical intuition for the superconducting qubit without too much trouble. Even better, we can re-use much of this intuition when we examine the other types of qubit implementations.

Before we talk about superconductors, silicon, quantum states, or anything else, we're going to go back to high school physics. Consider, as in Fig. 3.3, a mass

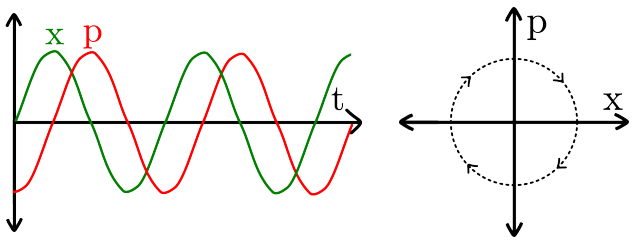


Figure 3.4: Momentum and position of a mass-spring oscillator plotted against time and against eachother.

mounted to a spring, in turn mounted to a fixed point. The mass is frictionless and does not lose energy as it slides across the ground – it's ideal! If we put energy into this system, it will oscillate forever. This is usually a bad assumption for the real world, but in fact is a great assumption for the world of superconductors: These conductors are so 'super' because they don't dissipate energy. A copper wire at room temperature is a pretty good conductor (certainly better than air, or wood, or rubber), but has finite resistance. If I induce some current in a loop of copper, it will generate heat and die out. Not so in a superconductor! Superconducting qubits utilize metals which exhibit superconductivity at very low temperatures (single digit Kelvin, just above absolute zero). That's an important reason why these qubits are kept in dilution fridges.

I digress! We are just talking about springs and masses. We're going to care about a few properties of the system: its energy, position, and momentum (which is equivalent to its velocity). If the system is sitting still: there's zero energy and zero momentum, and we will call this position zero. If we perturb the system, it will oscillate! The energy in the system will slosh back and forth between the mass's position and its momentum. When the position reaches an extreme, the instantaneous velocity & momentum will be zero. Conversely, when the position is zero, the instantaneous momentum will be at its highest. We can graph position versus time and momentum versus time, and get two out of phase sine waves. We can also graph position versus momentum, and get a circle, as in Fig. 3.4.

In each of these cases, if we perturb the system more and put additional energy into it, the amplitudes of the sines get bigger – i.e. higher energy means larger peak momentum and position. We can add energy to that position vs. momentum graph and get a conical shape in 3D. We can then take a cross section of that cone to see position versus energy, getting a parabolic shape, shown in Fig. 3.5.

OK! That was a lot of preamble to say something very simple: If a mass on a spring has a position far away from its equilibrium point, it has more energy! Now... let's make the system scarier... instead of a mass on a spring... we now have a quantum mass on a quantum spring. Everything is very very tiny, and obeys the laws of quantum physics. What changes? First and foremost, our energy levels are quantized. Depending on your level of familiarity with quantum

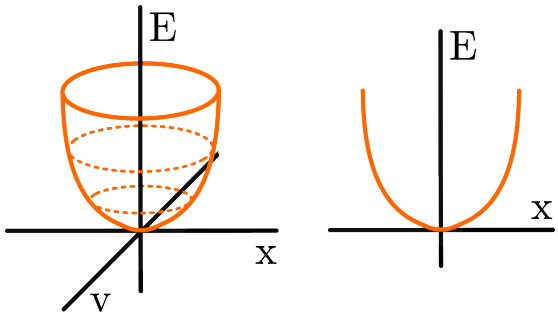


Figure 3.5: Momentum, position, and energy of the mass-spring oscillator.

physics, this may be a given or incredibly confusing. The following explanation is a gross oversimplification (sorry real physicists), but it is helpful: As we get to very small particles and systems, nature becomes discrete, not continuous. Our system can have 1 unit of energy, or 2 units of energy, but not 1.5 units of energy. It can only sit in certain discrete states. We can show this by drawing lines on our parabola, each representing a valid quantum state. We will also label each state with our special bra-ket notation.

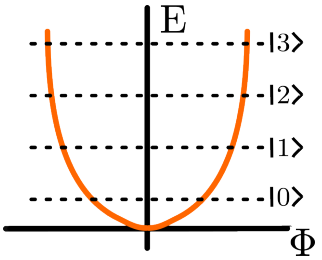


Figure 3.6: Quantum harmonic oscillator with qu(d)its encoded in its energy levels.

I mentioned this before, but it bears repeating. This system has more than 2 states! So while we pick the bottom two states to be $|0\rangle$ and $|1\rangle$, we also have $|2\rangle$ through $|N\rangle$ other states. Almost always, we don't actually want our computer to be in those states, as that can cause uncorrectable errors. It's very important for our system to be limited to states $|0\rangle$ and $|1\rangle$.

One more quantum property we care about: the system is now probabilistic. We cannot directly measure its velocity and its position at the same time, due to Heisenberg's famous uncertainty principle. As such, we can't have total certainty about the state that it's in. To make an analogy to a truly classical system: If we could only measure the position of the mass on a spring, observing it as position zero means it's likely in a lower energy state, but it could be the mass briefly and rapidly moving through center position as it oscillates while in a higher energy state. We simply can't be sure!

This is confusing, but should be a bit more intuitive given that you've just read quite a bit about probabilistic models of computation. Just keep in mind that just like our theoretical qubits from section 1, we can put the system into superpositions of states and that, upon measurement, the system will collapse to a singular valid quantized state.

Let's now put our microscopic mass and spring into a dilution fridge, and cool it down to 40 millikelvin. Brrr! It ends up being in its ground state - $|0\rangle$. There's virtually no motion. How do we excite the system – i.e., move it from $|0\rangle$ to $|1\rangle$? We have to inject some energy. In the mass and spring example, we'd have to give it a very careful, precisely calibrated microscopic tap. After all, we want to be in $|1\rangle$, not $|2\rangle$ or $|3\rangle$. Keep this in mind as we move forward.

Unfortunately, we are running into the useful limits of our mass and spring resonator example. We can keep most of our mental model the same, but we need to introduce how these systems are actually built – with electronic circuit elements. We are going to substitute our mass and spring for an inductor and a capacitor in parallel. These are two circuit elements that store energy. The capacitor is represented by two parallel lines, and the inductor by a coiled line. A capacitor stores energy in an electric field (often between two parallel plates of conductive material) and an inductor stores energy in a magnetic field generated by current running through loops of conductors. While we are drawing them as symbols in our little circuit model, these are in fact physical objects that we can design and manufacture by depositing or etching metal in certain shapes on a silicon wafer.

The inductor and capacitor form what is known as an LC oscillator. At a certain frequency (called the resonant frequency, determined by the value of each of the components), energy perfectly sloshes back and forth between the inductor and the capacitor. It is directly analogous to the mass on a spring, where energy moves back and forth between the momentum of the mass and the potential energy in the compressed or stretched spring. We can reuse the diagrams from before and just change the variables from position (x) and momentum (p) to flux in the inductor (Φ) and charge on the capacitor (q). We've changed our physical example, but ended up in the same place.

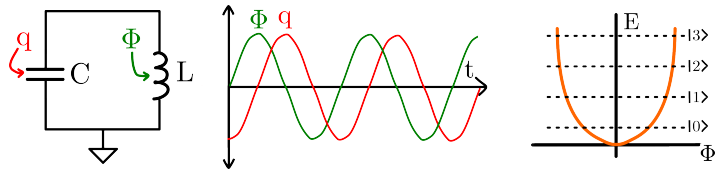


Figure 3.7: Waveforms showing charge and flux from a LC oscillator.

Now, if we want to excite the system, we need to add some energy into it. We do this by blasting the oscillator with some high frequency microwave radiation. This is roughly the same as putting leftovers into your microwave and having the water inside be excited at its resonant frequency, heating the meal up. Importantly, the system only wants to resonate at a certain frequency (its

resonance), so we have to tune the waves to excite the system at that frequency. Again, based on certain properties of the oscillator, we can figure out what that frequency needs to be. So we are almost at the qubit!

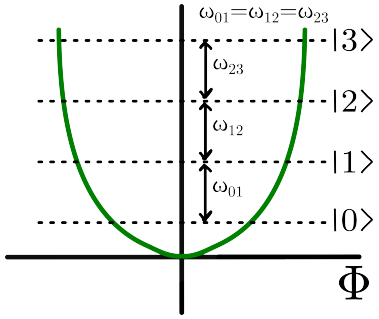


Figure 3.8: Circuit diagram and energy levels of a quantized harmonic oscillator, showing equal level-to-level spacing.

But one problem arises: Remember, we want very good controllability. We need to be able to put energy in, and be reasonably sure the qubit goes from $|0\rangle$ to $|1\rangle$, not $|0\rangle$ to $|2\rangle$. With our designed system, we can't get that! The reason is because the ideal quantum harmonic oscillator is linear. The spaces in between $|0\rangle$, $|1\rangle$, and $|2\rangle$ are all the same in our energy diagram. Thus, the frequencies of the waves needed to transition the system from state to state are all the same. Putting in the frequency ω could excite the system from $|0\rangle$ to $|1\rangle$ or to $|2\rangle$ - this is unacceptable! We need to make the energy levels in the system non-linear (i.e., space them unevenly). The term quantum engineers use for this is anharmonocity.

We get this by substituting our inductor with something known as a Josephson junction. This is a fancy microscopic structure that A) can be made with semiconductor manufacturing methods; B) at superconducting temperatures, has non-linear quantum behavior; and C) looks almost like an inductor, but not quite. With a Josephson junction, our energy curve changes, and the distances between states become different! This means that we would need a different frequency of microwave to excite the system from $|0\rangle$ to $|1\rangle$ than for $|1\rangle$ to $|2\rangle$. We can therefore address the exact state transition we want.

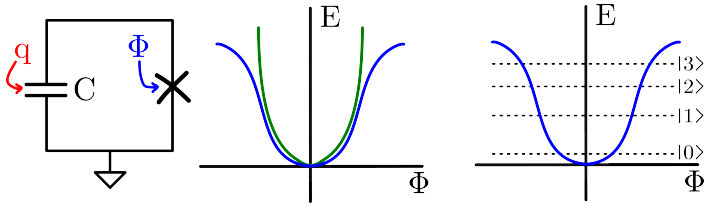


Figure 3.9: Circuit diagram and energy levels for a transmon superconducting qubit.

This is called the transmon superconducting qubit. There are several different kinds of superconducting qubits (fluxonium, phase, the delightfully named quantronium, etc.), but this is the one that most research groups are now pursuing, for various good reasons. As I've been mentioning, this is a system which can be manufactured on silicon wafers using relatively standard manufacturing principles. Some images of transmons are shown in Fig. 3.10. This ostensibly means that making these devices should be relatively straightforward, scalable, and repeatable! Also, the fact that these are not atomic devices, and have physical properties measured in the nanometers to millimeters, should make it easier to interact with the qubit. Functionally, a superconducting quantum computer looks like a big grid of these qubits, each which can interact with its nearest neighbors*.

However, this manufacturability has its downsides: the qubits are not exactly identical! Some of the qubits on a given chip are built faulty, and might be broken or have uneven performance (compare this to other approaches, which use particles that are definitionally identical). Even when systems are fabricated well, qubits are not necessarily identical and may have unique resonance frequencies, requiring intensive device and qubit-specific tuning.

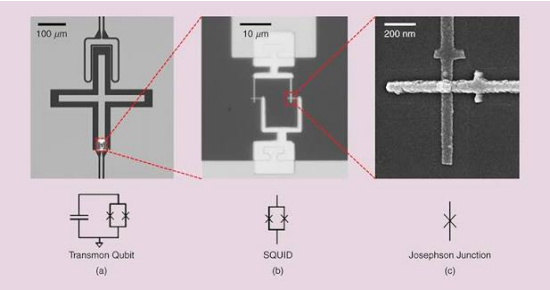


Figure 3.10: Images of a transmon qubit, with magnified insets showing a Josephson junction. Reproduced from[36].

Speaking of interaction – how do we implement gates? I've already told you how to do 1-qubit gates! We blast the qubit with microwaves of a specific frequency and phase, changing its state. We can calculate how much time that wave needs to completely change the qubit state from $|0\rangle$ to $|1\rangle$, and irradiate the qubit for shorter amounts of time to put it into intermediate superposition states. Furthermore, by varying the phase of the incoming microwave pulses, we can change the phase of the system, equivalent to performing gates like X, S, and T.

Two qubit gates are unfortunately more complicated. The most common way of implementing a 2-qubit gate on superconductors is to blast two adjacent qubits with an external magnetic field, changing their resonant frequencies. Depending on the state of the qubits, this process conditionally moves the qubits into

∗. It should be apparent why the surface code is very applicable to this kind of hardware.

a leakage state (i.e., one of the qubits is in $|2\rangle$) then back down, with a phase difference. This implements a controlled phase, or CPHASE, gate. The CPHASE gate can be easily combined with some single qubit gates to be equivalent to the more recognizable CNOT gate.

Remember that we don't need to be able to implement every kind of single and multi-qubit gate to be able to get universal quantum computation. As long as a quantum computer has access to a universal gate set, it can always compile any program down to the gates it does have (although it might use more operations than we hoped for!).

Finally, we need to be able to measure the state of the qubit. Here's the crude, roughly correct version: inject a waveform into the qubit and measure the phase shift of the reflected signal. Because the qubit has a slightly different resonant frequency depending on its state, that phase shift should be either negative or positive, corresponding to $|0\rangle$ or $|1\rangle$. This process is called 'dispersive readout'.

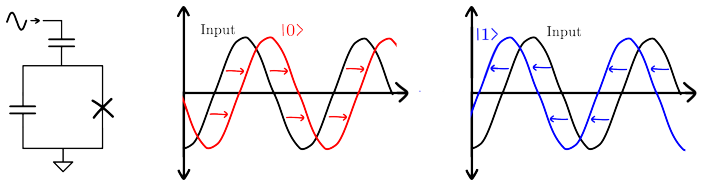


Figure 3.11: Cartoon showing a simplified example of superconducting qubit readout.

The reason why that description is not quite correct connects very directly to the challenges of superconducting qubit engineering. It turns out that readout is extremely hard to do well. The qubit is very sensitive to noise of all kinds. Measurement of the qubit can put it into a leakage state (such as $|2\rangle$) which will then propagate to neighboring qubits, ruining calculations. Thus, the readout process is carefully engineered, and all the circuits upstream must be designed to not inject any heat back into the qubit itself. This functionally looks like A) additional resonators integrated into the chip itself, which do filtering; B) amplifiers and isolation that sit in different parts of the dilution fridge. Even with these precautions, anti-leakage propagation methods must be inserted into our error correction circuits*. Oh, and by the way – all of these protections make our readout slower! It's sort of a massive balancing act trying to optimize qubit coherence times (how long they last in a desired state); noise and leakage; and speed – all of which are critical to building a functioning quantum computer.

With regards to speed and lifetime: Superconducting qubits are much less stable than other qubit implementations. They decohere on the order of milliseconds, which implies aggressive error correction is needed to keep computations accurate until completion. This is significantly shorter than most other kinds of qubits, as we'll see.

The flip side of this is that superconducting qubits are expected to be much

∗. This is called multilevel reset (MLR). See this excellent talk by Google's Daniel Sank.

faster at computing than other qubit implementations. Gates should take roughly tens of nanoseconds, and readout somewhere in the tens to hundreds of nanoseconds. This translates to operating frequencies in the MHz*, well below the multi-GHz clock speeds of conventional CPUs, but well above what trapped ion and neutral atom qubits can achieve today. This should reinforce our intuition about needing to find problems for quantum computers to solve that have very large speedups over classical computers, because doing an $O(n^2)$ computation on a quantum computer will still take much longer than an $O(n^2)$ computation on a CPU.

Let's add another challenge to this mix: How do we even build this thing? Well, I've mentioned already that we can make superconducting qubits on silicon. These things look like computer chips! However, on a regular chip, we can use multiplexers, memory, and various communication protocols to interface with lots of bits at once. If you tell your CPU to add two numbers, it just needs two indices in memory and an ADD command. We unfortunately don't have that degree of flexibility with superconducting qubits. We have lots of qubits, and need individual connections to each to be able to control them. This problem manifests in space constraints on the chip and in the dilution refrigerator, since all that control circuitry, amplifiers, and isolation take up a lot of volume too! Moreover, the more qubits you have, the greater risk that they interact with each other in ways you aren't planning for (we call this crosstalk). And to make matters worse, it's actually hard to make identical qubits at scale. Even successful chips often have several malfunctioning qubits which either operate with higher error rates or simply can't be used[37]. So far, labs have successfully scaled up to the ∼100s of qubits range, but breaking past that barrier will require either successful scaling using multiple interfaced chips, multiplexing on single chips†, or by some fundamental improvements that increase how many qubits can be put on one chip.

We're not to the point where we understand multiple qubit architectures, but let's attempt to keep track of some of the strengths and weaknesses of the superconducting qubit:

- We can make them relatively easily. That means more testing, more iterations, and more data. However, because they are human-built, they aren't always manufactured perfectly, and qubit-specific tuning is usually required.
- We build them on a grid and can get good two qubit gate operations with their neighboring units. However, they are on a grid, and can't be moved around!
- They're fast, with gates approximately 100-1000x faster than other qubit architectures. They also decohere 100-1000x faster than other types of

∗. Clock frequency isn't quite as simple as 1/gate speed, since A) there's significant amounts of time spend in readout; and B) it takes multiple rounds of gates to do any individual operation due to error correcting codes.

†. This is an active field of research. So far, multiplexing typically results in some tradeoff with gate fidelity, readout fidelity, or crosstalk. See a representative paper like[38].

qubits! They are incredibly sensitive to noise of all kinds. They need to be kept at sub-1 Kelvin temperatures* and continuously monitored with error correction schemes.

3.6.2 Trapped Ions

If superconducting qubits are the 1A of most developed qubit technology, trapped ions would be 1B. This is the favored approach of many academic groups and several high-profile startups, including IonQ and the Honeywell-spinout Quantinuum. In many ways, they are the yin to superconducting's yang: Long coherence times, high qubit interconnectivity and mobility, relatively balmy (2-4 Kelvin) operating temperatures, and slow operations. But this isn't to say that a trapped ion quantum computer is a simple-to-implement affair. Just like their superconducting brethren, there are significant engineering challenges to scaling these systems up.

Trapped ion quantum computers use, as their name suggests, ions as their qubits. Think back to high school chemistry class and you might remember that an ion is an atom that has a positive or negative charge due to the loss or gain of electrons. Ions are nice because they are identical and fungible (this will be a common theme among trapped ions and the next few qubit approaches). No engineering needs to go into making the qubits themselves 'better'. They simply exist in nature! Instead, efforts go toward interfacing with and manipulating these particles. That's where the 'trapping' comes in.

Ions are definitionally charged particles. Absent any sort of control, a bunch of identical atoms would all move away from each other (since alike charges repel), and we'd have no hope of interacting with them. The good news is that we can use that charged nature to our advantage and apply external electric fields to manipulate the positions of the ions. These electric fields are switched on and off extremely quickly to keep the ions in a stable position, and can be specially controlled to move the ions around.

The way this practically works is that labs and companies will use silicon chips (made with standard semiconductor processes) that have carefully designed electrodes on them. Then, ions around the system become 'trapped' as the electrodes start switching. The ions don't actually get stuck onto the surface of the chip! They actually float above just above (50 microns) the electrode surface, in a particular zone of manipulated electric field, as shown in Fig 3.12.

Note that we only want to trap the ions that we want to do computation with. As such, ion trapping is typically done in a controlled and supercooled vacuum (4 K), to limit outside particles and interference with the qubits.

These ions are typically metals like calcium, barium, and ytterbium. When these elements lose one of their outer electrons, they are left with a singular

∗. Several of the other platforms we will look at also need to be supercooled, typically to something like 2-4 Kelvin. It doesn't seem like there's a big gap between 2 Kelvin and 40 milliKelvin, but cooling gets much harder the closer to absolute zero you get. Practically, superconducting qubits require a lot more cooling infrastructure and engineering than other approaches.

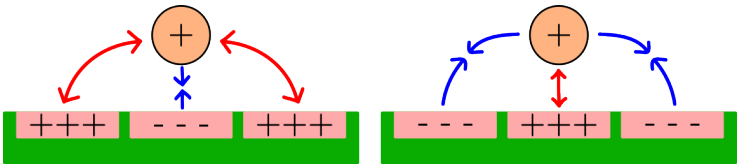


Figure 3.12: Cartoon showing how alternating electric fields can 'trap' a charged particle.

electron in their outermost orbital. Outermost orbital, you ask? Again, remember high school chemistry. In an atom, electrons exist in discrete spatial bands, called orbitals. Each orbital can only be occupied by a certain number of electrons. Once a given orbital is full, electrons have to start filling the next one up. Lower orbitals have lower energies. An atom or ion in its 'ground state' will generally have all its electrons in their lowest available orbitals.

If an ion is perturbed with some sort of external energy though, some of its electrons can move to higher orbitals – we call this an excited state. Having only one electron in the outermost orbital means that it is simpler to analyze the ion, since most of its excited states will just involve that single electron hopping between higher energy orbitals. Hopefully you see where this is going: We can encode quantum information in these different states of the ion. The basic energy diagram for a prototypical trapped ion qubit looks something like Fig. 3.13.

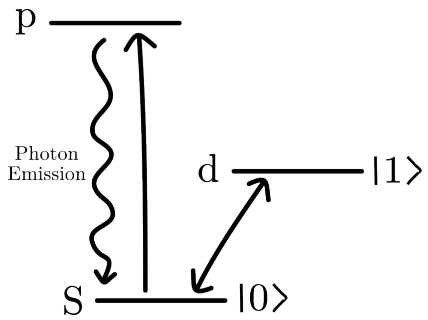


Figure 3.13: Energy levels of a prototypical ion used as an optical qubit.

The ground state, where all electrons are in their lowest energy levels (and the highest occupied orbital is s), encodes $|0\rangle$. If an electron jumps to the d orbital, the qubit is now in the $|1\rangle$ state. Those two levels are separated by a very well-defined amount of energy, and a laser with a specific wavelength can stimulate that transition. This should feel very similar to superconducting qubits, but note that these ions naturally have very high anharmocity! That laser only excites the $|0\rangle/|1\rangle$ transition, and nothing else. The time duration and phase of that laser pulse determine what kind of single-qubit gate the ion experiences. Note also that the times necessary for that laser to drive gates are on the order of milliseconds, much slower than the superconducting systems

previously discussed.

To do readout, the qubit is excited with a different laser. This laser drives an electron transition from the s orbital ($|0\rangle$) to the p orbital. If the qubit is in $|1\rangle$, nothing happens. If the qubit is in $|0\rangle$ (or collapses to it), then it is driven to p, then rapidly decays back to $|0\rangle$. In that process, the qubit emits some photons (light!), which can be detected. So readout is quite straightforward: blast the qubit with a laser, then try to detect light. This does end up being practically tricky (as just about all the details of any qubit implementation tend to be), because those emitted photons might scatter everywhere, disturbing the states of all the other qubits in the system. There are various schemes that mitigate this effect.

Now this is almost correct, but real-world implementations differ slightly. What I've described above is called an 'optical' qubit, since the wavelength of light separating $|0\rangle$ and $|1\rangle$ is in the visible spectrum. The lifetime of the optical $|1\rangle$ state is fairly long, about a second. This is already miles better than superconducting qubits, but for various reasons, many of the trapped ion companies actually use a different implementation called a 'hyperfine' qubit.

While optical qubits operate on energy scales in the hundreds of THz*, states in a hyperfine qubit are separated by energies in the GHz, and require microwave (also known as radio frequency) pulses to drive transitions. This is because the $|0\rangle$ and $|1\rangle$ states here are encoded in two different narrowly separated energy levels of the ground state (where the outermost electron is in the s orbital), known as hyperfine levels. The advantage of these qubits is A) extremely long coherence – hours to days to weeks; and B) being able to control them using electronics rather than precisely tuned optical lasers. In most other aspects, they're similar to optical qubits.

Just because qubit coherence is so long for ions, doesn't mean error correction is unnecessary. These qubits may remain in the same state for a long time absent noise, but there is unavoidable noise in any quantum system. Moreover, just like in superconducting systems, the gates themselves and the readout process may introduce errors. So error correction is still vital to successful large scale computation for trapped ion systems. That being said, it's not as critical to small scale tests, and seems to be a lower priority for most of the trapped ion players. In particular, Oxford Ionics (formerly an independent company but now an IonQ subsidiary with a unique technological approach) seems to be focusing all of its efforts on increasing gate fidelity and somewhat punting on error correction, at least for the time being. This does seem to be working, given that they have demonstrated 4 9s (99.99%) of two-qubit gate fidelity[39].

Speaking of two-qubit gates... trapped ion systems have a wonderfully elegant way of implementing two qubit gates. It turns out there's a 'hidden qubit' lurking in the middle of this system of ions. So far, we've been talking about these ions as if they are isolated particles: I point my laser at one qubit, do a gate, point it at another one, and do another gate. Well, these ions are sitting

∗. Physicists often refer to energies in units of frequency ($E = hf$), temperature ($E = kT$), or even mass ($E = mc^2$), as long as there is a fundamental constant that can be set to 1.

in a line and interact with one another (once again, remember that particles of like charge repel each other). There are certain stable 'modes' of interaction or motion that can exist in this line. Think of this like standing waves on a string or spring, as shown in Fig. 3.14.

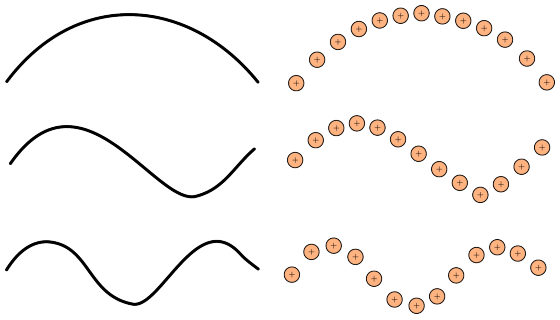


Figure 3.14: Cartoon showing harmonic standing waves that can be generated from lines of trapped ions.

This motion is another system that behaves quantumly! Thus, we could use it as a qudit of its own (not a qubit, since there are many possible states). However, this is a shared property of groups of ions in the system, so it is practically more useful to use it as a method of sharing quantum information – i.e. performing two qubit gates.

The mechanism for this is very similar to 1-qubit gates: driving the ions with lasers. However, in two-qubit gates, these laser pulses are tuned to have additional components slightly off the wavelength they would typically use for $|0\rangle$ to $|1\rangle$ transitions. These are known as 'sideband' pulses. The sideband excites (or attempts to excite) the motional state of the ion system. With a red (lower energy) sideband, we could go from [qubit state $|0\rangle$, motional state $|1\rangle$], to [qubit state $|1\rangle$, motional state $|0\rangle$], as shown in Fig. 3.15. However, trying to excite a qubit in [qubit state $|0\rangle$, motional state $|0\rangle$] with that same laser will do nothing.

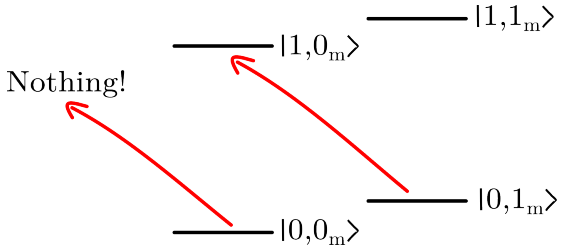


Figure 3.15: Energy level diagram showing how utilizing motional sideband excitation can create conditional logic in a qubit system.

The simplest implementation of a two-qubit ionic gate relies on the asymmetry in this system: A red sideband will do nothing if the qubit is already in its

lowest energy motional state. As a result, a sort of conditional logic is created by applying such laser pulses to multiple qubits, then letting them interact. Most real systems use a more complicated, but more generally applicable two-qubit gate called a Molmer-Sorenson gate[40]. Note that this gate performs its operations natively in the X basis, rather than the Z basis, so some rotations are necessary to properly implement a CNOT.

Importantly, these multi-ion interactions have a sort of locality to them. In a chain of 100 qubits, it may be possible to make the 40th and 50th qubits entangled via a two-qubit gate, but impossible to do so with the 1st and 99th qubits. This may seem 'bad', but it's already quite a bit better than superconducting qubits, who only have nearest neighbor connectivity. This interconnectivity could be made even better by making the ions mobile, as all the companies in this space plan to do. A key selling point of trapped ion systems is this ability to have greater qubit-to-qubit interactivity. This may enable easier implementation of certain algorithms and error correcting codes (think specifically of LDPC codes, which require non-nearest neighbor interactions).

This all sounds totally hunky-dory. We've got great fidelity and connectivity! What's the catch? Well, laser control of these qubits is a very difficult engineering challenge, both in terms of figuring out all the multi-qubit interactions and in terms of physically building and controlling the laser systems. As a result, we really don't want to have a line of hundreds to thousands of qubits, each with its own laser. Instead, most proposals for scalable trapped ion systems involve moving ions around from active computation zones to storage zones. See Fig. 3.16 for an image from Quantinuum's newest Helios processor for an example of this: Computation gets done by fixed laser arrays pointing at the two strips, while qubits get shuttled in and out of the storage ring.

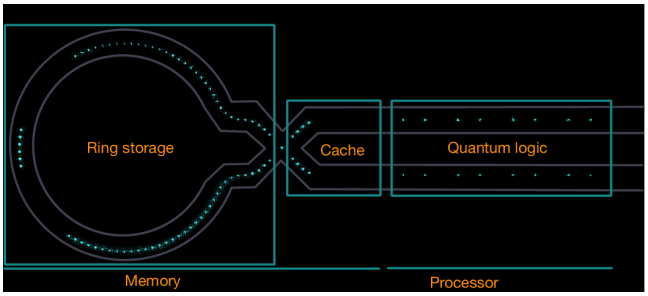


Figure 3.16: Labeled image of Quantinuum's 98 qubit trapped ion computer, Helios. Reproduced from[42].

But perhaps the most important catch is timing. Transport takes time, currently on the order of tens of milliseconds. In addition, after every transport, ions must be cooled down, which also adds tens of milliseconds. After all is said and done, gate operations can be performed on the order of milliseconds as well. Functional clock speeds may then be in the kHz range, significantly slower than superconducting systems (and comparitively glacial to a normal classical

processor). Clearly, optimizing all aspects of the system for higher speed will be necessary to get to scalable quantum computing. It's unlikely that trapped ion systems will ever be as fast as superconducting ones, but there is progress left to be made.

Next problem: Controlling lots of lasers is really hard and very bulky! Different groups have different approaches to solving this problem: Lots of folks are betting on integrated photonics – essentially building laser optics into the semiconductors themselves[43]. This is still a nascent approach, but could drastically reduce the complexity of building these systems. The previously mentioned Oxford Ionics is taking a very orthogonal tack, and choosing to minimize laser usage altogether. Their goal is to use integrated optics to do state initialization and measurement, but use electric fields to perform gate operations[44]. It's far too early to tell which approach, if any, will win.

The long-term vision for trapped ion systems is to have multiple chips, each with hundreds of ions, networked together[45]. This is ultimately the most feasible way to achieve scalability to systems with millions of qubits. Individual chips would be able to generate entangled photons via ion/ion interactions, then send those photons (with quantum information embedded in them) to interconnected chips via fiber optic cables. Trapped ion systems are far from this stage of development, but it is a critical piece of their future development roadmap.

Phew! Another one bites the dust. Once again, let's summarize some of the high-level tradeoffs of this modality:

- Advantages: Trapped ions are totally identical and extremely controllable. They are quite resistant to noise, and extremely stable and long lived. They have good inter-qubit connectivity, which enables lots of great things from a computation perspective.
- Disadvantages: They are extremely slow, and scaling to larger qubit-count systems will require huge amounts of infrastructure and difficult engineering.

3.6.3 Neutral Atoms

Much of the following will rhyme with the trapped ion situation. Atoms are extremely similar to trapped ions, mostly because ions are just atoms with an electron removed or added. These systems encode qubits in the same way, between different energy states of the atom (where its electrons are in different orbitals). This leads to similarly long coherence times. In general, the gate operations are similar too. And just like trapped ions, atoms can be shuttled around to enable non-local entanglement or storage.

Perhaps it's more edifying to focus on the differences. Neutral atoms are definitionally neutral. That's both good and bad: trapped ion systems have to do active control to contain their charged particles, because without it, those like charges would repel. But that electric field control lends itself nicely to implementing chips with integrated electrodes that can trap and shuttle ions around. On the other hand, atoms are happy to be near each other, and most quantum

computer proposals & demonstrations in this space end up implementing systems as a large rectangular grid of atoms. As for containing and manipulating those atoms, the solution is lasers! We've glossed over it before, but lasers are used for cooling in both trapped ions and neutral atom systems (see box below). We add one more level of laser application for trapped ions, known as 'optical tweezers'.

These tweezers are focused beams of laser light. At their focal points, there is a location of low energy where the atoms will naturally sit. These beams can pick atoms up and move them around. Controlling those beams at scale is a unsurprisingly difficult problem; the most common approach relies on a device called an acoustic optical deflector (AOD)[46]. An AOD is a special kind of crystal that can break a single incident laser into multiple output beams, each adjusted by a certain angle, all controlled by an input acoustic signal. It's incredibly cool engineering!

This control mechanism is advantageous for scaling on both fronts: We can have arrays of many atoms that can be moved around (potentially in bulk!) and it is not impossibly difficult to generate lots of tweezers that do the moving. But because these atoms are not 'trapped', they do happen to have a problem getting lost in the middle of operations. As such, a fully realized atom-based computer likely needs to be able to actively replace atoms in its system[47].

Laser Cooling

In our popular imagination, lasers lay somewhere between death rays and fun cat toys. In both cases, the laser shoots out light and hits something. In the weapon application, that laser has a lot of energy and heats its target up, perhaps in explosive fashion. In many neutral atom quantum computing systems, not only do lasers hold the qubit in place, they actually cool the atom down all the way to several micro-Kelvin (this technique is also used in trapped ion systems too!). This is pretty unintuitive!

The primary laser cooling process used is known as 'Doppler cooling'. To understand how it works, we need to know a few prerequisite facts:

- Heat in a system is equivalent to the motion of the particles in the system. If particles are moving more slowly, that system is cooler.
- An atom/ion will absorb an incoming photon if the photon is of a particular frequency. If a photon is absorbed, the atom/ion will go into an excited state, then emit another photon when it drops back into its ground state.
- The Doppler effect explains why an observer can perceive different frequencies depending on the relative motion of the observer and the incident particle/wave. The classic example is listening to a police siren as a cop car drives by: The wail is higher pitched (higher frequency) as the car drives toward you, and lower pitched as it drives away from you.

Put these concepts together and you get laser cooling: Tune a laser to be slightly lower than the excitation frequency of the atoms/ions, and point it at the bunch of particles you want to cool. Then, the particles moving toward the laser actually experience photons at their excitation frequency (because of the Doppler effect), while particles moving in any other directions will be unaffected. So this subset of particles absorbs photons, which lessens their momentum. Crucially, when they later emit photons, they do so in a random direction, so on average, whatever momentum that the emission contributes is less than the initial slow-down from the absorption. Over time, this cools down our mass of particles down to the 'Doppler limit', which is good enough for many quantum applications. There are advanced techniques that can get down even lower. Cool!

Another difference relates to the implementation of two-qubit gates. Neutral atom implementations rely on something called 'Rydberg states'[48]. When an atom is in a Rydberg state (also called a Rydberg atom), its outermost electron is in a much higher orbital, farther away from the nucleus. These Rydberg atoms

are more prone to interactions with other atoms, and it is these interactions which enable two-qubit gates. The standard CNOT gate elevates one qubit to a Rydberg state, and it conditionally interacts with another qubit, depending on the second's state (the interested reader may want to look up this process, known as a Rydberg blockade). In general, this is a pretty fast operation, occurring under a microsecond. This is significantly faster than two-qubit operations in a trapped ion computer.

One very exciting quality of neutral atoms is that it may be possible to perform gate operations en masse – i.e. making large blocks of atoms experience the same gates[49]. While it doesn't seem like this has been implemented in a commercial quantum computing system yet, this quality has the ability to further improve neutral atom-based approaches' scalability.

There's a subtle distinction between different implementations of these schemes. Some have a more static grid of qubits and can implement gates between non-local qubits (of course, with some limitation in range). However, most schemes have segmented regions for two-qubit operations, readout, and storage. This is easier in some regards, but does require more qubit transport. There's always a tradeoff!

The different atomic startups differ mostly in how the encode their qubits. The most popular choice is in the hyperfine states of rubidium atoms, although Atom Computing uses nuclear spin states of strontium, and Infleqtion uses two different kinds of atoms (a 'dual species' approach). Some of the startups (and parts of academia) are also interested in building 'quantum simulators' (which are akin to analog computers) with their atoms. This is a more natural fit for atom based systems because even today, they have lots of controllability over atom positioning. As such, you could imagine (and if you have the right equipment, actually conduct) experiments wherein atoms are arranged in particular 2D or 3D patterns, then allowed to evolve over time. Some measurement of the system's properties could then reveal something about the problem you are attempting to solve.

Let's do our review then: Neutral atoms have long coherence times and have fundamental advantages in system scaling, especially at the 100-1000 qubit level. Their gate operations are pretty fast too! Perhaps their biggest disadvantage now is that the approach simply isn't mature enough. There are very difficult control problems, especially at high qubit counts, that need to be dealt with in these systems. Experimental data has shown very similar learning curves for two-qubit gate fidelity for superconducting, trapped ion, and neutral atom experiments. However, since neutral atom experiments started later than the other technologies, their gate fidelities still lag behind! Time will tell if these companies catch up, or get left behind.

3.6.4 Photonic

You may be seeing a pattern: Find (or engineer) a system with quantized energy states, isolate it from its environment as much as necessary, use specially tuned pulses of energy to transition it from state to state, and a quantum computer

is created! Unfortunately, photonic quantum computers render this hard-earned intuition virtually useless. To make matters worse, photonic approaches not only differ radically from their superconducting/ion/atom brethren, they also have a great deal of internal variance – the approaches of the various photonic companies are not necessarily simpatico. All of this is because photons are so fundamentally different than the other qubit building blocks we've covered. They have very unique strengths and shortcomings that influence how a computer utilizing them should work.

First of all, photons are extremely mobile. Superconducting qubits are big and stationary; atoms will rest in place; even ions can be trapped and manipulated. Photons literally never stop moving. From the moment they are generated to the moment they are destroyed, they are moving at the speed of light! This is sort of exciting in some sense – we've talked a little bit about the necessity for quantum communication between different processing units, and something that can move while having quantum properties could be very useful. It's also, on another level, incredibly inconvenient for computation. A quantum computer has to make decisions on the fly about what gates to apply to a system (quintessentially, this is part of error correction). That's all well and good when your qubits are always in the same spot, but practically much harder when they are flying through the system at $3 * 10^8$ meters per second.

That is, if they even are in your system. Photons are extremely mobile, to the point where at any moment they may just... get lost. Photon loss is an intrinsic part of any optical system and strongly constrains these types of quantum computers.

Finally, for better and for worse, photons don't like to interact with other things. This makes them extremely resilient to noise and decoherence, to the point where these systems may be able to operate at room temperature. It also makes them very hard to force into two-qubit interactions. The standard way to implement a two-qubit photonic gate has a roughly 50% chance of failing in theory, let alone in practice! That's obviously miles away from the previously discussed approaches, which have had measured two-qubit gate fidelities with multiple 9's of fidelity.

Let's establish some physical intuition before we start getting too theoretical though. Photons are light, and light can be moved around and manipulated using linear optical components. These optics can be standalone, discrete objects or (as is the plan of all the companies in this space) integrated onto silicon chips. Something generates the photons, it flows through the chip on waveguides and hits various optical components, then gets measured by a photodetector at the end. There are two primary kinds of linear optical components: phase shifters and beam splitters. A phase shifter does exactly what it says it does – takes the input photon(s) and adjusts its phase. A beam splitter takes an input, and splits it into two output beams. If a single photon enters a beam splitter, it will go down one path with a certain probability (which can be tuned).

It's not intuitive to see how those building blocks can be used to build a quantum computer. But in 2001, Knill, Laflamme, and Millburne came up with their KLM scheme[50], which was the first real, although impractical, photonic

proposal. It's not quite how things are being developed now, but it is intellectually quite useful to think through this model first.

Their proposed approach is what is known as a 'dual rail' scheme, wherein a single qubit is represented by a photon traversing two optical paths. If the photon is measured on one path, that's a $|0\rangle$, if on the other, a $|1\rangle$. Single qubit gates can be implemented via phase shifters (changing phase, duh) and by bringing the two rails close together (which implements gates acting on the computational basis, since with some probability photons will jump over to the other rail). A trivially simple path, shown below, could easily implement a qubit being subjected to several single qubit gates, then being measured.

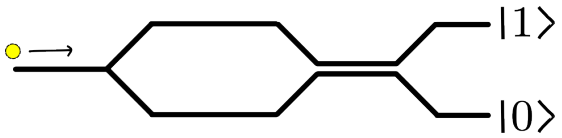


Figure 3.17: Simplified diagram of how a dual-rail, KLM-inspried scheme might work, wherein a single photon is fed through linear optics to eventual measurement on one of two channels.

These single qubit gates can be made to be reconfigurable via 'thermo-optics.' Essentially, heat up the waveguide, and its properties change. Hence, gates can be re-programmable in a more general chip, akin to a conventional computer. Thermo-optics are problematic since heat A) spreads between different gates, causing correlated errors; and B) takes time to... heat up, limiting bandwidth. That's why several of the leading photonic startups are investigating electro-optic schemes, wherein gate properties can be modified with fast-acting electric signals. PsiQuantum in particular seems to be at the cutting edge of this[51].

The tricky thing is how to implement a two-qubit gate. KLM's proposed a specific scheme (now superseded by better approaches) that involved several beamsplitters interacting between two qubits and some ancillas, followed by measurement. The details aren't terribly important, but there is one very big caveat with their implementation: it is non-deterministic. That is to say, any two-qubit gate was not guaranteed to work. Their proposal only had a 25% chance of succeeding (although that probability could be boosted with the addition of ancilla photons and additional work). I want to emphasize this. If a two-qubit gate has a 95% chance of succeeding and a 5% chance of destroying both qubits, after 1000 gates (a relatively small quantum circuit), the system has approximately a 1 in 10^{23} chance of getting an output. So this non-determinism is really a tremendous problem for large-scale computation! It's nigh impossible to get this approach to scale. You'd have to run computations repeatedly until you got a result for unfeasibly long times.

Nevertheless, the KLM proposal was a very important milestone for the field and stimulated a lot of incremental improvements and novel development. Better schemes for two-qubit gates were developed, and much attention was spent on photon generation. It is in fact extremely difficult to generate high quality, single

photons on demand. As we will discuss later, this is a rate-limiting factor for several of the current leading approaches in the field. Bits of the KLM scheme end up even in the more advanced, current approaches to quantum computing, and implementing a KLM-style processor is an attractive intermediate proof of concept that several startups have built[52].

The big 'a-ha' moment in the field came from the concept of 'cluster states' and measurement based quantum computing (MBQC). Remember way back in Chapter 2 when I told you that circuit-based quantum computing isn't the end-all-be-all of the field? Well, that chicken has come home to roost. The photonic people realized that if you have a system where it is really hard to do two-qubit gates, it might behoove you to pick a style of computation where you try to minimize the use of those gates. Thus, the selection of MBQC.

MBQC is not exclusive to photonic based computing, but it is very well-suited to this modality. The fundamental idea is that the system prepares a large, entangled, multidimensional state, called a cluster state. That state then undergoes a series of conditional measurements in varying bases*. These measurements (which have a kind of logic built in) actually do the computation.

Conceptually, it is a very different way of thinking about doing computation, but it is mathematically equivalent to the circuit-based model: anything that can be done with gates, can be implemented with MBQC. As a result, you can still think about algorithms for photonic quantum computing in terms of gates, but their compiled implementation will be very different.

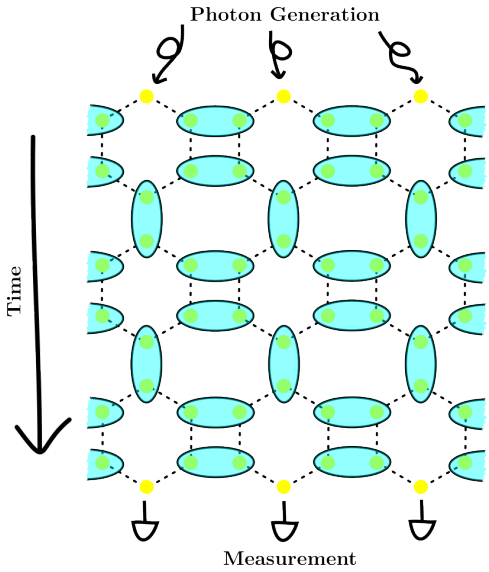


Figure 3.18: Cartoon of a MBQC approach to photonic quantum computing, wherein sets of photons are generated, entangled, then measured.

∗. Note that performing a single qubit gate/rotation, then measuring, is equivalent to measuring in a different basis.

The MBQC model is useful for photons for a few reasons:

- You do the really lossy two-qubit gates and entanglement first, and can potentially just post-select the photons in the correct state from the failed goes.
- That cluster state is multi-dimensional, and one of those dimensions can be time! So if you can generate a continuous stream of photons, delay some of them, then entangle them with the next 'generation', that sufficiently implements such a state. So a very large cluster state can be generated over time, which is very suitable for an implementation where qubits are constantly getting destroyed. Moreover, that cluster state can be imperfect, with failed operations or missing qubits, and computation can still be done.
- The lifetime of a qubit is relatively short: It gets created, entangled, then measured. That's it! That reduces the opportunities for the photon to get lost or to have some sort of error in a gate.

Somewhat important asterisk here: PsiQuantum, one of the leading photonic companies, has their own special flavor of MBQC they call 'fusion based quantum computing' (FBQC)[53]. FBQC is structurally very similar to MBQC, but has some additional tradeoffs. The MBQC method requires a large cluster state to begin with, which might require a lot of entanglement and happen with lower probability. The FBQC folks say look, what if we create simpler, smaller entangled states, then do a more complicated measurement scheme after the fact (with their destructive 'fusion' measurements). That lets them build a more complicated full calculation without having to make extremely large, entangled cluster states. The trouble is that the measurement scheme (consisting of fusion operations, hence FBQC), is non-deterministic. So they probably will need more resources, comparatively, to implement that scheme.

We've covered most of the system so far: we know how the systems do computation, how the qubits are encoded, and how gates and measurement are implemented. There's one more important detail – how do we make these photons in the first place? It's easy to make light: we can do that with lasers no problem. But these schemes require a very particular kind of light – identical single photons! Most light sources make lots and lots of photons at a time, often not typically identical ones. We want very high purity and high precision. There are a few varying approaches of doing this, but none are quite at high scalability yet. PsiQuantum uses these things called spontaneous four wave mixers (SFWM), that do pretty well, but yet again, they are *probabilistic*[54]. That's problematic, because we'd like all of our photons to be ready to go at the same time, so that we can entangle them and inject them into the system. Their plan is to multiplex lots of these units with lots of switches and delay lines to get things into the right state. That's somewhat plausible, but going to be difficult to implement. Quandela, another startup, takes a different tack: They use quantum dots* to generate photons[55]. The quantum dots, when excited with a laser, emit

∗. Think engineered, macroscopic atoms.

single photons with high precision. The trouble with this approach is that it's really hard to make identical quantum dots en masse. So in both cases, photon generation is the limiting factor – and you need lots of photons!

Yet again, there's another startup with distinct approach. Xanadu, a Canadian startup, thinks single photons are too hard to implement at scale. So instead, they use lots of photons, in the form of laser pulses[56]. These pulses are put into what are known as 'squeezed vacuum states'. Unfortunately while these squeezed states can be used to implement much of quantum computation, they are not universal; as a result Xanadu also has to create a special state (called a GKP qubit) probabilistically. Their computational approach bakes in the fact that they won't have only GKPs, but otherwise proceeds more or less like the above MBQC methods.

And so we're left with our photonic quantum computer: Blazingly fast, resistant to decoherence, (mostly*) room temperature, but hard to scale and control!

3.6.5 Other approaches

The above four platforms are by no means the only ways people have tried to (or are trying to) implement quantum computers. In fact, the list of possible qubit candidates is too long for me to cover. I'm choosing to highlight a few other high profile (or high potential) alternative platforms below.

Quantum Annealers

Quantum annealing is a strange beast. It's kind of a technology platform and kind of a method of computation. It's frankly ill-defined theoretically, so perhaps it's best to just describe the systems that perform it, known as annealers. After initial academic proposals, the company D-Wave was founded to develop commercial quantum annealers in 1999 – well ahead of any other dedicated quantum computing startups! For all intents and purposes, D-Wave is the quantum annealing field all by itself. For over 20 years, they've pressed forward with the idea that their limited version of quantum computing could get to market faster, demonstrate value, and win over customers.

The D-Wave system consists of a grid of superconducting 'flux' qubits (the newest version of their annealer has about 5000 of them)[57]. These are different than the transmon qubits we learned about earlier – they have a potential diagram that can form two different energy wells, shown in Fig. 3.19. One of these energy wells represents binary 1, and the other represents binary 0. The system is slowly transitioned from something that looks more like a harmonic oscillator (like the transmon qubit) to this binary well system.

On the grid, each qubit has some physical connections to neighboring qubits, called couplers. These couplers are programmable, and can set different relationships between their qubit pairs. The qubits and the couplers are initialized to some state that represents a problem of interest (think of this as some sort of generic optimization problem). The problem is encoded in a way that the lowest

∗. Some of the photon generation methods require cryogenics.

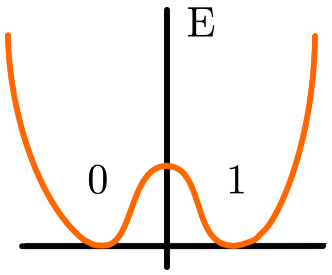


Figure 3.19: Representative energy diagram from a qubit in a quantum annealing device, where two potential wells are separated by a barrier.

energy state of the annealer corresponds to the solution of the problem. The system is evolved over time, with the goal of slowly getting the system to that minimum energy state. Because the system is quantum, it can hopefully get out of local minima that would trap classical approaches (like in Fig. 3.20).

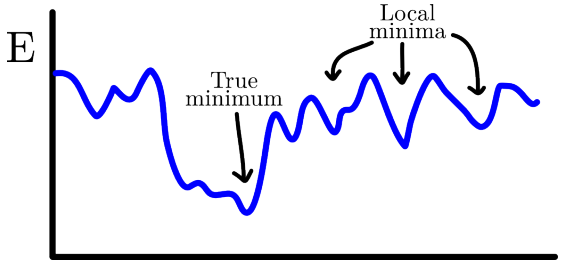


Figure 3.20: Cartoon showing an energy landscape, where an optimization process should find a global, rather than a local, minimum.

This process is closely related to the concept of adiabatic quantum computing, where a system is initialized into a starting Hamiltonian, slowly evolved over time with the goal of getting to some sort of final/solution energy state[58]. Adiabatic quantum computing is a fully universal quantum computing paradigm, and is mathematically equivalent to the gate-based model. However, truly adiabatic computing is hard to do! It requires very low temperatures (lower than even most superconducting systems operate at) and could require very long evolutions of state to get to an answer (which could push well past the decoherence times of currently available hardware). The quantum annealing concept essentially implements a version of adiabatic computing with limited scope. It makes less strict requirements regarding temperature & state evolution, and can only operate with particular kinds of Hamiltonians (those flux qubits have to end up in a binary state).

D-Wave's story is appealing in some ways: Limit flexibility in problem solving, but simplify the hardware requirements and make things feasible faster. And they have gotten to high qubit counts (thousands!) and attracted lots of funding in the

process. However, the quantum annealing method is fundamentally a heuristic one, and it's completely unproven if it can reach an exponential speedup in any sort of practical optimization problem[59]. D-Wave has done some simulations of 'spin glasses' which may be intractable classically, but these experiments seem to lack practical utility as well. As a result, the quantum annealing approach has totally fallen out of favor, especially as gate-based qubit platforms reach greater maturity, and the prospect of fault-tolerant computation seems possible.

Perhaps the most telling sign of the times is D-Wave's acquisition of Quantum Circuits Inc. (QCI) in early 2026[60]. QCI is a company building superconducting qubits for error-corrected, gate-based computation. D-Wave says they are going to push forward with both approaches, but it's clear that quantum annealing is destined to be an also-ran.

Topological

I need to make a confession here: I am totally out of my intellectual depth when it comes to topological qubits. In my defense, I'm in good company with my confusion. Every layman's explanation of topological qubits[61] ends up saying something like the following:

Topological qubits are a proposed qubit implementation that encodes quantum information in the structure/shape of the system that composes the qubit. As a result, these qubits are expected to be much more resilient to noise than other approaches.

Which frankly, tells you absolutely nothing. There is then some diagram that looks like a bunch of pieces of string hanging from holes, like in Fig. 3.21.

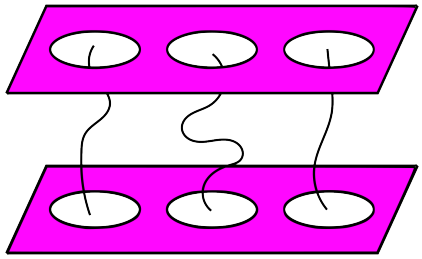


Figure 3.21: Cartoon of a topological qubit, allegedly.

Apparently, a topological qubit is just like a piece of string. Noise might wiggle that string around, or create a loop, but it's not going to change what holes the string comes out from or goes into. Thus, noise doesn't affect the quantum state! Or at least, it affects it less?

These qubits can be implemented via something called a Majorana quasi-particle, which is a special quantum particle (never actually observed, only theorized!) that has this topological property. Since we've never seen one of these Majoranas, we instead have to make one with semiconductors, akin to how we make transmon superconducting qubits.

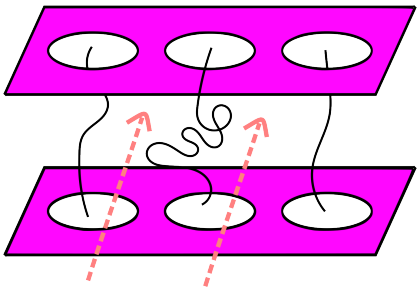


Figure 3.22: Cartoon of a topological qubit being resilient to noise, allegedly.

In 2025, Microsoft announced to much fanfare that they had successfully created this quasi-particle, or at least that they had created something that behaved in the way that such a quasi-particle would[62]. The paper was accompanied by a huge PR blitz, heralding the arrival of a quantum future led by our overlords at Microsoft. Unfortunately, the academic community immediately began pushing back against both the extreme claims made by the press releases and the actual data in the paper[63]. Like I said, I can't really make heads or tails out of the actual physics here, but there's enough smoke in the air to make me reasonably convinced of a fire: Some of the co-authors on a prior 2021 paper asked to be taken off[64] and several claims of data manipulation were made[65]. Almost a year after the initial hullabaloo, Microsoft actually published a new set of results[66] - which of course were met with another round of skepticism[67].

My opinion about topological qubits, which I suggest you adopt, is to not worry too much about them.

Silicon Spin

In contrast to the previous two approaches, silicon spin qubits are not necessarily fated to be an intellectual dead-end or technical boondoggle. Rather, they are an up-and-coming qubit modality with appealing characteristics and great promise!

Here's the short story: It's appealing to encode qubits in the magnetic spins of electrons, because it's a naturally binary quantum system (electrons are either spin up or spin down). This does necessitate capturing and controlling single electrons, a non-trivial task. The solution is to use quantum dots, which are sort of like artificial macroscopic atoms. Each quantum dot can be controlled to confine a single electron, convenient for quantum computation. The electrons can then be manipulated through microwave radiation (just like superconducting systems) or voltage pulses on the chip[68]. Quantum dots can be manufactured in a variety of substrates, but the most suitable medium seems to be silicon, in which A) electrons can remain coherent for relatively long amounts of time; B) be manufactured via standard semiconductor processes[69]. Add in fast gate operations (on the same timescale as superconducting qubit platforms), and you've got a very attractive technology for scalable quantum computation.

What's the catch? Well, these spin qubits are simply a less mature technology. Qubit counts are hovering around single digits today, and gate fidelities are not yet up to snuff[70]. Manufacturing is tricky, due to high precision and tolerance needed to successfully make the quantum dots (and if the process isn't quite right, then the qubits may have significant and problematic variation). If these problems can be addressed though, spin qubits could become a dominant qubit modality in the future! This approach is being pursued by several startups as well as American semiconductor giant Intel.

3.7 System Engineering

We now know how to build qubits, and maybe a little bit about how to enclose and control them (cryogenics, vacuums, RF fields, lasers, etc.). But in the same way that having some transistors does not make a computer, having some qubits does not make a quantum computer. There's a system that needs to be built around the qubits to make them into something that can do computation! Let's walk through some of these non-quantum components, starting at the user:

- Compiler: A quantum programmer is thinking in some sort of algorithm, and is choosing to implement that in some sort of high level way. After all, if computers can't hope to simulate the states of a hundred qubits, there's no way for a person to hold those states in their mind! So the programmer has some sort of programming language, and some sort of compiler translates those instructions into actual qubit gates. There might be multiple layers of this, where a platform agnostic compiler makes generic gates, and then a platform-specific compiler translates those generic gates into what is feasible to implement on a given piece of hardware.

- Calibration & Monitoring: This is all very dependent on the qubit modality in question, but in general there needs to be some sort of automated supervisory system for the quantum computer. This system might be automatically calibrating superconducting qubits or checking if atoms have leaked out of their optical traps, then mitigating these issues.

- Gate Control: Again, platform dependent. In general, a system must generate lots of high frequency laser, microwave, or voltage pulses with high precision, often at the same time. This is no trivial problem, and in some cases must be coupled with lots of isolation & cryogenics to reduce noise.

- Integration with classical computing: This is important on two fronts.

  - First, many algorithms are hybrid quantum/classical. While the quantum algorithm is running, the system may need to do classical computation, or vice versa.
  - Second and more crucially, error correction generally necessitates mid-circuit measurements and feed-forward operations, where future gate

operations depend on the results of stabilizer measurements. Especially in superconducting systems, this may require extremely low latency connections and logic, implemented on an FPGA, GPU, or other specialized chip. This is tricky to actually implement, and we'll likely see much innovation in this space moving forward.

Another axis for system integration (and seems to be the topic du jour), is the heterogeneous combination of different qubit architectures. In many ways, this is a pretty reasonable idea. Superconducting qubits can do lots of operations fast, but are bad at long term stability. Ion or atom based platforms sit on the other side of the spectrum, and may be suitable for longer term storage or some type of 'quantum memory'. If you integrate both on a single system, maybe you can get the advantages of each. The important caveat is that system complexity will increase significantly, and communicating quantum information between different qubit modalities is a very difficult engineering problem. Nevertheless, there's been significant buzz around this concept, with Google branching out into neutral atom approaches[71], funding programs being announced for the concept[72], and startups announcing plans for heterogeneous architectures[73].

So now you know how to build a quantum computer. What comes next?

Chapter 4

Application

We currently live in the 'Noisy Intermediate Scale Quantum' (NISQ) era[74]. This means that our quantum computers aren't fully error corrected; at some point in any computation, errors will build up and turn our results into noisy, random mush. One day, the industry may produce a computer that can perform calculations of arbitrary length and scale without accumulating errors. Once that's achieved, we will all be raptured into heaven... or err... excuse me, I meant that we will be in the 'Fault Tolerant Quantum Computation', or FTQC, era.

Let's now consider what that real-world, honest-to-god, fault-tolerant quantum computer will be used for. I will proceed through four categories of applications, touching on the current state of affairs and prospects for future innovation. These are all applications that are actively being worked on and are cited as reasons to build a quantum computer (except for the first category, which is more academic).

I should clarify that I am primarily interested in the long term, i.e. what algorithms we will run and what will be accomplished in the FTQC era. Many (and we will touch on this) are working on algorithms that are implementable in the near term, in the hopes of achieving useful quantum advantage on today's devices. Scott Aaronson likes to show this Venn diagram with three rings: Verifiable; Useful; Achievable on today's hardware. As of today, nothing sits in the middle of those three[75]! I'm personally happy to throw out that last criterion and focus on theoretical impacts way down the road, although many companies are urgently trying to prove value to industry.

Another word of caution: while last section may go out of date (new technology developed, other approaches become dominant, etc.), this one has the chance of being wrong factually! The opinions that follow are my own, and while I think they are reasonable, others certainly disagree. You've been warned![†]

†. Not to overwhelm you with caveats, but I want to be clear on another point: There's a sub-field of the broader quantum industry known as 'quantum sensing'. Essentially these are advanced sensors and techniques that utilize quantum mechanical properties to achieve higher performance in some way. I'm not going to cover these at all, since this is a book about

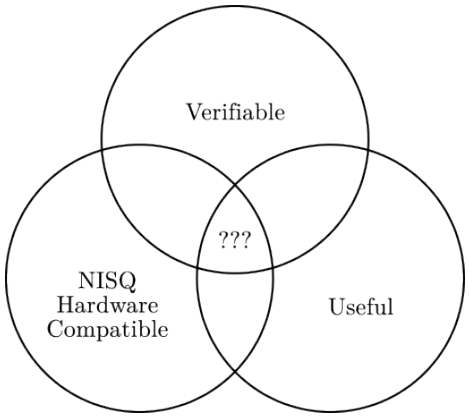


Figure 4.1: Venn diagram showing the holy grail of near term quantum computing algorithms. Adapted from Scott Aaronson.

4.1 Proving Quantumness

Our first application is incredibly boring. Imagine an evil company called Qeranos run by Qelizabeth Qolmes, who has no qualms about making a fake quantum computer. Qeranos claims their device has a hundred physical qubits and is available for public use on the cloud. They then take user requests and run them on a classical computer, with artificial noise added! If word of this were to leak, it would be a total disaster for Qeranos and the quantum computing industry as a whole. But what if Qeranos keeps its secrets safe? Is there still a way for a third party to verify that a 'quantum computer' is actually doing quantum computing? Thus, we reach our first application: Proving quantumness.

In these discussions, you'll often hear people bandy about buzzwords like 'quantum advantage' and 'quantum supremacy'. These are ill-defined terms that are generally used to refer to a quantum computer being able to perform tasks a classical computer could not. Practically, any PR puff-piece touting a new demonstration will claim a quantum advantage, but it takes quite a bit of time to understand if that really is the case. As we'll see, it may be possible for optimized classical algorithms to catch up, and perhaps outperform, quantum algorithms.

What is needed here? Well, we have those three criteria I mentioned above: First, the algorithm must be able to be run on today's NISQ devices; second, it must be verifiable – that is, we can check the answer with our classical machines and confirm the computer got it right; third, we couldn't have done it (or spoofed a made up answer) with our classical computer.

We are aware of algorithms that a FTQC machine could do that could fulfill the latter requirements. Take Shor's for example: We cannot factor large numbers efficiently on a classical computer, and any proposed answer can be

computing.

quickly verified via simple algebra. Unfortunately, we are orders of magnitude away in qubit counts and gate fidelity to implement Shor's.

There are a few approaches that are currently being used for this task. The first and most mature is called 'random circuit sampling'[76]. The idea is extremely simple: Prepare a bunch of qubits, then subject them to as many gate operations as the machine possibly can perform (this is limited by how error-prone the gates are). Then, the qubits are then in a very complicated entangled state. Given the same gates, it would not be efficient for a classical computer to generate that state. On the other hand, it should be easy for a real quantum computer to execute those gates and get the output state. This is really good at satisfying our first criterion – we can execute these circuits with existing hardware! Hooray! It is perhaps less excellent at the other criteria. Say the circuit runs perfectly on a quantum computer: how do we verify it is in the right state? We first have to measure the system, since the state is entangled and complicated. We in fact must run the circuit lots of times, to get a probability distribution of results. However, the whole point of the random circuit is that it would take an extremely long time (years and years) for a classical computer to run. As a result, we don't have a ground truth probability distribution to compare against! What is actually done to verify the results is the calculation of a property called 'linear cross entropy'[77]. This metric distinguishes between results that are uniformly random and those that are random in a very specific, quantum-enabled way. There have been quite a few RCS experiments from various groups, and they've all shown sufficient cross entropy scores that indicate that their computers are indeed quantum!

Nevertheless, this whole approach is sort of aesthetically displeasing. It's a given that it is totally useless (everything in this section is). More problematic is that it has an extremely indirect verification method. It relies on, as one very nice paper puts it, a 'proxy of a proxy of a benchmark'[78]. It's like being blindfolded and deafened, carried to an unknown place, and verifying that you're in Disneyland by the smell of the churros. Moreover, that cross entropy metric can still be spoofed by a dishonest actor! So there's still a need for other approaches.

Next up on the list is boson sampling. I'm sort of cheating here – boson sampling is sort of a quantum computing paradigm (like gate-based or adiabatic), and sort of its own application. It's complicated. The original pitch from Scott Aaronson and Alex Arkhipov was to shoot single photons through a linear optical network, then measure the distribution on the other side[79]. Trying to simulate that distribution classically is actually inefficient, but can be done efficiently experimentally (i.e., with quantum computation!). That proposal was interesting, but difficult to implement physically, because as we've discussed, generating single photons is hard. So a different team proposed using 'squeezed states' of light to do 'gaussian boson sampling'[80]. This is essentially the same thing, but easier to physically implement. You'll remember that the photonic quantum computing startup Xanadu is using these squeezed states in their approach. This style of experiment has been done, and achieved some form of quantum advantage![81] Some of the photonic platform companies have also used this as an intermediate milestone on their way to full-scale quantum computation.

This boson sampling style of computation is not universal, but potentially could be applied to some problems regarding simulation of 'molecular vibroic spectra of complicated molecules'[82]. That's a niche, but real application! Unfortunately more general simulation requires universal quantum computation, as we'll discuss later.

Another somewhat promising avenue of research is 'peaked circuit sampling'[83]. The idea is pretty similar to random circuit sampling. The quantum computer is given a circuit to run that is far too deep and complex for a classical computer to run in reasonable time. The catch here is that instead of the output distribution being random/unknown, the circuit is designed to produce a specific output with higher probability (the peak!). This is desirable because there's a verifiable ground truth in the results. If your computer produces the peak, then it's quantum advantage! The difficulty here is designing such a circuit in a way that classical approaches can't figure out the desired peaked output. Recent events demonstrate the challenge here: Quantum software startup BlueQubit proposed a peaked circuit scheme, touting it as the state of the art for quantum advantage. However, their collaborators/intellectual sparring partners at IBM successfully developed a new classical method for finding the output of a peaked quantum circuit[84]. This new method cut classical runtime by an estimated 11 orders of magnitude (huge!), thus outperforming the quantum algorithm. And that's a crosscutting theme we will see in several applications: Quantum algorithms can propose drastic speedups, but classical algorithms can fight back! This is an actively evolving field.

Finally (but not exhaustively) I'd like to discuss Google's recent (late 2025) results on an approach varyingly referred to as 'out of time order correlators' (OTOC) or the 'quantum echoes' algorithm[85]. Again, the core of this approach is applying a relatively large circuit that conventional computers can't hope to simulate. The OTOC experiments apply a random circuit (call it U) on a set of qubits, then a specific operation to a single qubit, then they reverse the U operations (U^T). They repeat this a few times, then measure a specific qubit. The idea is that without that middle perturbation (and without noise and error), the system would just return to its initial state. However, that perturbation causes some higher order random effects as it propagates through the reversing process, leaving the system in some other state. Google claims that this has potential practical applications for molecular simulation.

I think this experiment is interesting for two reasons:

- First, Google says they can verify it! The circuit is too complicated and big to simulate with a classical computer, but they ran the same circuit on two different quantum computers, getting the same result. This indicates that the quantum computers are probably functioning properly.
- Second, it is a very representative example of the quantum hype machine. Google announced the experiment and its quantum advantage results, performed on 65 qubits. Their announcement noted their optimism using this for molecular simulation tasks, and concurrently published a different paper that used their computer to simulate molecular geometry for some

simple molecules[86]. The trouble is that second paper[87] did not use the OTOC algorithm, and only used ∼10 qubits for a task that was much easier to do with classical approaches. As is often the case in the quantum world, the fine print (the actual papers) were factually correct and intellectually honest, but the non-technical summaries conflated separate results and inflated the impact of the work. It's a valuable lesson that one should approach claims with skepticism in this field.

4.2 Factoring, Cryptography

While speculation about the potential practical uses for quantum computation began with Feynman and quantum simulation in the late 1970s, it wasn't until Peter Shor's algorithm for factoring in the mid 1990s that funding started to really pour into the field. Many parties, perhaps chiefly the military industrial complex, are interested in using quantum computers for various cryptographic applications. Today, as we've covered above, Shor's is still perhaps the only definitive quantum algorithm that can achieve proven exponential speedups on a real-world problem. And yet, I personally think the potential impacts of quantum computing on encryption are relatively staid.

Let's do a quick overview of modern cryptography: There are broadly two categories of encryption – symmetric and asymmetric[88]. In a symmetric scheme, two parties have the same secret key, and use it to both encrypt and decrypt messages. This is fast and easy, but relies on both parties having the same key. Asymmetric schemes require no shared private information, but are more computationally expensive. In most modern communication schemes (such as the internet), an asymmetric scheme is used to exchange a key used in symmetric encryption of the ensuing communication. These are the protocols that keep your messages private, your financial data secure, and your passwords safe.

Quantum computing is not some magic bullet that can be used to break any encryption scheme. If there's anything you should remember from Section I (and will become a theme in this section as well) is that quantum algorithms need some sort of 'structure' in order to access exponential speedups, and thus practical quantum advantage. Absent any sort of structure, a quantum computer can only achieve a Grover-type quadratic speedup. Now that we've reviewed quantum hardware, we understand that these computers are going to operate with clock speeds orders of magnitude lower than conventional CPUs and GPUs. That theoretical quantum quadratic advantage would likely translate to a practical thrashing by the classical computing team.

So in the realm of encryption, quantum computers are unlikely to be helpful in breaking symmetric codes[89]. There's no structure to exploit, just some secret key! They could attempt to brute force that key using a Grover-type search scheme, but once again, that is going to be much slower than a classical machine.

However, the most commonly used asymmetric schemes – the previously discussed RSA; elliptic curve cryptography (ECC); Diffie-Hellman – are all vulnerable to Shor's. Each has some internal structure that Shor's factoring

algorithm can exploit, leading to exponential speedups over classical approaches.

This has rightfully freaked out security specialists for the past two and a half decades. The term of art for when a quantum computer can break real-world security schemes is 'Q-Day.'[90] Visionary prophets (or opportunistic fearmongers), spurred by recent advancements in hardware and algorithms, have been telling the world that Q-Day is approaching rapidly, and that preparations must be made. And to be fair, if there is a risk (even if it's small) of RSA and its ilk being cracked, then it is critical to mitigate that risk!

The quantum industry is obviously very aware of public interest in encryption applications, and various researchers spend time thinking about exactly what physical resources are necessary to break encryption schemes. These days, the best estimates come from the aforementioned Craig Gidney. His latest paper tackling this issue (published in 2025), assuming the characteristics of fault tolerant superconducting quantum computer (specifically its operating frequency & connectivity), finds that RSA-2048 could be broken in about a week utilizing around 1 million physical qubits[91]. While adding more qubits could reduce that time, that seems to be a pretty reasonable bound. Recently, Iceberg Quantum, an Australian software startup, published a much-ballyhooed paper that reduced estimated qubit count by another factor of ten, to 100,000[92]. The catch is that they assume implementation of QLDPC codes (low density parity checks, discussed briefly above). These codes have far better logical qubit per physical qubit counts, but require non-nearest neighbor connectivity. The Iceberg folks seem to have their cake & eat it too – assuming non-local connectivity (not typically a hallmark of superconducting qubit implementations), but also fast, superconducting level cycle times. Yet another reminder to take every result with a grain of salt.*

With all this smoke, the cryptography community is not standing still. Leading the way is the US National Institute of Standards and Technology (NIST) – an extremely strange organization housed in the Department of Commerce that regulates weights and measures, is a center for quantum research, and manufactures very expensive peanut butter†. NIST has been looking at post-quantum encryption schemes for well over a decade now and has selected five to-date as future standards[94]. These schemes are all based on fundamentally different math than RSA and ECC, and should be resistant to Shor-based attacks. There's no free lunch though – each comes with more overhead than conventional schemes, either via more data transmission or more necessary computation to (de/en)crypt[95]. NIST has selected so many because, frankly, it's worried! Our conventional encryption schemes are very much battle-tested, and we have significant evidence that says they are secure. These novel codes are not yet proven, so backups and alternatives have been made.

∗. Since I initially wrote this section, a flurry of new papers have come out that attempt to further reduce resource requirements for encryption cracking. These are primarily focused on elliptic curve protocols like the one that bitcoin uses[93], and obviously have some caveats, but do represent meaningfully progressed estimates. I've chosen not to rewrite this section simply because A) I'd like to highlight just how fast the rate of progress is in this field; B) I'm lazy.

†. Available for purchase right now!

While NIST's recommendation is to deprecate non-quantum secure standards in 2030, the technology industry is actively adopting these PQS methods today. Internet infrastructure provider Cloudflare reported that over half of its traffic utilized post-quantum encryption as of October 2025[96]. This begs the question – what's the point? By the time quantum computers are developed, our systems will be secure from them!

There are a few directions of discussion from here: First is the obvious point, which is that quantum algorithms could improve. Shor's is a starting point, not an endpoint. It is entirely possible that a new scheme could be developed, potentially significantly decreasing the effort needed to attack classical or post-quantum encryption methods. We simply don't know yet!

Second, many are worried about a family of attack known as 'harvest now, decrypt later' (sometimes abbreviated to HNDL)[97]. The idea, as the name suggests, is to collect and save encrypted data today, then decrypt it once a sufficiently powerful quantum computer has been developed. This is one reason why it's prudent that technology infrastructure providers adopt PQS schemes today. Data harvested 15-20 years before Q-Day is much less valuable than data harvested a year before Q-Day. In general, I think this is a reasonable but rather overblown fear, especially considering the relatively rapid (and very in advance) rate of adoption of PQS.

Third, one might have a reasonable concern if some sensitive information – say financial transactions – were protected by asymmetric encryption and said information was easily viewed on a publicly available ledger which intermediated trustless transactions between anonymous counterparties. Good thing we do financial transactions via trusted intermediaries and a regulated payment system! Oh, wait. Some people use (invest in? hold? hodl?) bitcoin. Unfortunately, bitcoin and certain cryptocurrencies of its ilk are vulnerable to quantum attacks. The bitcoin blockchain publishes transactions with user public keys. A sufficiently large quantum computer could then use that public information to find the user's private key, then initiate a transfer of their bitcoins to some attacker's wallet. Users can work around this by cycling public keys after each transaction, but there's some percentage of stranded bitcoins/unsophisticated parties who will not do this, leaving them vulnerable. Perhaps more worryingly, the blockchain itself relies on a factorization problem to mine new blocks (which adds to the ledger & mints new bitcoin). There's a future where a quantum computer could beat all the miners to the punch and monopolize the world's bitcoin supply/infrastructure[98]! Bitcoin could solve this via somehow integrating a PQS algorithm, but it would require a 'hard fork,' where the fundamental operation of the blockchain changes. It seems like something that will be a terrible pain in the ass for lots of people, but ultimately won't change the world too much. The more centralized Ethereum community has already begun thinking about formalizing post-quantum standards into their blockchain[99]. In general though, I'm a crypto pessimist, so I don't really care about the impacts here.

Quantum computing may have certain niche uses for creating encryption, though. It's useful, both for encryption and other applications, to have a source of truly random numbers. It's remarkably hard to generate random numbers:

computers are very deterministic, physical phenomena and human behavior both have underlying patterns, and truly random phenomena are hard to access. Quantum mechanics has an excellent source of true randomness. Very simply, if you initialize a qubit in $|0\rangle$, perform a Hadamard gate, and measure, you will get 0 50% of the time and 1 50% of the time, with pure randomness! This is overly simplistic – more complex randomness creating schemes have been made – but directionally correct.

Additionally, you could imagine using quantum-native encryption schemes. This could take the form of a specific, quantum enabled algorithm, but most proposals focus on using quantum properties for secure key distribution. The problem statement is just like the asymmetric encryption schemes discussed above: We'd like a very secure way for two parties to determine a secret key to use in subsequent encrypted communications. The idea here, initially proposed by Charles Bennett and Gilles Brassard with their 'BB84' scheme[100] (and shown in Fig. 4.2), is to have one person (Alice) entangle pairs of qubits and then send half of each entangled pair to the other party (Bob). Alice and Bob then measure each of their qubits in one of two bases, which they select randomly. They then tell each other which basis they measured in, but not the results. They throw away all the bits where they measured in a different basis. The bits that remain, because they are entangled and measured in the same basis, should be the same, forming the basis of their shared secret key! They take some selection of their approved matched bitstring and share, to make sure that there wasn't some sort of eavesdropper (whose intermediate measurements would destroy the system's entanglement, revealing their interception!). Ultimately, this approach and others like it have to compete with very established and resource efficient classical key distribution techniques, but may be useful in niche applications.

I reckon the story about quantum computing and encryption would be much more interesting if post-quantum encryption schemes didn't exist (or couldn't be made!). Then we'd have a ticking time bomb, making quantum computer development into a fascinating moral choice. I have no doubt that actors (probably the US government) are planning to utilize these data harvesting schemes, but I simply can't see a world where Shor's algorithm fundamentally changes data privacy. This is objectively a good thing, but leaves me wondering what good quantum computers will actually do.

4.3 Optimization, Machine Learning, Quantum AI, and other Miscellanea

This is a hodgepodge of topics that I hope to convince you should be treated as more-or-less the same in your mental model of quantum computing applications. Depending on where you're coming from, optimization, machine learning, and AI might seem very distinct or virtually identical. My rough boundary lines between them (at least for quantum computation) are as follows: Optimization is a problem with defined constraints where we are looking for one or many

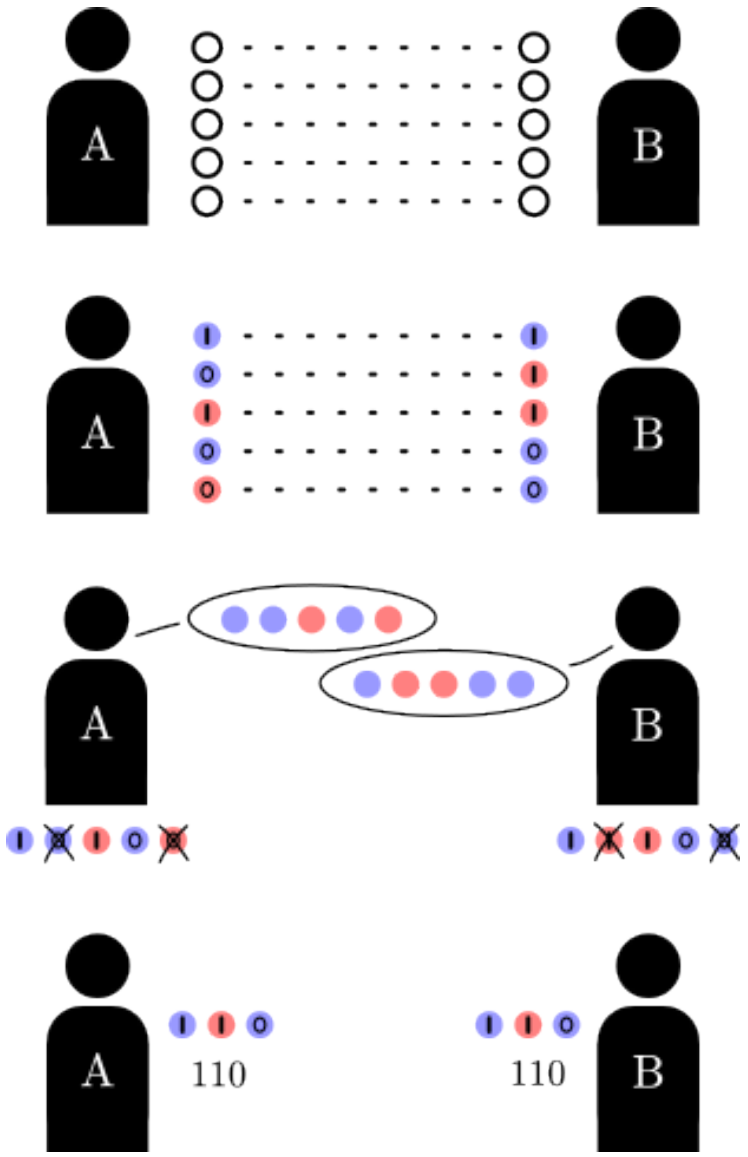


Figure 4.2: A rough schematic of a BB84 quantum key exchange. Alice and Bob start with entangled qubit pairs, and choose to measure each in one of two bases randomly (red or blue). If they measure in the same basis, they will get the same value (since the qubits are entangled). They then tell each other which bases they measured in for each qubit, and discard the measurements in differing basis. The remaining measurements represent a private key, exchanged without sharing the key information. They could choose to share some bits of the private key to verify against a 'man in the middle' attack (not shown).

solutions with superior characteristics (i.e., lie on the pareto front); machine learning is a set of methods that attempt to solve numerical problems, often optimization, via a process of training on existing data, then applying the trained model to new problems; AI could really mean anything, but let's say it's more specifically some sort of transformer based model like today's LLMs. AI is a specific kind of machine learning, and we could potentially use both AI and ML to solve optimization problems.

These problems are often tied to specific applications, perhaps the most salient of which is 'finance,' broadly speaking. The financial industry (hedge funds, banks, proprietary trading firms, etc.) relies on algorithms across a wide variety of timescales to make trading decisions. The prospect of faster and/or better solutions to these problems via quantum computation could potentially translate to return on investment for financial firms[101]. Hence, this is an prospective area of interest for the quantum industry.

Before we discuss any actual algorithms or specific applications, I need to beat a dead horse one more time: a quadratic algorithmic speedup is not going to result in a practical acceleration of anything with a quantum computer. Pitted against a classical computer trying to solve the same problem (but taking quadratically more operations), not only will a quantum computer lose, it will lose badly. Let's make it very plain why, borrowing some analysis from this excellent Matthias Troyer paper[102]. They compare a highly performant, hypothetical fault tolerant quantum computer (10,000 qubits, 10uS gate times, full qubit-to-qubit connectivity) with a single, commercially available GPU. This is an extremely unfair comparison: that quantum computer is extremely optimistic in its assumptions, since it's unclear how to achieve fast operations and all-to-all gate interactions; parallelling GPUs is a common practice that typically accelerates classical problem solving; and GPU performance is increasingly rapidly over time - their comparison, an NVIDIA A100, was state of the art at time of publication, but now is something like 6 times less performant than the comparable NVIDIA GPU of 2026!

Nevertheless, we can take a few useful conclusions from the work: The first is that math operations on a quantum computer will take something like 10^{10} times longer on a quantum computer (the exact factor depends on the type of data being operated on). That's mind-bogglingly slow. As a result, while a Grover/quadratic speedup may reduce the number of function calls by a factor of n^2, each function call may take radically longer. The authors find that a quadratic speedup is totally impractical for realized quantum advantage. They do think that a quartic (n^4) speedup might result in real gains though! Keep that in mind as we move through the section. Second, they identify data input/output as a huge bottleneck for practical speedups. Even with their optimistic assumptions, it's going to be about 10,000 times slower to load data into or out of a quantum computer. This, in and of itself, can turn an exponential algorithmic speedup into a practical slowdown for problems with lots of input data. So what are we looking for in an application or algorithm? Exponential (or at the very least, high order polynomial, like n^4) speedups, and small input data problems. Unfortunately, for problems of optimization and machine learning, these criteria are not frequently

met.

With these constraints mind, let's examine the tools in the proverbial toolbox when it comes to these kinds of problems. First of all, we have Grover's algorithm. Basically any optimization function can be constructed such that it is the oracle in a Grover's implementation. As such, if you are naïvely brute force searching over the possible input space, a simple Grover's implementation can result in a quadratic speedup. There are some details in how to actually implement this, and these schemes are usually termed 'Grovers Adaptive Search' (GAS)[103][104][105]. So in a generic optimization problem, we can generally access a quadratic speedup over brute force search (but remember, we may have a better way of solving this problem classically than brute force!). This is intellectually quite interesting, but, as we've discussed, too slow to have practical impact in the world.

Our next family of approaches are called 'variational' or 'heuristic' based algorithms. Their most common manifestations are the variational quantum eigensolver (VQE) or the quantum approximate optimization algorithm (QAOA), which are very closely related[106]. These variational algorithms are, in a sense, a workaround to the realities of modern, NISQ-era quantum hardware. We certainly do not have systems that can run arbitrarily long series of gates, precluding the use of Grover-based schemes. The variational approaches were developed with the express intention of running them on noisy currently available hardware, which means they run relatively constrained and short gate sequences. They are also very explicitly quantum/classical hybrid algorithms.

In any variational algorithm implementation, the optimization problem has to first be encoded in a quantum way. The variable inputs that we want to optimize are mapped onto qubits, and the cost function (the thing that we are trying to optimize for) is mapped onto a Hamiltonian – the equation that represents the entire quantum state. The algorithm's goal is to end up with quantum state that maximizes or minimizes that Hamiltonian.

The actual process of running a variational algorithm, shown in Fig. 4.3 is pretty simple: We start with an initial guess of what an approximately good solution might be. The quantum computer runs a short quantum circuit consisting of several parameterized gates, creating an entangled quantum state that corresponds to that solution. That state is then measured, perhaps re-running multiple times to get the information it needs to evaluate the Hamiltonian/cost function. The classical computer then evaluates the cost function of the state, then adjusts the parameters of the quantum circuit (e.g., changes the angles of the gate rotations). This proceeds iteratively until an optimal solution is found!

This approach can be applied to a variety of optimization & math problems. However, variational algorithms are no magic bullet for these applications. First of all, they are heuristic based: their performance is highly dependent on the initial guess. If the initial guess is good, the solver might find an optimal solution quite rapidly. If it's bad, perhaps it will never find that solution! This makes it very difficult to establish whether variational methods can achieve meaningful speedups over classical approaches. They are so dependent on classical computing not just for performing mid-algorithm optimization, but also creating this all-important initial guess. This also does raise the question: if the algorithm is so

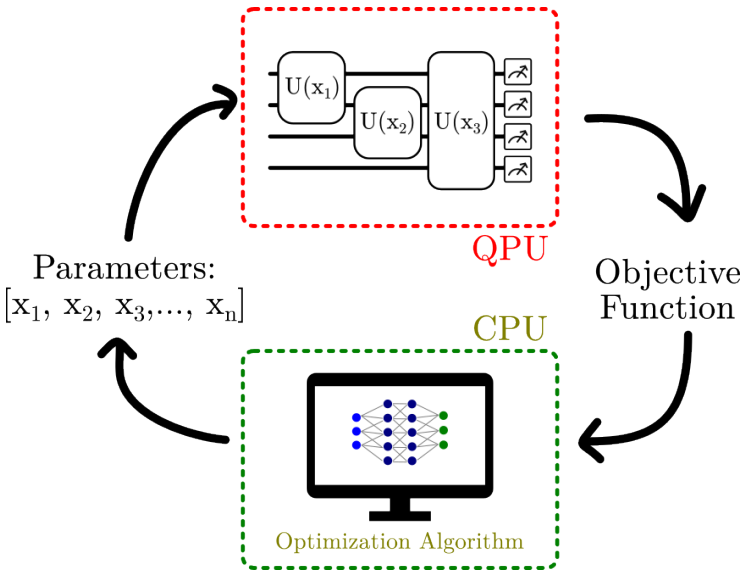


Figure 4.3: High level schematic of a variational algorithm, which involves a classical computer running some sort of optimization function on a set of parameters which are then fed into a quantum processor. The output of the quantum circuit represents the objective that the system is trying to minimize or maximize.

dependent on a good initial guess, why not just devote more classical resources to the problem? In the extreme case, you would just use a classical computer to perform the full optimization, and the quantum process would be left vestigial. For such an approach to be maximally effective, it needs to be easy to get a good guess of the solution, but very hard to get the exact solution. That's a very specific and difficult circumstance to find.

It does not seem like the VQE or the QAOA has found a killer app, let alone practical usage just yet. Once again, I can point to a narrative of 'team finds quantum algorithm (QAOA in this instance) outperforms classical approach for problem (Low Autocorrelation Binary Sequences)[107], until better classical approach is found[108]. One interesting theoretical result[109] found that QAOA scales better than the best currently known approaches for the k-SAT problem, a NP-hard problem dealing with Boolean logic.

So Grover's speedups are insufficient, variational algorithms are on shaky ground, what's left? Well, the HHL (Harrow, Hassidim, Lloyd) algorithm[110] has a proven exponential improvement over classical methods for solving certain sets of linear equations! Wait, what? Is this the great hope coming true? Well, not quite. Solving sets of linear equations is indeed a very useful tool, broadly applicable to many optimization problems, machine learning tasks, and various real-world applications. It takes polynomial time to solve systems of equations with classical methods, but the HHL algorithm can do so in logarithmic time

(exponentially faster!).

Unfortunately, there's a catch (or rather, several)[111]. First, the HHL algorithm only has an exponential speedup when you ignore how you load those systems of linear equations onto the quantum computer. Loading all those entries into a superposition naïvely takes too much time, and drops the algorithm out of the logarithmic regime. Workarounds are possible, but difficult: One could generate the system on the quantum computer using some other algorithm (but this doesn't apply to any generic problem) or one could load the system in superposition from a 'quantum memory' or QRAM (although it's unclear if building such a thing is possible). To make matters worse, HHL does impose some restrictions on the types of systems that can be solved. And finally, the output of the algorithm is not actually the answer to the system of linear equations – it's the quantum state of that answer, and cannot be directly measured without destroying some information! We might have to re-run HHL multiple times to get all the information needed to truly solve the system of equations, again destroying our speedup. This ultimately makes many schemes that rely on the HHL algorithm (most saliently, quantum machine learning) infeasible.

These are of course, the algorithms we have available today. It is entirely possible that a new algorithm could be created that bests these unwieldy beasts and leads to quantum dominance. However, those new potential algorithms still have to grapple with some of the structural challenges of trying to get quantum advantage on these sorts of tasks. First and foremost, it is challenging to load lots of data into or out of a quantum computer. In many cases, that is a rate limiting step, and may limit advantage.

Moreover, we've seen several times that successful exponential speedups require some sort of special 'structure' in the problem (e.g., Shor's). Arbitrary big data problems without structure will also have a difficult time achieving more than quadratic speedups, at least based on our current knowledge of quantum computer capabilities.

This seems to significantly limit the utility of quantum computing for optimization and machine learning applications. Classical computers, and especially GPUs, have proven themselves incredibly capable at handling problems that deal with bulk operations on massive sets of data. It seems that quantum computers 'desire' the opposite – limited data input/output, but complex and long calculations on said inputs. Hence we arrive at the 'killer app' of quantum computing: simulation.

4.4 Simulation

This is the original quantum computer application, and certainly the most exciting. Feynman outlined the concept best when he said: "nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical."[112] It intuitively makes sense that simulating complicated molecules, chemical interactions, and properties like superconductivity should be hard to do! Moreover, it's clear that classical computers, while having made

great strides, haven't yet cracked the code of molecular simulation. We do utilize simulators for drug design, material development, and fundamental physics research, but they are neither fast nor perfect. And the potential impacts are sky high when it comes to simulation: Superconductors[113]! Low-carbon fertilizer production[114]! Drug discovery[115]! Nuclear fusion and limitless energy[116]! Wowee!

As you can imagine, this is a wonderful story that the quantum computing industry loves to tell. But, is it true? I'd like to spell the chain of logic out here, and examine each link. Here's the high-level argument:

1. Classical computers cannot efficiently simulate molecules (or inter-molecular interactions) to a sufficient level of accuracy.
2. Quantum computers can efficiently simulate molecules to a sufficient level of accuracy.
3. Simulating molecules will lead to incredible, world-changing impacts.

I think point 3 is particularly important. While we will cover the status of the quantum computing 'market' in Chapter 5, the point of the applications I'm covering here is to provide value to someone. As I've discussed above, I don't find quantum computer usage for encryption to be particularly value-adding, and I don't think they will be useful at all for optimization or machine learning. So for me, the entire 'value' of the quantum computing industry (around 10 public pure-play quantum companies, dozens of startups, and some fraction of the market cap of Google, IBM, and Microsoft) rests on its ability to do something useful with simulation.

4.4.1 Classical capabilities

So let's start with point 1, and try to understand the classical state of the art. Chemical simulations run the gamut from routine to impossibly complex. Typically processes that involve macro-scale, classical interactions are more straight forward to model and solve (think back to modeling reaction rates in high school chemistry). However, a great deal of interest in the world of chemical simulation lies in properties that are inherently quantum.

Most important is the 'molecular ground state energy'. We've previously talked about ions and atoms in the context of qubit platforms, and noted that we typically encode $|0\rangle$ as the lowest energy state of the system, AKA the ground state. The system here is a single particle (ion or atom), but it has some Hamiltonian which specifies all its valid states (and their respective energies). More complex systems – i.e. molecules – also have Hamiltonians that describe their energy levels. We focus on the ground state energy because it's practically the most useful: Systems naturally go to their ground state energies, and understanding if a structure is stable; or if a reaction is favorable, depends on understanding the relative ground state energies of the system and its components. This propagates to virtually every practical and theoretical application of chemical simulation, classical and quantum:

- Drug discovery practically involves huge numbers of estimations of ground state energy to predict the relative efficacy and stability of drug candidates.
- Earlier, I mentioned low-carbon fertilizer production as a potential product of quantum computer-enabled simulation. Why? Well, humanity currently makes ammonia (fertilizer) using the fossil-fuel based Haber-Bosch process, which requires huge energy inputs (and associated emissions). Nature performs nitrogen fixation (turning atmospheric nitrogen into ammonia, just like Haber-Bosch) at room temperature and pressure using an enzyme called nitrogenase. We sort of know how nitrogenase makes the magic happen – it involves a helper molecule (called a cofactor) that catalyzes the reaction[117]. But we don't understand the actual steps by which nitrogenase and this helper, commonly called the FeMo cofactor, or FeMoco, do the fixation! If we could solve for the ground state energy of FeMoco, we could potentially help solve this mystery, and perhaps, unlock low-emissions fertilizer!
- High temperature superconductors have long been a holy grail for physicists, but understanding the phenomenon of superconductivity is notoriously difficult theoretically. The most common angle of attack is known as the Fermi-Hubbard model, which describes the highly-coupled interactions between electrons in a lattice. We'd love to be able to solve these models better, potentially unlocking how to engineer superconducting materials.

But I digress. We're talking about the capabilities of classical simulation. Clearly, we can already do some ground state energy estimation, since there are thousands of computational chemists in the world today, armed with nothing but classical computers, working on problems such as drug discovery, chemical synthesis, and molecular characterization. Unfortunately, these simulations may not always be straightforward. Calculating a molecule's ground state energy ostensibly requires taking into account the relative positions & energies of all of its component electrons. Those electrons are fundamentally quantum beasts, and thus have (potentially) quantum mechanically mediated interactions with their peers. To properly and completely simulate every one of those interactions is a very difficult game, and scales exponentially with the number of electrons or orbitals in a system. Computational chemists today have a variety of tricks to help simplify this process: the most simple of which is to ignore quantum effects altogether, looking only at 'molecular mechanics'[118]. This tends to be fast, but obviously less accurate than including quantum mechanical effects (how much less accurate is highly dependent on the system in question).

If quantum mechanical effects are to be included in a simulation, an array of computational tactics can be used, each with varying degrees of complexity. Again, the tradeoff is between speed and accuracy. The most base of these is called the Hartree Fock (HF) method. An HF simulation makes a simplifying assumption, wherein each electron only interacts with the average field of the other electrons (rather than having individualized interactions with each other particle). This assumption radically accelerates computation, at the cost of

accuracy. The relative accuracy penalty however, really depends on the kind of molecule! Chemists make distinctions between molecules with different kinds of inter-atomic correlations, some of which can be quite well-simulated by classical methods[119]. More complex molecules, however, may require much more advanced approaches. Various other methods essentially build on top of Hartree Fock and the closely related Density Functional Theory (DFT), culminating in the intimidating 'CCSD(T)' method*. CCSD(T) is often called the gold standard of computational chemistry[120], but it's no free lunch. It's application comes at a steep computational cost: Implementing the approach scales with $O(N^7)$†. For large molecules, that does in fact become impractical! But, these wily chemists have more tricks up their sleeves. They can segment larger molecules into sections, and perform more robust, time-consuming simulations on sectors where more local quantum interactions occur. After all, (this is a point that computational chemists like to make) quantum effects such as entanglement are delicate! To first order, two people are not entangled, and neither are two molecules. Generally, in large systems we wouldn't expect two electrons from opposite parts of the system to have quantum-mediated interactions. Thus you could imagine some sort of approach (and indeed, this is what chemists do today) that combines fast classical simulation with more precise, selective, quantum-aware aspects.

Moreover, there's a lot of industry interest in approaches which are even faster than these HF/DFT style strategies. We have enormous amounts of compute deployed in data centers for various machine learning and artificial intelligence applications. A natural question is to ask whether some of these ML approaches (and the copious infrastructure that is deployed to support them) can be used for chemical simulation. And indeed, there are a host of startups that are investigating neural network potentials as an enabling tool, given their favorable linear scaling and reasonable levels of accuracy[121]. This is still a relatively nascent approach, so we might expect to see significant gains on this front in the near future.

But what of this headline point that current day simulators are hamstrung and limited? Well, the truth is that it's complicated. Even with all our tricks and computational power, there are simulations we cannot do – there are any number of enzymatic reactions we've never been able to simulate. Moreover, some problems, such as simulating superconductivity, involve long-range electron interactions that are not well-suited for HF/DFT-style approaches.

And crucially, we'd like to be able to do many of these simulations fast. In drug design especially, chemists often need to perform huge quantities of iterative simulations; it's often an acceptable tradeoff to have lower accuracy but much faster simulations to increase throughput[122].

Even in other, more complicated simulation domains, the computational chemists are not sitting still. For a long time, this problem of simulating FeMoco has been a sort of north star for the field, especially for the quantum side. Any number of papers[123] have been written describing how many qubits would

∗. Coupled Cluster Singles and Doubles, with a perturbative correction for Triples

†. N here is technically the 'basis set size', but it's more useful to think of it as a proxy for how many atoms are in the molecule in question.

be needed to solve the problem, considered to be intractable with classical methods. And yet, in 2026, a group from Caltech led by Professor Garnet Chan, published results for their classical estimation of FeMoco's ground state energy (with chemical accuracy!)[124]. We'll return to the implications of this startling result shortly, but I think it's worth underscoring this point yet again: Classical approaches are always getting better!

4.4.2 Quantum computer enabled simulation

Claim 2 is that quantum computers will be able to efficiently do accurate chemical simulations. At least right now, the best known approach for quantum-enabled simulations is known as 'quantum phase estimation,' or QPE. Some of the variational algorithms discussed above could also be used, but at the expense of probability of success, total runtime, and/or accuracy. As a result, we will focus our interest on QPE, and assume its utilization in a world with fault tolerant quantum computers.

The operational principle of QPE (which is in fact used as a subroutine in Shor's algorithm) in chemistry applications is as follows[125]: Prepare a guess (often called an ansatz) of the ground state of a system. That is then encoded into qubit form. Run the QPE with the ansatz as an input, and it will project the ansatz state onto the real ground state, with a probability of success proportional to the overlap between the ansatz and the ground state. After that, measurement of the output qubits gives you the ground state energy! There are some complications I'm glossing over here (how do we encode molecular states into qubits?!), but this is schematically correct.

An additional wrinkle is 'time evolution' – i.e. how does the Hamiltonian evolve over time. This might be relevant to thinking about reactions actually occur, like a drug binding to its target. In many cases, where molecules are not strongly entangled, it may be straightforward to evolve the ground state over time with classical methods. However, some cases (photo-chemistry is an oft-cited example) nuclear dynamics become very important, and classical approximations break down[126]. In these cases, a quantum computer would be able to evolve the system natively by applying gates, in a process called Trotterization[127][128]. Unfortunately, things are a bit trickier than standard QPE on the read-out front, since it's not possible to extract the full quantum state in one-shot. You'd have to run the circuit over many shots to extract the full picture of the quantum state after evolution.

One potential problem with this whole QPE story is easy to see: We need a good guess of the ground state to be able to produce the ground state! If we start with a totally random ansatz, then the QPE will only succeed after exponential time, leaving us practically far behind the classical approach. The typical approach is to use a HF approximation as the ansatz, and proceed from there. A classical proponent would say, A) that HF (or a DFT approximation with similar computational effort) is probably good enough for a lot of applications; B) even getting to that HF ansatz is computationally difficult for lots of highly coupled systems like these Fermi-Hubbard models[129]; C) for these very difficult

problems (such as FeMoco) the HF approximation is quite poor, leading to much longer hypothetical quantum run-times[130]! A more subtle problem is that it might be difficult or time consuming to actually load that ansatz into the quantum computer, again eating away at any quantum advantage[131].

The quantum proponents say, look, there are lots of molecules where classical approaches today simply aren't tractable, but with a reasonably good ansatz we can use QPE (or even variational approaches) to get highly accurate ground states. This in fact is a really crucial element of the argument for quantum computers in chemical simulation: QPE makes a guarantee about how accurate its result is. However, the probability of getting to that result (corresponding directly to its runtime) is directly proportional to the quality of the ansatz. So we still run into the problem of ansatz quality! Nevertheless, these papers predicting theoretical run-times often make guarantees about chemical 'accuracy' (e.g., to X milli-hartrees)[132]. Our classical methods don't have such a guarantee built in[*].

I do think the quantum vision is cogent here, assuming the hardware is capable enough. The more-accurate CCSD(T) and its ilk have very unfavorable scaling (N^7, remember!) and we've already reviewed that making cubic or quartic polynomial speedups could result in practical quantum advantage. This seems to spell out an opportunity for real quantum speedups! Of course, both camps believe in the important role of classical simulation here. A successful quantum-enabled system would have tight integration of classical HPC resources with its quantum processor[133][134].

Time estimates of these quantum algorithms obviously vary depending on approach and molecule complexity. This paper[135] estimates that with qubit counts in the low thousands, the ground state energy for molecules like FeMoCo could be simulated in around ten hours. This one[136] says about 5 million qubits can solve cytochrome P450's (a liver enzyme) ground state in about a hundred hours. This seems to indicate that quantum computational approaches are probably more useful for situations where one desires high accuracy rather than high throughput.

But there are alternative visions for quantum-enabled simulation besides this QPE-based approach: First and foremost, there are many people working on analog simulators, typically utilizing ultracold atoms or molecules[137]. We've discussed analog approaches briefly before, but it's worth emphasizing how they are different then the digital quantum world that most 'quantum computers' live in. In the QPE approach above, a critical step is encoding the problem of interest into some sort of binary notation: one common approach is to represent each orbital as a qubit, such that if an orbital is occupied, that qubit would be in the $|1\rangle$ state. Then, the actual algorithm runs as a series of 1- and 2-qubit gates on those qubits (and some set of ancillas). The analog approach says hold on, why do that encoding at all? We're talking about continuous systems!

∗. Of course, if we were to continue the hypothetical dialogue, the classical computational chemist would probably say that cases where conventional approaches are unsuitable and high accuracy is needed are few and far between; AND that these GPU-accelerated approaches may soon make new classes of problems tractable.

Their approach leans on some physics I can't hope to understand, wherein when atoms/molecules get super cold, they begin to behave like smaller quantum particles. In their systems, they cool down particles, position them in a specific way that represents a problem of interest, then apply fields and make some measurements. This allows them to attempt to simulate these Hamiltonians and find ground states of interest, among other things. One particular hot topic for the analog simulators is the Fermi-Hubbard model we discussed earlier. This is a semi-idealized model of electrons moving through a lattice. The model is A) quite hard to simulate with conventional approaches, especially as its size increases; B) related to interesting physical phenomena such as superconductivity. You could also attempt to solve Fermi-Hubbard with 'standard', digital QPE. However, given the relative maturity and capability of gate-based computation (particularly limited circuit depth), analog approaches are currently in the lead here.

And then there are digital, gate-based approaches that aren't based on standard QPE. The aforementioned Garnet Chan has an interesting paper[138] proposing a high-order polynomial speedup for certain coupled cluster simulations using a novel quantum encoding technique. This is sort of orthogonal to typical QPE proposals since it is targeting speeding up simulations in a regime where classical methods have typically excelled.

And of course, I have to end with the ever present caveat – new algorithms might be developed! Who knows what might be possible when better hardware capabilities are available to researchers.

4.4.3 Quantum simulation & impact

Finally, let's say we do get these ground state energies. Do they change the world? This is a tricky question, and I'll tell you immediately that I won't be able to give a satisfying answer. Certainly, some people think yes. My personal belief is that accurate ground state simulation of complex systems (whether that be classical or quantum based) is potentially very valuable, but not world-changing.

Today, there are legions of computational chemists who routinely do ground state estimation in service of drug discovery. This is one step in a complex workflow, wherein candidate drugs are proposed then winnowed down until something is manufacturable, effective, safe, and commercially viable. For hit identification, throughput is a priority; huge quantities of candidates need to be screened. This seems to be a very good fit for more heuristic, tensor based approaches which excel in speed to solution rather than accuracy. On the other hand, different parts of the development process (i.e. structure based drug discovery) rely on more in-depth analysis of a smaller subset of candidates. If a highly accurate and sufficiently fast quantum approach was available, it could be useful. Continuing along this line of thinking: low-throughput, hyper-accurate simulations could be helpful for fundamental understanding of protein interactions (thus feeding into better selection of targets for future efforts).

There's this concept of 'Amdahl's law', which says that the speedup associated with optimizing a portion of a process is proportional to the time that portion is

actually used. Put otherwise, bottlenecks move! I'm sure just about every drug manufacturer in the world would take a better simulation tool, but it certainly wouldn't solve all their problems. There are many other bottlenecks in the drug discovery process, from target selection (what do we even need to simulate in the first place) to trials (do our drugs even work). So I think this all sort of eats into the story of quantum computing being revolutionary for this market, but leave the possibility of it being an important tool open.

On other fronts, things get murkier. There's a chance that simulation reveals something fundamental and clear about superconductivity or nitrogenase operation that immediately leads to groundbreaking new materials or processes. I can't rule that out! But it seems more likely that the information will be much more granular and hard to utilize. I'm going to quote Professor Garnet Chan directly here, from his blog post about his team's classical FeMoco simulation work:

> *The [...] proposition, that elucidating the reaction mechanism of nitrogenase will lead to a transformative societal impact, is [...] nuanced. The claim originates in the observation that the competing industrial process for fertilizer production via nitrogen reduction, namely, the Haber-Bosch process, takes place at high temperatures and pressures and consumes a significant percentage of the world's energy. Bacteria, on the other hand, can do this process at room temperature.*
>
> *While it is true that the nitrogenase enzyme functions at ambient temperature and pressure, it is simply false that it consumes much less energy. This is because the large amount of energy required for nitrogen fixation mainly originates from thermodynamics, i.e. one needs energy to break the strong nitrogen triple bond. In fact, taking into account the physiological conditions and the ATP cost, bacteria arguably expend more energy to reduce ammonia*[139] *than a modern efficient industrial implementation of the Haber-Bosch process. Thus the real hope behind trying to understand the nitrogenase mechanism in the context of societal impact is that we may one day engineer a variant of it with more desirable properties, e.g. with higher turnover, or with a lower carbon footprint, or which is more selective for nitrogen reduction. Whether this is actually possible remains to be seen, and certainly requires much more than knowing the ground-state of FeMo-co, or even the full reaction mechanism.*[140]

If you are at all interested in this question, I highly recommend reading the full blog post.

Where does this leave us for the question of quantum utility for simulation? I am loathe to give you an opinion to adopt whole cloth. To me, it seems possible that a fully developed, fault tolerant quantum computer could outperform classical computers at certain simulation problems, particularly when it comes to accuracy. The relative probability of that outcome though, depends on how

fast such a computer could be developed. Classical simulation approaches seem to be advancing quite rapidly, especially with interest from the ML/AI world overflowing into other application spaces. I think it is just as possible that in the next decade, we end up with extremely practical classical approaches for even large systems of interest, eating away at potential applications for a quantum computer. In the domain of Fermi-Hubbard, I think there's a lot of reasons to be optimistic about the scaling and utilization of analog quantum simulators to solve these problems on time-scales much sooner than FTQC.

This is the application space for quantum computing I feel most optimistic about, but that's more a reflection of my bearishness for spaces like encryption and machine learning rather than my excitement for quantum computer-enabled chemical simulation. Perhaps one way to frame my hesitancy is this: If all the money flowing to quantum computing went to computational chemists, where would the field be today?

4.5 On Value

I reject categorically the notion that all research and development has to be done with a specific consideration for applications and end-user value.

Application-unoriented research and experimentation has yielded many world changing innovations (electromagnetism, the internet, X-rays, etc.). Similarly, research intended for one use case has often gone to result in significant impacts for other industries: today's semiconductor manufacturing processes rely on extreme ultraviolet (EUV) lithography machines which use laser technology originally developed for defense, fusion, and medical device appications[141].

And yet, I think at some point you do have to do a cost-benefit analysis of the impacts of funding a direction of research, especially when it is becoming more mature. Quantum computing is at a strange place, where it has decades of theoretical & algorithmic research; ever-increasing hardware capabilities; a still uncertain and long timeline to fault tolerant operation; and a cloud of financialized hype growing around it. We'll discuss this last point further in the next chapter, but I want to take a moment to review our survey of applications and try to draw some conclusions.

Going down the line:

- It's interesting to perform quantum-enabled calculations for their own sake, but they have no real world impact.
- Breaking encryption schemes certainly is a real, quantumly-advantaged application. But the world is rapidly moving toward post-quantum encryption schemes that will make quantum computers inefficient at decryption. Perhaps there will be some impact for hackers and bad actors trying to harvest old data for value, or exploit vulnerabilities in cryptocurrency protocols. But it's not very... exciting.
- Machine learning and optimization are unnatural fits for quantum computers, which have problems with data throughput and no general way to

access exponential or high-order polynomial speedups for these kinds of problems. There might be advantage on the table for extremely specific, quantum-native problems, but it's certainly not generic. Meanwhile, classical GPU-accelerated methods are getting so good here, and show no rate of slowing their rate of progress. It seems doubtful that this will provide long term value!

- Quantum computing for simulations is a real application, but not a panacea to every difficult chemistry/physics problem. In general, classical methods are sufficient for the vast majority of problems that computational chemists deal with today. Quantum approaches may be able to get higher accuracy solutions than conventional computers, but not at high speed. This will probably be an unfavorable tradeoff for applications that demand high throughput, such as drug design or materials discovery. It may be transformational for problems that are (currently) intractable for classical approaches, such as studying superconductivity. However, even for these: A) analog quantum simulators show a lot of near-term promise, perhaps eating away into the impact of quantum computers; B) classical approaches are always getting better; C) solving the specifically scoped mathematical problems in question does not translate directly into some sort of world-changing discovery. The most likely future here is that (if successfully developed) quantum computers get integrated into high performance computing nodes at some universities and national labs, and are a tool in the proverbial toolbox for computational chemistry.

4.6 Not-quite-parting Thoughts

And so, we've learned all we're going to learn on the technical side. We've covered theoretical computer science, semiconductor physics, and computational chemistry, among other topics. I'm reasonably convinced that if you've read and internalized the above, you'll be able to be 'quantum literate'. Will you be able to understand the proof on some quantum algorithm paper? No, probably not. But can you parse through headlines and understand things at a reasonable level? Hopefully!

Let's try using our noggins and working through a case example: At the time of writing, the top-voted paper on Scirate (a sort of community voting platform for quantum computing papers) is "Fault-Tolerant Quantum Computing with Trapped Ions: The Walking Cat Architecture ". OK! The title is pretty straightforward: this is a proposal for a quantum computer architecture that uses trapped ions that the authors think could enable fault-tolerant operation. We can read the abstract and go deeper:

> *We propose a fault-tolerant quantum computer architecture for trapped-ion devices, which we call the walking cat architecture. Our blueprint includes a compiler, a detailed description of all the quantum error-correction protocols, a micro-architecture, a sufficiently fast decoder,*

> *and thorough simulations. The backbone of the architecture is a cat factory, producing cat states distributed throughout the machine, which are consumed to perform logical operations. The walking cat architecture is based entirely on a modern quantum error-correction approach called low-density parity-check (LDPC) codes. [...] Our dense architecture provides a design with 110 logical qubits executing about one million T gates per day using only 2,514 physical qubits. We estimate that the quantum Hamiltonian simulation of a Heisenberg model on 100 sites can be executed within one month with 10,000 physical qubits, including all shots required to achieve chemical accuracy, suggesting that such a device could enter the regime of classically intractable physics simulations. [...]*[142]

This makes things a bit clearer: they're going to use LDPC codes, which we've covered (and certainly it makes sense for a ion-based computer to use a LDPC approach, which would utilize non-local interactions). They figure out some decoding schemes for error correction, do resource estimates for a fault tolerant system, and do some modeling of system throughput for a real-world problem. A headline takeaway might be the density of their logical qubit encoding: a ratio of approximately 23 physical qubits per logical qubit is a very impressive figure, and one that might merit some additional scrutiny or skepticism. Doing a close reading might be out of our comfort zone, but we now have the tools to parse what's going on!

Not every paper or press release will be so simple. But I hope that this work has given you the tools and mental models to think about quantum computing in a structured and critical way.

Chapter 5

State of Play

It's time to leave the science and engineering behind, and turn towards the 'real' world, full of companies and employees and investors and financialization. Quantum computing is no longer a pet project of academia; it is a proper industry. Companies in this space have become increasingly mature and capable of raising large amounts of capital. In 2025 alone, the quantum industry raised $4.9 billion in venture capital money[143], and around $12.6 billion in total capital, a roughly 6x increase from 2024[144]. There are several public quantum computing companies, with several more coming along the way. At the time of writing, the Trump administration has pledged an additional $2 billion in investment[145], making US government support of the industry explicit. Governments in the EU and Australia have also made quantum computing a key plank of future technological development strategy[146][147]. There are enormous expectations and stakes for the field, a far cry from the situation from even 5 years ago.

Why is this happening, and why now? Why is this sleepy, academic field suddenly an enormous sink for public, venture, and retail capital? The short answer is that the field A) has tremendous capacity for absorbing capital, given the massive technological hurdles it must overcome; B) has an enormous amount of hype swirling around it.

But quantum computing is not a 'natural' fit for most of these types of capital. Despite the growing maturity and size of the industry, the technology is extremely nascent, the applications uncertain, and the revenues miniscule! Let's go down the list: funding high-risk technological bets is indeed the purview of government agencies like the NSF, DOE, and DOD (through innovation arms like DARPA). As a result, much of the initial research funding for quantum academic labs came from these organizations - and government funding remains an important capital source to this day. Then, quantum startups and spin-outs began to be founded in the 2000s, and attracted both this public funding and venture capital investments.

But historically, VC money hasn't typically gone to industries like quantum computing. Despite its roots in funding the California semiconductor companies (Intel, Fairchild, etc.), the vast majority of venture dollars have gone to

capital-light software companies, which promise investors huge returns per dollar invested. Hardware companies, by comparison, have significantly greater capital needs, considering the cost of fundamental R&D, equipment, manufacturing, etc. Quantum computing is an extreme example of this, where the technical and market risk of the industry is still extremely high, even decades in. And yet, in the past two years especially, massive amounts of capital have piled into the field from venture capital firms. Consequently, the universe of quantum startups has exploded, consisting of over 200 firms targeting different hardware platforms, applications, and places in the software/hardware stack.

It's also a bit of a historical aberration that there are several public, pure play quantum computing companies. Historically, public markets have been reserved for companies that have a history of profitability, and going public was a way to simultaneously reward existing investors with liquidity; raise additional capital from the bigger public markets; and expand access to the company's growth to the common retail investor (sounds a bit socialistic, no?). Over time, these expectations have shifted. In the mid-2010s, startups like Facebook, Uber, and Snap went public with no profits! Uber's IPO enabled the company to finally consolidate market share, raise prices, and become profitable. Snap meanwhile, has tread water, never really achieving profitability.

In the early 2020s, we saw companies with even more speculative and uncertain business models go public, typically via a 'Special Purpose Acquisition Company', or SPAC. A SPAC is a public entity which raises money from investors, typically at $10 per share. It then goes out and tries to use its big pot of cash to find a target to take public. The SPAC often also finds some private investors who will co-invest in the target company, and brings it to its shareholders for a vote. If the vote succeeds, the firms 'de-SPAC', merging into one publicly traded entity. The SPAC shareholders can either get shares of the company, or their original money back, plus interest[148]. Advocates for SPACs tout that the process is lower cost and lower risk compared to the traditional IPO. Detractors point to a number of facts: That SPAC sponsors (the people organizing the SPAC) get a huge bucket of shares for a very small initial investment, misaligning their incentives from the SPAC shareholders[149] and that historically de-SPAC'ed companies have performed very poorly once they've gone public[150]. Ostensibly this all happens because SPACs have lightened regulatory scrutiny than IPOs, and generally target more speculative and less mature businesses.

Several quantum companies went public during this 2021-22 SPAC boom – IonQ, Rigetti, and D-Wave, raising about a billion dollars cumulatively. To me, what makes these companies singular, especially in comparison to their public cohort, is their lack of revenue. There are commonly accepted ways to translate a company's income into estimates of its total enterprise value: most simplistic is the idea of a price to earnings ratio (P/E ratio). A typical P/E ratio might be something like 10 to 30x, which sort of makes sense. If you were to buy the company for a price that was 20x its earnings, you'd expect to return your investment in 20 years with no growth, and hopefully faster if the business continues to grow.

These public quantum companies not only have negative earnings (they

are unprofitable), but have virtually no revenue, and likely won't be making substantial amounts anytime soon. This is in stark contrast to even the most rickety tech IPOs of recent years. Snap, which went public in 2018, had about half a billion dollars in revenue its first year of being public (it was unprofitable, but that's a different story - investors were paying for growth!); IonQ had about $10 million.

This trend has continued even today. After a few quiet years for IPOs and SPACs, 2026 has been a boon for new entrants into public markets: Infleqtion, Xanadu, Horizon Quantum, and Quantinuum have all gone public (Quantinuum notably via a traditional IPO!). While their revenue figures are not substantial, their enterprise valuations range between single digit and tens of billions of dollars.

So quantum computing is an outlier in both public and private markets. I can point to a few interrelated reasons why this is the case:

- For venture capitalists, we are at a unique time where their typical bread and butter – software companies – are being devalued due to the rapid deployment of AI. Silicon valley VCs have quickly reoriented toward all things hardware (deep-tech, hard-tech, or 'American Dynamism' are all hallmarks of this). Quantum computing, with its incredible technical hurdles, huge potential market size, and possible national security implications, becomes catnip to a VC in 2026. Furthermore, VCs have a documented pattern of (and incentive to partake in!) trend following: The VC profit model depends on one out of ten (or so) of their portfolio companies becoming huge, runaway successes. They are perfectly happy investing in nine stinkers if that tenth company is a hit, and can return the value of the fund and then some. This naturally means they are highly motivated by FOMO, or the fear of missing out. If their peers are clustering into an industry, a VC would see it as being extremely high risk not to also be in that industry. This helps explain the waves of VC money flowing into crypto, then AI, and now quantum.

- For governments, quantum computing clearly has applications for the intelligence community, with its possible applications for encryption and code-breaking. But there's more than just that. Much of the west, America especially, experiences existential dread about the concentration of the semiconductor industry's expertise in Taiwan, and the downstream national security and economic implications of this. I reckon that, in the eyes of many in the US government, the development of quantum computing could present a strategic hedge against this dependence. I think this is a hard story to buy, given that quantum computation will always work in concert with conventional computing, and certainly never displace it for general tasks – but I do think it's a story that Washington DC believes. For the EU and Australia, the incentive toward developing a local, highly-skilled tech industry is much more straightforward (and plausible).

- I hesitate to translate the behavior of retail investors into a coherent nar-

rative. I couldn't tell you why the average retail investor chooses to invest in IonQ instead of NVIDIA (let alone IonQ versus D-Wave), but I can make some guesses. It's been well documented that we live in a time of increased perception of economic precarity, due to (pick one of) AI deployment/increased inequality/stagnant wages/poor consumer sentiment. This then translates into people wanting to take big asymmetric bets in order to try and make huge profits (and escape the so-called permanent underclass). We see this in the increasing popularity of sports gambling, prediction markets, and short-dated options trading, all of which present high leverage opportunities to strike it rich[151]. Couple this with the public valorization and high profile of tech venture capitalists, whose business model is commonly associated with high risk bets on early-stage technologies. To me, putting money into speculative quantum computing stocks is spiritually akin to all of these activities. Sure, you can't go back in time to get into NVIDIA when it was $10 a share – but maybe IonQ will be the NVIDIA of tomorrow? Worth a shot!

- The quantum computing companies, who have an almost limitless ability to absorb new capital and direct it toward their formidable technical challenges, do their part as well. This is a bit of a hot-button topic, but it is well-understood that quantum computing companies (especially the public ones) are prone to exaggeration about both their technical capabilities and their relevance for real-world use cases. Here's an extremely non-exhaustive list:
  - We've already covered the debacle that is Microsoft's topological qubit program.
  - There are myriads of papers and press releases that claim some sort of practical business use case for near term quantum computers, none of which stand up to any real scrutiny. Take this HSBC/IBM paper[152], which found some sort of advantage of using a quantum algorithm on predicting financial data, but only because of the noise in the system? Scott Aaronson has a nice takedown here[153] but the short story is that it's absolutely bogus, and does not represent any real advantage over classical methods! There's no shortage of papers like this: IonQ claiming accelerated finite element analysis with Ansys[154] and applications for image processing[155]; D-Wave optimizing Ford production lines[156]; Quantinuum saying that quantum natural language processing has promise[157]; etc., etc. The purpose of these papers are for external consumption, to convince third parties that business adoption of quantum computing is imminent.
  - The industry has at times been extremely aggressive about qubit count projections, typically failing to meet them and adjusting them downwards after the fact. Famously, initial SPAC presentations from companies like IonQ and Rigetti were overly optimistic, predicting thousands of qubits by 2025/6, which certainly hasn't happened[158][159].

IBM also fell victim to this, with a 4,000 qubit 'Kookaburra' system slated for 2025 (according to a 2022 timeline)[160] – that has not come to fruition. Given the embarrassing nature of missing deadlines, timelines for these companies and others have been pushed back and de-emphasized. That being said, even today, IonQ says they'll have 10,000 physical qubits by 2027[161]! That's roughly two orders of magnitude of progress in 18 months. We'll see if it happens.

- Short reports* have indicated that IonQ has at times outright lied about its technical capabilities, financial situation, and market prospects. Rumors abound that the company's acquisition of Oxford Ionics in 2025 was driven primarily by poor internal technical results.[162][163]

It's very difficult for the average person (or even the sophisticated investor) to parse all of this and determine what's BS or not.

I also want to take a second to explain why all of these companies with substantial technical risk are going public now. Rigetti, IonQ, and D-Wave are sitting on huge amounts of cash from their SPAC events, even years later. This is a great luxury to have when you aren't selling any quantum computers. But simply being public can be a great asset for a business. It allows a company to use its stock as a component of acquisition deals. We've seen IonQ acquire a number of potentially complimentary businesses (most notably Oxford Ionics, a prominent competitor, and Skywater Technology, a semiconductor fab). Additionally, these quantum stocks tend to be incredibly volatile. Since option prices reflect the underlying volatility of a stock, this volatility can be monetized by hedge funds. We've thus seen transactions like IonQ's 2025 issuance of $2 billion in warrants[164].

5.1 State of Play

So where has all this investment gotten the field? Let's do an overview of the state of the industry. We've got a handful of different platforms and dozens (if not hundreds) of startups, all jockeying for position.

In superconducting land, things seem dominated by the big tech giants. Google's Willow chip has 105 functioning qubits, and they seem to be pretty open about their technical results. IBM's qubit count is slightly higher (∼150 qubits on their Heron chips), and they are unique in their ability to offer public API access to their systems. IBM has historically used honeycomb connectivity for its qubits, but is moving toward a more standard square grid in its next generation of chips, called Nighthawk. D-Wave recently jumped into the superconducting ring as well, acquiring Quantum Circuits Inc., a superconducting startup that promises lower error rates by utilizing a 'dual rail' architecture, which combines two transmon qubits into one unit. There are also several well-funded superconducting startups.

∗. A short report is an exposé about a company by an investor, detailing some sort of poor performance or shady business practices which should hurt the company's value. The investor publishes the short report, discloses a short position, and hopes that the market responds and pushes the stock down.

Particularly notable is Qolab, a new startup started by several UCSB and Google Quantum alumni, including Nobel prize winner John Martinis. They seem to have a particular vision for scalability in superconducting systems[165], which has historically been an enormous challenge. French startup Alice and Bob is also quite prominent – their superconducting qubit architecture (called a 'cat qubit') is supposed to be more naturally resilient to bit-flip errors, thus reducing the overhead necessary for error correction[166].

In trapped ion world, the elephant in the room is obviously IonQ. They've recently bolstered their technical capabilities by acquiring Oxford Ionics, who holds the record for two qubit gate fidelities, utilizing their electronic control scheme. I'm not particularly clear on how the company will integrate its legacy approach with OI's very unique electronic architecture (perhaps abandoning their initial efforts altogether?), but it does seem like they hold both some interesting technology and a large reserve of capital they are willing to deploy. Several other trapped ion startups exist, with varying degrees of maturity. Quantinuum, the Honeywell spin-out, is probably the most significant. As an industry outsider, I simply see what people post online (and have discussions with folks in the industry occasionally), but I perceive a bit of cooling enthusiasm for the trapped ion approach. Recent papers working to do time estimates for various decryption tasks have attempted to benchmark for both 'fast' and 'slow' systems (with ions falling into the later), finding that the time penalty for slow platforms can significantly degrade their utility.[167]

On the other hand, it feels like neutral atoms are on their cultural ascendency. They seem to be a good middle ground between the stable but slow operation of ions, and the fast but difficult to control and scale superconducting domains. Google recently announced it is putting resources into building a neutral atom based team, and startups abound in this space. Atom Computing (to my knowledge) currently holds the record for qubit count, at 1,180[168] – although it seems that their competitor QuEra has more impressive error correction experimental results[169] (due to better gate fidelities). I noted above that atoms have lots of potential, but have lagged behind their peers simply because of their relative immaturity. This may change soon!

The photonic companies sort of seem to exist in their own world. As I wrote above, there are a great variety of technical approaches here. The most far-along seem to be PsiQuantum, a very well-funded Australian startup, and Xanadu, a Canadian startup. Some of their technical progress has been very impressive, but to my knowledge, no true measurement based quantum computing system has been deployed as of yet. The technical challenges for this subset of the industry are really quite daunting, from single photon generation to reducing photon loss to extremely high bandwidth control.

On the spin qubit front, Intel is the established tech company working in this space, although it's unclear the extent of their resources they are putting in this direction. There are a variety of spin startups, including Australia's SQC and Diraq, and the EU's Groove. Qubit counts here are relatively low (low tens)[170].

I also want to mention the hodgepodge of startups working on other parts of the quantum stack. Some are focused on being infrastructure providers to

the industry: take Maybell Quantum, which makes dilution refrigerators. Others focus on how to translate software to actual gate operations (akin to a compiler), like Q-CTRL. Others sit on the applications side, trying to use existing quantum computers to solve business problems. I generally am extremely skeptical of this last camp, given that I think it's extremely unlikely that generic business problems are going to be better served by a quantum algorithm than a classical approach, especially in the near term. I don't want to name names, but there are some truly specious ventures out there.

Because progress is so nonlinear here, it's hard to tell who, if anyone, is in the proverbial lead. You could imagine a system that is very easy to take to 100 qubits, which then faces enormous challenges networking to systems with thousands of qubits (or perhaps the bottleneck is at 1000, or with control, or error correction, etc.). I think this makes for a very fascinating, if technically opaque, field to follow.

5.2 A Bearish Read

We seem to be at the peak of the hype curve for quantum computing. Public quantum stocks cumulatively sit at something around $40-80 billion in total market capitalization (with enormous volatility from day to day). New VCs and investors are pouring into the space, with expectations of getting returns on their investment in the next 5 to 10 years. All of this implies the creation of some short to medium term value being created by these companies. I think this is ultimately unlikely: A) the timeline for dealing with the very integral technical issues involved in scale-up and scale-out (across all platforms) is significant, and seems closer to a decade+ problem; B) real-world applications for these devices are few and far between, and while carefully crafted corporate partnerships can go a long way for generating hype, it's difficult to actually build revenue and profitability if your customers fundamentally don't get utility out of your product; C) even putting aside the last issue, who captures the value here? Let me clarify – when a big pharmaceutical company does a drug discovery project, they ostensibly use a bunch of supercomputing resources (chips from AMD and Intel, memory from lots of vendors, system assembled by HPE, etc.). They pay a price for that equipment, sure, but when they discover a new blockbuster drug, they accrue the vast majority of the value. It remains to be seen how that will play out in the quantum space.

I believe the market is suffering from some degree of irrational exuberance here. We have been collectively trained to revere and respect 'the science', and technologists are naturally optimistic about the potential of the approaches they are working on. This has made it relatively easy for an industry which is both very difficult to understand and has plausible sounding real-world impacts to become a giant sink for capital. I think what's interesting (and this is based on my discussions with scientists and engineers) is that many people working in the field either A) are narrowly focused on their particular niche, and assume that the technology will have massive impact; B) know that the larger narratives

around quantum computing are hype-based overselling. Unfortunately, in this world, everyone is strongly incentivized to keep this momentum going: The raised profile of the industry enables more funding, more jobs, more research, and more development. If you're a true believer in the long-term impact or a physicist looking for a stable job, these billions help! No one has the incentive to pull the alarm and talk about the long-term issues facing the industry.

I furthermore worry that the purpose of these recent SPAC and IPO events is not to provide these companies with sustainable long-term funding, but to enable founders, leadership, and investors with easy off-ramps for liquidity. Certainly venture investors are not typically used to operating on timescales of multiple decades, and recognize that today's market environment is a somewhat unique opportunity.

I don't mean to portray the folks working in this industry as being 'evil' (whatever that really means), although I do think some, especially in leadership positions, are acting in bad faith. I think there are a lot of people in the field trying to do good work and high-quality research. Unfortunately, financialization is a process that can damn even the best of intentions.

5.3 On Conceptual Art

Despite being an engineer, I have more than a passing interest in 'art', especially the kinds which are (I would say unfairly) maligned by the STEM crowd. There's one artist I'd like to discuss right now: Richard Serra. Serra was a post-war American minimalist sculptor, famous for his large steel installations. Neither words nor images properly convey the feeling of standing next to (or within) a Serra, particularly his more giant works. These are huge panels or blocks made up of solid steel, both hulking and delicately formed. Sometimes the panels form waves or arcs that loom over you or invite you inside to explore. There's been a lot written about Serra's work, its influences, and purposes (cliff notes: he loved going to the San Francisco shipyard as a child, he worked at steel plants as a young man, the sculptures manipulate how a viewer interacts with the environment and space, the art is physical and immediate, etc). I have always appreciated his work not just for these reasons, but also because of how absurd it is. Steel is this tremendously useful material, with human production dating back several thousands of years. We've harnessed raw elements to create an industrial, multinational supply chain to generate this multi-purpose material at scale. It goes into our buildings, our machines of war and transport, and our conceptual art. This mad artist has to go and say, what if we use this industrial process to create these unique, hulking forms that provoke thought; that confuse; that delight. Then the supply chain goes and bends to his will – some set of engineers and laborers works to manufacture these incredible shapes, then they get shipped around the world and installed in museums and public spaces for the public to puzzle at. It strikes me as this phenomenal trick that Serra is playing on the world. And that's not to say that I don't like it – I love it! What greater dominance can man have over the elements than to take raw materials

Figure 5.1: Richard Serra - Four Rounds: Equal Weight; Unequal Measure, housed at the Glenstone Museum. Image from the David Zwirner gallery.

and turn them into something (pseudo-)decorative? It's glorious! Both the works themselves and the narrative of their creation is beautiful.

And maybe that's how I feel about quantum computing. If the applications never pan out; if the algorithms never find real world uses; even if the technical advances end up so niche that the investment was all for naught; to me, it is equal parts awe-inspiring and beautiful that we can manipulate the quantum world with such precision – that we could even think about entangling two particles at will, or fight back against the process of decoherence. Is it worth doing? Perhaps; perhaps not. Nevertheless, I will appreciate the outcomes, not dissimilarly to how I might look at Serra's steels.

Notes

1. Charles H. Bennett and Gilles Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical Computer Science* 560 (December 2014): 7–11, ISSN: 03043975, accessed April 26, 2026, https://doi.org/10.1016/j.tcs.2014.05.025, https://linkinghub.elsevier.com/retrieve/pii/S0304397514004241.

2. Ignacio F. Graña et al., *Materials Discovery With Quantum-Enhanced Machine Learning Algorithms*, arXiv:2503.09517, March 12, 2025, accessed March 6, 2026, https://doi.org/10.48550/arXiv.2503.09517, arXiv: 2503.09517[cond-mat], http://arxiv.org/abs/2503.09517.

3. Michal Krelina, "Quantum technology for military applications," *EPJ Quantum Technology* 8, no. 1 (November 6, 2021): 24, ISSN: 2196-0763, accessed February 26, 2026, https://doi.org/10.1140/epjqt/s40507-021-00113-y, https://doi.org/10.1140/epjqt/s40507-021-00113-y.

4. Manuel S. Rudolph et al., "Generation of High-Resolution Handwritten Digits with an Ion-Trap Quantum Computer," *Physical Review X* 12, no. 3 (July 15, 2022): 031010, accessed May 9, 2026, https://doi.org/10.1103/PhysRevX.12.031010, https://link.aps.org/doi/10.1103/PhysRevX.12.031010.

5. Dylan Herman et al., "Quantum computing for finance," *Nature Reviews Physics* 5, no. 8 (July 11, 2023): 450–465, ISSN: 2522-5820, accessed March 20, 2026, https://doi.org/10.1038/s42254-023-00603-1, arXiv: 2307.11230[quant-ph], http://arxiv.org/abs/2307.11230.

6. Axel Ciceri et al., *Enhanced fill probability estimates in institutional algorithmic bond trading using statistical learning algorithms with quantum computers*, arXiv:2509.17715, September 22, 2025, accessed June 17, 2026, https://doi.org/10.48550/arXiv.2509.17715, arXiv: 2509.17715[quant-ph], http://arxiv.org/abs/2509.17715.

7. "This Month in Physics History," accessed June 17, 2026, https://www.aps.org/archives/publications/apsnews/200805/physicshistory.cfm.

8. "40 years of quantum computing," *Nature Reviews Physics* 4, no. 1 (January 2022): 1–1, ISSN: 2522-5820, accessed May 29, 2026, https://doi.org/10.1038/s42254-021-00410-6, https://www.nature.com/articles/s42254-021-00410-6.

9. Paul Benioff, "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines," *Journal of Statistical Physics* 22, no. 5 (May 1980): 563–591, ISSN: 0022-4715, 1572-9613, accessed February 23, 2026, https://doi.org/10.1007/BF01011339, http://link.springer.com/10.1007/BF01011339.

10. Richard P Feynman, "Simulating Physics with Computers."

11. Yuri Manin, *Mathematics as Metaphor: Selected Essays of Yuri I. Manin* (American Mathematical Society, 2007).

12. David Deutsch, "Quantum theory, the Church–Turing principle and the universal quantum computer," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*

400, no. 1818 (July 8, 1985): 97–117, ISSN: 0080-4630, accessed May 29, 2026, https://doi.org/10.1098/rspa.1985.0070, https://doi.org/10.1098/rspa.1985.0070.

13. Peter W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Journal on Computing* 26, no. 5 (October 1997): 1484–1509, ISSN: 0097-5397, 1095-7111, accessed June 17, 2026, https://doi.org/10.1137/S0097539795293172, arXiv: quant-ph/9508027, http://arxiv.org/abs/quant-ph/9508027.

14. William Kretschmer et al., *Demonstrating an unconditional separation between quantum and classical information resources*, Version Number: 2, 2025, accessed June 18, 2026, https://doi.org/10.48550/ARXIV.2509.07255, https://arxiv.org/abs/2509.07255.

15. A. Holevo, "Bounds for the quantity of information transmitted by a quantum communication channel" (1973), accessed June 18, 2026, https://www.semanticscholar.org/paper/Bounds-for-the-quantity-of-information-transmitted-Holevo/0733393a545b62e49e9fc4a76d53d843cfb72010.

16. Ehud Altman et al., "Quantum Simulators: Architectures and Opportunities," *PRX Quantum* 2, no. 1 (February 24, 2021): 017003, ISSN: 2691-3399, accessed November 20, 2025, https://doi.org/10.1103/PRXQuantum.2.017003, https://link.aps.org/doi/10.1103/PRXQuantum.2.017003.

17. David Deutsch and Richard Jozsa, "Rapid Solution of Problems by Quantum Computation," *Proceedings: Mathematical and Physical Sciences* 439, no. 1907 (1992): 553–558, ISSN: 0962-8444, accessed June 18, 2026, https://www.jstor.org/stable/52182.

18. Dorit Aharonov and Wim van Dam, "Adiabatic Quantum Computation is Equivalent to Standard Quantum Computation."

19. Lov K. Grover, *A fast quantum mechanical algorithm for database search*, arXiv:quant-ph/9605043, November 19, 1996, accessed April 19, 2026, https://doi.org/10.48550/arXiv.quant-ph/9605043, arXiv: quant-ph/9605043, http://arxiv.org/abs/quant-ph/9605043.

20. W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," ADS Bibcode: 1982Natur.299..802W, *Nature* 299 (October 1, 1982): 802–803, ISSN: 0028-0836, accessed May 6, 2026, https://doi.org/10.1038/299802a0, https://ui.adsabs.harvard.edu/abs/1982Natur.299..802W.

21. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer."

22. Peter Gutmann and Stephan Neuhaus, "Replication of Quantum Factorisation Records with an 8-bit Home Computer, an Abacus, and a Dog."

23. "More Quantum Computing with Fewer Qubits? Meet our New Error-Correction Code," Alice & Bob, accessed May 6, 2026, https://alice-bob.com/blog/more-quantum-computing-with-fewer-qubits-meet-our-new-error-correction-code/.

24. "Introducing Helios: The Most Accurate Quantum Computer in the World," accessed May 6, 2026, https://www.quantinuum.com/blog/introducing-helios-the-most-accurate-quantum-computer-in-the-world.

25. Victor V. Albert and Philippe Faist, "The Error Correction Zoo," 2026, accessed April 19, 2026, https://errorcorrectionzoo.org/list/quantum_transversal.

26. Pedro Sales Rodriguez et al., "Experimental demonstration of logical magic state distillation," *Nature* 645, no. 8081 (September 2025): 620–625, ISSN: 1476-4687, accessed April 19, 2026, https://doi.org/10.1038/s41586-025-09367-3, https://www.nature.com/articles/s41586-025-09367-3.

27. Craig Gidney, Noah Shutty, and Cody Jones, "Magic state cultivation: growing T states

as cheap as CNOT gates," arXiv.org, September 26, 2024, accessed April 19, 2026, https://arxiv.org/abs/2409.17595v1.

28. N. Lacroix et al., "Scaling and logic in the colour code on a superconducting quantum processor," *Nature* 645, no. 8081 (September 2025): 614–619, ISSN: 1476-4687, accessed April 19, 2026, https://doi.org/10.1038/s41586-025-09061-4, https://www.nature.com/articles/s41586-025-09061-4.

29. Rajeev Acharya et al., "Quantum error correction below the surface code threshold," *Nature* 638, no. 8052 (February 2025): 920–926, ISSN: 1476-4687, accessed April 19, 2026, https://doi.org/10.1038/s41586-024-08449-y, https://www.nature.com/articles/s41586-024-08449-y.

30. Vladislav D. Kurilovich et al., "Correlated Phase Error Bursts in a Gap-Engineered Superconducting Qubit Array," *Phys. Rev. X* 16 (2 2026): 021025, https://doi.org/10.1103/1bl4-b2f7, https://link.aps.org/doi/10.1103/1bl4-b2f7.

31. *NVIDIA Ising Introduces AI-Powered Workflows to Build Fault-Tolerant Quantum Systems* [in en-US], April 2026, accessed July 4, 2026, https://developer.nvidia.com/blog/nvidia-ising-introduces-ai-powered-workflows-to-build-fault-tolerant-quantum-systems/.

32. Craig Gidney, "Quantum Error Correction goes FOOM," accessed May 28, 2026, https://algassert.com/post/2503.

33. Xavier Waintal, "The Quantum House Of Cards," *Proceedings of the National Academy of Sciences* 121, no. 1 (January 2, 2024): e2313269120, ISSN: 0027-8424, 1091-6490, accessed April 28, 2026, https://doi.org/10.1073/pnas.2313269120, arXiv: 2312.17570[quant-ph], http://arxiv.org/abs/2312.17570.

34. Gil Kalai, *The Argument against Quantum Computers*, arXiv:1908.02499, August 7, 2019, accessed October 21, 2025, https://doi.org/10.48550/arXiv.1908.02499, arXiv: 1908.02499[quant-ph], http://arxiv.org/abs/1908.02499.

35. Thomas Roth, Ruichao Ma, and Weng Chew, "The Transmon Qubit for Electromagnetics Engineers: An introduction — IEEE Journals & Magazine — IEEE Xplore," June 2022, accessed May 16, 2026, https://ieeexplore.ieee.org/document/9789946.

36. Roth, Ma, and Chew.

37. Gavin S. Hartnett et al., *Fast, accurate, high-resolution simulation of large-scale Fermi-Hubbard models on a digital quantum processor*, 2026, arXiv: 2605.04025 [quant-ph], https://arxiv.org/abs/2605.04025.

38. R. Acharya et al., "Multiplexed superconducting qubit control at millikelvin temperatures with a low-power cryo-CMOS multiplexer," *Nature Electronics* 6, no. 11 (November 2023): 900–909, ISSN: 2520-1131, accessed May 6, 2026, https://doi.org/10.1038/s41928-023-01033-8, https://www.nature.com/articles/s41928-023-01033-8.

39. A. C. Hughes et al., *Trapped-ion two-qubit gates with ¿99.99% fidelity without ground-state cooling*, arXiv:2510.17286, October 20, 2025, accessed February 19, 2026, https://doi.org/10.48550/arXiv.2510.17286, arXiv: 2510.17286[quant-ph], http://arxiv.org/abs/2510.17286.

40. C Monroe, "Primer on Mølmer-Sørensen Gates in Trapped Ions."

41. Anthony Ransford et al., *Helios: A 98-qubit trapped-ion quantum computer*, arXiv:2511.05465, November 7, 2025, accessed January 9, 2026, https://doi.org/10.48550/arXiv.2511.05465, arXiv: 2511.05465[quant-ph], http://arxiv.org/abs/2511.05465.

42. Ransford et al.

43. Carmelo Mordini et al., "Multizone Trapped-Ion Qubit Control in an Integrated Photonics QCCD Device," *Physical Review X* 15, no. 1 (February 24, 2025): 011040, ISSN: 2160-3308, accessed May 6, 2026, https://doi.org/10.1103/PhysRevX.15.011040, https://link.aps.org/doi/10.1103/PhysRevX.15.011040.

44. C. M. Löschnauer et al., *Scalable, high-fidelity all-electronic control of trapped-ion qubits*, arXiv:2407.07694, July 10, 2024, accessed October 21, 2025, https://doi.org/10.48550/arXiv.2407.07694, arXiv: 2407.07694[quant-ph], http://arxiv.org/abs/2407.07694.

45. Kenneth R Brown, Jungsang Kim, and Christopher Monroe, "Co-designing a scalable quantum computer with trapped atomic ions," *npj Quantum Information* 2, no. 1 (November 8, 2016): 16034, ISSN: 2056-6387, accessed February 16, 2026, https://doi.org/10.1038/npjqi.2016.34, https://www.nature.com/articles/npjqi201634.

46. Dolev Bluvstein et al., "A fault-tolerant neutral-atom architecture for universal quantum computation," *Nature* 649, no. 8095 (January 2026): 39–46, ISSN: 1476-4687, accessed May 9, 2026, https://doi.org/10.1038/s41586-025-09848-5, https://www.nature.com/articles/s41586-025-09848-5.

47. Yiyi Li et al., *Fast, continuous and coherent atom replacement in a neutral atom qubit array*, arXiv:2506.15633, June 18, 2025, accessed December 30, 2025, https://doi.org/10.48550/arXiv.2506.15633, arXiv: 2506.15633[quant-ph], http://arxiv.org/abs/2506.15633.

48. M. Saffman, *Quantum computing with atomic qubit arrays: confronting the cost of connectivity*, arXiv:2505.11218, November 9, 2025, accessed March 6, 2026, https://doi.org/10.48550/arXiv.2505.11218, arXiv: 2505.11218[quant-ph], http://arxiv.org/abs/2505.11218.

49. Simon J. Evered et al., "High-fidelity parallel entangling gates on a neutral-atom quantum computer," *Nature* 622, no. 7982 (October 2023): 268–272, ISSN: 1476-4687, accessed March 6, 2026, https://doi.org/10.1038/s41586-023-06481-y, https://www.nature.com/articles/s41586-023-06481-y.

50. E. Knill, R. Laflamme, and G. J. Milburn, "A scheme for efficient quantum computation with linear optics," *Nature* 409, no. 6816 (January 2001): 46–52, ISSN: 1476-4687, accessed March 3, 2026, https://doi.org/10.1038/35051009, https://www.nature.com/articles/35051009.

51. Koen Alexander et al., "A manufacturable platform for photonic quantum computing," *Nature* 641, no. 8064 (May 2025): 876–883, ISSN: 1476-4687, accessed February 20, 2026, https://doi.org/10.1038/s41586-025-08820-7, https://www.nature.com/articles/s41586-025-08820-7.

52. Nicolas Maring et al., "A versatile single-photon-based quantum computing platform," *Nature Photonics* 18, no. 6 (June 2024): 603–609, ISSN: 1749-4893, accessed March 3, 2026, https://doi.org/10.1038/s41566-024-01403-4, https://www.nature.com/articles/s41566-024-01403-4.

53. Sara Bartolucci et al., *Fusion-based quantum computation*, arXiv:2101.09310, January 22, 2021, accessed February 28, 2026, https://doi.org/10.48550/arXiv.2101.09310, arXiv: 2101.09310[quant-ph], http://arxiv.org/abs/2101.09310.

54. Alexander et al., "A manufacturable platform for photonic quantum computing."

55. Maring et al., "A versatile single-photon-based quantum computing platform."

56. Daiqin Su et al., "Implementing quantum algorithms on temporal photonic cluster states," *Physical Review A* 98, no. 3 (September 14, 2018): 032316, ISSN: 2469-9926, 2469-9934, accessed February 25, 2026, https://doi.org/10.1103/PhysRevA.98.032316, arXiv: 1805.02645[quant-ph], http://arxiv.org/abs/1805.02645.

57. Andrew D. King et al., "Quantum critical dynamics in a 5,000-qubit programmable spin glass," *Nature* 617, no. 7959 (May 2023): 61–66, ISSN: 1476-4687, accessed April 21, 2026,

https://doi.org/10.1038/s41586-023-05867-2, https://www.nature.com/articles/s41586-023-05867-2.

58. Niel de Beaudrap, "Answer to "What are the models of quantum computation?"," Quantum Computing Stack Exchange, September 14, 2018, accessed February 23, 2026, https://quantumcomputing.stackexchange.com/a/4228.

59. user1271772 No more free time user1271772, "Answer to "Did D-Wave show quantum advantage in 2023?"," Quantum Computing Stack Exchange, September 1, 2023, accessed March 27, 2026, https://quantumcomputing.stackexchange.com/a/34012.

60. "D-Wave Announces Agreement to Acquire Quantum Circuits Inc., Establishing World's Leading Quantum Computing Company," accessed April 21, 2026, https://www.dwavequantum.com/company/newsroom/press-release/d-wave-to-acquire-quantum-circuits-inc-establishing-world-s-leading-quantum-computing-company/.

61. "Microsoft Quantum Chip: Majorana-1 and the Case for Topological Qubits," accessed May 8, 2026, https://www.bluequbit.io/blog/microsoft-quantum-chip.

62. Morteza Aghaee et al., "Interferometric single-shot parity measurement in InAs–Al hybrid devices," *Nature* 638, no. 8051 (February 2025): 651–655, ISSN: 1476-4687, accessed November 7, 2025, https://doi.org/10.1038/s41586-024-08445-2, https://www.nature.com/articles/s41586-024-08445-2.

63. Matteo Rini, "Microsoft's Claim of a Topological Qubit Faces Tough Questions," *Physics* 18 (March 21, 2025): 68, accessed May 8, 2026, https://doi.org/10.1103/Physics.19.s59, https://physics.aps.org/articles/v18/68.

64. Hao Zhang et al., "Editorial Expression of Concern: Ballistic superconductivity in semiconductor nanowires," *Nature Communications* 16, no. 1 (April 3, 2025): 3185, ISSN: 2041-1723, accessed May 8, 2026, https://doi.org/10.1038/s41467-025-58136-3, https://www.nature.com/articles/s41467-025-58136-3.

65. Henry F. Legg, *Comment on "InAs-Al hybrid devices passing the topological gap protocol", Microsoft Quantum, Phys. Rev. B 107, 245423 (2023)*, arXiv:2502.19560, February 26, 2025, accessed February 23, 2026, https://doi.org/10.48550/arXiv.2502.19560, arXiv: 2502.19560[cond-mat], http://arxiv.org/abs/2502.19560.

66. Morteza Aghaee et al., *20 Second Parity Lifetime in an InAs–Pb Tetron Device*, 2026, arXiv: 2606.03884 [cond-mat.mes-hall], https://arxiv.org/abs/2606.03884.

67. Henry Legg, *Today Microsoft announce "Majorana 2", claiming they will build a "practical" quantum computer by 2029. This is of course massive PR bullshi...*, Post, Author Handle: henrylegg.bsky.social, June 2026, accessed July 4, 2026, https://bsky.app/profile/henrylegg.bsky.social/post/3mndbin46xs2z.

68. Aaron J. Weinstein et al., "Universal logic with encoded spin qubits in silicon," *Nature* 615, no. 7954 (March 2023): 817–822, ISSN: 1476-4687, accessed March 29, 2026, https://doi.org/10.1038/s41586-023-05777-3, https://www.nature.com/articles/s41586-023-05777-3.

69. Guido Burkard et al., "Semiconductor Spin Qubits," *Reviews of Modern Physics* 95, no. 2 (June 14, 2023): 025003, ISSN: 0034-6861, 1539-0756, accessed April 22, 2026, https://doi.org/10.1103/RevModPhys.95.025003, arXiv: 2112.08863[cond-mat], http://arxiv.org/abs/2112.08863.

70. Brennan Undseth et al., *Weight-four parity checks with silicon spin qubits*, arXiv:2601.23267, January 30, 2026, accessed February 26, 2026, https://doi.org/10.48550/arXiv.2601.23267, arXiv: 2601.23267[cond-mat], http://arxiv.org/abs/2601.23267.

71. "Building superconducting and neutral atom quantum computers," Google, March 24, 2026, accessed April 18, 2026, https://blog.google/innovation-and-ai/technology/research/neutral-

atom-quantum-computers/.

72. "For quantum computing, different qubits are better together — DARPA," accessed April 18, 2026, https://www.darpa.mil/news/2026/quantum-computing-different-qubits-better-together.

73. Pranav S. Mundada et al., *Heterogeneous architectures enable a 138x reduction in physical qubit requirements for fault-tolerant quantum computing under detailed accounting*, arXiv:2604.06319, April 7, 2026, accessed April 9, 2026, https://doi.org/10.48550/arXiv.2604.06319, arXiv: 2604.06319[quant-ph], http://arxiv.org/abs/2604.06319.

74. John Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum* 2 (August 2018): 79, ISSN: 2521-327X, https://doi.org/10.22331/q-2018-08-06-79, http://dx.doi.org/10.22331/q-2018-08-06-79.

75. *Q2B25 Silicon Valley — Scott Aaronson, Professor, The University of Texas at Austin*, in collab. with QC Ware (January 7, 2026), accessed June 18, 2026, https://www.youtube.com/watch?v=oclXVtQuQE4.

76. Frank Arute et al., "Quantum supremacy using a programmable superconducting processor," *Nature* 574, no. 7779 (October 2019): 505–510, ISSN: 1476-4687, accessed April 26, 2026, https://doi.org/10.1038/s41586-019-1666-5, https://www.nature.com/articles/s41586-019-1666-5.

77. Dominik Hangleiter, *Has quantum advantage been achieved?*, Version Number: 1, 2026, accessed March 14, 2026, https://doi.org/10.48550/ARXIV.2603.09901, https://arxiv.org/abs/2603.09901.

78. Hangleiter.

79. Scott Aaronson and Alex Arkhipov, *The Computational Complexity of Linear Optics*, arXiv:1011.3245, November 14, 2010, accessed April 18, 2026, https://doi.org/10.48550/arXiv.1011.3245, arXiv: 1011.3245[quant-ph], http://arxiv.org/abs/1011.3245.

80. Craig S. Hamilton et al., "Gaussian Boson Sampling," *Physical Review Letters* 119, no. 17 (October 23, 2017): 170501, ISSN: 0031-9007, 1079-7114, accessed April 18, 2026, https://doi.org/10.1103/PhysRevLett.119.170501, arXiv: 1612.01199[quant-ph], http://arxiv.org/abs/1612.01199.

81. Lars S. Madsen et al., "Quantum computational advantage with a programmable photonic processor," *Nature* 606, no. 7912 (June 2022): 75–81, ISSN: 1476-4687, accessed February 25, 2026, https://doi.org/10.1038/s41586-022-04725-x, https://www.nature.com/articles/s41586-022-04725-x.

82. Andrew Pizzimenti, "Gaussian Boson Sampling and its Applications."

83. Scott Aaronson and Yuxuan Zhang, *On verifiable quantum advantage with peaked circuit sampling*, arXiv:2404.14493, May 21, 2024, accessed April 26, 2026, https://doi.org/10.48550/arXiv.2404.14493, arXiv: 2404.14493[quant-ph], http://arxiv.org/abs/2404.14493.

84. "Quantum Advantage Tracker: the race to advantage — IBM Quantum Computing Blog," accessed April 26, 2026, https://www.ibm.com/quantum/blog/quantum-advantage-tracker.

85. Dmitry A. Abanin et al., "Observation of constructive interference at the edge of quantum ergodicity," *Nature* 646, no. 8086 (October 2025): 825–830, ISSN: 1476-4687, accessed April 26, 2026, https://doi.org/10.1038/s41586-025-09526-6, https://www.nature.com/articles/s41586-025-09526-6.

86. "A verifiable quantum advantage," accessed January 16, 2026, https://research.google/blog/a-verifiable-quantum-advantage/.

87. C Zhang et al., "Quantum computation of molecular geometry via many-body nuclear spin echoes."

88. Bill Young, "Foundations of Computer Security - Lecture 44: Symmetric vs. Asymmetric Encryption."

89. Filippo Valsorda, "Quantum Computers Are Not a Threat to 128-bit Symmetric Keys," April 20, 2026, accessed April 20, 2026, https://words.filippo.io/128-bits/.

90. "What Is Q-Day, and How Far Away Is It—Really?," Palo Alto Networks, accessed April 26, 2026, https://www.paloaltonetworks.com/cyberpedia/what-is-q-day.

91. Craig Gidney, *How to factor 2048 bit RSA integers with less than a million noisy qubits*, 2025, arXiv: 2505.15917 [quant-ph], https://arxiv.org/abs/2505.15917.

92. Paul Webster et al., *The Pinnacle Architecture: Reducing the cost of breaking RSA-2048 to 100 000 physical qubits using quantum LDPC codes*, 2026, arXiv: 2602.11457 [quant-ph], https://arxiv.org/abs/2602.11457.

93. Ryan Babbush et al., "Securing Elliptic Curve Cryptocurrencies against Quantum Vulnerabilities: Resource Estimates and Mitigations."

94. "NIST Selects HQC as Fifth Algorithm for Post-Quantum Encryption," Last Modified: 2025-03-20T11:07-04:00, *NIST*, March 11, 2025, accessed April 26, 2026, https://www.nist.gov/news-events/news/2025/03/nist-selects-hqc-fifth-algorithm-post-quantum-encryption.

95. "Post-quantum cryptography is too damn big.," David Adrian, Section: blog, March 22, 2024, accessed March 15, 2026, https://dadrian.io/blog/posts/pqc-signatures-2024/.

96. "State of the post-quantum Internet in 2025," The Cloudflare Blog, October 28, 2025, accessed March 15, 2026, https://blog.cloudflare.com/pq-2025/.

97. Jillian Mascelli and Megan Rodden, ""Harvest Now Decrypt Later": Examining Post-Quantum Cryptography and the Data Privacy Risks for Distributed Ledger Networks," *Finance and Economics Discussion Series*, no. 2025 (September 30, 2025), ISSN: 1936-2854, 2767-3898, accessed January 26, 2026, https://doi.org/10.17016/FEDS.2025.093, https://www.federalreserve.gov/econres/feds/harvest-now-decrypt-later-examining-post-quantum-cryptography-and-the-data-privacy-risks-for-distributed-ledger-networks.htm.

98. Babbush et al., "Securing Elliptic Curve Cryptocurrencies against Quantum Vulnerabilities: Resource Estimates and Mitigations."

99. Matt Swayne, "Ethereum Foundation Elevates Post-Quantum Security to Top Strategic Priority," The Quantum Insider, January 26, 2026, accessed April 26, 2026, https://thequantuminsider.com/2026/01/26/ethereum-foundation-elevates-post-quantum-security-to-top-strategic-priority/.

100. Bennett and Brassard, "Quantum cryptography."

101. "Wall Street's Quantum Computing Divide: Goldman Retreats, JPMorgan In...," archive.ph, April 27, 2026, accessed April 27, 2026, https://archive.ph/CcUsk.

102. Torsten Hoefler, Thomas Haener, and Matthias Troyer, *Disentangling Hype from Practicality: On Realistically Achieving Quantum Advantage*, arXiv:2307.00523, July 2, 2023, accessed March 20, 2026, https://doi.org/10.48550/arXiv.2307.00523, arXiv: 2307.00523[quant-ph], http://arxiv.org/abs/2307.00523.

103. D. Bulger, W. P. Baritompa, and G. R. Wood, "Implementing Pure Adaptive Search with Grover's Quantum Algorithm," *Journal of Optimization Theory and Applications* 116, no. 3

(March 1, 2003): 517–529, ISSN: 1573-2878, accessed March 27, 2026, https://doi.org/10.1023/A:1023061218864, https://doi.org/10.1023/A:1023061218864.

104. Christoph Durr and Peter Hoyer, "A Quantum Algorithm for Finding the Minimum," arXiv.org, July 18, 1996, accessed March 27, 2026, https://arxiv.org/abs/quant-ph/9607014v2.

105. W. P. Baritompa, D. W. Bulger, and G. R. Wood, "Grover's Quantum Algorithm Applied to Global Optimization," _eprint: https://doi.org/10.1137/040605072, *SIAM Journal on Optimization* 15, no. 4 (2005): 1170–1184, https://doi.org/10.1137/040605072, https://doi.org/10.1137/040605072.

106. M. Cerezo et al., "Variational Quantum Algorithms," *Nature Reviews Physics* 3, no. 9 (August 12, 2021): 625–644, ISSN: 2522-5820, accessed March 26, 2026, https://doi.org/10.1038/s42254-021-00348-9, arXiv: 2012.09265[quant-ph], http://arxiv.org/abs/2012.09265.

107. Ruslan Shaydulin et al., "Evidence of Scaling Advantage for the Quantum Approximate Optimization Algorithm on a Classically Intractable Problem," *Science Advances* 10, no. 22 (May 31, 2024): eadm6761, ISSN: 2375-2548, accessed March 20, 2026, https://doi.org/10.1126/sciadv.adm6761, arXiv: 2308.02342[quant-ph], http://arxiv.org/abs/2308.02342.

108. Zhiwei Zhang et al., *New Improvements in Solving Large LABS Instances Using Massively Parallelizable Memetic Tabu Search*, arXiv:2504.00987, August 14, 2025, accessed March 20, 2026, https://doi.org/10.48550/arXiv.2504.00987, arXiv: 2504.00987[cs], http://arxiv.org/abs/2504.00987.

109. Sami Boulebnane and Ashley Montanaro, "Solving Boolean Satisfiability Problems With The Quantum Approximate Optimization Algorithm," *PRX Quantum* 5, no. 3 (September 10, 2024): 030348, accessed March 29, 2026, https://doi.org/10.1103/PRXQuantum.5.030348, https://link.aps.org/doi/10.1103/PRXQuantum.5.030348.

110. Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd, "Quantum Algorithm for Linear Systems of Equations," *Physical Review Letters* 103, no. 15 (October 7, 2009): 150502, accessed March 29, 2026, https://doi.org/10.1103/PhysRevLett.103.150502, https://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

111. Scott Aaronson, "Quantum Machine Learning Algorithms: Read the Fine Print."

112. Feynman, "Simulating Physics with Computers."

113. Ramil Nigmatullin et al., "Experimental demonstration of breakeven for a compact fermionic encoding" [in en], *Nature Physics* 21, no. 8 (August 2025): 1319–1325, ISSN: 1745-2481, accessed July 3, 2026, https://doi.org/10.1038/s41567-025-02931-8, https://www.nature.com/articles/s41567-025-02931-8.

114. Austin Lin, *The tide that is entering: Quantum Computing and impossible Problems in Chemical engineering*, 2025, https://www.aiche.org/sites/default/files/cep/20251230.pdf.

115. Raffaele Santagati et al., "Drug design on quantum computers," *Nature Physics* 20, no. 4 (April 2024): 549–557, ISSN: 1745-2473, 1745-2481, accessed November 21, 2025, https://doi.org/10.1038/s41567-024-02411-5, arXiv: 2301.04114[quant-ph], http://arxiv.org/abs/2301.04114.

116. Nicholas C. Rubin et al., "Quantum computation of stopping power for inertial fusion target design," *Proceedings of the National Academy of Sciences* 121, no. 23 (June 4, 2024): e2317772121, ISSN: 0027-8424, 1091-6490, accessed April 28, 2026, https://doi.org/10.1073/pnas.2317772121, arXiv: 2308.12352[quant-ph], http://arxiv.org/abs/2308.12352.

117. Thomas Spatzal et al., "Nitrogenase FeMoco investigated by spatially resolved anomalous dispersion refinement," *Nature Communications* 7, no. 1 (March 14, 2016): 10902, ISSN: 2041-1723, accessed April 28, 2026, https://doi.org/10.1038/ncomms10902, https://www.nature.com/articles/ncomms10902.

118. Corin Wagen, "Quantum Chemistry in Drug Discovery," Rowan Documentation, accessed May 20, 2026, https://www.rowansci.com.

119. Maximilian Mörchen et al., *Classification of electronic structures and state preparation for quantum computation of reaction chemistry*, arXiv:2409.08910, September 13, 2024, accessed April 7, 2026, https://doi.org/10.48550/arXiv.2409.08910, arXiv: 2409.08910[physics], http://arxiv.org/abs/2409.08910.

120. Jan Řezáč, Lucia Šimová, and Pavel Hobza, "CCSD[T] Describes Noncovalent Interactions Better than the CCSD(T), CCSD(TQ), and CCSDT Methods," *Journal of Chemical Theory and Computation* 9, no. 1 (January 8, 2013): 364–369, ISSN: 1549-9618, accessed June 18, 2026, https://doi.org/10.1021/ct3008777, https://doi.org/10.1021/ct3008777.

121. Benjamin Rhodes et al., "Orb-v3: atomistic simulation at scale."

122. Wagen, "Quantum Chemistry in Drug Discovery."

123. "Elucidating reaction mechanisms on quantum computers — PNAS," accessed April 28, 2026, https://www.pnas.org/doi/10.1073/pnas.1619152114.

124. Huanchen Zhai et al., *Classical solution of the FeMo-cofactor model to chemical accuracy and its implications*, arXiv:2601.04621, January 8, 2026, accessed February 26, 2026, https://doi.org/10.48550/arXiv.2601.04621, arXiv: 2601.04621[physics], http://arxiv.org/abs/2601.04621.

125. Ivan Kassal et al., "Simulating Chemistry Using Quantum Computers," *Annual Review of Physical Chemistry* 62, no. 1 (May 5, 2011): 185–207, ISSN: 0066-426X, 1545-1593, accessed November 15, 2025, https://doi.org/10.1146/annurev-physchem-032210-103512, https://www.annualreviews.org/doi/10.1146/annurev-physchem-032210-103512.

126. Pauline J. Ollitrault, Guglielmo Mazzola, and Ivano Tavernelli, "Nonadiabatic Molecular Quantum Dynamics with Quantum Computers," *Physical Review Letters* 125, no. 26 (December 31, 2020): 260511, ISSN: 0031-9007, 1079-7114, accessed June 17, 2026, https://doi.org/10.1103/PhysRevLett.125.260511, https://link.aps.org/doi/10.1103/PhysRevLett.125.260511.

127. Sabina Dragoi, "Analysis of the Trotter Method for Hamiltonian Simulation."

128. I. M. Georgescu, S. Ashhab, and Franco Nori, "Quantum Simulation," *Reviews of Modern Physics* 86, no. 1 (March 10, 2014): 153–185, ISSN: 0034-6861, 1539-0756, accessed June 3, 2026, https://doi.org/10.1103/RevModPhys.86.153, arXiv: 1308.6253[quant-ph], http://arxiv.org/abs/1308.6253.

129. Wagen, "Quantum Chemistry in Drug Discovery."

130. Seunghoon Lee et al., "Is there evidence for exponential quantum advantage in quantum chemistry?," *Nature Communications* 14, no. 1 (April 7, 2023): 1952, ISSN: 2041-1723, accessed November 11, 2025, https://doi.org/10.1038/s41467-023-37587-6, arXiv: 2208.02199[physics], http://arxiv.org/abs/2208.02199.

131. Vivek V. Shende, Stephen S. Bullock, and Igor L. Markov, "Synthesis of Quantum Logic Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25, no. 6 (June 2006): 1000–1010, ISSN: 0278-0070, 1937-4151, accessed June 3, 2026, https://doi.org/10.1109/TCAD.2005.855930, arXiv: quant-ph/0406176, http://arxiv.org/abs/quant-ph/0406176.

132. Yudong Cao et al., "Quantum Chemistry in the Age of Quantum Computing," *Chemical Reviews* 119, no. 19 (October 9, 2019): 10856–10915, ISSN: 0009-2665, 1520-6890, accessed April 9, 2026, https://doi.org/10.1021/acs.chemrev.8b00803, https://pubs.acs.org/doi/10.1021/acs.chemrev.8b00803.

133. Davide Castaldo and Markus Reiher, *Utility-scale quantum computational chemistry*, arXiv:2603.19081,

March 19, 2026, accessed April 6, 2026, https://doi.org/10.48550/arXiv.2603.19081, arXiv: 2603.19081[quant-ph], http://arxiv.org/abs/2603.19081.

134. Narjes Ansari et al., *The Convergence Frontier: Integrating Machine Learning and High Performance Quantum Computing for Next-Generation Drug Discovery*, arXiv:2603.17790, April 2, 2026, accessed April 6, 2026, https://doi.org/10.48550/arXiv.2603.17790, arXiv: 2603.17790[quant-ph], http://arxiv.org/abs/2603.17790.

135. Guang Hao Low et al., "Fast Quantum Simulation of Electronic Structure by Spectral Amplification," *Physical Review X* 15, no. 4 (October 31, 2025): 041016, ISSN: 2160-3308, accessed January 12, 2026, https://doi.org/10.1103/pb2g-j9cw, https://link.aps.org/doi/10.1103/pb2g-j9cw.

136. Joshua J. Goings et al., "Reliably assessing the electronic structure of cytochrome P450 on today's classical computers and tomorrow's quantum computers," *Proceedings of the National Academy of Sciences* 119, no. 38 (September 20, 2022): e2203533119, ISSN: 0027-8424, 1091-6490, accessed April 13, 2026, https://doi.org/10.1073/pnas.2203533119, arXiv: 2202.01244[quant-ph], http://arxiv.org/abs/2202.01244.

137. Altman et al., "Quantum Simulators."

138. Jielun Chen and Garnet Kin-Lic Chan, *A framework for robust quantum speedups in practical correlated electronic structure and dynamics*, arXiv:2508.15765, August 21, 2025, accessed April 7, 2026, https://doi.org/10.48550/arXiv.2508.15765, arXiv: 2508.15765[quant-ph], http://arxiv.org/abs/2508.15765.

139. Garnet Kin-Lic Chan, "Spiers Memorial Lecture: Quantum chemistry, classical heuristics, and quantum advantage," *Faraday Discussions* 254, no. 0 (November 6, 2024): 11–52, ISSN: 1364-5498, accessed June 18, 2026, https://doi.org/10.1039/D4FD00141A, https://pubs.rsc.org/en/content/articlelanding/2024/fd/d4fd00141a.

140. Garnet Chan, "The FeMo-cofactor and classical and quantum computing," Quantum Frontiers, March 13, 2026, accessed March 15, 2026, https://quantumfrontiers.com/2026/03/12/the-femo-cofactor-and-classical-and-quantum-computing/.

141. Substack, *How to Build a Moat: The Vietnam War, Playing Frisbee, and Brute Force Engineering* [in en], accessed July 4, 2026, https://substack.com/home/post/p-201766378.

142. Felix Tripier et al., *Fault-Tolerant Quantum Computing with Trapped Ions: The Walking Cat Architecture*, arXiv:2604.19481, April 21, 2026, accessed April 29, 2026, https://doi.org/10.48550/arXiv.2604.19481, arXiv: 2604.19481[quant-ph], http://arxiv.org/abs/2604.19481.

143. Matt Swayne, "Global Quantum Computing Market to Double by 2028, Reaching 3 Billion in Revenue, QED-C State of the Global Quantum Industry 2026 Report Finds," The Quantum Insider, April 14, 2026, accessed May 29, 2026, https://thequantuminsider.com/2026/04/14/global-quantum-computing-market-to-double-by-2028-reaching-3-billion-in-revenue-qed-c-state-of-the-global-quantum-industry-2026-report-finds/.

144. "McKinsey Quantum Technology Monitor 2026 — McKinsey," accessed April 29, 2026, https://www.mckinsey.com/capabilities/mckinsey-technology/our-insights/mckinsey-quantum-technology-monitor-2026-a-commercial-tipping-point.

145. Ana Swanson, "U.S. Plans to Invest $2 Billion and Take Stake in Quantum Firms," *The New York Times*, May 21, 2026, ISSN: 0362-4331, accessed May 29, 2026, https://www.nytimes.com/2026/05/21/business/trump-quantum-computers-stake.html.

146. Mohib Ur Rehman, "France Adds €1.55 Billion for Quantum and Semiconductor Development," The Quantum Insider, May 25, 2026, accessed May 29, 2026, https://thequantuminsider.com/2026/05/25/emmanuel-macron-announces-1-55-billion-euros-more-for-quantum-and-semiconductors/.

147. "Australia leads developed world in govt quantum investment," Quantum Australia, accessed May 29, 2026, https://www.quantum-australia.com/news/australia-leads-developed-world-in-govt-quantum-investment.

148. Katrina Munichiello Full Bio Katrina Ávila Munichiello is an experienced editor et al., "Special Purpose Acquisition Company (SPAC) Explained: Examples and Risks," Investopedia, accessed April 29, 2026, https://www.investopedia.com/terms/s/spac.asp.

149. Felix Feng et al., "The incentives of SPAC sponsors," *Journal of Financial Economics* 177 (March 1, 2026): 104220, ISSN: 0304-405X, accessed May 30, 2026, https://doi.org/10.1016/j.jfineco.2025.104220, https://www.sciencedirect.com/science/article/pii/S0304405X25002284.

150. Saurav Chaudhari, *The SPAC Boom and Bust: An Analysis of Investment Banking Strategies and Investor Returns*, 5130749, Rochester, NY, February 10, 2025, accessed April 29, 2026, https://doi.org/10.2139/ssrn.5130749, https://papers.ssrn.com/abstract=5130749.

151. Kyla Scanlon, "The Ozempicization of Everything," Kyla's Newsletter, March 26, 2026, Substack newsletter, accessed June 1, 2026, https://kyla.substack.com/p/the-ozempicization-of-the-economy.

152. Ciceri et al., *Enhanced fill probability estimates in institutional algorithmic bond trading using statistical learning algorithms with quantum computers*.

153. Scott Aaronson, "HSBC unleashes yet another "qombie": a zombie claim of quantum advantage that isn't," Shtetl-Optimized, September 26, 2025, accessed June 2, 2026, https://scottaaronson.blog/?p=9170.

154. Willie Aboumrad et al., "Accelerating large-scale linear algebra using variational quantum imaginary time evolution," arXiv.org, March 17, 2025, accessed June 2, 2026, https://arxiv.org/abs/2503.13128v2.

155. *Explaining Quantum: Machine Learning Image Recognition Application*, in collab. with IonQ (December 15, 2025), accessed June 2, 2026, https://www.youtube.com/watch?v=jGXTgNmPkps.

156. "In Production: Ford Otosan Deploys Vehicle Manufacturing Application Built with D-Wave Technology," accessed June 2, 2026, https://www.dwavequantum.com/company/newsroom/press-release/in-production-ford-otosan-deploys-vehicle-manufacturing-application-built-with-d-wave-technology/.

157. "Quantum Computers Will Make AI Better," accessed June 2, 2026, https://www.quantinuum.com/blog/quantum-computers-will-make-ai-better.

158. *Investor Presentation: 2021*, IonQ, 2021, https://s28.q4cdn.com/828571518/files/doc_presentation/2021/03/IonQ-Investor-Presentation-030721-vFF.pdf.

159. *Investor Presentation: October 2021*, Rigetti, 2021, Available:%20https://www.rigetti.com/uploads/Rigetti-Investor-Presentation.pdf.

160. "IBM Unveils 400 Qubit-Plus Quantum Processor and Next-Generation IBM Quantum System Two," IBM Newsroom, accessed May 29, 2026, https://newsroom.ibm.com/2022-11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two.

161. *The World's Only Quantum Platform and Merchant Supplier - Investor Overview – May 2026*, IonQ, 2026, https://s28.q4cdn.com/828571518/files/doc_presentations/2026/May/06/IONQ-Investor-Overview-May-2026-vF-2026-05-20.pdf.

162. "IONQ HAS LOST FUNDING FOR VITAL PENTAGON CONTRACTS THAT HAD REPRESENTED UP TO 86% OF REVENUES," Wolfpack Research, accessed June 2, 2026, https:

//www.wolfpackresearch.com/items/ionq-has-lost-funding-for-vital-pentagon-contracts-that-had-represented-up-to-86%-of-revenues-.

163. *IonQ, Inc. (IONQ): Trapped in the Hype*, Kerrisdale Capital, 2025, https://www.kerrisdalecap.com/wp-content/uploads/2025/03/IonQ-%E2%80%93-Kerrisdale.pdf.

164. Andrew Walker, "IONQ and the evolution of meme stock financing," January 17, 2024, accessed December 18, 2025, https://www.yetanothervalueblog.com/p/the-evolution-of-meme-stock-financing.

165. Masoud Mohseni et al., *How to Build a Quantum Supercomputer: Scaling from Hundreds to Millions of Qubits*, arXiv:2411.10406, March 13, 2026, accessed March 20, 2026, https://doi.org/10.48550/arXiv.2411.10406, arXiv: 2411.10406[quant-ph], http://arxiv.org/abs/2411.10406.

166. "Concept Cats: Designing Better Qubits," Alice & Bob, accessed June 2, 2026, https://alice-bob.com/blog/concept-cats-designing-better-qubits/.

167. Ryan Babbush et al., *Securing Elliptic Curve Cryptocurrencies against Quantum Vulnerabilities: Resource Estimates and Mitigations*, arXiv:2603.28846, April 15, 2026, accessed June 2, 2026, https://doi.org/10.48550/arXiv.2603.28846, arXiv: 2603.28846[quant-ph], http://arxiv.org/abs/2603.28846.

168. Matt Swayne, "Quantum Startup Atom Computing First to Exceed 1,000 Qubits," The Quantum Insider, October 24, 2023, accessed June 2, 2026, https://thequantuminsider.com/2023/10/24/quantum-startup-atom-computing-first-to-exceed-1000-qubits/.

169. Sales Rodriguez et al., "Experimental demonstration of logical magic state distillation."

170. J. J. Dijkema et al., "Simultaneous operation of an 18-qubit modular array in germanium," arXiv.org, April 1, 2026, accessed June 2, 2026, https://arxiv.org/abs/2604.01063v1.

About the Author

Michael Solomentsev is an engineer, entrepreneur, and sometimes writer. He holds a PhD in electrical engineering from the University of Texas at Austin, and recently spent two years attempting to commercialize his research in advanced data center power architectures (via his startup, Palanquin Power). Michael is interested in novel technology and its diffusion to the market.

He has a website and blog. Inquiries, comments, or corrections are welcome via email.

Colophon

This book was typeset in LaTeX, with images generated in Inkscape. The cover design is inspired by a woodblock print by Maki Haku, called Proportion-1.