

Implementation Aspects of Quantum Computing
Prof. Debabrata Goswami
Department of Chemistry
Indian Institute of Technology, Kanpur

Lecture - 04
Computational Tools

(Refer Slide Time: 00:17)

What Is Computing?

Most people have a very concrete and very limited idea of what computing is—work done by some metal-encased electrical machine with a disk and mouse attached!

Fact: notion of computing is a generalized mathematical concept...

- In 1936, even before computers existed, Alan Turing proved that a device with a limitless memory, and a scanner to scan the memory backward and forward, read it symbol by symbol, and write additional symbols, can execute any computation—and that holds true even today.

Present-day classical computers are equivalent to universal Turing machines!

The slide includes a diagram of a modern desktop computer (monitor, tower, keyboard, mouse) and a diagram of a Turing machine (a vertical tape with a head and a scanner).

First we have all ready looked at this, the idea of computing and in that we are used to these kinds of computers, where the basic idea remains the same which was concede way back in 1936, before the computers actually existed, showing that a computer essentially is a device which uses limitless memory and a scanner to scan the memory backward and forward and read it by symbol to symbol and write additional symbols to execute any computation that is the Turing machine. The whole idea of this principle still works and it is the same one which is given for this computer. That is the basic idea behind the concept of any classical computer.

(Refer Slide Time: 01:13)

Classical Computing Limitations

- Even "parallel" computers are really complex Turing engines employing multiple computing modules which deal with pieces of incoming data (chunks, bytes, instructions, etc.).

But...

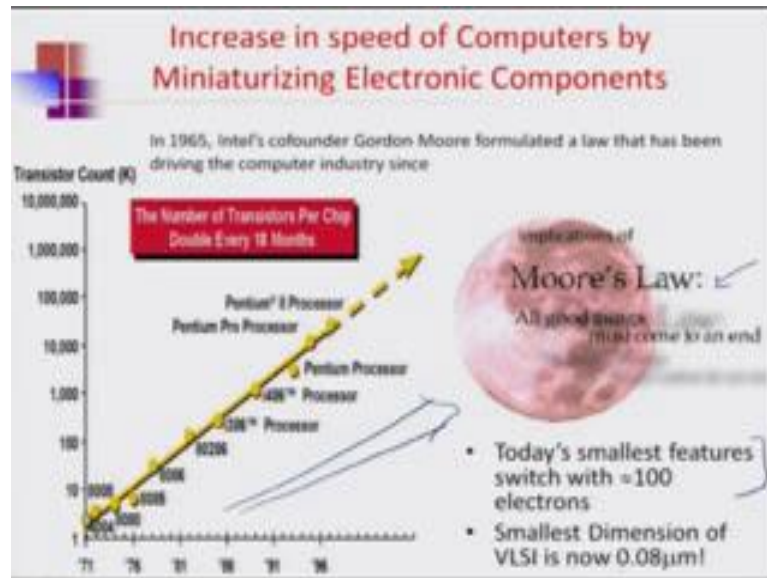
- Many problems beyond the competence of a universal Turing machine are known.
 - For example it cannot predict whether a program will terminate. Program compilers cannot anticipate all possibility of program run crashes—as we know!
 - Many, many classes of problems have been delineated that are solvable, for example, in polynomial or exponential time and those whose answers are checkable in polynomial time.

N-polynomial
NP

The classical computer has some limitations, even the fact you can have limitless parallel computers they, which can do really complex work, but they are basically complex Turing engines which employ multiple computing modules dealing with pieces of incoming data which are chunks of bytes instructions etcetera, but there are problems which are beyond the competence of a universal Turing machine. For example, we cannot predict whether a program will terminate and this is the halting problem, famous one and then there is classes of problems which have been delineated and are solvable for example, in polynomial or exponential time and those whose answers are checkable in polynomial time.

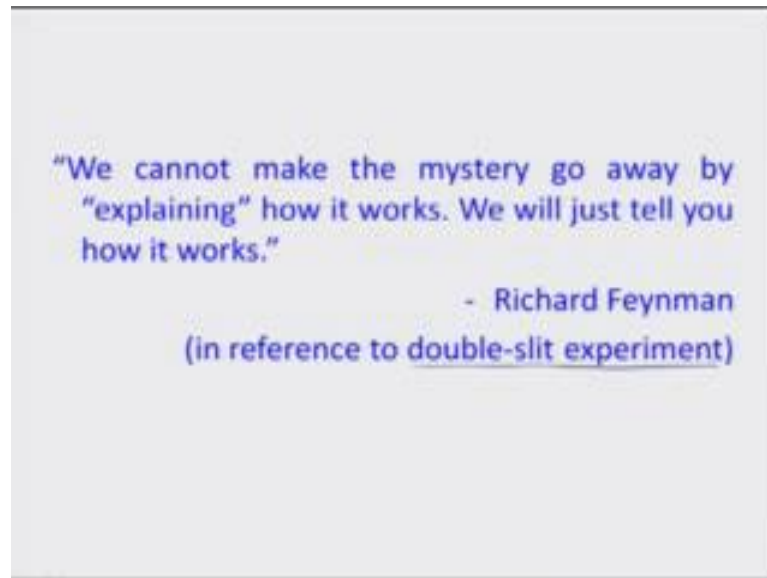
These are the classes which will be looking at. These are in, if anything can be solved in N polynomials then it is a problem which is to be looked at in this fashion, but as these, this is an $N P$ problem, but as these become larger and larger, it becomes more difficult.

(Refer Slide Time: 02:32)



We will be looking into these things more and more and we have also discussed about Moore's law before which was given by Gordon Moore who is a founder of Intel cofounder of Intel who made an implicit statement based on the fact that he saw that the number of Transistors per chips, essentially doubled every 18 months and this tendency of the computer world has kept on going since the beginning of the computers for a very long time. And that is kind of very interesting because now we are talking about switching which are at very small scales and the current VLSI, the very small very large scale integrated circuits that is VLSI can be as small as a few tenths of a micron and even smaller actually they have reached nanometer scales.




(Refer Slide Time: 03:30)



The principles of quantum mechanics are going to come and in the spirit of Feynman, the mystery associated with quantum mechanics is something which you perhaps cannot really explain, but what you can do is you can just look at it how it works because you cannot really, in his words we cannot make the mystery go away by explaining it. That is how you look at it and this was done, this is his famous statement with respect to the double slit experiment, where interference fringes and things were found for the first time.

(Refer Slide Time: 04:12)

The Beginning

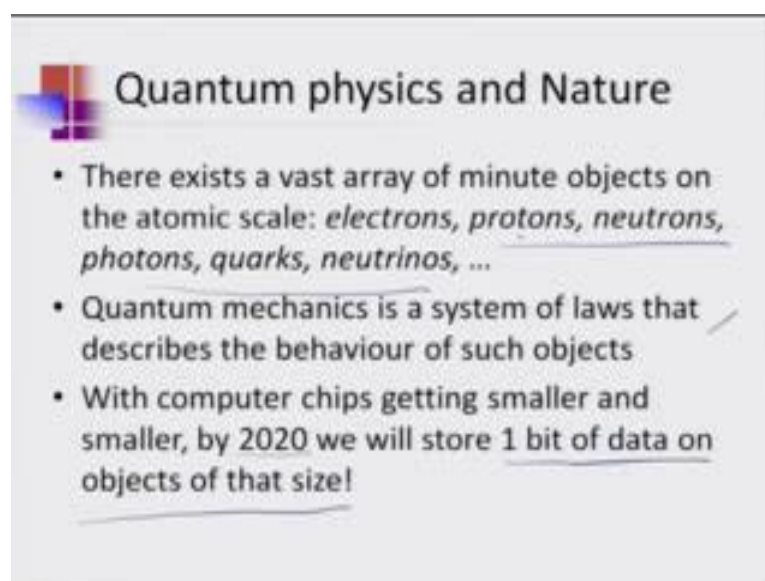
- Richard P. Feynman (7th May, 1981) 
• Classical Turing Machine would probably experience an exponential slowdown
• A quantum system of many particles is described by a Hilbert Space whose dimension is exponentially large in the number of particles
- David Deutsch (13th July, 1984) 
• Proposed Universal Quantum Computer
• Qubits required ~ No. of particles in system
- Seth Lloyd (23rd August, 1996) 
• A standard quantum computer can be programmed to simulate any local quantum system efficiently
• Has been extended to larger classes of quantum systems

CLEO 1981

In terms of quantum mechanics, the beginning or basically the use of quantum mechanics in computers started in 1981, where this statement was made that the classical Turing machine would probably experience an exponential slowdown because of the probability issues as you become smaller. And the immediate other side of the story was there the quantum system with many particles is described by Hilbert Space whose dimensions are exponentially large in the number of particles which means that it can be utilized for its advantages and this statement that this has its benefits, I do not know what happened sorry; this has its benefits where described in a conference called Cleo in this 1981. It is in 1981; Cleo conference Feynman first coined the term which is quantum computer.

Then David Deutsch in 1984, proposed the idea of universal quantum computer, it was quite soon that this idea was brought out and the Qubits required were the number of particles in the system that is how it was defined and then it when through a lot of developments and about a decade later, satellite talked about a standard quantum computer which can be programmed to stimulate local quantum systems efficiently. This was one of the important statements that he was able to make and he was able to show he was a able to take a stock of all the developments over the more than a decade by that time and showed that it quantum computers has been extended to larger classes of quantum systems.

(Refer Slide Time: 06:22)

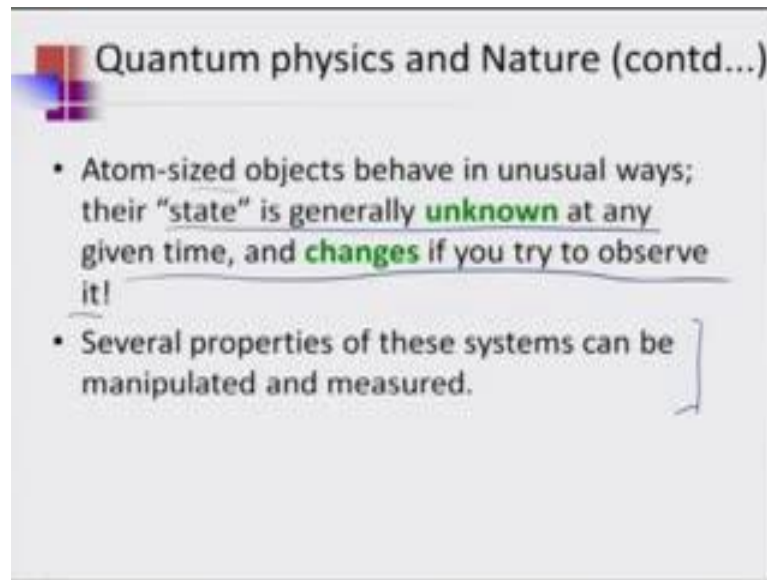


Quantum physics and Nature

- There exists a vast array of minute objects on the atomic scale: *electrons, protons, neutrons, photons, quarks, neutrinos, ...*
- Quantum mechanics is a system of laws that describes the behaviour of such objects
- With computer chips getting smaller and smaller, by 2020 we will store 1 bit of data on objects of that size!

Well quantum mechanics essentially are systems which involve electrons, protons, neutrons, photon, quarks, neutrinos, any of them can be our potential Qubit in some sense and it is a system of laws that describes the behavior of such objects.

(Refer Slide Time: 06:59)



This was predicted way back then that by 2020, we will store one bit of data on objects of that size. Not sure we are there yet, but we are not too far from 2020 either. Let see what happens.

The one of the interesting concepts of quantum mechanics is that the atoms size objects behave in unusual ways because their state is generally unknown at any given point of time and changes if you try to observe it. Now this is a very important aspect of quantum mechanics which is different from classical works. In a classical object whatever you have, the solution is there. In a quantum object unless you measure it, you do not know what it is. Secondly, if you measure it you cannot be sure that it is the same thing that you measured because the act of measurement changes it. There is also this principle of standardization which exists in this quantum world.

And simultaneously several properties of the systems of such systems can be manipulated and measured simultaneously that is one of the other advantages of this.

(Refer Slide Time: 08:03)

Why Go Quantum?

- As we go smaller to avoid waste in time, energy etc. We inevitably go quantum.
- To what extent can a computer act reversibly, with minimal energy loss? A Turing machine is not reversible. But quantum evolution is reversible.
- A classical bit is either 1 or 0. A two-state quantum system can be in an arbitrary superposition. All such states can be processed at once in quantum operations. *Just as a quantum system cannot be explained by hidden variables, so quantum processing cannot be reduced to a Turing machine.*

Diagram illustrating superposition states. On the left, two classical states are shown: 'spin up' (a black triangle pointing up) and 'spin down' (a black triangle pointing down). On the right, a 'Continuous tipping' diagram shows a black triangle rotating around a vertical axis, representing a superposition state. The text 'Superposition states' is written below the diagram.

Anyway these are the motivations as to how you would be going and why you would be going quantum and we have done this in class, I am just going to re iterate here, it to bring you up to speed – one is that as we go smaller to avoid wastage in time energy etcetera because as we go smaller the time taken to travel goes lesser, wastage of time is lesser. Energy required in dissipation goes down and eventually go to quantum. That is the way of looking at it. The other option is that in order to be energetically most favorable you want to act reversibly that comes from thermo dynamic principles; that gives minimum energy loss and the biggest important point is Turing machine is not reversible, but the quantum evolution is reversible.

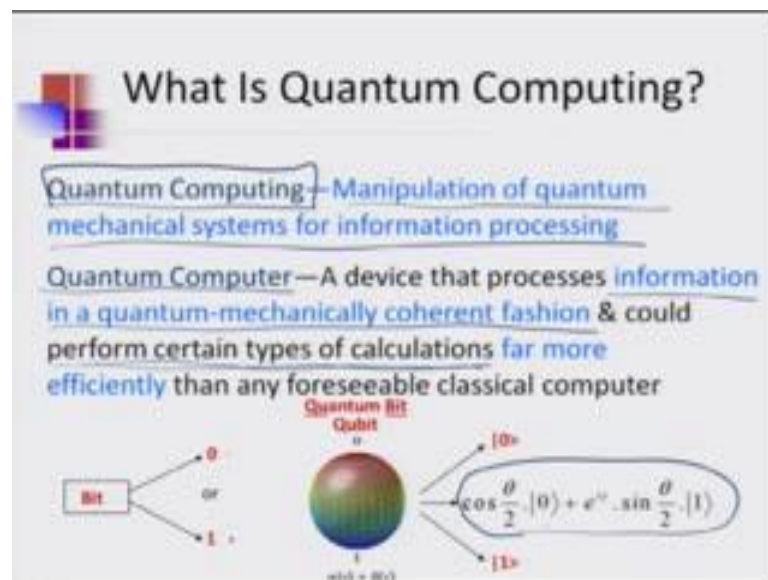
A classical bit again, the other important thing which is different is a classical bit can either be 1 or 0; however, a 2 state quantum system can be in an arbitrary number of super positions. All such states can be processed at once in quantum operations. For example, this continuously the system getting tipped is an example of a superposition principle. While classical essentially means either one or the other in a spin system, in quantum systems it is a continuous tipping kind of a situation, where in the middle wherever whatever you are you can see them. So, those are our superposition states and we can use them.

There is this concept of hidden variables in physics which have been used quite often to even talk about these a superposition and in between conditions, but we need not worry

about this right now. It was assumed that quantum systems essentially meant that they were lots of variables which you cannot see and they were called hidden variables is one of the concepts of quantum physics or foundations of quantum mechanics, but it was found that just by thinking that there are hidden variables you can explain quantum mechanics that did not work. Similarly a quantum process also therefore, will not perhaps be reduced to a Turing machine.

But on the other hand, we will be discussing something called a quantum Turing machine which will be an analog to the world of computers that we know it.

(Refer Slide Time: 10:49)



The definition of quantum computing which will use is manipulation of quantum mechanical systems for information processing and there are 2 terms here. Let us be clear, one is quantum computing and the other one is quantum computer. Please remember, they are not the same thing. It is like noun and adjective. You do not want to use a noun in place of noun and objective and vice versa. A quantum computer is the noun and the computing is actually the adjective. The act of manipulation where do have actually, act of manipulation is the quantum mechanical system for information processing, that is quantum computing whereas, quantum computer is a device that processes information in a quantum mechanically coherent fashion and it could perform certain types of calculations. Now it is important to mention this certain types of calculations not that we have going to at least at this point of time say that for every class

of calculation it will be better, but for certain parts it can be far more efficient and this is again the example of how they look different, in bits you can either be here or there whereas, in quantum bit or Qubit you could be anywhere in between and all possibilities of these between exist, which is sort of not an issue for the bit case.

(Refer Slide Time: 12:16)

What can a QC do?

- **Simulate quantum systems** (Feynman 1982)
 - quantum parallelism: superposition of k -qubits means parallel evolution of 2^k states, i.e. exponential speedup over state-by-state classical simulations
- **Factor integers in their primes** (Shor 1994) *Cryptography*
 - exponential speedup
- **Search databases** (Grover 1997)
 - quadratic speedup
- Finding quantum algorithms is a research field in itself

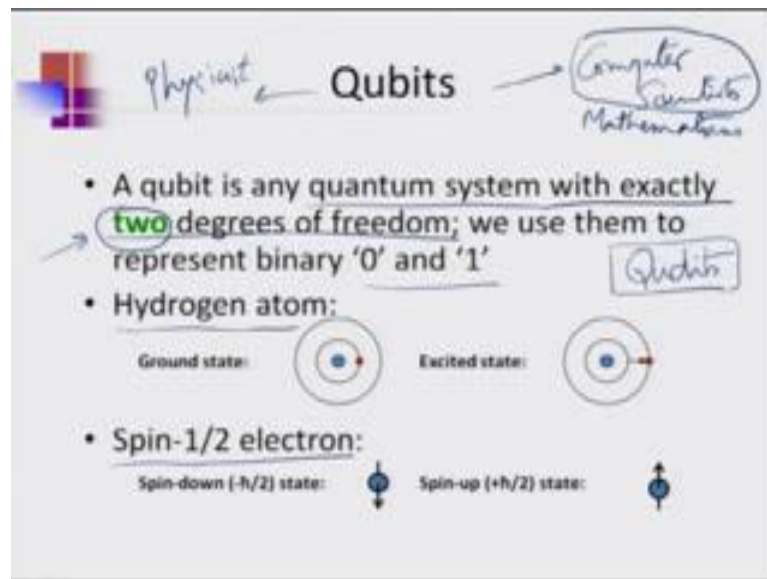
Interestingly way back in 1982, Feynman essentially looked at quantum computing from a different perspective. He was not thinking in terms of a computer that can be faster or something. He basically looked at it saying that since all natural processes are at the base or at the fundamental level, quantum mechanical, you can by using quantum computing realistically stimulate quantum systems. That was his basic principle and his basic interest being a physicist of his kind. That was his main idea that you could stimulate quantum systems in the most appropriate fashion once a quantum computer is here. That is his way of looking at it. The first realistic application came as late as 1994 by Shor, Peter Shor showed that it is possible to actually find the primes or basically factorize integers into their primes in a much faster way than it is possible by any foreseeable classical computer because there is the place where you can see an exponential speedup by using a quantum computers. We will do this particular problem.

And the other important place right after this which became very popular is the Grover's algorithm, which basically searches database. Now you might think that you know this speedup of this is not as much as a Shor's algorithm, but it is usually popular because

most of the problems that you have in a quantum computer finally, boils down to a such problem because in most cases what happens is you come down to a point when you would say the answer exists. Once you say an answer exists its essentially a search problem of finding the answer and the best part is when you can say that there is only a single answer and if you say that then this is perhaps the most important problem the most important algorithm that you need to use and therefore, Grover's algorithm although maybe not the most effective way of quantum computing, but it is also one of the most popular approaches of quantum computing.

However the, I should mention that these 1994 work of Shor is a by far the most complete application of quantum computing and also is one of the most important basis of cryptography. Some of the applications today that is being already done in the defense labs is some of the countries around the world is due to the fact that this factorization to give raise to this extreme security is possible because of this particular principle of Shor's algorithm which gives raise to exponential speedup.

(Refer Slide Time: 15:37)



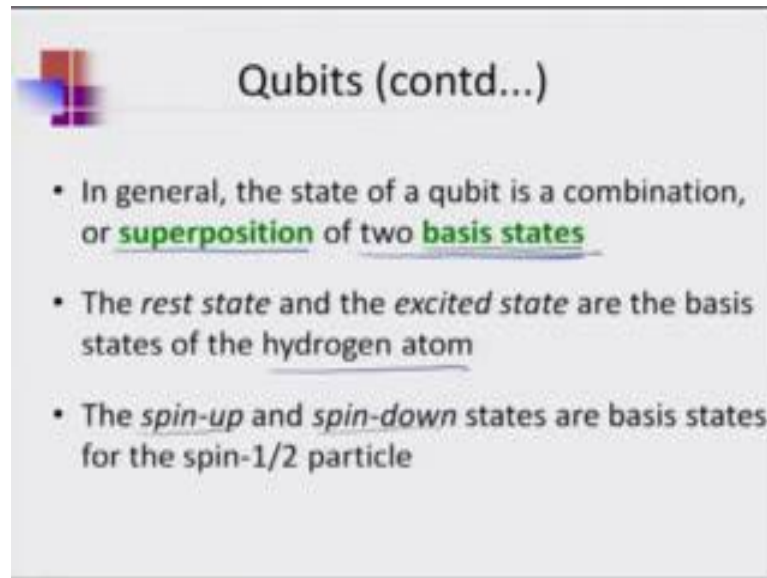
Now the basic idea of qubit is that the case where the quantum system with exactly 2 degrees of freedom. Now let me also point out that I am giving you the picture which is an evolved picture. In the 1980s people were not really concerned about just the 2 degrees of freedom. When the first idea came and the ideas of superposition and entanglement became clear, people wanted to use many degrees of freedom, but soon it

turned out that in order to maintain the language, which can be continuous between the computer scientist as well as the rest of the community developing this. This has a lot of people in this area. Your QC essentially is represented in one side by computer scientist, mathematicians and on the other side by mostly physicist and some very odd people who are interested in physics. For example, Setloit, he is a mechanical engineer.

All of these different people, who have an inclination of physics understanding or interest in physics have also contributed hugely in this, but in order to make sure that the community understands each other quite well, more to say that the computer scientist are able to finally, able to use and make it or compare it with the classical computing. It was decided that the 2 degrees of freedom should be the benchmark because classical computers essentially still realize this based on that binary principle. We will be using the same principle to go between 0 and 1. I mention this right away here because that is a large and we will perhaps look at it in some point of the other. There is a large number of body of work which is also parallely developed into having multiple states or multiple degrees utilized for quantum computing very often those are known as qudits.

Anything more than 2, when it goes is called qudit. That is a typical area of development which is also happened and people have shown that you could use qudits to do computing without any problem. It is like saying if you have is like having a decimal system verses a binary system and so on and so forth. You can have more than just to, just to clarify. Now once you say 2 degrees of freedom then there are some very common ones to look at 1 is say for example, the hydrogen atom ground in the excised state, these are the quantum states spin half system and that can be in electron, positron, a nuclei, any of these would work.

(Refer Slide Time: 19:03)



Qubits (contd...)

- In general, the state of a qubit is a combination, or superposition of two basis states
- The *rest state* and the *excited state* are the basis states of the hydrogen atom
- The *spin-up* and *spin-down* states are basis states for the spin-1/2 particle

That is the situations. The state of the qubit is generally as we mentioned superposition of 2 basis states.

Here the starting point is our basis states. That is coming from quantum mechanics. We have been looking at the basics of quantum mechanics before. We have all ready know that everything starts off with basis states and here we are basically confining ourselves to the 2 basis states that we start off with. The rest state and the excited state are the basis states of hydrogen atom for example, the rest or initially ground it can be anything. For example, again in the spin case the spin up and spin down are the states that a basis states of the spin half electron.

(Refer Slide Time: 19:50)

The State Vector

- The state of a quantum system is described by a **state vector**, written $|\psi\rangle$
- If the basis states for a qubit are written $|0\rangle$ and $|1\rangle$, then the state vector for the qubit is $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$

where α and β are complex numbers with $\alpha^2 + \beta^2 = 1$

Relative

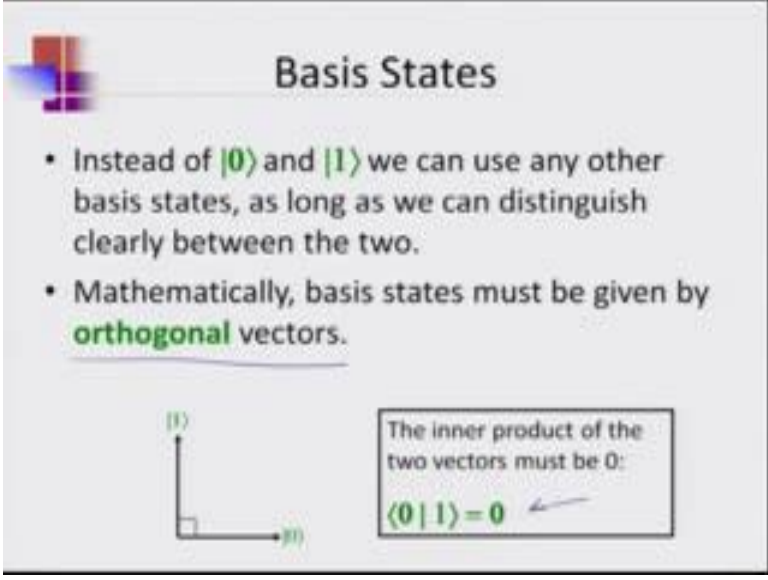
Now, based on these, you can have the vector picture, which we discussed, which is essentially the idea of Dirac and he used this bracket notation a essentially representing the arrow going one way verses the other and very effectively using this pictorial principles to develop the entire shorthand math associated with this entire process. The reason for the rap to develop this was, he was the first want to do the entire relativistic quantum mechanics and with the help of using relativistic quantum mechanics, he was able to for the first time, show the spin quantum number to appear and not adhoc, put in as was done in the non-relativistic scheme.

Those of you who have done relativistic quantum mechanics, you already know what I am talking about. For relativistic quantum mechanics this was critical to be able to develop the ideas of shorthand notation. It was important to develop the shorthand notations to take care of the entire math that was going to go ahead with it. Simply put it is a much easier way of looking at the entire problem. There are certain notions that you go by doing this one is to say that the basis states for each qubit can be written as 0 and 1 for instance and then each of them is associated with some probability or some amplitude factor and they can be all complex. For example, alpha and beta as long as they the square of them add up to 1, you are allowed to take any complex number for them.

That is how you bring in quantum mechanics. At the very beginning whoever is entering this field, for them it is important to notice the main difference between the classical

systems versus a quantum system. Even here in terms of writing out the linear way of writing everything out is the fact that we invoke complex numbers for the amplitudes and these numbers by themselves really have no physical understanding, no physical meaning. Only when you square them then they start having some meaning.

(Refer Slide Time: 22:26)



Basis States

- Instead of $|0\rangle$ and $|1\rangle$ we can use any other basis states, as long as we can distinguish clearly between the two.
- Mathematically, basis states must be given by **orthogonal** vectors.

The inner product of the two vectors must be 0:
 $\langle 0 | 1 \rangle = 0$

The slide features a diagram of two orthogonal vectors, $|0\rangle$ and $|1\rangle$, represented as a vertical and a horizontal line meeting at a right angle. A small square symbol at the vertex indicates orthogonality. To the right, a text box contains the statement 'The inner product of the two vectors must be 0:' followed by the equation $\langle 0 | 1 \rangle = 0$ with a blue arrow pointing to the right-hand side of the equation.

You get the feeling as to how probability is associated with the square and not just the amplitudes. We can use any kind of a basis sets. Instead of 0 and 1 we can use any other one as long as we can distinguish them. The basis states are typically are chosen in terms of orthogonal vectors. The reason being the inner product of the 2 vectors then are determined or are confirmed to be always 0, when you take orthogonal. This is always going to work.

(Refer Slide Time: 23:08)

Basis states (contd...)

Operation

- For example, we could use the basis $\{|+\rangle, |-\rangle\}$ to describe the state of a qubit:

Now: $|\psi\rangle = \gamma |+\rangle + \delta |-\rangle$

orthogonality: $\langle + | - \rangle = 0$

You setup the problem this way. You could also use a plus minus notations for this and you can go from 1 basis set to the other. This is basis set transforms again something which you have done, quantum mechanically many times for simple cases, it is just a rotation. For instance here, we are showing an example of a 45 degree rotation from the original orthogonal states of 1 and 0 to plus and minus which are off by 45 degrees and you can just easily rotate 1 to get to the other. This is the qubit rotation in some sense. This itself, whenever you look at in an answer or situation like this, you are going to visualize what it means. Here is our first example of an operation. Why is it important? Because these are the operations that form the basis of how you are going to develop your computer? Just this very simple idea that you can actually rotate a basis set gives you the first principle idea of an operation in a quantum sense.

(Refer Slide Time: 24:30)

Systems of many qubits (contd..)

- The state of a composite quantum system, when all the component states are known, is their **tensor product**:

$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes |\psi_3\rangle$

- This is the "outer product" of vectors
- Note that this is **different** from the inner product $\langle\phi|\chi\rangle$

Handwritten notes on the slide include: $|\psi_1\rangle \otimes |\psi_2\rangle \otimes |\psi_3\rangle$ with arrows pointing to the equation above; $\langle\phi|\chi\rangle$ with an arrow pointing to the text below; and $|\phi\rangle \otimes |\chi\rangle$ with an arrow pointing to the text below.

Now, when you have more than 1 qubit, then their internally arrangement gives raise to many more conditions. For instance, if we know the individuals states of electrons of a system given below. These are 3 electrons, let say they are individual representation on the basis of how they are, would give raise to the 3 different possibilities of representing them. I could always write my different way functions, psi 1, psi 2, psi 3 with amplitude factors in such a way that will give me raise to 3 different way functions. They can be as simple as 0 and 1 amplitudes which give raise to either one of the other state as it is being represented here or it could be much more complex. If we write it like this very simply and we ask the question what is the overall state of the 3 particle system then will be able to look at their composite products as their solution.

And these composite products are tensor products, they are not anything else, but tensor products. This is again a definition here. The inner product was the term where we used the basis states to tell that they were orthogonal. 0 and 1, when they formed orthogonal set their inner product was 0, but when I actually taken outer product which is representing the final state of all these different qubits that I have taken, that is my outer product and this is my final state psi is essentially an outer product of all these 3 different states that I have which is a tensor product and that gives raise to the state and this is very different from the inner product.

For any 2 states, if I take an inner product, that is not going to be the same as the outer product. If I want to write, if I take the phi and psi for instance, if I want to write this is my inner product as I have mentioned here, then the outer product will essentially be phi and psi. This is my representation which corresponds to that is what it means. The tensor products are written in this way I could also as well represent this psi 1, psi 2, psi 3, this is the shorthand notation which goes. Outer product, inner product and this is how they go.

(Refer Slide Time: 27:45)

Systems of many qubits (contd...)

- We have

$$\begin{aligned}
 |\psi\rangle &= |\psi_1\rangle \otimes |\psi_2\rangle \otimes |\psi_3\rangle \\
 &= (0|0\rangle + 1|1\rangle) \otimes (1|0\rangle + 0|1\rangle) \otimes (0|0\rangle + 1|1\rangle) \\
 &= |1\rangle \otimes |0\rangle \otimes |1\rangle
 \end{aligned}$$

By convention, we write $|1\rangle \otimes |0\rangle \otimes |1\rangle$ as $|101\rangle$

Now, in this particular case where this was just set it up like this, we will finally find that my state psi, is going to be just represented by a state which is 1 0 1, which is a tensor product of the states corresponding to each of them. Now that is how you represent states next important thing which we would like to understand is the concept of gates.

(Refer Slide Time: 28:14)

Quantum gates

- As in classical computing, **a gate is an operation on a unit of data**, here: a qubit
- A **quantum gate** is represented by a **matrix** that may be applied to a state vector
- We will talk about this in more detail; for now we will look at some examples of commonly used quantum gates:
 - the Hadamard gate (**H**)
 - the Pauli gates (**I, $\sigma_x, \sigma_y, \sigma_z$**)
 - the Controlled Not (**CNot**)

Handwritten annotations on the slide include a circled 'H' and a box labeled 'Hadamard' with an arrow pointing to the Hadamard gate text.

The first thing we discussed was qubits, first thing is quantum mechanics and classical mechanics concept to computer, the very basics of it. These are how we built. Then we looked at what is the qubit, we defined qubits and we specifically mentioned the way we are going then we came to a point where we looked at the way the qubits are defined in terms of when you have multiple of them and then it is also looked at to say whether we are going to take in a products on outer products what does they mean? What do they mean? Inner products typically are projections, outer products are basically the final state, when you have multiple of them and that is what we looked at.

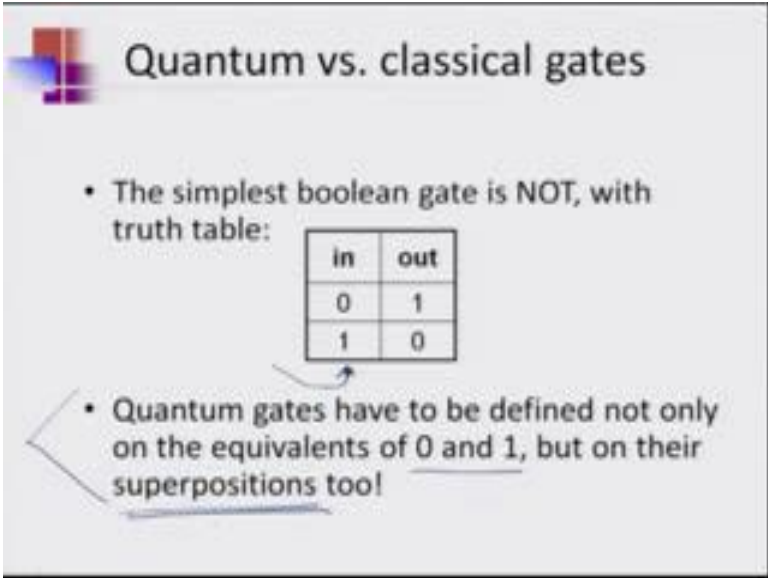
Now, when we are going to see, what can you do with them, that is when we are going to do the quantum gates. Now it can be parallel to the idea of a classical gate. In terms of the classical computing a gate is an operation on a unit of data where in our particular case, it is a qubit in this particular case. A quantum gate is represented by a matrix that may be applied to a state vector. Now you can already see why it is a matrix because you are now looking at states which were being represented as vectors which are also essentially single order matrices. If you want to operate on them and change to another one, you need a matrix most of in this square matrices.

Quantum gates are essentially often square matrices which can give raise to the vectors changing into the states that you would like. Now we will talk about this is more detail later, but for now let us look at some of the examples of commonly used gates. Now this

is important case. Let us start with some of the gates here. One of them is called the Hadamard gate and they are represented by different notations unfortunately the Hadamard gate representation is H which is very close to a Hamiltonian. Therefore, please note for this class we will be using the Hamiltonian in the cursory manner.

This will represent Hamiltonian because we are used to using Hamiltonian for generating energy whereas, for the Hadamard gate, we will be using the H , regular H , capital H , then there are these very important Pauli gates. What are these Pauli gates? I will come to them. The next one is the Pauli gate which is also very important and the first gate which is of consequence to an actual computing is the controlled NOT gate. We will go by them quickly to understand how these goes.

(Refer Slide Time: 31:39)



Quantum vs. classical gates

- The simplest boolean gate is NOT, with truth table:

in	out
0	1
1	0

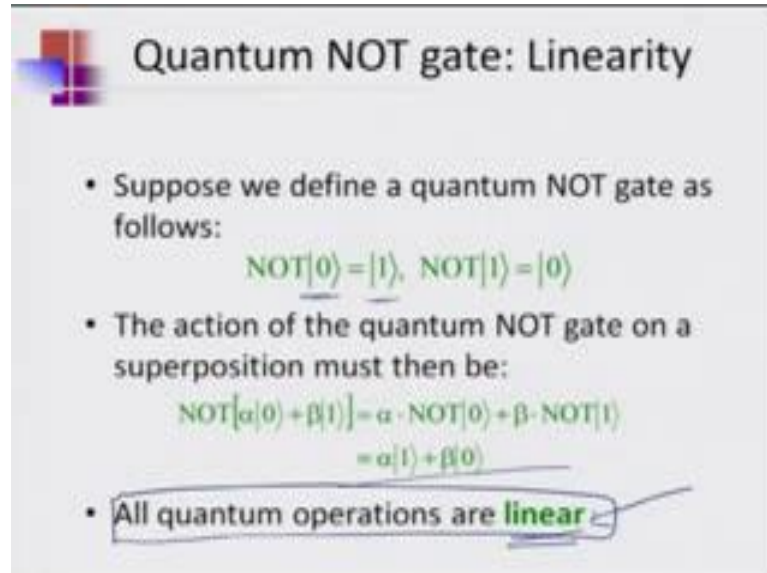
- Quantum gates have to be defined not only on the equivalents of 0 and 1, but on their superpositions too!

As an example for the classical case, this is a parallel that you are building the simplest Boolean gate is not. For the simplest case of a single qubit, what is it? We put in 1 case and we get out the opposite of them as a NOT gate whatever you put in the opposite comes out. If you go up, it will come out as down and so on and so forth. 0 1 goes in and you get a 1 0. Now one of the very important things to remember is that in quantum gates; however, you have defined it only, not only on the basis of the equivalents of 0 and 1, but also on the basis of their superposition.

Now, this is one very important point that you have to remember. Just by looking at the final outcome, which is the classical case you could have done or made your gates, but in

quantum mechanics, that is not just the case. Every other possible superposition also coming from the same condition should also be following this rule. That is the idea.

(Refer Slide Time: 32:58)



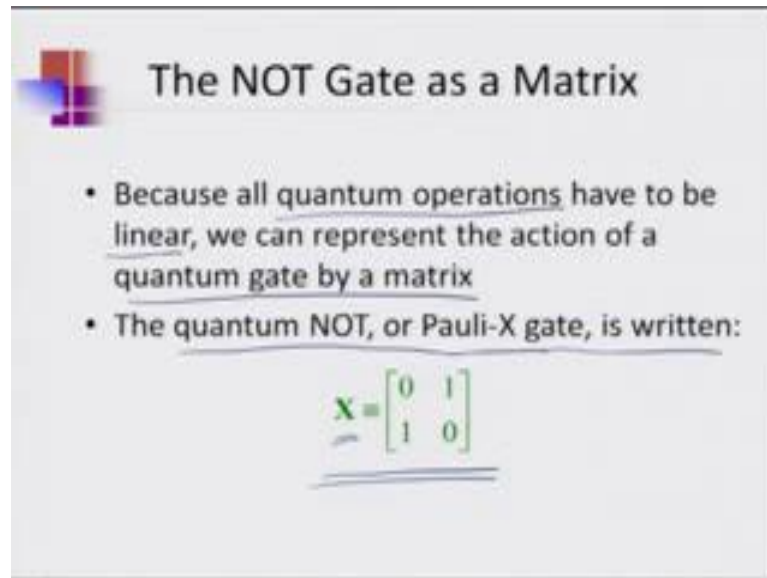
Quantum NOT gate: Linearity

- Suppose we define a quantum NOT gate as follows:
$$\text{NOT}|0\rangle = |1\rangle, \text{NOT}|1\rangle = |0\rangle$$
- The action of the quantum NOT gate on a superposition must then be:
$$\begin{aligned} \text{NOT}[\alpha|0\rangle + \beta|1\rangle] &= \alpha \cdot \text{NOT}|0\rangle + \beta \cdot \text{NOT}|1\rangle \\ &= \alpha|1\rangle + \beta|0\rangle \end{aligned}$$
- All quantum operations are **linear**

Here is an example, you have a state 0 and you have state 1. If you want to define a NOT gate, what you want to do is you want to have say not operating on 0, it should give 1 and not operating on 1, should give 0. That is what you expect.

The action of quantum NOT gate on a superposition must then also have a situation where if I do this then it should work, which means that this very important point is going to be always maintained in case of quantum mechanics, there all the quantum operations are going to be linear. This linearity, now it should not come to you as a surprise because you know that when we did quantum mechanics or when we have been learning quantum mechanics, we have always mentioned the quantum mechanics is always going to work in a linear way. All the operations in quantum mechanics are linear. The allowed operations or the ones which you actually call are measurable operations always linear. Whenever you use a Hamiltonian, the Hamiltonian is linear. Whenever you use a momentum operator, the momentum operator is linear. The position operator is a linear. That is the point, that if it works for any particular set then all the superposition states also should be following that which means that a requirement of quantum operations are that they are going to be linear.

(Refer Slide Time: 34:29)



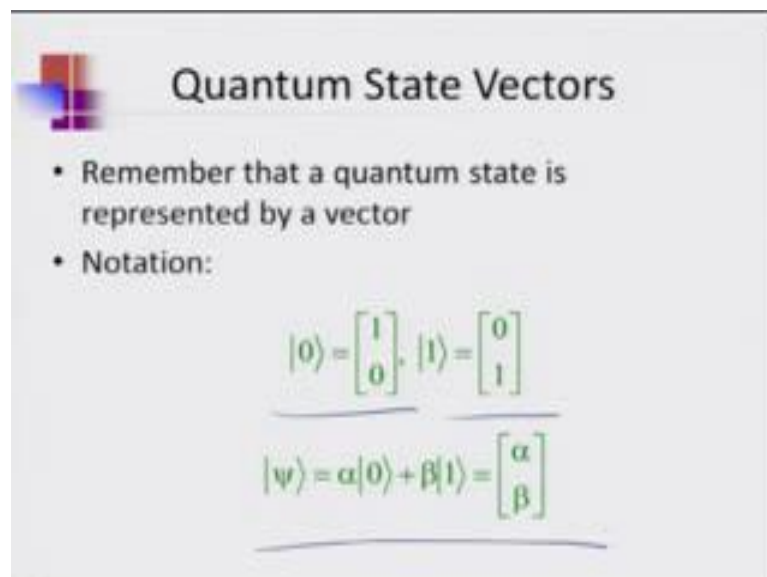
The NOT Gate as a Matrix

- Because all quantum operations have to be linear, we can represent the action of a quantum gate by a matrix
- The quantum NOT, or Pauli-X gate, is written:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Now, how do you get that? This is the logic behind why do you have a matrix for a quantum gate. The logic being that the quantum operations have to be linear. We can represent the action of a quantum gate by a matrix. The quantum NOT gate or it is also known as the Pauli X gate is written by this form, the X is equal to a matrix is 0 1 0 1 0 this 1.

(Refer Slide Time: 35:04)



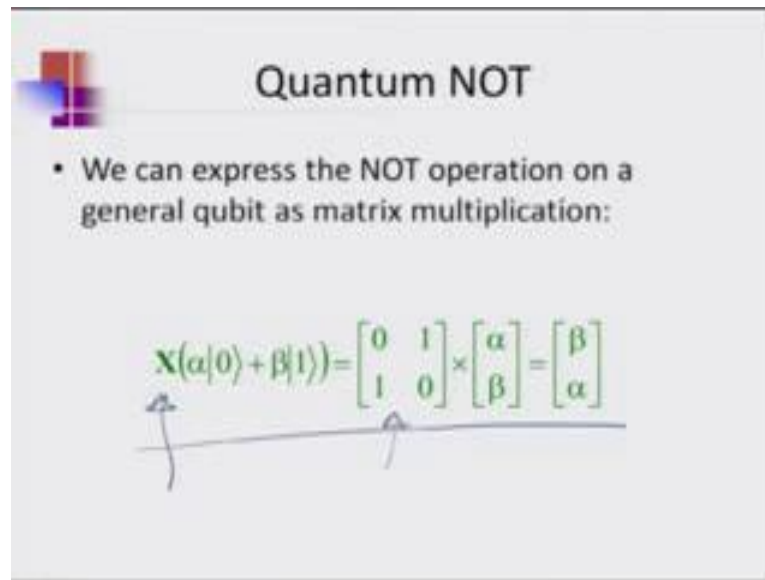
Quantum State Vectors

- Remember that a quantum state is represented by a vector
- Notation:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

Whenever you operate, here is a way; how it works the quantum state is represented by a vector. 0 is 1 0 whereas, 1 is 0 1 and the any psi which is a linear combination of this 2 can be written in this form.

(Refer Slide Time: 35:23)



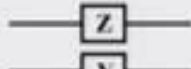
The slide is titled "Quantum NOT" and features a small logo in the top left corner. Below the title, there is a bullet point: "We can express the NOT operation on a general qubit as matrix multiplication:". The main content is a mathematical equation: $X(\alpha|0\rangle + \beta|1\rangle) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$. The equation is written in green. Below the equation, there is a horizontal line with two upward-pointing arrows, one under the input vector and one under the output vector, indicating the mapping of the input state to the output state.

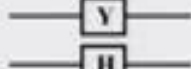
You can now express the not operation as on a generalized qubit, by using a matrix manipulation method and you will find that this always works. That is why your NOT operator X is working properly as a quantum gate and you can immediately see that NOT was a good one to take because NOT is one of the classical gates which is reversible. I specifically choose the one to start with which is the reversible one. It makes sense. Everything goes well.

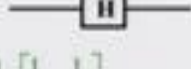
(Refer Slide Time: 36:05)

Other Single Qubit Gates

- The Pauli-X gate works on only **one** qubit
- Other common single qubit gates are:

– Pauli-Z gate: $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ 

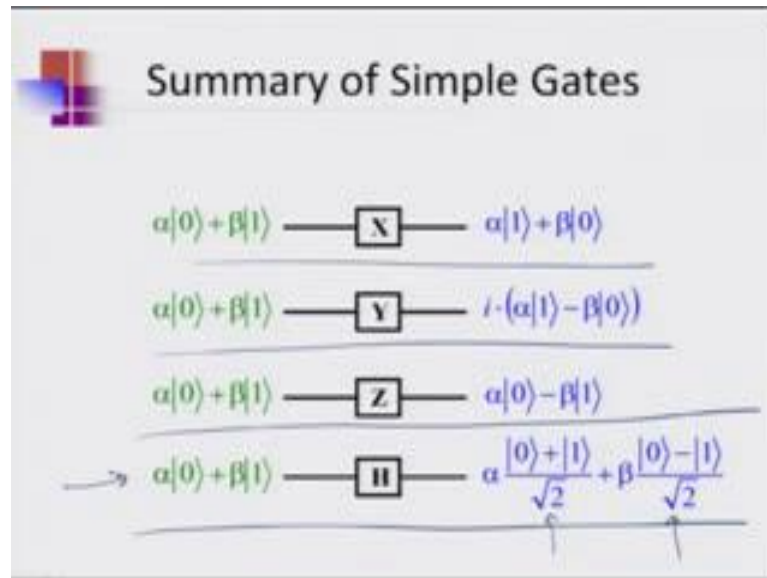
– Pauli-Y gate: $Y = Z \times X =$ 

– Hadamard gate: $H = \sqrt{\text{NOT}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ 

Let us look at another important single qubit gates. This was a single qubit gate. We will just use 1 qubit. The other single qubit gates which are important, that was the Pauli X gate and this works on 1 qubit. The other common single qubit gates are the Pauli Z gate. The representation therefore, would then be in terms like similar to the way that, you write the classical ones. You will be writing out how they are Pauli Z gate is 0 1 0 0 minus 1 then Pauli Y gate is a multiplication of the Z and the X and the Hadamard gate is an essentially a square root of the NOT gate.

You can for your own; you can do these molecule excises to prove it to you that this is how they work. If you just apply Y, what does the Y look like? You should write it out similarly the Z is anyway given, but the fact that the Hadamard gate can be actually written in this form by taking a square root of not just simple exercise, you should just refresh your matrix manipulation by doing this X.

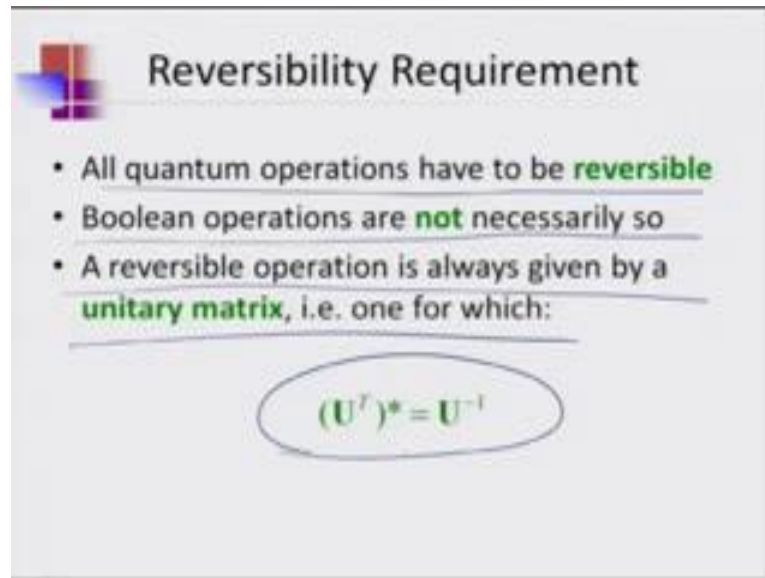
(Refer Slide Time: 37:27)



Now, the summary of the simple gates therefore are these X gates, basically the NOT gate alpha 1 0 converts to 1 beta 1 converts to 0, then the Y gate which basically is a complex conversion. Let us a part which takes you like this and then the Z gate gives you that finally, the Hadamard gate now this Hadamard gate is actually very important as we will find out later because what it is producing is a superposition of the 2 states. That is involved with the individual amplitudes.

Basically your sort of equalizing the 2 states, I mean if I take my basis now to be 0 and 1, superposition of 0 and 1 anyone of them, then I will get a 50-50 combination of them. That is the reason why it becomes useful. Now the reversibility requirement is we have being doing that.

(Refer Slide Time: 38:41)



Reversibility Requirement

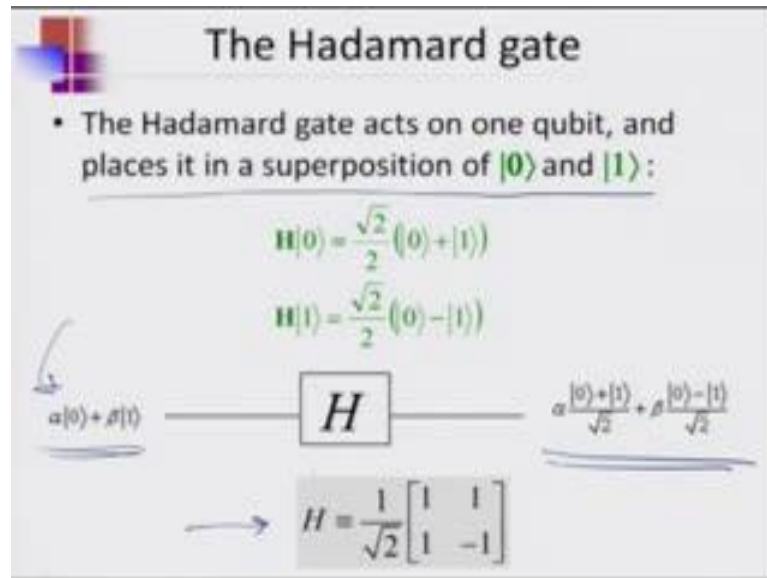
- All quantum operations have to be **reversible**
- Boolean operations are **not** necessarily so
- A reversible operation is always given by a **unitary matrix**, i.e. one for which:

$$(U^T)^* = U^{-1}$$

But now this is the point where we should look at it. Since all the quantum operations have to be reversible, we have this principle that our gates have to be designed in a particular way that they are always reversible. This is not true for the classical case and that is what is written in next line which is that, Boolean operations are not necessary reversible. A reversible operation is always given by a unitary matrix for which, all our quantum gates will be written will be possible to be written in terms of this kind of a unitary transforms.

That is one other point, which we have. Any gate that we device will have a unitary transform. It will be possible to get this inversion very easily. Any operations that you have, you have its inverse available. That is the advantage of it.

(Refer Slide Time: 39:52)



Now let us look at the Hadamard gate because it is an important gate and as I said, it places it in a superposition of 1 and 0. This is the gate which takes 1 qubit and puts it in the superposition state of its own 2 states.

This is important because you will find many practical uses of this Hadamard gate later on and this is the representation that you will in shorthand notation when you write circuits which is analogous to computer circuits, you will be having this kind of scenario where we will be feeding in the states and you will be having an output state like this with just the circuit diagram given like that. You should be able to interpret these kinds of simple circuit diagrams to you. That is one of the learning that we do in this because we will be soon going to cases where I will be simply showing you pictures like this and will not be writing down these other states. If I give you this then you should be immediately able to recognize that, this is how it is or if I just give you this, there should be able to understand how this is happening. This is important because this is the language that will be following for building up quantum circuits.

(Refer Slide Time: 41:09)

Hadamard Gate (contd.)

Input	Action of Hadamard Gate	Output
$ 0\rangle$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$	$\frac{ 0\rangle+ 1\rangle}{2}$
$ 1\rangle$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$	$\frac{ 0\rangle- 1\rangle}{2}$
$a 0\rangle + b 1\rangle$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}}\begin{bmatrix} a+b \\ a-b \end{bmatrix}$	$\frac{a+b}{2} 0\rangle + \frac{a-b}{2} 1\rangle$ = $\frac{ 0\rangle+ 1\rangle}{2} \cdot a + \frac{ 0\rangle- 1\rangle}{2} \cdot b$
$\frac{ 0\rangle+ 1\rangle}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$	$ 0\rangle$
$\frac{ 0\rangle- 1\rangle}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$	$ 1\rangle$

Basis for Transform

The Hadamard gate for instance, can give you very interesting results. With 0 as an input, you will be getting a superposition of the sum. With 1 you will get the superposition of the difference when you put in the superposition state in there, you actually get a superposition of the individual components with equal probabilities of theirs or another way of writing is this one where we are actually taking in some sense, this is the very important gate for basis transform. This is very important because in quantum mechanics, is all about taking the problem to a basis where it is easy. That is the reason why quantum mechanics is quantum computing is also going to take advantage of that and the design of a quantum algorithm or anything else that you can think of is 90 percent relying on this idea as to how smoothly can you come up with ideas with which you can do your basis transform to a condition where it is the easiest to solve the problem. And then you come back to the state or the transform back to the case where you were, that you can get your result. That is the point of this entire place. Hadamard is almost 99.9 percent. You will always find a Hadamard gate existing in any computing you cannot avoid it because you always need to do basis transforms. That is one thing.

Similarly, if you input this superposition you will get back 1. This is basically complement, you its shows that it is a completely reversible condition. If you believe in this, then you can immediately get the other and so on.

(Refer Slide Time: 43:16)

The Pauli gates

- The Pauli gates act on one qubit, as follows:
 - phase shift, σ_z :

$$\sigma_z(\alpha |0\rangle + \beta |1\rangle) = \alpha |0\rangle - \beta |1\rangle$$
 - bit flip, σ_x :

$$\sigma_x(\alpha |0\rangle + \beta |1\rangle) = \alpha |1\rangle + \beta |0\rangle$$
 - phase shift and bit flip, σ_y :

$$\sigma_y(\alpha |0\rangle + \beta |1\rangle) = \alpha |1\rangle - \beta |0\rangle$$
 - identity, I , does not change the input

σ_z
σ_x
σ_y
I

If you believe in this, then you can immediately get the other and so on so forth. The other important gate is the Pauli gate, once again it is important because the Pauli gate is in important for phase shift which means that the phase of the 2 states can be reversed for example, this is their 0 and 1 are in the same direction sum, but the reversal will make 1 is plus and 1 is minus. This is the phase shift whereas; the sigma X is a bit flip. We started off by saying these were the Z X and Y gates, more formally in terms of the Pauli gates they are more known as the sigma Z sigma X and sigma Y and to complete the identity goes along with it. These are our final set of four sigma four Pauli gates and these are all very important in terms of anything to do with spin.

That is the part, which is very important about these Pauli gates. One of them, you can see there is an understand the Z is only 1 dimension Z sigma Z that is only phase shift if you do a bit flip that is along the X axis if you do both phase shift and bit flip, which means that you are doing Y which is a composite of Z and X will find this and identity does not do anything, but this is necessary to complete the gate Z, that is what the Pauli gates are.

(Refer Slide Time: 45:08)

Pauli Matrices

$$\sigma_0 = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$\sigma_1 = \sigma_x = X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$
$$\sigma_2 = \sigma_y = Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$
$$\sigma_3 = \sigma_z = Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

These are the corresponding Pauli matrices, they are also often written as 0 1 2 and 3 - 0 is nothing but the identity, sigma 0 then, sigma 1 is the X, sigma 2 is the Y and sigma 3 is the Z. I have given you all the different notations which are used in different books, each books uses different notation and to be clear I have put them all in this particular slides that there is no confusion wherever you see these you know these are the most popular single qubit gates.

(Refer Slide Time: 45:45)

The Controlled NOT Gate

- The **CNOT** gate is the standard **two-qubit** quantum gate
- It is defined like this:

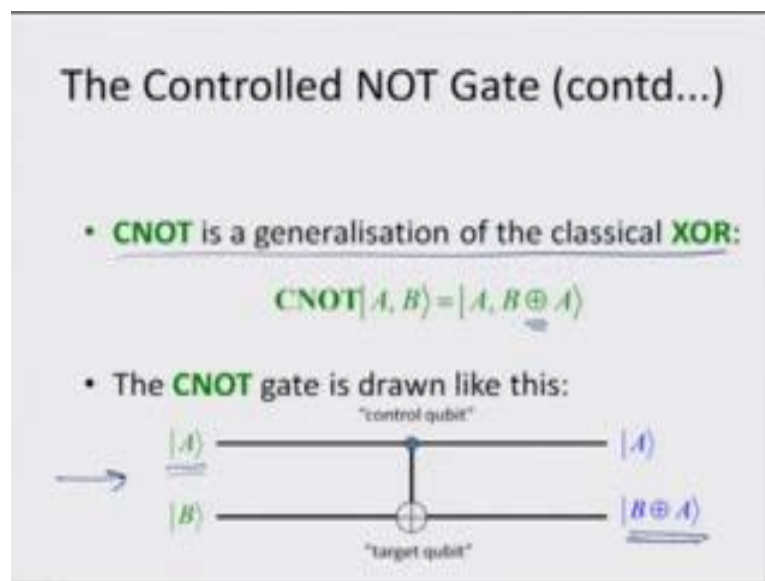
\rightarrow	$\text{CNOT} 00\rangle = 00\rangle$	\leftarrow
	$\text{CNOT} 01\rangle = 01\rangle$	
	$\text{CNOT} 10\rangle = 11\rangle$	
	$\text{CNOT} 11\rangle = 10\rangle$	

Handwritten annotations: "No change |0>" with an arrow pointing to the first row; a circled "flip" with an arrow pointing to the second row of the table.

Now, the final, the first gate which we will do, which is more than the single qubit gate is the control NOT gate. This is a 2 qubit gate. Why you need more than one in this case is because you need a bit for the control. It is more often known as this. This is our other bit requirements. The control NOT gate or the CNOT gate as is popularly known is the standard 2 qubit quantum gate and it is defined in this fashion when you are only operating on 0, 0 then it is just going to give back the same one because your control parameter essentially the 0. Whenever you have 0s, then it is leaving it. Your operation and the control essentially give raise to what is going to happen.

The 1 flips the bit and the 0 is no change. That is your control the first bit is your control that is working on the other one to do the not right. That is why it is a control not. The state of the first if it is going to be 1 will give rise to a flip, if it is going to be 0 it will not let it will not make any changes. That is why, it is a control the first bit decides as to whether it will act like a NOT gate or NOT. That is why it is a control NOT.

(Refer Slide Time: 47:46)



It has a notation, this is its notation a plus with 0 inside the 0 there is a plus. It is a generalization of the XOR, classical XOR. If you, they call it X or XOR, whatever way you want to call it. The classical X or is essentially these where you have this principle and here you are using this 2 qubit case where the second 1 is, second bit is going to behave based on what the first bit is going to be. It is drawn in this fashion. This is also important to know because you would like to see. Basically the first bit remains as it is.

Whether it is 0 or 1 remains as it is, the second one is going to change and depending on whether the control is going to do something or not, it is going to show off as a result that is how it is.

(Refer Slide Time: 48:43)

The Controlled NOT Gate (contd.)

- The matrix corresponding to the **CNOT** gate is:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

4x4

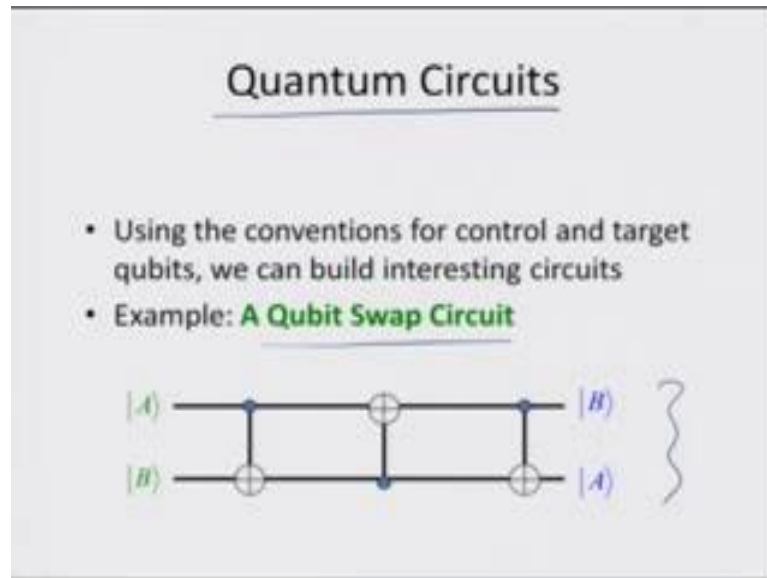
Output matrix Y

$$Y \begin{bmatrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{bmatrix}$$

- The **CNOT** together with the single qubit gates are **universal** for quantum computing.

The first obviously, if it is a 2 qubit gate, then it has to be a 4 by 4 matrix because for the 1 qubit gates we were using 2 by 2 matrices. This is a 2 qubit gate; this will be a 4 by 4 matrix. The simplest matrix corresponding to the CNOT will therefore be this. Here you can see that, this is my CNOT gate and here is an example of applying the CNOT gate for a matrix which is y for a state, which is Y and you get the solution. Now these CNOT gates along with the single qubit gates are universal for quantum computer irrespective of how you formulate your problem. These will be always a complete closed set as you go and make bigger gates, they can often be broken down into these basic gates, but these gates cannot be broken down further. That is what is known that you know if you can go to the very basic state that is how they are.

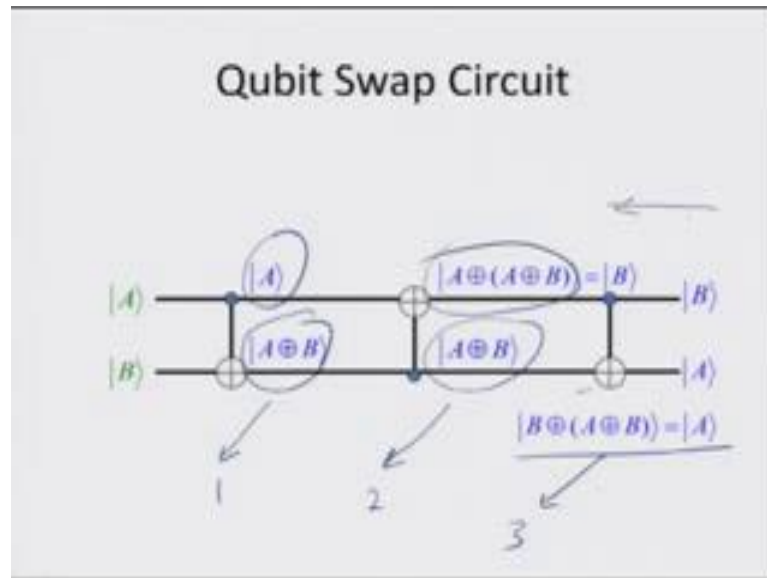
(Refer Slide Time: 50:26)



How can you actually use them? The first important thing about the quantum computing is that once you have learnt your qubits and you have learnt some operations which are you gates, and then you can actually write out your circuits. We have been doing that, but here is a basic principle of how to go about doing something which is combination of more than 1 applied gate to get to where you want to. For example, we would like to just show a quantum swap circuit. I would like to go from A to B and B to A. The point is when you if you apply only 1 gate, it will be not possible to swap both is a CNOT will only swap one of them and the other one remains constant, but if you apply it twice then you end a producing a swap gate a quantum swap circuit.

This is exactly an example of that. By using just control NOTs in series, you will be able to get this happen. You can actually start building your interesting circuits from these very basics that we have just learnt. So, this is a circuit.

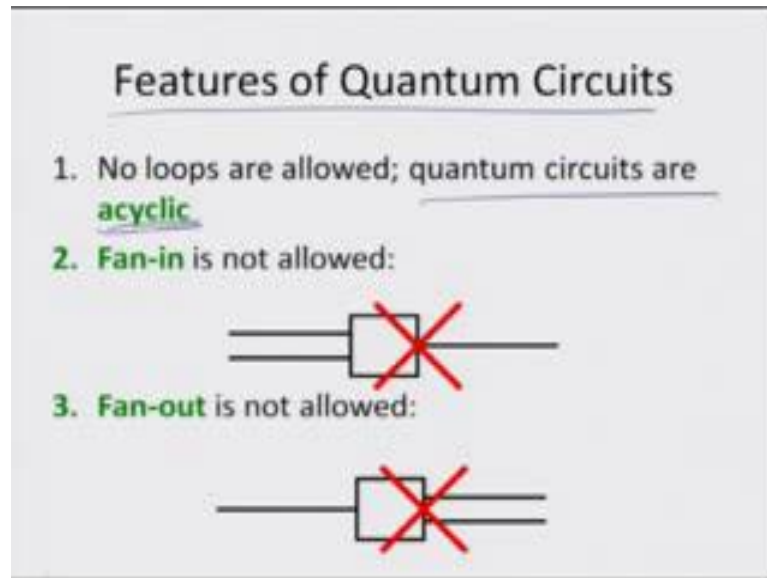
(Refer Slide Time: 51:42)



Let see, here is the point that is being shown step by step. What does the first one do? First one generates a CNOT. You get this next one you have not applied anything on this part. It is only going to be the next and then the second one is going to be this, which is going to go to become B and then that one again remain same whereas, the other one finally, goes and applies this one to get back there. How many CNOTs did I use? 1 2 and then the third one; now if you tell somebody, who does classical computing will just look at and you say come on you know is a swap qubits, they equal swap qubits for that you have gone through 3 not gives 3 controls and 3 gates to do this, but one very important thing to note here; however, is that this is reversible. I can as well come from this side and I will have the same result that is not true when you do a classical computing.

In classical computing, yes you can actually swap them just like that, but you are expending energy, you want to go the other way round is expending energy because it is not reversible. It is not the same way how it works.

(Refer Slide Time: 53:08)



That is the idea. That brings it to a very important point. What are the basic features that we are now looking for in a quantum circuit? We cannot do a few things that we have just now realized. No loops are allowed. Quantum circuits are acyclic, you know why? There are no loops which are allowed, quantum circuits are cyclic acyclic because you cannot paralyze loop circuits, even we when you do parallel procing in a classical computer, the most difficult part is the nested loops. Generally speaking you are not really supposed to have nested loops if you have trying to do parallel procing.

For example, fan in you can understand that why fan-in is not allowed you are going to loose qubits, you cannot have that and similarly you cannot gain qubits, you cannot have fan-out. What we are going to do is that we are going to take on more from this point on into further lectures. For today let us stop here and we will continue on with this in the next lecture.

Thank you.