Slide 1:

Quantum computation is the study of the information processing tasks which may be accomplished using quantum mechanical systems. At the heart of inquiries to ascertain the potential of quantum computation is the question of whether it is possible for a quantum computer to efficiently solve computational problems which have no efficient solution on a classical computer. Pioneering work done by David Deutsch, Peter Shor, Lov Grover and others in the 1980s and 1990s established a great deal of the essential theoretical framework and results which provided answers to this question which has justified continued research and interest in quantum computing ever since. Today we will provide a brief historical and functional overview of the development of quantum computing and its applications during this time, and contrast it with classical computing, and in particular, we will examine Grover's algorithm as an illustrative example of the potential benefits of quantum search algorithms.

Slide 2:

Computer technology today is based on the silicon microchip. First introduced in the 1960s, it is the font from which which countless electronic and technological innovations have sprung ever since. In 1965, Gordon Moore, co-founder of Intel, first articulated his now famous law, which predicted that the number of transistors on a chip would double about every 2 years, for at least the subsequent 10 years. In fact, that exponential growth has so far held effectively until the present. (Slide Fig. 1)

Slide 3:

There are, however, reasons acknowledged by Moore himself, why the law must eventually begin to fail. So far, improvements in microchip technology have paralleled improvements made in the fabrication methods, and it in this way that we have been able to produce transistors of ever decreasing size. Current commercially available transistors are as small as

5 nm. Maintaining this trend has become more and more difficult, owing to production costs, as well as to much more fundamental physical constraints. (Slide Fig. 2)

Slide 4:

As transistor sizes decrease, the more serious problem of quantum effects begin to manifest as the size of the transistors become comparable to the de Broglie wavelength of the electrons which carry the signals. At these scales, the classical electromagnetic principles that govern circuit design, such as Ohm's law, no longer hold. Although the development of semiconductor nanostructures and quantum transport theory has made effective and extremely fascinating contributions to prolonging the natural life of Moore's Law, inevitably it must fail when the size of transistors become comparable to the size of the individual atoms. To quote Moore himself, *"In terms of size of transistors you can see that we're approaching the size of atoms which is a fundamental barrier, but it'll be two or three generations before we get that far—but that's as far out as we've ever been able to see. We have another 10 to 20 years before we reach a fundamental limit. By then they'll be able to make bigger chips and have transistor budgets in the billions."* [Techworld Magazine, April 13, 2010] (Slide Fig. 3)

These fundamental limits on how much computational power that can be increased through improvements to existing technologies in turn impose limits on the scale of the computational problems that can be tackled, as they are ultimately limited by the computer processing power that is available.

Slide 5:

The complexity of problems, or tasks, are classified according to the asymptotic behavior of solutions. That is, in terms of lower bounds on the number of iterations of a process required to obtain a solution. If the number of computer operations increases as a polynomial power of the size n, then the problem is said to belong to the polynomial complexity class, or class P. If the number of operations increases faster than a

polynomial function, then the problem is said to belong to the non-polynomial complexity class, NP. For instance, problems in the P class are generally tractable for conventional computers; historically, when such problems have been too hard to solve, in time their solution became practical due to exponential increases in processing power. NP class, however, are a different matter. For such problems, the general issue is that a small increase in the size of the problem leads to a very large increase in the amount of computing power required to solve it. A classic example of an NP problem is the factorization of large numbers. For a given integer n, the number of operations required to find its prime factors increases exponentially as n increases. From a computational perspective, the required computational time exponentially increases by increasing n—hence the difficulty that computer scientists face when dealing with problems in the NP class. Such problems are often impractical, even with the use of the most powerful conventional computers. And it was against the background of such problems that the idea of quantum computation was conceived. [Fig. 4]

Slide 6:

Conventional computers which run according to classical principles. Mathematically speaking, according to the operations of universal Church–Turing machine. The birth of quantum computation came from the realization that it might be possible to construct other types of computers which operate based on different principles than those of Turing machines. Like many late 20th century developments in theoretical physics, the idea of constructing a computer explicitly based on the principles of quantum mechanics was first put forward by Richard Feynman in 1982. Then, as now, it becomes progressively harder to model quantum systems with a conventional computer due to the problem of exponential increases in processing power needed as the size of the system being modeled increased. What he proposed was that computer hardware be built to explicitly take advantage of quantum mechanical phenomena. In this way, the computer's computational power would directly scale with the complexity of the system under investigation. Then, in 1985, David Deutsch

wrote his famous paper in which he identified the basic principles of quantum computation. In analogy to Turing machine, he introduced the concept of a universal quantum computer, and he demonstrated that in principle it could solve problems that could not be efficiently solved with a classical computer. [Fig. 5]

Slide 7:

The underlying physics of conventional computers are electromagnetic and so of course have always been, implicitly, quantum mechanical. Yet the way in which data is encoded—as "bits", "on and off", and in a deterministic fashion—is about as classical as it gets. What is conceptually novel about quantum computation was Deutsch's idea to encode information itself as a quantum states—as a superposition of eigenstates—for which there is no classical analogue. In so doing, an exponential increase in computing power is obtained as the system gets larger. In 1985, in fact, it would be fair to describe the idea as revolutionary, and in the years following Deutsch's landmark paper, discourse on the subject was limited mainly to theoreticians working to understand the basic principles and prospective advantages to be gained from quantum computing over conventional computing. Some such efforts were directed toward finding specific examples that would explicitly demonstrate the ability of quantum computers to outperform classical ones. Others sought to design experiments to prove the principles and establish the plausibility of quantum computing as a material reality. [Fig. 6]

Slide 8:

A major breakthrough came in 1994 when Peter Shor devised the algorithm which now bears his name. In examining the application of quantum computing to the problem of determining the prime factors of large numbers, Shor demonstrated with his algorithm that it was in fact possible for a quantum computer to determine these factors in polynomial time rather than exponential time, reducing the factorization problem from the NP complexity class to the P complexity class! Since then, other instances have been identified of quantum computers possessing a clear

computational advantage over the best classical computers. In this spirit, we examine in some greater detail another prominent example, known as Grover's Algorithm. [Fig. 7]

Slide 9:

In 1996, Lov Grover devised the quantum algorithm which now bears his name in order answer to the question of how to efficiently search a database. It is this quantum search algorithm which is the particular subject of our discussion here today. As an exhaustive description of Grover's algorithm is rather complicated and beyond the scope of the present discussion, we aim to provide a somewhat simplified description which achieves the goal of conveying a good intuitive sense of its advantages. In essence, the algorithm describes a process for finding a single object of interest in an otherwise unorganized collection of objects.

Slide 10:

Grover's Algorithm is a way of performing an exhaustive search for an object with a given property—for simplicity, let us suppose it is a particular number. We know whether the number we find is the one we want or not, and we have our algorithm, F, that delivers an output +1 or -1, where -1 means that we have found our desired number, and +1 if we have not. We denote our desired value t, the unique value for which $F(t) = -1$. We consider a case where there are N possible values that have to be searched through for this t, and suppose for the sake of illustration that $N = 2^L$ for some integer L, so that we may encode each possible value uniquely in a register of L-qubits. A qubit is just the quantum computation analogue of a classical bit—in fact, in qubits, by convention the qubit +1 corresponds to 0 in binary, and the qubit -1 corresponds to 1 in binary. Hence, why $F(t) = -1$ for our desired t, and $F(x) = 1$ for all other N-1 values.

Slide 11:

By convention, we represent our algorithm, our function F, by a so-called "Oracle"—which is just an abstraction of some computer operating coherently on however many qubits that we can give it as input. So in our

case, the Oracle computes F on an L-qubit input, and gives us a 1-bit output—namely, whether the input is the value we are looking for or not. For inputs x and y, where x can take one of 2^L values, and y can take one of two values, ±1, it gives an output x, and y·F(x). So the Oracle as a whole operates on L+1 qubits. So whether classically or quantum computationally, the question is how many evaluations of F, how many iterations of the Oracle, must we perform to find the unique value t so that F(t) = -1? Clasically, we may need to invoke it as many as N-1 times. Using quantum computation, we will show that we can do a great deal better.

Slide 12:

Grover's algorithm consists of three subroutines. The first subroutine involves the use of what is called a Hadamard gate, the particulars of which we need not delve in to for our present purpose. We apply this Hadamard gate to each of L-qubits, and denote it H in our diagram. The algorithm begins with all L of the qubits in their blank initial state, where they all have value +1, which we denote with the zero-ket-- a state of L-qubits, all in state +1. The effect of H on the zero-ket is to produce the L factors in the tensor product, which we identify as μ. In the simplest terms, what the Hadamard transformation does is place the zero-ket into the equal superposition state. The Hadamard gate is its own inverse, and so $H|\mu\rangle = 0$. If we expand μ, we see that it consists of a superposition of states with all N 2^L possible sequences of plus and minus values, representing the 2^L possible values of x. Hence, if $|x\rangle$ is any basis state, then the usual orthonormality means the scalar product of x with μ is $\frac{1}{\sqrt{N}}$.

Slide 13:

The next subroutine involves the oracle with its L-qubits to hold the argument of F, and its one auxiliary qubit. We denote this "marking" subroutine, M. It begins with the auxiliary bit being put through a not-gate and then a Hadamard gate, in order to prepare it in the state shown. With that as the input for the auxiliary qubit, the effect of the Oracle on $|x\rangle$ is

that |x> is unchanged, with the overall effect being that of just multiplying the overall state by F(x). That means that the auxiliary qubit is left unchanged. M could run repeatedly just recycling the same auxiliary qubit, which never changes. So the first L-qubits evolve autonomously, undergoing coherent evolution during the process M, and the effect of M is that of a unitary matrix acting on the state of those L qubits alone. Overall, the effect of M on the L-qubits holding |x> is M|x> = F(x)|x>. For all but one value of x, F(x) = 1; only F(t) = -1, and this is why we refer to this as the marking operation. It marks the target term in the superposition by changing its sign. And we can simply write M|μ> compactly as shown.

Slide 14:

The third subroutine, B, just marks everything but the zero-ket. This means that for whatever basis inputs, the auxiliary qubit is flipped unless the first L inputs are all +1. The effect of B on the blank state 0 produces 0, and B on any other state |x> produces -|x>.

Slide 15:

Together these subroutines comprise Grover's algorithm—each of them are operations on L-qubits: The operation H transforms the zero-ket to the superposition μ and vice-versa; M marks the target state, and B marks all basis states except the zero-ket. Grover's algorithm then consists of starting with a blank initial state, perform H, which makes the state μ, then continues some number of iterations of the sequence MHBH. The net effect of such an iteration is called the Grover iteration, and is denoted by the unitary matrix HBHM. That is how the algorithm works—but what does it do?

Slide 16:

There's a beautiful geometric interpretation of what it's doing. We imagine a two dimensional plane containing both the target state |t> and the superposition |μ>. For large N, t and μ are very nearly orthogonal, but though not exactly-- their scalar product is $\frac{1}{\sqrt{N}}$. We define a state that is

orthogonal to t, denoted φ, and consider a family of states lying in this plane. Any such state may be parameterized according to $\cos(\theta)|\varphi\rangle + \sin(\theta)|t\rangle$, for some $\theta \in \mathbb{R}$. The target state t is, of course, in this family for $\theta = \frac{\pi}{2}$. Prior to our very first Grover iteration, we are in state |μ>, which his also in this form, but with $\theta = \arcsin(\frac{1}{\sqrt{N}})$.

Slide 17:

For a general state of this form, what is the effect of a Grover iteration? The Grover iteration begins with M, which changes the sine of the coefficient of the target ket |t> , and leaves all other basis states unchanged. So the effect of M on a general state is merely a reflection about the horizontal axis.

Slide 18:

After M, we apply HBH. Looking directly at what HBH's effect is, it has no effect on the state |μ>, but it changes the sign of any state perpendicular to |μ>. On an arbitrary state, the effect then is to reflect it in the line |μ>.

Slide 19:

Cumulatively, these two reflections are equivalent to a single rotation. In fact, a rotation through an angle $2\cdot \arcsin(\frac{1}{\sqrt{N}})$, towards t, and each successive iteration rotates the state a little closer to t. Too many iterations, we pass t. Hence, we must choose the right number of iterations, which is clearly $\frac{\frac{\pi}{2}}{2\cdot\arcsin(\frac{1}{\sqrt{N}})}$. Since the number of iterations is of course an integer value, the appropriate number of iterations which brings us closest is just $\text{Floor}(\frac{\frac{\pi}{2}}{2\cdot\arcsin(\frac{1}{\sqrt{N}})})$. Which, for large N, is proportional to $\sqrt{N}$. So Grover's algorithm completes its search using about $\sqrt{N}$ iterations, compared to the order N iterations for a classical search.

Slide 20:

In fact, it has been proven that the algorithm is optimal. No algorithm, quantum or classical, and perform an exhaustive algorithmic search faster than Grover's. Though the proof of this applies to an idealization, as is usually the case in such proofs, it remains of course an essential fundamental result demonstrating the potential benefits quantum computing can offer in solutions to exhaustive searches. In fact, for cases where there are M "acceptable" objects in the collection being searched, Grover's algorithm completes its search in order $\sqrt{N}/\sqrt{M}$ iterations, compared to the classic N/M order. In concluding, it is worth pointing out that, in general, the algorithm does not produce t exactly after the appropriate number of iterations. However in practice, a winnowing process is undertaken in tandem with classical computers—the quantum algorithms probabilistically reduce the collection of objects to be searched to a size which can then be deterministically searched by a classical algorithm. This basic cooperation is the power of quantum search algorithms—to "do the heavy lifting", and reduce problems utterly intractable for a classical computer to one which they can handle. Quantum computing is still very much in its infancy, and it will be exciting to see the potential it develops over the coming years.