Introduction

Quantum computing begins with a simple but surprising question:

What if the rules of computation were built from the rules of quantum physics?

At first, this question may sound mysterious. Most of us meet computers as laptops, phones, websites, games, spreadsheets, or programming languages. We learn that computers store information as bits, values such as 0 and 1, and that they process those bits using logical steps. That picture is powerful. It explains why a phone can play a video, why a search engine can find a document, and why a program can calculate millions of operations in a second.

But nature itself is not made of ordinary bits.

At the scale of atoms, electrons, photons, and molecules, the world follows the rules of quantum mechanics. Quantum mechanics is the physical theory used to describe very small systems. It is not merely a strange philosophical idea; it is one of the most accurately tested frameworks in science and is essential to technologies such as lasers, semiconductors, magnetic resonance imaging, and modern electronics. Quantum computing asks whether information can be stored and processed directly using those quantum rules.

This book is about learning that idea from the ground up.

You do not need to begin with prior quantum physics. You do not need to already know advanced linear algebra. You do not need to believe that quantum computers are magical. In fact, this book will work against that myth. Quantum computers are not magic machines that “try all answers at once” and then hand us the correct one. They are physical information-processing devices whose behavior can be described using precise mathematics. The aim of this book is to help you understand that mathematics, the physical intuition behind it, and the algorithms that make quantum computing interesting.

Why quantum computing is worth studying

A classical computer is a machine that processes information according to rules. The information is represented using bits, and the rules are implemented through logic gates, circuits, and programs. This model has transformed the world. It is also flexible: the same general-purpose computer can run a word processor, simulate a bridge, train a neural network, or solve an equation.

Yet some problems remain extremely difficult for ordinary computers.

For example, suppose we want to understand the behavior of a molecule. A molecule is made of atoms, and atoms contain electrons. Electrons obey quantum mechanics. To describe the exact quantum state of a system with many interacting particles, the amount of classical information needed can grow extremely quickly as the system grows. Richard Feynman famously argued that simulating quantum physics with ordinary classical computers appears inefficient in general, and that computers built from quantum systems might be the natural tools for simulating quantum systems (Feynman, 1982).

This is one of the deepest motivations for quantum computing:

If nature is quantum, perhaps the most natural computer for some tasks is also quantum.

That does not mean a quantum computer is better at every task. If you want to write an essay, browse the web, or add two small numbers, a quantum computer is not expected to replace your laptop. The power of quantum computing is more specific. It appears in certain kinds of problems where quantum structure can be used to shape probabilities in useful ways.

Two famous examples show why the field became so important.

First, Peter Shor discovered a quantum algorithm that can factor large integers and compute discrete logarithms efficiently on an ideal quantum computer (Shor, 1994). Factoring means breaking a number into prime factors. For example,

$$21 = 3 \times 7.$$

That example is easy. But factoring numbers with hundreds or thousands of digits is much harder for known classical methods, and the difficulty of related number-theoretic problems supports widely used public-key cryptography. Shor's result showed that a sufficiently large, fault-tolerant quantum computer would have major consequences for cryptographic systems based on factoring and discrete logarithms.

Second, Lov Grover discovered a quantum algorithm for unstructured search that gives a quadratic speedup over classical search (Grover, 1996). "Quadratic speedup" means that if a classical method needs about N checks, Grover's algorithm needs about $\sqrt{N}$ checks. This is not an exponential miracle, but it is still powerful. If $N = 1{,}000{,}000$, then $\sqrt{N} = 1{,}000$. The difference between one million checks and one thousand checks can matter.

These examples teach an important lesson:

Quantum computing is not about making every computation faster. It is about discovering which computational structures can be transformed by quantum mechanics.

The central idea: information can have quantum form

To understand quantum computing, we must first separate two ideas that are often mixed together:

1. Information: what is being represented.
2. Physical representation: how that information is stored in the world.

A classical bit is information with two possible values: 0 or 1. Physically, a bit might be represented by a voltage level in a circuit, a magnetic region on a disk, or a tiny charge in memory. The abstract bit is simple, but the physical implementation can vary.

A quantum bit, or qubit, is the quantum version of a bit. Like a classical bit, it has two standard basis states, usually written

$$|0\rangle \quad \text{and} \quad |1\rangle.$$

The symbols |0⟩ and |1⟩ are called ket notation. This notation comes from quantum mechanics and is used to represent quantum states. For now, you can read |0⟩ as “the quantum state corresponding to 0” and |1⟩ as “the quantum state corresponding to 1.”

But a qubit can also be in a state such as

$$\text{frac1}\sqrt{(2)}|0\rangle + \text{frac1}\sqrt{(2)}|1\rangle.$$

This expression is called a superposition. A superposition is not simply ignorance about whether the qubit is really 0 or really 1. It is a genuine quantum state described by amplitudes, the numbers placed in front of the basis states.

In the example above, the amplitudes are both frac1√(2). When we measure this qubit in the usual way, we get a classical result: either 0 or 1. The probabilities are found by squaring the magnitudes of the amplitudes. Since

$$\left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2},$$

this state gives 0 with probability (1)/(2) and 1 with probability (1)/(2).

At this early stage, the key point is not to memorize formulas. The key point is this:

Quantum information is described by amplitudes, and amplitudes can combine in ways ordinary probabilities cannot.

That combination is called interference. Interference is one of the main resources of quantum algorithms. Amplitudes can reinforce each other, making some outcomes more likely, or cancel each other, making some outcomes less likely. A good quantum algorithm is not a machine that blindly creates many possibilities. It is a carefully designed process that makes wrong answers tend to cancel and useful answers tend to appear with higher probability.

A small example of interference

Imagine two water waves meeting in a pond. If the crest of one wave meets the crest of another, the water rises higher. This is constructive interference. If the crest of one wave meets the trough of another, they can partly cancel. This is destructive interference.

Quantum amplitudes behave mathematically more like wave quantities than like ordinary probabilities. This is why signs and phases matter.

For example, consider two amplitudes:

(1) □ (2) and (1) □ (2).

If they add, we get

$$\frac{1}{2} + \frac{1}{2} = 1.$$

But if one amplitude has a negative sign,

$$\frac{1}{2} + \left(-\frac{1}{2}\right) = 0.$$

A probability cannot be negative, but an amplitude can have a sign or, more generally, a complex phase. This allows cancellation. Quantum algorithms use this feature with great care.

This does not mean we can directly observe amplitudes. When we measure, we get ordinary classical outcomes. The art of quantum algorithm design is to arrange the computation so that measurement is likely to reveal useful information.

What this book will build

This book follows a gradual path. It starts with ordinary computation because quantum computing only makes sense when compared with classical computing. Before asking what a qubit changes, we need to know what a bit does. Before studying quantum circuits, we need to understand ordinary circuits. Before discussing quantum speedup, we need to understand what computational cost means.

Then we build the mathematical language. Quantum states are described using vectors. Quantum gates are described using matrices. Probabilities are computed from amplitudes. These tools may look abstract at first, but they will become practical. You will learn them by using them.

The path of the book is roughly this:

We begin with the motivation: why quantum computing exists, what classical computers do well, and where quantum ideas enter. Then we study classical bits, logic gates, circuits, and algorithms. After that, we introduce the linear algebra needed for quantum states: vectors, complex numbers, inner products, matrices, and basis representations.

Once the language is ready, we study qubits, measurement, and single-qubit gates. Then we move to multiple qubits, tensor products, and entanglement. Entanglement is a special kind of quantum correlation that cannot be explained as merely two hidden classical values waiting to be revealed. It is one of the features that makes quantum information different from classical information.

After that, we learn how quantum circuits are built. We study controlled gates, reversible computation, ancilla qubits, uncomputation, and interference. These ideas lead naturally to the first quantum algorithms, including the Deutsch and Deutsch-Jozsa algorithms. Later chapters develop the quantum Fourier transform, phase estimation, Shor's factoring algorithm, Grover's search algorithm, quantum simulation, error correction, quantum programming, near-term algorithms, complexity theory, and quantum communication.

The goal is not only to recognize famous algorithms. The deeper goal is to understand how they are assembled.

By the end of the book, you should be able to look at a basic quantum circuit and ask meaningful questions:

- What state enters the circuit?
- Which gates act on which qubits?
- Where do amplitudes change?
- Where can interference occur?
- What happens if we measure?
- What probability distribution should we expect?
- What problem is the circuit trying to solve?

These questions are the beginning of real understanding.

What quantum computers are not

Because quantum computing is often surrounded by dramatic language, it is useful to remove a few misconceptions early.

A quantum computer is not simply a classical computer that runs faster. It uses a different model of computation. Some tasks may gain large speedups, some may gain smaller speedups, and many ordinary tasks may gain no useful advantage.

A qubit is not just a bit that is both 0 and 1 in the everyday sense. That phrase can be a rough metaphor, but it becomes misleading quickly. A qubit is described by a vector of amplitudes, and measurement produces a classical outcome with probabilities determined by those amplitudes.

A quantum computer does not automatically test every possible answer and choose the right one. If that were true, quantum computing would be easy. The challenge is that measurement gives only one outcome, and most naive quantum procedures produce useless randomness. Quantum algorithms work only when their structure causes useful interference.

A quantum computer also does not violate the known rules of computation by solving every hard problem efficiently. Complexity theory, the study of how computational resources scale with problem size, suggests a more careful picture. The class of problems efficiently solvable by quantum computers is usually called BQP, and it is not believed to contain every difficult problem, such as all problems in NP (Nielsen and Chuang, 2010).

This careful view makes quantum computing more interesting, not less. The field is not a collection of miracles. It is a disciplined search for the boundary between what classical information processing can do and what quantum information processing can do.

Ideal quantum computers and real quantum devices

In the first part of the book, we will often study ideal quantum circuits. An ideal circuit is a mathematical model where gates are perfect, qubits remain coherent, and measurements follow the exact probabilities predicted by quantum theory.

Real devices are harder.

A real qubit is a physical system. It may be built using superconducting circuits, trapped ions, photons, neutral atoms, or other platforms. Such systems are delicate. They interact with their environment. They suffer from noise. Gates are imperfect. Measurements can be wrong. Qubits may only interact with certain neighboring qubits, depending on the hardware design.

Modern quantum processors are often described as NISQ devices, meaning “Noisy Intermediate-Scale Quantum” devices, a term emphasized by John Preskill for machines with enough qubits to be scientifically interesting but not enough error correction to run long, fully reliable algorithms (Preskill, 2018). This matters because the beautiful circuits drawn in early chapters are not automatically practical on current hardware.

Still, ideal models are not useless. They are like frictionless planes in introductory physics. They let us understand the core principles before adding real-world complications. Later in the book, we will return to noise, decoherence, error correction, and practical workflows.

A first mental model

Here is a useful first mental model for quantum computing:

A classical computation moves through definite classical states.

A quantum computation evolves a state of amplitudes.

A measurement converts part of that quantum state into a classical outcome.

An algorithm is successful when the evolution of amplitudes makes useful outcomes likely.

This model is incomplete, but it is a good starting point. Let us unpack it gently.

Suppose a classical bit is 0. If we apply a NOT gate, it becomes 1. If we apply NOT again, it becomes 0. The bit has a definite value at every step.

Now suppose a qubit is in a superposition. A quantum gate changes the amplitudes. It may turn one pattern of amplitudes into another. Before measurement, the state is not best understood as a hidden classical value. It is a quantum state. When we measure, we get a classical bit, and the probabilities come from the amplitudes.

This is why quantum computing requires a new language. Ordinary Boolean logic is not enough. We need vectors, complex numbers, matrices, and probability. But each of these will be introduced from first principles.

How to stay motivated when the mathematics begins

Quantum computing has a reputation for being difficult. Some of that reputation is deserved: the subject combines computation, mathematics, and physics. But difficulty is not the same as impossibility.

The most important habit is to keep asking what each mathematical object represents.

When you see a vector, ask: What quantum state does this describe?

When you see a matrix, ask: What gate or transformation does this represent?

When you see an amplitude, ask: How will this contribute to a measurement probability?

When you see a circuit, ask: How does information move through it?

When you see an algorithm, ask: Where is the interference?

Mathematics becomes easier when it is connected to purpose. In this book, equations are not decorations. They are tools for saying exactly what a quantum computer is doing.

The promise of the journey

Quantum computing is young compared with classical computing, but its foundations are already rich. David Deutsch formulated the idea of a universal quantum computer in the 1980s, showing how quantum theory could be connected to the general idea of computation (Deutsch, 1985). Since then, the field has grown into a major area involving physics, computer science, mathematics, engineering, chemistry, and information theory.

The subject matters for several reasons.

It matters scientifically because it gives us a new way to think about nature and simulation.

It matters computationally because it changes our understanding of algorithms and complexity.

It matters technologically because building reliable quantum computers requires new hardware, control systems, software tools, and error-correction methods.

It matters socially because some quantum algorithms affect cryptography, security planning, and future infrastructure.

But perhaps most importantly for a student, quantum computing matters because it teaches a powerful lesson:

Computation is physical.

Information is not an abstract ghost floating outside the universe. Every bit, every memory cell, every signal, and every measurement is represented by some physical system. Classical computing uses physical systems that behave well enough like classical bits. Quantum computing asks what becomes possible when we use systems that preserve quantum behavior.

That question will guide the rest of this book.

We will begin slowly. We will define bits before qubits, gates before quantum gates, vectors before state spaces, and probability before measurement. Step by step, the strange will become structured. The goal is not to make quantum computing seem easy by hiding its depth. The goal is to make it learnable by building every idea on a clear foundation.

Let us begin with the question that started the whole path:

Why should quantum computing exist at all?

References

Deutsch, D. (1985). Quantum theory, the Church-Turing principle and the universal quantum computer. Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 400(1818), 97–117.

Feynman, R. P. (1982). Simulating physics with computers. International Journal of Theoretical Physics, 21, 467–488.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, 212–219.

Nielsen, M. A., & Chuang, I. L. (2010). Quantum Computation and Quantum Information (10th anniversary ed.). Cambridge University Press.

Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. Quantum, 2, 79.

Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. In Proceedings 35th Annual Symposium on Foundations of Computer Science, 124–134. IEEE.

Document information

Introduction

Project	Quantum Computing from First Principles
Document	Document 1.4
Author	mujirin
Verifier	Not verified
Downloaded	July 05, 2026 22:40 KST
Status	Working
Document link	https://theorytrace.com/projects/quantum-computing-from-first-principles/documents/introduction/