How Does Quantum Computing Work?

BY DAVID CARDINAL

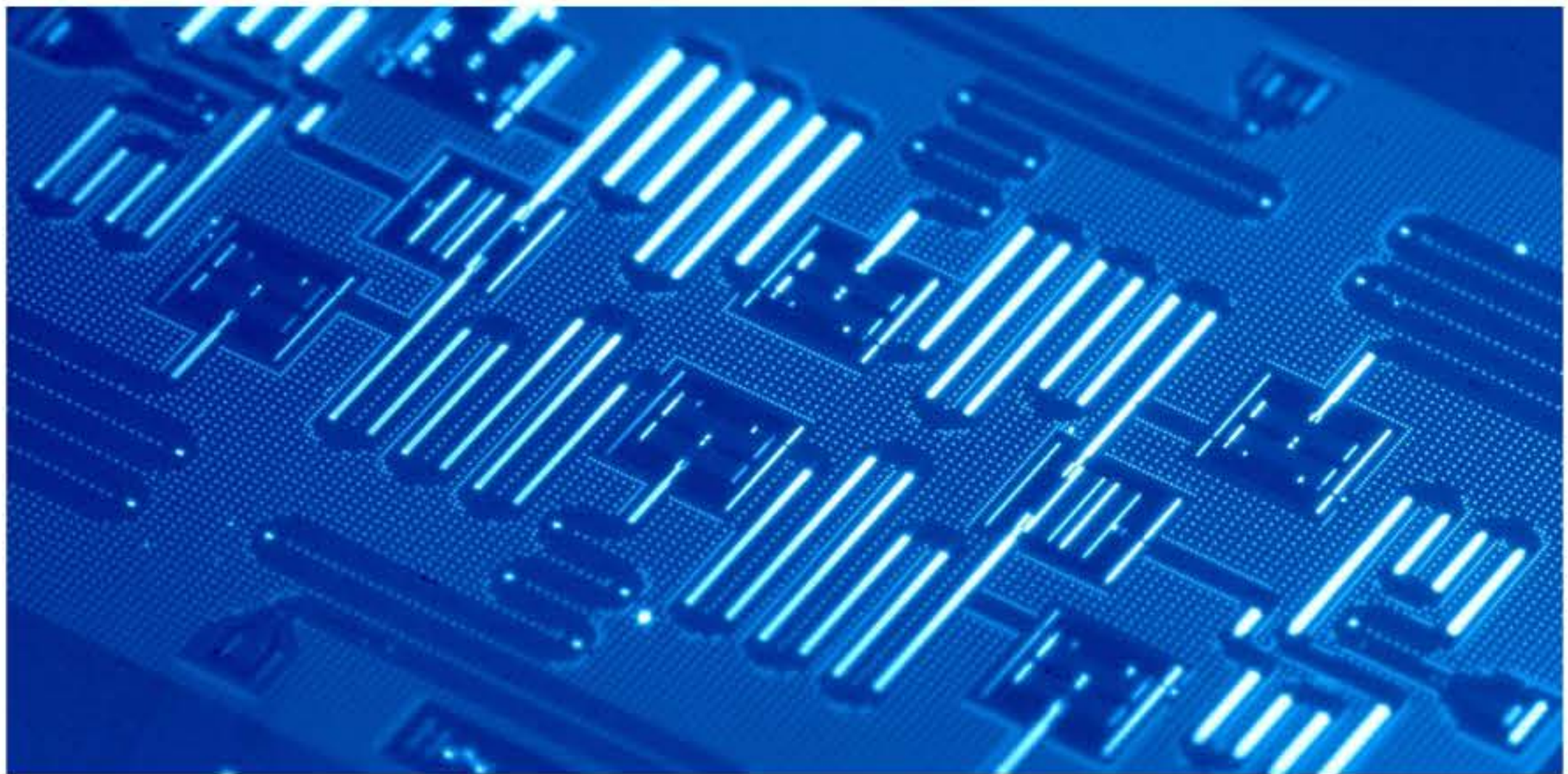
Quantum computing just plain sounds cool. We've all read about the massive investment in making it a reality, and its promise of breakthroughs in many industries. But all that press is usually short on what it is and how it works. There's a reason: Quantum computing is quite different from traditional digital computing and requires thinking about things in a non-intuitive way. Oh, and there is math. Lots of it.

This article won't make you an expert, but it should help you understand what quantum computing is, why it's important, and why it's so exciting. If you already have a background in quantum mechanics and grad school math, you probably don't need to read this article. You can jump straight into a book like *A Gentle Introduction To Quantum Computing* ("gentle" is a relative term). But for those (most of us) who don't have that background, we'll do our best to demystify one of the most mystical topics in computing.

QUANTUM COMPUTING CONCEPTS

Here are the basics. First, quantum computers use qubits instead of traditional bits (binary digits). Qubits are different from traditional bits because until they are read out (meaning measured), they can exist in an indeterminate state in which we can't tell whether they'll be measured as a 0 or a 1. That's because of a unique property called *superposition*.

Superposition makes qubits interesting, but their real superpower is *entanglement*. Entangled qubits can interact instantly. To make functional qubits, quantum computers have to be cooled to near absolute zero. Even when supercooled, qubits don't maintain their entangled state (coherence) for very long.

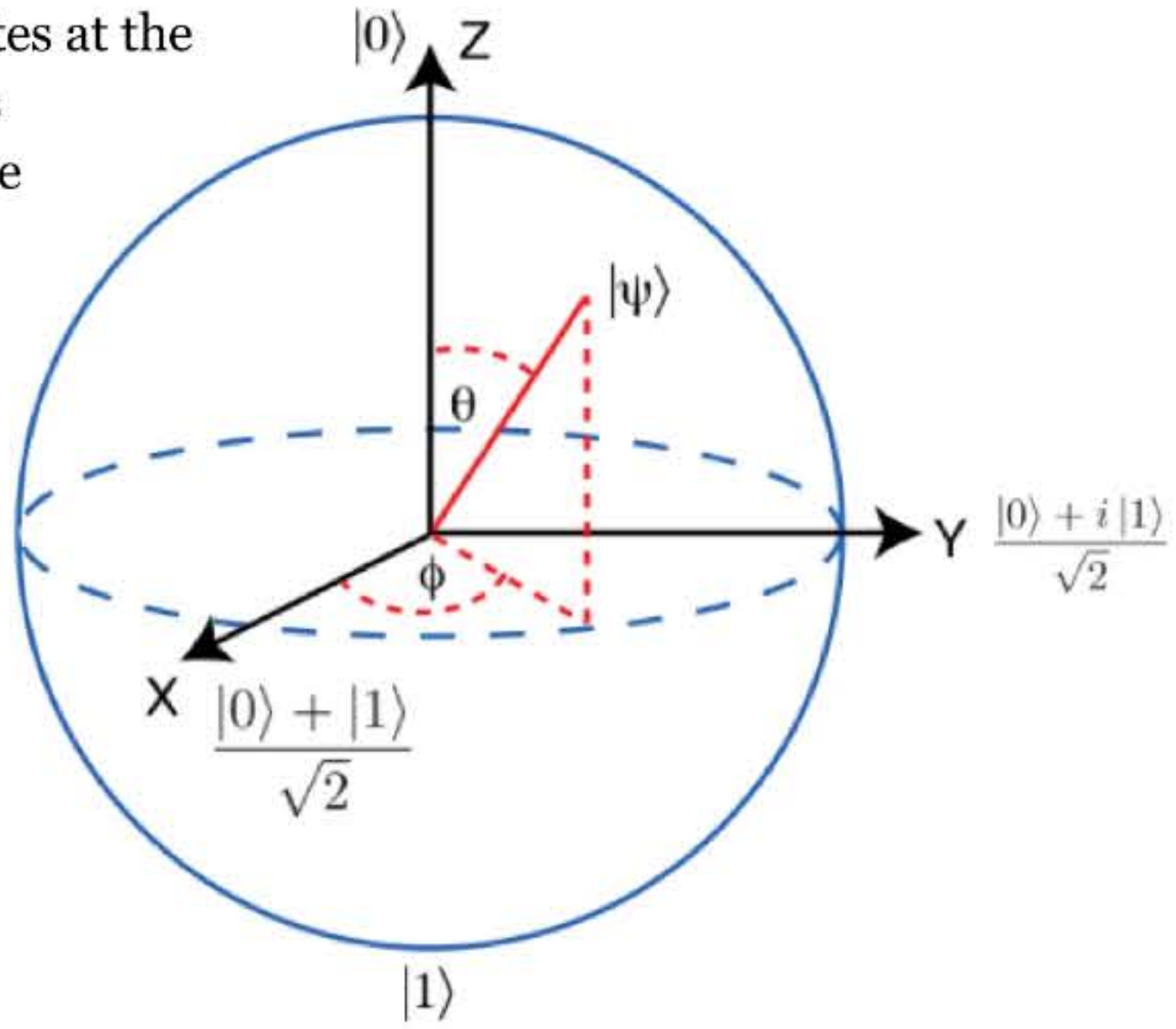


That makes programming them extra tricky. Quantum computers are programmed using sequences of *logic gates* of various kinds, but programs need to run quickly enough that the qubits don't lose coherence before they're measured. For anyone who's taken a logic class or done digital circuit design using flip-flops, quantum logic gates will sound somewhat familiar, although quantum computers themselves are essentially analog. But the combination of superposition and entanglement make the process about a hundred times more confusing.

QUBITS AND SUPERPOSITION

The ordinary bits we use in typical digital computers are either 0 or 1. You can read them whenever you want, and unless there's a flaw in the hardware, they won't change. Qubits aren't like that. They have a probability of being 0 and a probability of being 1, but until you measure them, they may be in an indefinite state. That state, along with some other state information that allows for additional computational complexity, can be described as being at an arbitrary point on a sphere (of radius 1), that reflects both the probability of being measured as a 0 or 1 (which are the north and south poles).

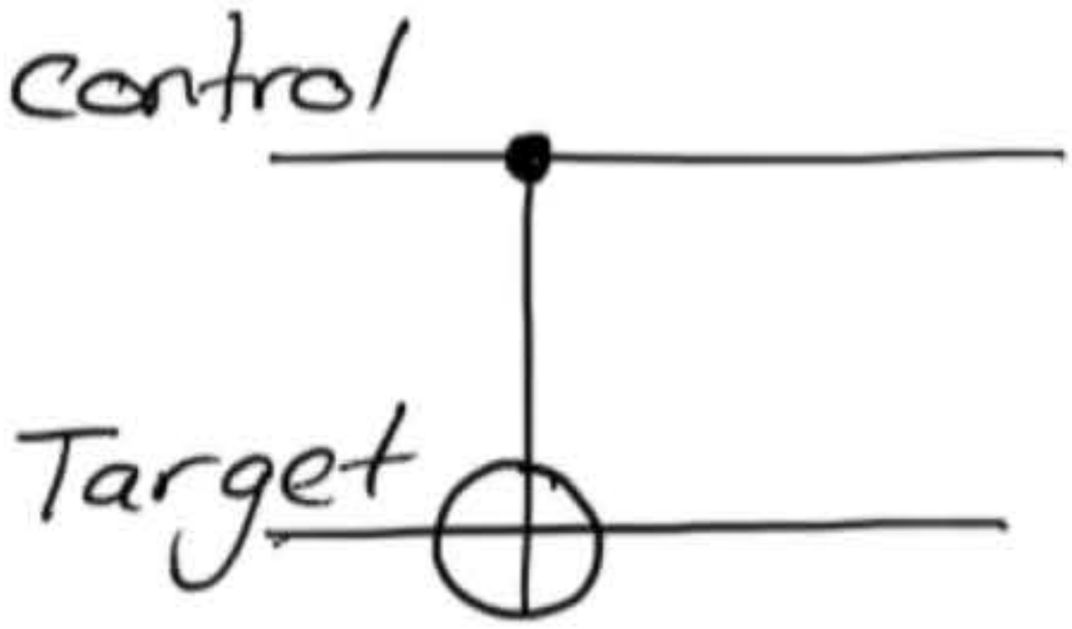
The qubit's state is a combination of the values along all three axes. This is superposition. Some texts describe this property as being in all possible states at the same time, while others think that's somewhat misleading and that we're better off sticking with the probability explanation. Either way, a quantum computer can actually do math on the qubit while it is in superposition—changing the probabilities in various ways through logic gates—before eventually reading out a result by measuring it. In all cases, though, once a qubit is read, it is either 1 or 0 and loses its other state information.



Qubits typically start life at 0, although they are often then moved into an indeterminate state using a Hadamard Gate, which results in a qubit that will read out as 0 half the time and 1 the other half. Other gates are available to flip the state of a qubit by varying amounts and directions—both relative to the 0 and 1 axes, and also a third axis that represents phase, and provides additional possibilities for representing information. The specific operations and gates available depend on the quantum computer and toolkit you're using.

ENTANGLEMENT: WHERE THE ACTION IS

Groups of independent qubits, by themselves, aren't enough to create the massive breakthroughs that are promised by quantum computing. The magic really starts to happen when the quantum physics concept of entanglement is implemented. (One industry expert likened qubits without entanglement as being a "very expensive classical computer.") Entangled qubits affect each other instantly when measured, no matter far apart they are, based on what Einstein euphemistically called "spooky action at a distance." In terms of classic computing, this is a bit like having a logic gate connecting every bit in memory to every other bit.



You can start to see how powerful that might be compared with a traditional computer needing to read and write from each element of memory separately before operating on it. As a result, there are multiple large potential gains from entanglement. The first is a huge increase in the complexity of programming that can be executed, at least for certain types of problems. One that's creating a lot of excitement is the modeling of complex molecules and materials that are very difficult to simulate with classical computers. Another might be innovations in long-distance secure communications—if and when it becomes possible to preserve quantum state over large distances. Programming using entanglement typically starts with the C-NOT gate, which flips the state of an entangled particle if its partner is read out as a 1. This is sort of like a traditional XOR gate, except that it operates only when a measurement is made.

QUANTUM ALGORITHMS WILL CHANGE CRYPTOGRAPHY

Superposition and entanglement are impressive physical phenomena, but leveraging them to do computation requires a very different mindset and programming model. You can't simply throw your C code on a quantum computer and expect it to run—certainly not to run faster. Fortunately, mathematicians and physicists are way ahead of the computer builders here, having developed clever algorithms that could take advantage of quantum computers decades before the machines started to appear.

Some of the first quantum algorithms created—and honestly, some of the few useful ones I've found that you can understand without a graduate degree in math—are for secure cryptographic-key distribution. These algorithms use the property of entanglement to allow the key creator to send one of each of many pairs of qubits to the recipient. The full explanation is pretty long, but the algorithms rely on the fact that if anyone intercepts and reads one of the entangled bits en route, the companion qubit at the sender will be affected. By passing some statistics back and forth, the sender and receiver can figure out whether the key was transmitted securely or hacked on the way.

You may have read that quantum computers could one day break most current cryptography systems. That's because clever algorithms designed to run on quantum computers can solve hard math problems, which in turn can factor very large numbers; one of the most famous is Shor's Factoring Algorithm. The difficulty of factoring large numbers is essential to the security of all public-private key systems, the most commonly used today. Current quantum computers don't have nearly enough qubits to attempt the task, but experts predict they will within three to eight years. That leads to some potentially dangerous situations: for example, imagine if only governments and the super-rich had access to the ultra-secure encryption provided by quantum computers.

WHY BUILDING QUANTUM COMPUTERS IS HARD

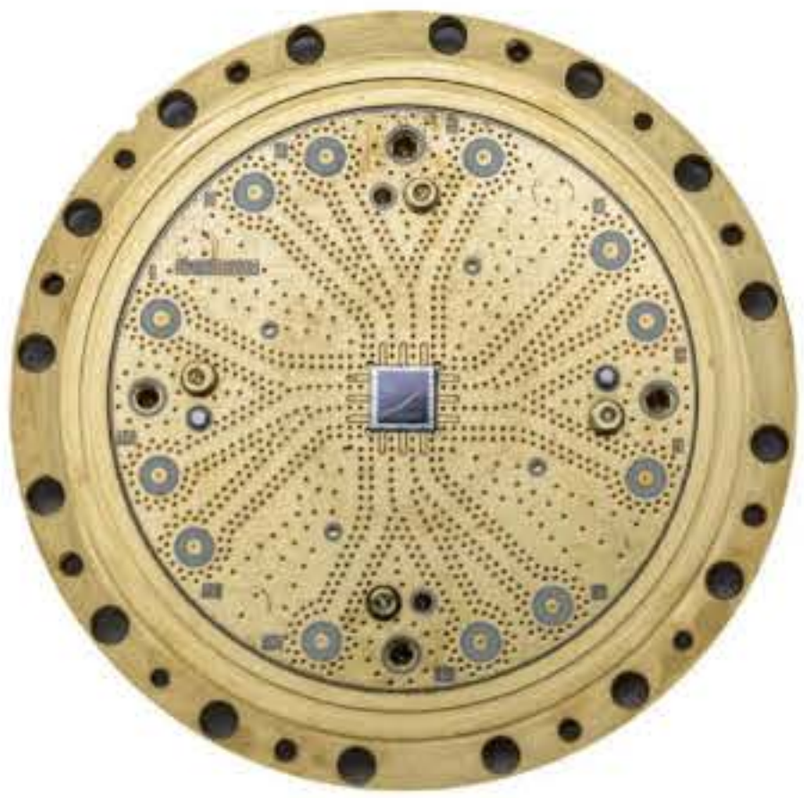
There are plenty of reasons quantum computers are taking a long time to develop. For starters, you need to find a way to isolate and control a physical object that implements a qubit. That also requires cooling it down to essentially zero (as in .015 degrees Kelvin, in the case of IBM's Quantum One). Even at such a low temperature, qubits are stable (retaining coherence) for only a very short time. That greatly limits programmers' flexibility in how many operations they can perform before needing to read out a result.

Not only do programs need to be constrained, but they also need to be run many times, as current qubit implementations have a high error rate. Additionally, entanglement isn't easy to implement in hardware. In many designs, only some of the qubits are entangled, so the compiler needs to be smart enough to swap bits around as needed to help simulate a system where all the bits can potentially be entangled.

GETTING STARTED WITH QUANTUM COMPUTING

The good news is that trivial quantum computing programs are actually pretty easy to understand, if a bit confusing at first. Plenty of tutorials are available that will help you write your first quantum program, as well as let you run it on a simulator and possibly even on a real quantum computer.

One of the best places to start is with IBM's QISKit, a free quantum toolkit from IBM Q Research that includes a visual composer, a simulator, and access to an actual IBM quantum computer after you have your code running on the simulator. Rigetti Quantum Computing has also posted an easy intro application that relies on their toolkit and can be run on their machines in the cloud.



But the trivial applications are just that: trivial. So simply following along with the code in each example doesn't really help you master the intricacies of more sophisticated quantum algorithms. That's a much harder task.

Thanks to William Poole and Sue Gemmell for their thoughtful input.