A Brief Overview of Quantum Computing

C. E. Coppola
Department of Physics, North Carolina State University, Raleigh, NC 27695-8202

The theoretical benefits of quantum computation have been known for decades, but it was not until the discovery of algorithms that showed exponential decreases in computation time on quantum computers that they received significant attention as a field of research.[1] The potential power of quantum parallelism as applied to computation and the difficulties posed by decoherence are reviewed. The background of quantum computation and the basic operation of a quantum computer are described and contrasted to classical computing, in order to emphasize the advantages of quantum computers and the unique difficulties in constructing them.

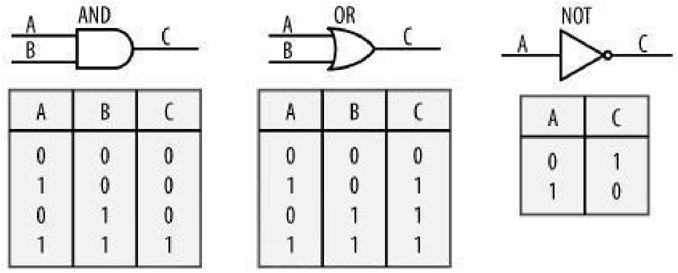
I. Introduction

The concept of a quantum computer has been around since at least 1982, when Richard Feynman proposed that a device operating to exploit quantum phenomena could be used to perform computations. Since then, computer scientists and physicists have worked to discover the properties of this kind of machine with the hope of applying its power to classically intractable problems. Considerable attention has been given to the problem of figuring out how to use a quantum computer to perform useful calculations, and there has been significant achievement in this area. The difficulties in actually building such a computer are clear, and progress in construction of these machines has lagged considerably behind the understanding of their behavior and theoretical limitations.

Quantum computers must be constructed out of atom-sized elements, and therefore require refined methods of storage and manipulation that are still being engineered. They are probabilistic, meaning that accurate solutions require a computation be performed many times. Quantum computers are also strongly susceptible to decoherence due to the scale of their components, and addressing this problem through quantum error correction is a critical aspect of actually constructing them.

The most important property of quantum computers, and the one which has caused an entire field to grow around these nonexistent devices, is their potential power. Classical computing power scales linearly with the size of the register. In order to double the computing power of a classical computer, the number of computing elements must be doubled. Quantum effects allow for the possibility of a computer whose power scales exponentially with the size of the register - only one additional element is necessary to double the power of a quantum computer. This power is difficult to access, however, since it is hidden in quantum effects, and accessing it requires ingenious construction and programming methods. Some impressive algorithms and techniques have already been discovered.

Quantum computation is best examined in contrast to classical computation, which is well understood. Quantum effects alter the methods of computation and storage. The comparison allows for a direct understanding of quantum computers, and offers insights into their significance.



A	B	C
0	0	0
1	0	0
0	1	0
1	1	1

A	B	C
0	0	0
1	0	1
0	1	1
1	1	1

A	C
0	1
1	0

FIG. 1: Three of the basic logic gates of classical computing.

II. Classical Computation

Classical computers are built out of bits, which are switches that store information in binary, the most commonly used language of computation. Information in binary is stored in strings composed of two values, represented as 0 and 1. These bits are collected into large arrays that allow the computer to hold large amounts of information between operations. Computations are done on these bit arrays using what are called Boolean operators. Boolean operators act on two input values to give a third output value, and these are the familiar logic gates of computer science AND, OR, NOT, and their derivatives. Logic gates are the functional building blocks of computation, and all operations that a computer performs are constructed out of these gates.

Boolean logic gates are not reversible, meaning that the final states do not uniquely define the initial ones. This is an inescapable consequence of deterministic computation, and one that has implications for the future of classical computers. Logically irreversible gates destroy information with each operation, increasing the entropy of the system and generating heat. Classical computers are composed of billions of transistors in spaces on the order of 10^{-4} m, that perform billions of operations per second. Heat generated in this environment can only be pumped away so quickly. Too much heat will interfere with the operation of the computer, and can permanently damage the components. The minimum amount of energy dissipated by a logically irreversible binary operation is given by the Landauer bound,

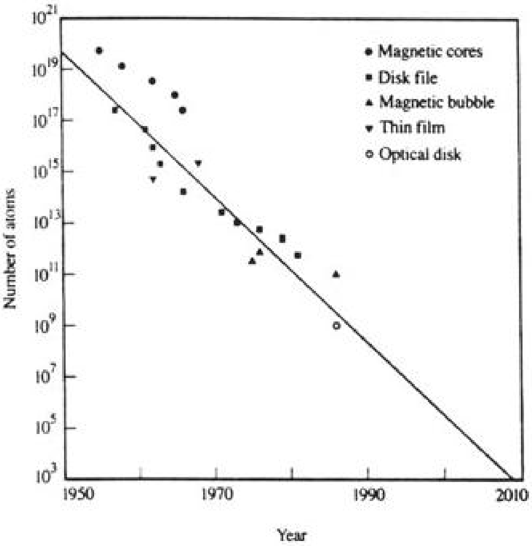


FIG. 2: A plot of the decrease in the number of atoms used to construct a bit in various technologies.

$$E \geq kTln2$$

which effectively limits the construction of classical computers. As the density of transistors increases, the density of heat dissipation in classical processors also increases, which interferes with the semiconducting properties of silicon-based materials. The rapid growth and miniaturization of processor technology has already brought classical computation to the edge of the classical regime of physics, and quantum effects will become even more important as commercial lithography approaches 10^{-9} m widths.[2]

Classical computers can only operate on one bit at a time. This is why a popular metric of the power of classical computers is their speed - the faster a computers core clock, the more operations per second it can perform, and the larger the set of problems it can solve in a useful time. To calculate an answer, a classical computer will perform a series of Boolean operations on a set of bits in a specific order, and then yield the result. Classical computers are called deterministic because without errors, the answer will be the same every time a specific series of operations is performed.

The time necessary to compute a problem is a fundamental measure of both the power of a computer and the difficulty of the problem. Just as classical computers are measured in the number of operations per second then can perform, problems are measured in the length of time necessary to solve them. Complexity theory classifies problems based on how the computation time scales as the size of the input increases. Two of the more relevant complexity classes are called P and NP, and the space in between them is perhaps the most explored in complexity theory due to the relevance of these problems to computer science. P stands for polynomial time, and problems in this class can be solved in a time that is some polynomial of the length of the input:

$$t = \sum_k c_k N^k \approx O(N^K)$$

These problems are considered to be solvable by classical computers, because the order of most interesting problems is low enough that polynomial time remains a practical limit. NP stands for non-deterministic polynomial time. For a problem to be in NP, it must be the case that no algorithm exists that can solve the problem in polynomial time, and also that all solutions to a problem can be verified within polynomial time. It is believed that classical computers are restricted to solving only problems in P due to their deterministic nature. Quantum computers would be able to solve difficult problems in P much more quickly than classical ones, but it is still unknown if they would be able to solve NP problems.[3]

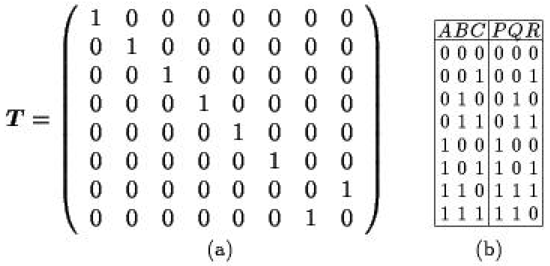
A classical bit array can store a unique string only as large as the number of bits it contains. Bits are composed of large numbers of atoms, so once a bit is stored it retains its exact state indefinitely. This is what we expect from a classical perspective. However, this reliability comes with a hidden price that was not discovered until quantum computers were investigated.

III. Quantum Computation

If elements that exhibit quantum behavior are used to store information instead of bits, the properties of computation change dramatically. Quantum bits are systems which have two possible values, like bits, but are small enough that observation and manipulation of them is limited by the Heisenberg uncertainty principle. This means that utilizing qubits as storage makes reading data more difficult, but by yielding our ability to monitor it continuously, we are allowing the system to contain exponentially more information than would be possible with a classical bit array.

An array of N bits can contain one string of length N, since once the values are written they remain constant. One real number is necessary to represent this state, which would be the decimal value of the string. However, an array of N qubits can contain 2^N strings of length N, since each qubit is in a superposition of its basis states. 2^N complex numbers, the amplitudes of each superposition of basis states, are necessary to represent this state.

Consider a system capable of storing three units of information. A classical computer with three bits could store a single string with three digits, or any of the numbers from 0 to 7. Once this number is written into storage, it is invariable unless there is a fault in the storage device. However, with qubits, it is possible to store a superposition of each number from 0 to 7 with its own probability amplitude. The qubits will maintain these superpositions unless disturbed, and therefore every possible state is available.



$$T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

ABC	PQR
0 0 0	0 0 0
0 0 1	0 0 1
0 1 0	0 1 0
0 1 1	0 1 1
1 0 0	1 0 0
1 0 1	1 0 1
1 1 0	1 1 1
1 1 1	1 1 0

FIG. 3: Toffoli's logic gate, as a) a matrix and b) a truth table.

An array of qubits constitutes a physical quantum state, which by Schrödingers equation must evolve unitarily. Unitary operators are logically reversible, and logic gates that can be represented by unitary operators are the only gates that can operate on the system without destroying the information contained in it. Unitary logic gates that can be combined to produce any operation are called universal gates.

There are many universal logic gates, but they require three input values, as opposed to Boolean gates which operate on two values. Toffoli's gate is an example of one of these gates. Computations in reversible operations can be brought arbitrarily close to physical reversibility. The efficiency of the gates is the only limitation on isentropic computation for logically reversible computations.[4]

The power of the quantum computer is due precisely to the superposition of all possible answers in the qubit array, including the correct one. Performing an calculation on a qubit array will operate on every state in the superposition at once, doing an exponential amount of work in linear time.[5] Any logic gate that executes a unitary transformation can be used, and these computations can be designed to change the probability amplitudes in precise ways. The difficulty in actually using a quantum computer to solve complex problems is constructing the set of operators that will amplify the probabilities of a particular subset of the total qubit space, so that when measured, the computer is more likely to output the correct answer to the problem.

Measuring the state of the qubit array at the end of the calculation will destroy the superposition, and yield each state with frequency according to the square of its probability amplitude. Minimizing, or even eliminating, the amplitudes of incorrect answers is the objective of constructing quantum algorithms, and therefore computations will certainly have to be performed multiple times.[6] The power of a quantum computer is not measured in the time taken to complete an operation, but in the probability that the computation will result in the correct answer.[7]

The accuracy of the computers results is only limited by the number of times the computation is performed. This is a fundamentally different way of describing computational power, and a new complexity class is necessary to describe quantum computing problems. BQP, or bounded quantum polynomial time, is the set of all problems that can be solved on a quantum computer in polynomial time with an arbitrarily small chance of error.

IV. Applications

Coming up with methods for isolating answers to general questions is difficult, but algorithms have already been found that would enable quantum computers to factor large numbers and searching lists much faster than classical computers can.

In 1992 David Deutch and Richard Jozsa published an algorithm that solved an oracle problem in a single operation. Classical computers can best solve this problem in $O(log(N))$ time. This algorithm was one of the first to demonstrate that quantum computer can offer significant improvements in computation time over classical computers.[8]

In 1994, Peter Shor discovered an algorithm for factoring integers in $O(log(N)^3)$ time, which is exponentially faster than $O(2^{\sqrt[3]{logN}})$,the time of the best classical algorithm. This was the first major discovery in the programming of quantum computers and attracted much attention to the field.

In 1996, Lou Grover discovered an algorithm that could search a randomized list for a particular entry in $O(\sqrt{N})$ time, which is much faster than the classical $O(N)$ for large lists. This was shown to be the fastest possible sorting algorithm for a quantum computer.

In 2001 IBM constructed a seven-qubit quantum computer and used Shor's algorithm to factor the number fifteen. Significantly larger quantum computers have not yet been built. Decoherence of quantum states is a difficult engineering problem that is occupying much attention in the field. Quantum error correction methods have been developed, but implementing them still requires a refined ability to isolate qubits from their physical environment. At present this technology is still being developed.

[1] Eleanor Rieffel, Wolfgang Polak, ACM Computing Surveys, Vol. 32, No 3., September 2000, 300-355.

[2] Samuel J. Braunstein, University of York.

[3] D. Deutsch, Proc. R. Soc. Lond. A **400**, 97-117 (1985).

[4] Adriano Barenco *et al.*, Phys. Rev. A, 52 (1995).

[5] Lou K. Grover, The Sciences, 24-30, 22 August 1999.

[6] Lou K. Grover, Phys. Rev. Lett. **80**, 4329 (1998).

[7] David Deutch and Richard Jozsa, Proc. R Soc. Lond. A **400**, 553-558, (1992).

[8] Brad Huntting, David Mertz, IBM.