THE SCIENCES
JULY/AUGUST 1999

QUANTUM COMPUTING

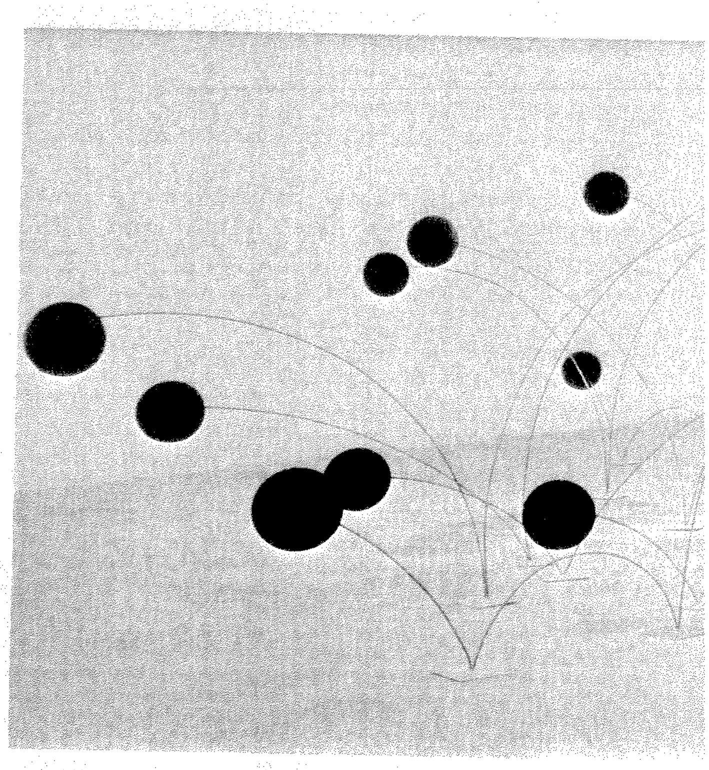
How the weird logic of the subatomic world could make it possible for machines to calculate millions of times faster than they do today

BY LOV K. GROVER

IN THE DARKEST HOUR OF THE SECOND World War, when France had fallen and German submarines prowled the Atlantic at will, the finest minds in Britain were clustered in a place known as Bletchley Park. There, in low wooden buildings thrown up on the grounds of a country estate fifty miles northwest of London, thousands of men and women worked furiously to decode the orders the Axis sent its ships and troops. As intercepted radio messages chattered out of the teleprinters, clerks would snatch them and copy them by hand onto standardized forms. Then the cryptanalysts would take over. Mostly, they worked by hand: comparing sheets of gridded paper, running through permutations, crossing off false starts, staring, shuffling, guessing. It worked. For days or weeks the fog of war would clear as the Bletchley Park team found the keys to the Red and Light Blue ciphers of the Axis air force, the naval and army Enigma ciphers and then, triumphantly, the super-secret Fish cipher of the German high command.

It seems astonishing that the fate of Europe may once have depended on pencil stubs and intuition. Today, when calculations become complex and the stakes are high, ordinary brainpower almost always gets an electronic assist from a computer—and the faster and more powerful the better. Many of the hardest calculations still have to do with codes—though they are intended mostly to test security, not compromise it. But today's codes make Fish seem like the simple letter-substitution ciphers beloved of puzzlers in children's magazines (*a* becomes *b*, *b* becomes *c* and so forth). One is the so-called RSA protocol, which makes electronic banking possible by assuring banks and their customers that a bogus transfer of funds or a successful forgery would take the world's fastest computer millions of years to carry out. Another is the widespread Data Encryption Standard (DES), which remains secure for most ordinary business transactions.

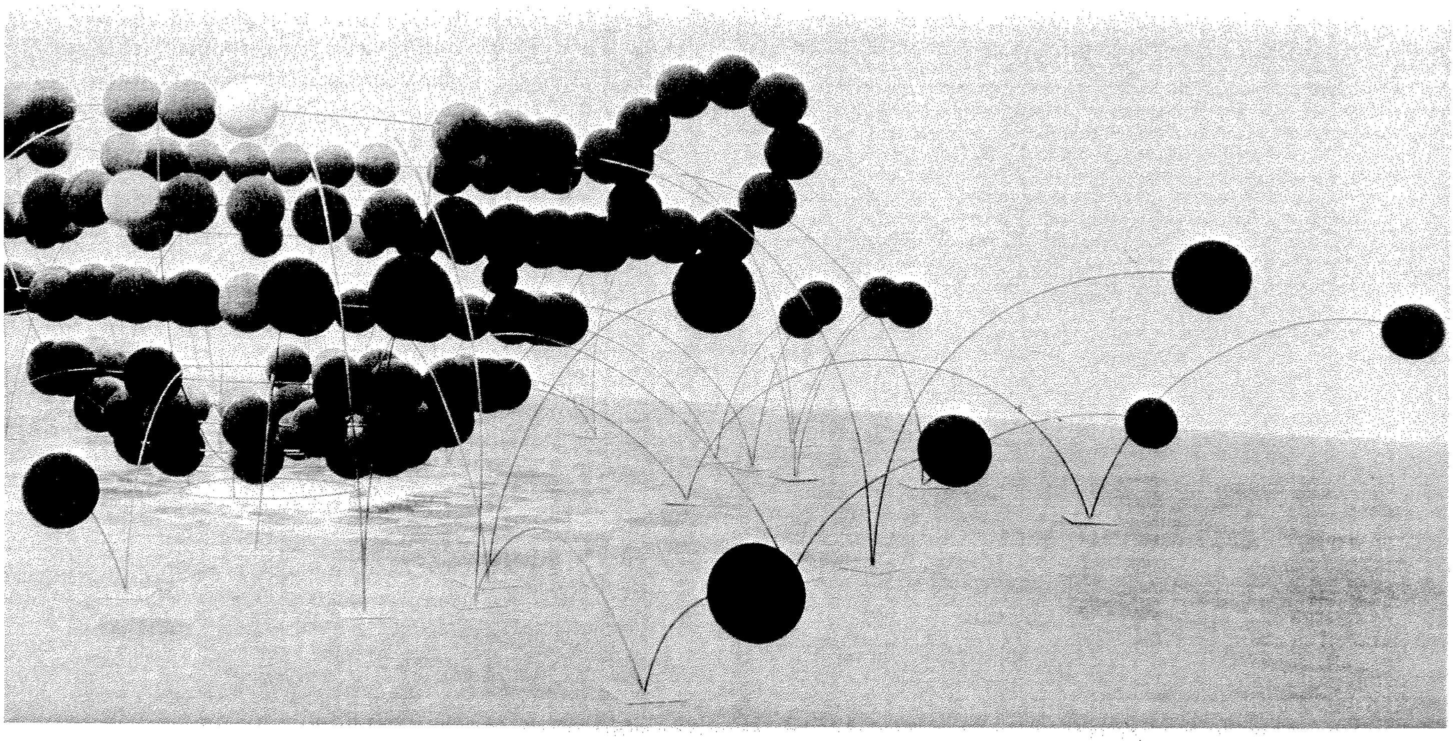
Other calculations—searching the Internet, modeling the national economy, forecasting the weather—likewise strain the capacities of even the fastest and most powerful computers. The difficulty is not so much that microprocessors are too slow; it is that computers are inherently inefficient. Modern computers operate according to programs that divide a task into elementary operations, which are then carried out serially, one operation at a time. Computer designers have tried for some time to coax two or more computers (or at least two or more microprocessors) to work on different aspects of a problem at the same time, but progress in such parallel computing has been slow and fitful. The reason, in large part, is that the logic built into microprocessors is inherently serial. (Ordinary computers sometimes appear to be doing many tasks at once, such as running both a word-processor and a spreadsheet program, but in reality the central processor is simply cycling rapidly from one task to the next.)



A truly parallel computer, in contrast, would have simultaneity built into its very nature. It would be able to carry out many operations at once, to search instantly through a long list of possibilities and point out the one that solves a problem.

Such computers do exist. They are called quantum computers—not so much because they are inherently small, but because they operate according to the bizarre rules of quantum mechanics, which do indeed govern the world of the very small: the waves and particles of subatomic physics. One quantum rule in particular creates an enormous incentive to apply quantum mechanics to computing: the startling discovery by twentieth-century physicists that elementary particles such as protons, neutrons and electrons can persist in two or more states at once. That makes it possible, at least in principle, for them to be harnessed as processing units in a machine more efficient than any conventionally designed "classical" computer could ever be.

In the past few years, simple quantum computers have been built in the laboratory. Meanwhile, in my field, the theory of quantum computation, investigators are eagerly describing what the more sophisticated machines will be like if they can ever be constructed. We are, if you will, writing the software for a device that does not yet exist. Yet on paper, at least, the prospects are stunning: an algorithm that could factor 140-digit-long numbers a billion (10^9) times faster than is currently possible with the best nonquantum methods; a search engine that could examine every nook and cranny of the Internet in half an hour; a "brute-force" decoder that could unscramble a DES transmission in five minutes.


Dennis Oppenheim, Image Dissonance—Coffee Cup, *1988–89*

PERHAPS THE MOST SURPRISING THING ABOUT quantum computing is that it was so slow to get started. Physicists have known since the 1920s that the world of subatomic particles is a realm apart, but it took computer scientists another half century to begin wondering whether quantum effects might be harnessed for computation. The answer was far from obvious.

Boiled down to its essentials, any computer must meet two requirements: it must be able to store information as strings of 1's and 0's, or bits, and it must have a way of altering the bits in accordance with instructions. A computer transforms its bits by means of gates, or devices designed to carry out simple operations in logic. For example, a NOT gate converts any input bit into its opposite (0 becomes 1, and 1 becomes 0). An OR gate, by contrast, converts two input bits into a single bit whose value is the higher of the two (0 OR 0 yields 0; any other combination gives 1). And an AND gate yields a 1 only if both input bits are 1's; otherwise, its output is a 0. Everything a computer does—whether synthesizing speech, calculating the billionth digit of pi or beating Garry Kasparov at chess—ultimately comes about through the transformation of bits by gates.

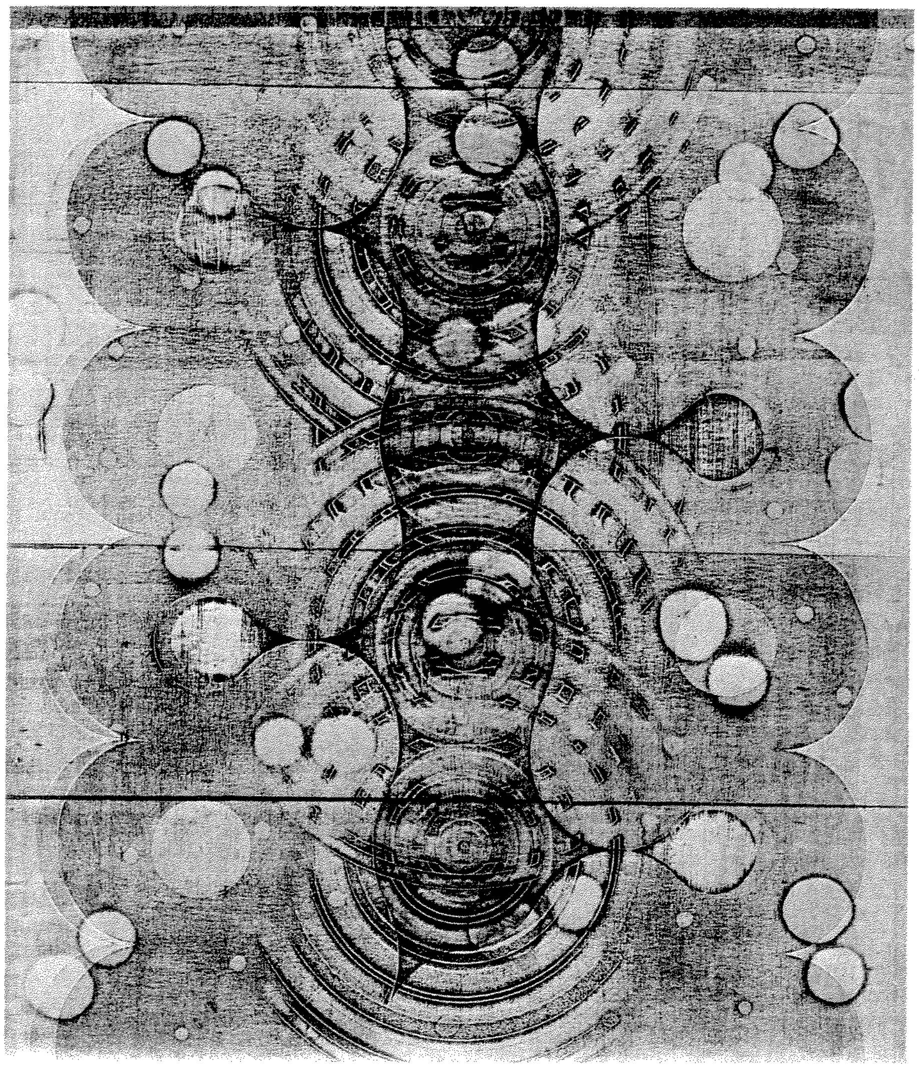
Could subatomic particles store bits? Could they form gates? Consider the electron. Every electron acts as if it were a little magnet, spinning about an axis, whose magnetic moment can point in only one of two directions, up or down. Thus the spin of the electron is quantized: it has just two possible states, which can readily be identified with the 0's and 1's of an ordinary computer processor. And you can flip the bit—that is, change a down, or 0, to an up, or 1, by adding just a smidgen of energy.

Suppose, however, you give it less energy than that—say, half a smidgen. Once again, when you observe the electron's spin state, you will find that the spin is quantized: it points either up or down. But now there is an important, though subtle, difference. According to the rules of quantum mechanics, the probability of observing the spin in one or the other state will change. That change arises from a qualitatively new state, with no analogue in the ordinary, nonquantum, laws of physics, called a superposition of the two spin states: a combined, in-between condition that can be, say, 60 percent up and 40 percent down, or 22 percent up and 78 percent down.

In the 1970s and early 1980s physicists and computer scientists began to investigate how the properties of quantum superpositions might be applied to computing. Early workers in the field—including the physicists Charles H. Bennett of the IBM Thomas J. Watson Research Center in Yorktown Heights, New York, Paul A. Benioff of Argonne National Laboratory in Illinois, David Deutsch of the University of Oxford and the late Richard P. Feynman—showed that particles in superposed states can function as quantum bits, or qubits, and can undergo operations analogous to the NOT, OR and AND operations of classical computing. But that is not all. Quantum computers, if they can be built, could achieve results that would seem almost magical—and quite different from anything a classical system has to offer.

Imagine a quantum computer made of two atomic nuclei acted on by an external magnetic field. Suppose the nuclei belong to the neighboring atoms of carbon and hydrogen in a single molecule of chloroform, $CHCl_3$. Just as electrons do, the nuclei align their spins with the magnetic field in the direction up (1) or down (0). One can now begin to compute with this toy system by tickling the nuclei with radio waves. By tuning the frequency and duration of a radio pulse in just the right way, it is possible to make one or the other nucleus flip its spin. It is even possible to ensure that the hydrogen nucleus flips over only if the carbon nucleus is already pointing up. In that case the quantized behavior of the two nuclei functions as what computer scientists call a controlled-NOT gate, with the carbon nucleus as the control. In symbols, with carbon in the first place and hydrogen in the second, there are four possible inputs, (1,1), (1,0), (0,1) and (0,0). Controlled-NOT can then operate in one of four ways: $(1,1) \rightarrow (1,0)$; $(1,0) \rightarrow (1,1)$; $(0,1) \rightarrow (0,1)$; $(0,0) \rightarrow (0,0)$.

Physicists and computer scientists have proved that, by stringing together single-qubit operations and two-qubit controlled-NOT gates, it is theoretically possible to build a quantum computer capable of doing anything a classical computer can do.


Robert Straight, P-289, *1996*

But the real magic transpires when a two-qubit gate acts on a particle that is in a superposition of spin states. First, place the chloroform molecule in a strong external magnetic field that aligns both atomic nuclei into the down, or 0, position. Then, with a pulse of tuned radio waves, tweak the carbon nucleus so that it does a partial flip, into a superposed state for which the probabilities for both spin directions are each 50 percent (a single-qubit operation). Finally, carry out a controlled-NOT operation with the carbon nucleus as the control qubit.

Because the second qubit (the hydrogen nucleus) started out in the zero state, only two of the operations are relevant: $(1,0) \rightarrow (1,1)$ and $(0,0) \rightarrow (0,0)$. In other words, if the carbon nucleus had initially gotten flipped to a 1, the controlled-NOT operation would flip the hydrogen nucleus into the 1 state, too. If the carbon had remained a 0, the controlled-NOT operation would have left the hydrogen in the 0 state, too. But the action of controlled-NOT on the superposed state of the carbon nucleus and the 0 state of the hydrogen nucleus leaves the two-qubit system as a whole in a more complicated superposition, with a 50 percent chance of being in the (1,1) state and a 50 percent chance of being in the (0,0) state. Such a superposition is called an EPR state, after the physicists Einstein, Boris

Podolsky and Nathan Rosen, who first studied it in 1935.

Intensely puzzling aspects of the EPR state arise when the two qubits are physically separated and independently operated on. Suppose you measure only the value of the first qubit, the spin state of the carbon nucleus. When you do that, you run up against one of the fundamental rules of quantum mechanics: If an interaction gives any information about the state of a quantum system, the rest of the system immediately readjusts itself to be consistent with that information. In general, any attempt to measure or observe a system in a superposition of two or more states immediately forces the system to make a decision. Instead of continuing in its intermediate, superposed state, the quantum computer jumps into just one of the possible quantum states open to it. In the language of quantum mechanics, it decoheres.

By observing the carbon qubit of the EPR state, then, you force decoherence and destroy the superposition; you had an even chance of observing a 0 or a 1, but you could observe only one or the other value. But that observation also implies that the system as a whole cannot continue to be in its superposition of the two states (0,0) and (1,1); instead, it too takes on a single, definite state, and so the hydrogen qubit assumes the same value as the carbon one. In quantum-mechanical terms, both states appear at the same time because the two nuclei have become entangled.

IF THAT WERE THE WHOLE STORY, QUANTUM computing might not seem particularly interesting. After all, the final outcome is just two identical qubits—with random values, at that. The exciting thing about entanglement, however, is that you do not have to measure those values right away. Instead, you can leave the system in its superposed state and carry out any number of delicate and intriguing operations on the qubits before you finally decide to make an observation. Meanwhile, any quantum operation on the system acts on all of the states simultaneously. If the number of qubits is q, the total number of possible states is 2^q.

Thus from, say, 500 particles you could, in principle, create a quantum system that is a superposition of as many as 2^{500} states. Each state would be a single list of 500 1's and 0's. Any quantum operation on that system—a particular pulse of radio waves, for instance, whose action was, say, to execute a controlled-NOT operation on the 175th and 176th qubits—would simultaneously operate on all 2^{500} states. Hence with one machine cycle, one tick of the computer clock, a quantum operation could compute not just on one machine state, as serial computers do, but on 2^{500} machine states at once! That number, which is approximately equal to a 1 followed by 150 zeros, is far larger than the number of atoms in the known universe. Eventually, of course, observing the system would cause it to collapse into a single quantum state corresponding to a single answer, a single list of 500 1's and 0's—but that answer would have been derived from the massive parallelism of quantum computing.

The consequence is that for some purposes quantum computers would be so much faster than classical computers are that they could solve problems the classical computers cannot touch. If functioning quantum computers can be built, harnessing their potential will be just a matter of creating algorithms that carry out the right operations in the right order.

A QUANTUM COMPUTER MADE UP OF 500 PARTICLES *could compute on more machine states simultaneously than there are atoms in the known universe.*

THAT REALIZATION HAS TOUCHED OFF AN explosion of research into the theory of quantum computing. The field poses some unique challenges. For one thing, the mathematics involved in analyzing the evolution of a system of particles is daunting. Just as you might fly from New York to Los Angeles either directly or via any of several hub airports, a subatomic particle changing from one state to another can take several possible paths. The difference is that, whereas your plane can take only one of the available routes, the particle acts like a wave that simultaneously takes them all. Furthermore, the probability of finding the particle along each path fluctuates from point to point and from moment to moment, as if it were a wave with crests and troughs. To evaluate the probability that any particular state will come to pass, one must sum the probabilities of all the paths leading to that state, being careful to keep the probability waves along each path in the proper phase.

Another obstacle is that quantum computers are extremely fragile. To remain in an intermediate, superposed state, a quantum-mechanical system needs to be almost totally isolated from its environment; the slightest interaction with anything outside itself will perturb the system, destroy the superposition and upset the computation. As a result, anyone who wants to build a quantum computer must be careful to shield it from heat, cosmic rays and other potential outside influences—including outside observers. Moreover, once your quantum computer has solved a problem for you, your own need to read out the answer forces you to destroy the system.

The quirkiness of quantum systems goes a long way toward explaining why the most impressive strides in quantum computing so far have taken place on paper. While experimenters struggle to build even rudimentary systems in the lab, theorists have raced ahead to anticipate the software such computers will need, should they ever become practical.

One essential piece of software for any computing technology is a way to correct errors. Machines make mistakes. Classical computers are designed to catch errors through redundancy. They perform each elementary computation several times and then accept the most frequent answer as correct. Such a "majority rules" approach would not work in quantum computers, though, because comparing answers would entail precisely what must be avoided: observing the system before the computation is done.

Surprisingly, however, quantum error correction is possible. In 1995 the mathematician Peter W. Shor, now at AT&T Labs in Florham Park, New Jersey, and the physicist Andrew M. Steane of the University of Oxford independently devised a scheme that deftly tiptoes through the quantum minefield. The scheme detects error in a way that reveals nothing about the ongoing computation (and thus does not ruin the superposition). Then it fixes the error by means of another quantum computation, which also keeps the quantum system intact.

A SIMILARLY SUBTLE APPROACH HAS BEEN devised for factoring large numbers. Factoring is what computer scientists call a one-way problem: hard in one direction but easy in the other. Suppose I asked you the question, "Which two integers can be multiplied to obtain the number 40,301?" Systematically testing all the candidates might keep you busy for fifteen minutes or so. But if I asked you to multiply 191 by 211, it would take you only about twenty seconds with pencil and paper to determine that the answer is 40,301.

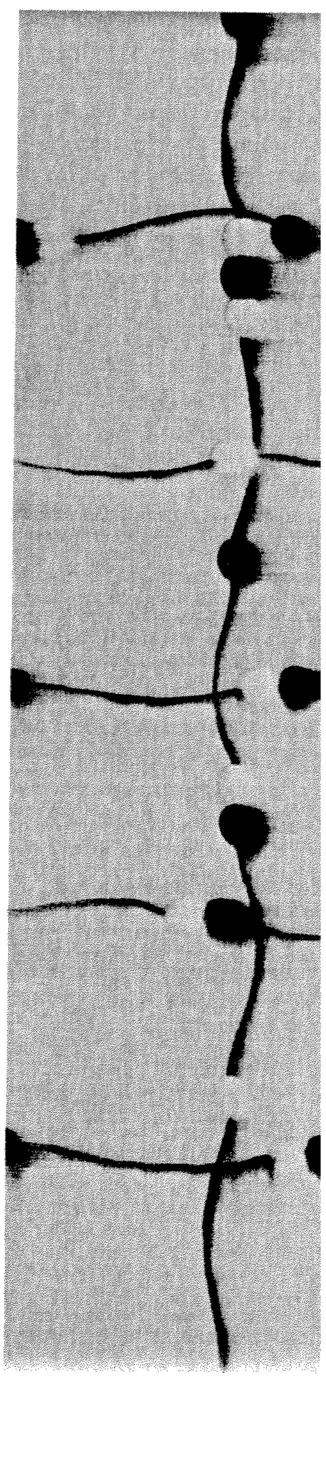
The lopsided difficulty of factoring compared with multiplication forms the basis for practical data encryption schemes such as the RSA protocol. Large prime numbers—say, a hundred digits each or so—make good "passwords" for such systems because they are easy to verify: just multiply them together and see whether their product matches a number that is already stored or that might even be made publicly available. Extracting the passwords from a 200-digit composite product of two large primes, however, is equivalent to factoring the large composite number—a problem that is very hard, indeed. The largest number that ordinary supercomputers have been able to factor with nonquantum algorithms is "only" 140 digits long.

Quantum algorithms, however, are another matter. In 1994 Shor discovered one that makes factoring almost as efficient as multiplication. In computer science, one often tries to solve hard problems by converting them into simpler problems that one already knows how to solve. In that spirit, Shor started by employing well-known results from number theory to convert the problem of factoring into one of estimating the periodicity of a long sequence.

Periodicity is the number of elements in the repeating unit of a sequence. The sequence 0, 3, 8, 5, 0, 3, 8, 5, . . . , for instance, has a periodicity of four. To estimate periodicity, a classical algorithm must observe at least as many elements as there are in the period. Shor's algorithm does much better. It sets up a quantum system made up of a large number of superposed states. Each state is identified with an element of the repeating sequence. A single quantum mechanical operation then transforms each superposed state in a way that depends on the value of the sequence to which the state corresponds. A series of such quantum mechanical operations, mathematically analogous to X-ray diffraction, is carried out on the superposed states.

The method works for much the same reason X-ray diffraction enables mineralogists to deduce the periodicity of the crystal lattice of an unknown solid substance. The periodic structure of the lattice allows only waves of certain wavelengths to propagate in any specified direction. Similarly, in Shor's algorithm, the quantum system of superposed states allows only certain of the wavelike probabilities associated with the quantum states to "propagate"; the rest are canceled out or damped away. The algorithm then calculates those propagating wavelengths, estimates the periodicity of the superposed states, and finally deduces the factors of the number. The result is the fastest factoring algorithm known.

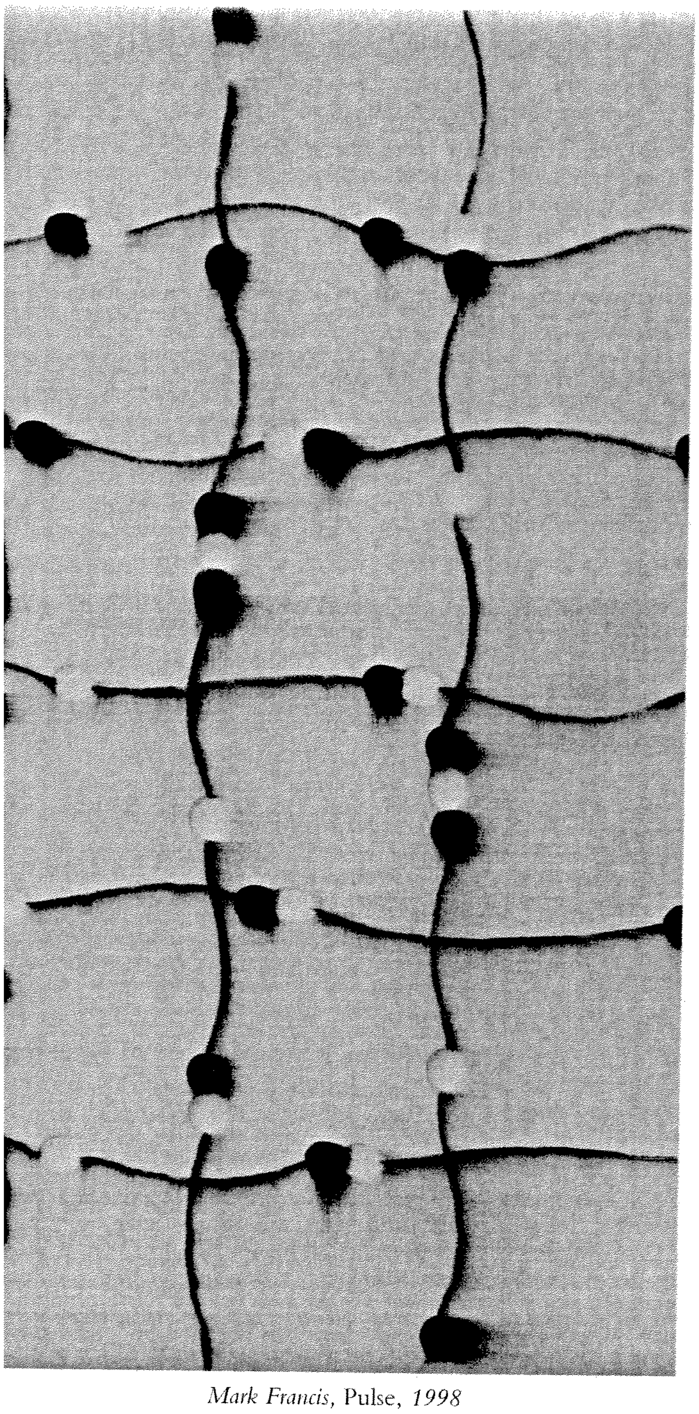
Factoring is a kind of search—a search for factors. For other searches, however, a more general-purpose algorithm is needed. My own most important contribution to quantum computation is an efficient quantum mechanical algorithm for searching unsorted data bases. That algorithm, which I discovered in 1996, is faster than any classical algorithm can ever be. More than that, it has been shown that no other quantum mechanical algorithm can ever beat it, either.



IMAGINE YOU WANT to look up a phone number in a telephone directory that has a million entries. Suppose, too, that you have forgotten the person's name; all the information you have to search with is an address. In that case, your only recourse is trial and error. On average, you will read the names of 500,000 strangers before you find the one you want; on a very bad day, you might have to look at 999,999 of them. A computer could search much faster, but algorithmically it would be in the same boat: in general, a list of N items takes, on average, $N/2$ steps to search.

A quantum computer could do much better, thanks to its ability to carry out many operations at the same time. Assuming you had access to a quantum system, here is how you could do the search: First choose enough particles (some number q) so that there are enough quantum states in the system (2^q) to assign at least one state to each name in the phone book. (To get a million names, for instance, you would need twenty particles, since 2^{20} is slightly more than a million.) Place the information from the phone book into quantum memory, and match each name to a different quantum state. Put the system into a superposition of the million or so states. Now the system can do a computation that checks whether each of the names is the right name. Thanks to quantum superposition, the computation takes place on all states simultaneously.

At this stage the answer is in the system; the trick is to get it out. Observing the system without further processing would instantly collapse the superposition into one of its million entangled states, but the chances that that state will be the one that gives you the name you need would be just one in a million. You might just as well have picked a name at random from the phone book.



Mark Francis, Pulse, *1998*

The way around the problem, then, is further processing: a sequence of quantum mechanical operations on the superposed states that amplifies the probability that when the superposition is observed, it will collapse only into the state corresponding to the desired name. That is what my search algorithm does. Like Shor's factoring method, the algorithm takes advantage of the wavelike nature of probability in a quantum computer.

To take a relatively straightforward example of how the process works, suppose you want to find a name in a phone book that has only four entries. To set up your quantum computer, you decide to compute with a pair of particles—for variety, choose protons this time—and you arrange things so that each name in the phone book corresponds to a distinct combination of spins: (0,0), (0,1), (1,0) or (1,1). Now suppose that, unbeknownst to you, the name you want to find corresponds to the third state, (1,0). That state is the target.

YOU BEGIN BY INITIALIZING THE SPINS OF THE protons with a strong magnetic field, aligning both of them in the up direction. Then you give each particle a fainter dose of magnetism, just enough energy to change the spin state to a superposition that is 50 percent up and 50 percent down (a "50 percent flip"). The two-particle system has now become a superposition of the four possible combinations of spins, each with a probability of 1/4.

In quantum mechanics each probability is treated mathematically as the square of a theoretical (but not directly observable) construct called the probability amplitude. Strictly speaking, what my search algorithm manipulates are probability amplitudes. The advantage of working with probability amplitudes is that, unlike actual probabilities, they can be either positive or negative, and so they can cancel one another out, just as waves do in water. The algorithm makes use of that property by canceling computational paths that initially seem promising but that later turn out to be dead ends.

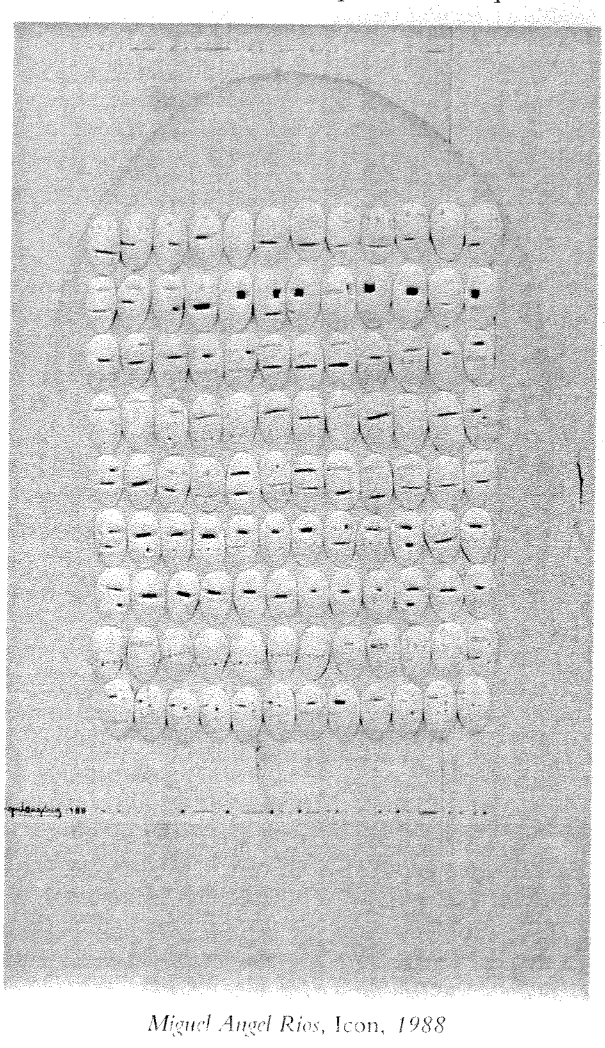
Since each of the four superposed states has a probability of 1/4, the probability amplitude of each state in my two-particle example can be either +1/2 or −1/2 (technically, in fact, it can even be a complex number). The algorithm ensures that all the probability amplitudes begin with the same value: (1/2, 1/2, 1/2, 1/2). Now comes the heart of the algorithm. The first operation changes the sign of the amplitude of the target state (in my example, the third state); thus the amplitudes change to (1/2, 1/2, −1/2, 1/2). That is possible because, in a sense, when the quantum computer is in the target state, it can verify that it is indeed in the right state and can then invert the phase in that state. Note that this operation reveals nothing to the outside world, because the probabilities—that is, the squares of the probability amplitudes—remain unchanged.

Next come three quantum operations: a 50 percent flip, an operation that inverts the phase of one of the states, and another 50 percent flip. The net effect is a maneuver called "inversion about the average." If you imagine the average value as a crossbar whose height is equal to the average value of the amplitudes, with the various individual amplitudes jutting up or dangling down from it, you invert each amplitude by flipping it over to the opposite side of the bar.

What is the net effect? The average of the amplitudes in the four states, after changing the sign of the amplitude of the target state, is (1/2 + 1/2 – 1/2 + 1/2)/4, or 1/4. The first state has an amplitude of 1/2, which is 1/4 above the average, and so after the inversion about the average its amplitude becomes 1/4 below the average, or 0. A similar calculation for each state shows that the amplitudes become (0, 0, 1, 0). The square of each of those numbers gives the probability of each state. In other words, the effect of the operations is to drive the quantum computer into the target state; the probability of the target state, (1,0), has reached certainty. If you now observe any of the spins of the protons, the quantum superpositions will collapse into a display of the right answer.

MOST SEARCHES, OF COURSE, WOULD SCAN a list longer than four items. To do so, the algorithm might repeat the three quantum operations many times, nudging the system toward the desired state with every pass through the loop. What makes quantum searching so powerful is that, for a list of *N* items, the algorithm requires only about the square root of *N* steps to find its quarry—not the *N*/2 steps of the classical trial-and-error search. Thus a quantum computer could search a million-name phone book in 1,000 tries instead of half a million. The longer the list, the more dramatically the quantum algorithm will outpace its classical rival.

The algorithm is good for more than merely looking up phone numbers. It can search through items not explicitly spelled out on a list, and so it can take a brute-force approach to any problem whose potential solutions can be systematically enumerated, trying them all until it stumbles on a solution that works. Nonquantum versions of the brute-force approach are already a staple of computing: such algorithms are applied in programs that play games such as chess. Because of its speed and versatility, it seems likely that the search algorithm will be a key component in the software of the future.


Miguel Angel Rios, Icon, *1988*

IF THE TWENTY-FIRST CENTURY IS TO BE AN AGE of quantum computers, some major advances will be necessary. Certainly the working models built to date come nowhere near their theoretical potential. Even factoring two-digit numbers remains beyond them. The most promising approach so far is a spin-off from the medical technology of nuclear magnetic resonance (NMR) imaging. The computers are molecules in a liquid, and information is encoded in atomic nuclei in the molecules. Instead of trying to coax results out of a few fragile qubits, the technique is based on manipulating, or, in effect, programming, enormous numbers of nuclei with radio-frequency pulses and then harnessing statistics to filter the right answers (about one result in a million) out of the background of noise.

In 1997 the physicists Isaac L. Chuang of the IBM Almaden Research Center in San Jose, California, Neil A. Gershenfeld of the Massachusetts Institute of Technology in Cambridge and Mark G. Kubinec of the University of California, Berkeley, built a simple two-qubit NMR quantum computer made of liquid chloroform. The "program" the computer ran was my search algorithm, applied to a list of four items. More recently, a group at the University of Oxford built a similar device out of two hydrogen nuclei in the organic chemical cytosine. Three-qubit NMR quantum computers running Shor's and Steane's error-correction routine were demonstrated in 1998 by Raymond Laflamme and his coworkers at Los Alamos National Laboratory in New Mexico.

Will quantum computers ever grow into their software? How long will it take them to blossom into the powerful calculating engines that theory predicts they could be? I would not dare to guess, but I advise all would-be forecasters to remember these words, from a discussion of the Electronic Numerical Integrator and Calculator (ENIAC) in the March 1949 issue of *Popular Mechanics:*

Where a calculator on the ENIAC is equipped with 18,000 vacuum tubes and weighs 30 tons, computers in the future may have only 1,000 vacuum tubes and weigh only 1.5 tons. ●

LOV K. GROVER is a member of the technical staff in physical sciences research at Bell Labs in Murray Hill, New Jersey. He is the inventor of what has been proven to be the fastest possible search algorithm that could run on a quantum computer.