NPTEL

NATIONAL PROGRAMME ON TECHNOLOGY ENHANCED LEARNING

IIT BOMBAY

CDEEP
IIT BOMBAY

Quantum Infromation and Computing

Prof. D.K.Ghosh
Department of Physics IIT Bombay

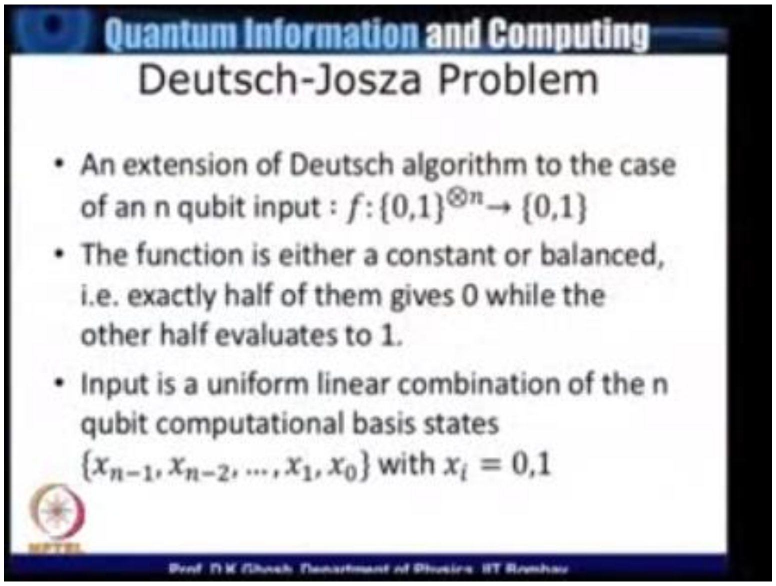
Modul No.04

Lecture No.17

Deutsch- Josza & Bernstein – Vazirani Algorithm

In the last lecture we had talked about a simple algorithm called Deutsch algorithm in which we said that if we have one qubit input with one qubit output the function being either constant or being a balanced function a quantum computer can determine it in one single query whereas a classical computer would take two equals we had just begun to introduce an extension of the same algorithm by Josza where the difference was that the input is n qubit input and the output is still a single qubit is output namely it is value could be either 0 or 1.

(Refer Slide Time: 01:07)



Quantum Information and Computing

Deutsch-Josza Problem

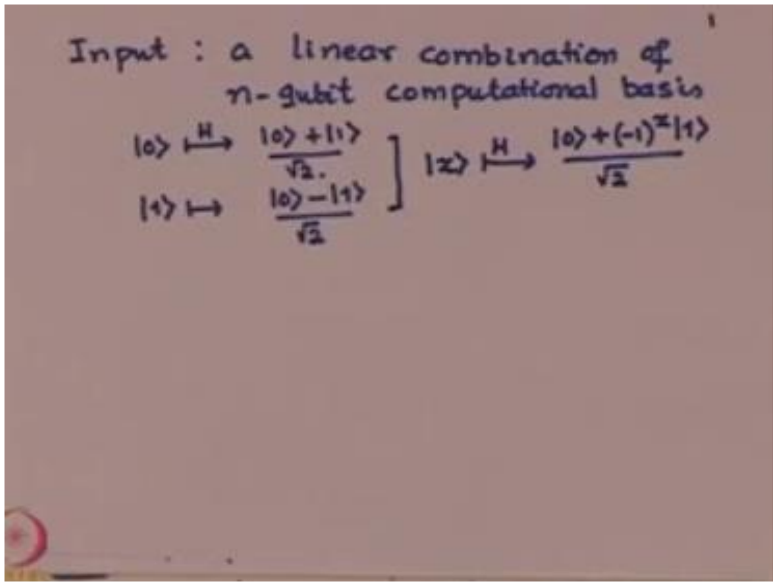
- An extension of Deutsch algorithm to the case of an n qubit input : $f: \{0,1\}^{\otimes n} \rightarrow \{0,1\}$
- The function is either a constant or balanced, i.e. exactly half of them gives 0 while the other half evaluates to 1.
- Input is a uniform linear combination of the n qubit computational basis states $\{x_{n-1}, x_{n-2}, \ldots, x_1, x_0\}$ with $x_i = 0,1$

Prof. D.K Ghosh, Department of Physics, IIT Bombay

So this is shown in this slide that your function is takes n qubit 0, 1 with that simple multiplication n shows that an n qubit input and that is evaluated to 0 or 1 so that is my that is the job of my Oracle and the function is either balance or constant job of the algorithm is to find out which type it belongs to. I also pointed out last time that classically in the worst-case scenario it will take n/ 2 + 1 number of queries to determine whether the function is balanced or constant though we did point out that probabilistically it will probably take much less.

But nevertheless the technically classical computation could involve of the order n step n/ 2 is same and let us look at what the quantum computer does. The first job is to prepare a input as a linear combination of the n qubit basics so my input is a linear combination.

(Refer Slide Time: 02:25)



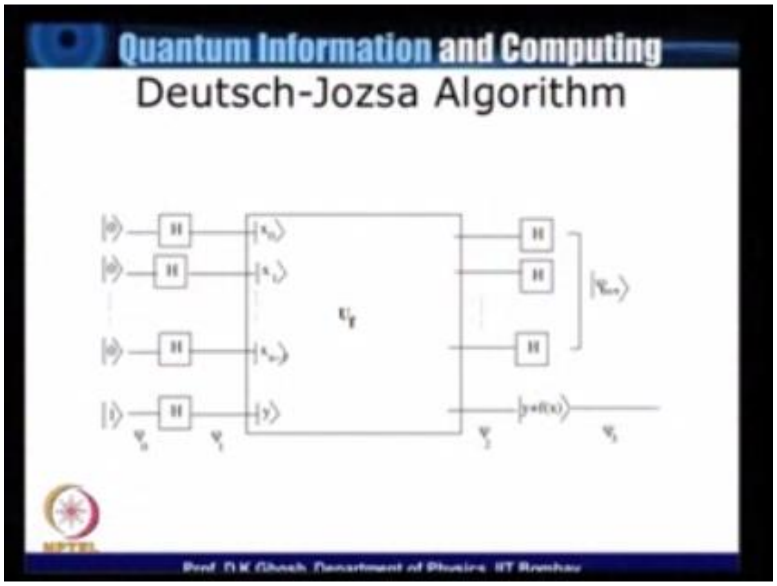
Input : a linear combination of n-qubit computational basis

$$|0\rangle \xmapsto{H} \frac{|0\rangle+|1\rangle}{\sqrt{2}}.$$
$$|1\rangle \mapsto \frac{|0\rangle-|1\rangle}{\sqrt{2}}$$
$$|x\rangle \xmapsto{H} \frac{|0\rangle+(-1)^x|1\rangle}{\sqrt{2}}$$

Of n qubit computational basic state now since this is the rather important step in many of the algorithms let me explain this a little bit. We have seen that a single qubit when it is passed through a Hadamard gate for example if it is 0, then I get 0 + 1 / √2 and if it is 1 you get 0 -1/ √2. Now I can combine this into a single step saying that if my input is x putting it through a Hadamard gate will give me 0 + minus 1^x 1/√2.

So that when x is equal to 0 it gives me 0 +1/ √2 and if x equal to 1 I get 0 – 1. Now what happens if I have a n qubit state and I pass all of them through Hadamard gate that is a sequence of Hadamard gates as is shown in the slide.

(Refer Slide Time: 04:03)

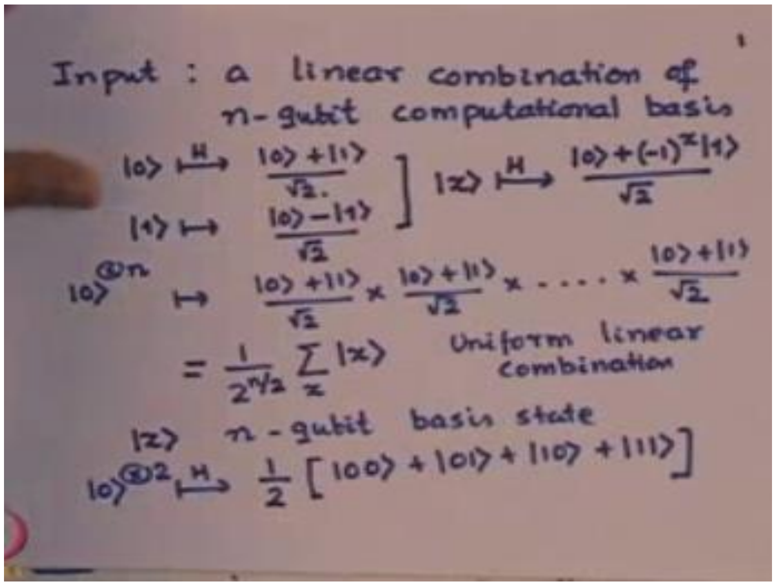

Quantum Information and Computing

Deutsch-Jozsa Algorithm

Prof. D.K.Ghosh, Department of Physics, IIT Bombay

So what I do is I start with 0 in each of my.

(Refer Slide Time: 04:10)



Input : a linear combination of n-qubit computational basis

$$\left.\begin{array}{l} |0\rangle \xrightarrow{H} \frac{|0\rangle+|1\rangle}{\sqrt{2}}. \\ |1\rangle \mapsto \frac{|0\rangle-|1\rangle}{\sqrt{2}} \end{array}\right] |x\rangle \xrightarrow{H} \frac{|0\rangle+(-1)^x|1\rangle}{\sqrt{2}}$$

$$|0\rangle^{\otimes n} \mapsto \frac{|0\rangle+|1\rangle}{\sqrt{2}} \times \frac{|0\rangle+|1\rangle}{\sqrt{2}} \times \cdots \times \frac{|0\rangle+|1\rangle}{\sqrt{2}}$$

$$= \frac{1}{2^{n/2}} \sum_x |x\rangle$$ Uniform linear combination

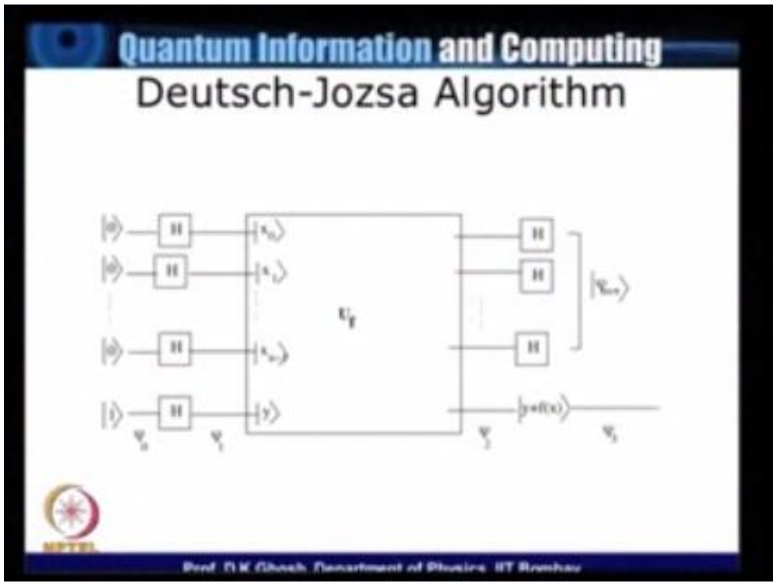
$|x\rangle$ n - qubit basis state

$$|0\rangle^{\otimes 2} \xrightarrow{H} \frac{1}{2}\left[|00\rangle+|01\rangle+|10\rangle+|11\rangle\right]$$

Register so I will write it as 0 n now if n of the n0 are passed through Hadamard gates then what do I get I simply get 0+1 /√2 for each one of them. Now if you expand this up what do you get there is a 1 /√2 for there so I will get $1/2^{n/2}$ there will be terms like 0, 0, 0, 0, 0, 1, 0, 0 etc…, in other words I will get a linear combination of each of the n qubit basis states and each one of them appearing with the same strength.

So this I will write as Σ_x x where x is n qubit basis state so this is simply a shorthand notation for 0000 etc. For example, supposing just to give you an idea of what happens when I start with 00 so what happens to this when it is pass through Hadamard gate I get 1 / n= 2 so I get 1 / 2 and this would then be 00 + 01 + 10 + 11 and a very similar statement I can make for any numbers. So this is what we will frequently refer to as the uniform linear combination of n qubit basis. The target in which my output will come is it is set.

(Refer Slide Time: 06:45)

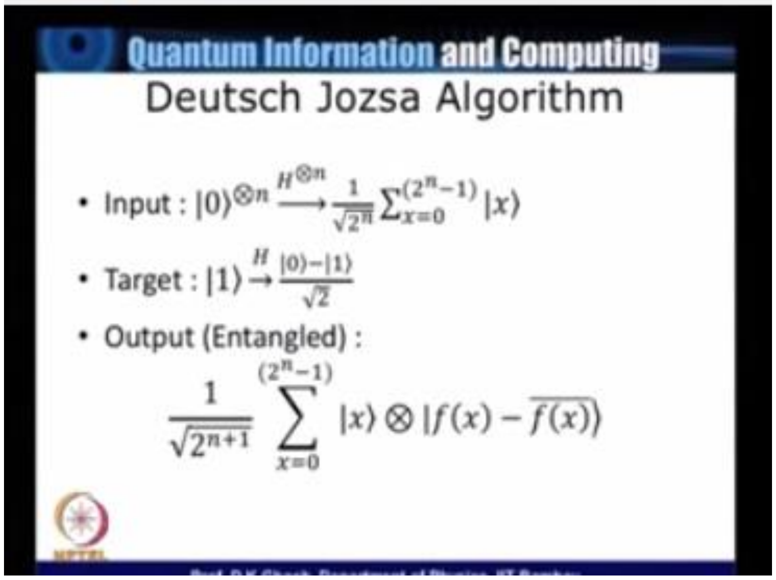

Quantum Information and Computing
Prof. D.K Ghosh, Department of Physics, IIT Bombay

Deutsch-Jozsa Algorithm

To 1 and I pass that through Hadamard gate as you know that this is 0 -1√2 now here in this picture I am indicating the state of the system by ψ_0, ψ_1, so what I do this is a very common nomenclature that since we have said that this is the direction of logic flow essentially it is a time diagram, so my initial for a starting point which is 0000 at last one being one is ψ_0 when all of them are put through the Hadamard gate my state is ψ_1.

Then this thing is passed through an oracle and so the out after the oracle what comes out the this will be the same for the inputs but there would be a difference in the output and then ψ_2 and then after the first qubits are put through the Hadamard gate I will call it as the ψ_4 or ψ_3 does not matter.

(Refer Slide Time: 07:52)



Quantum Information and Computing

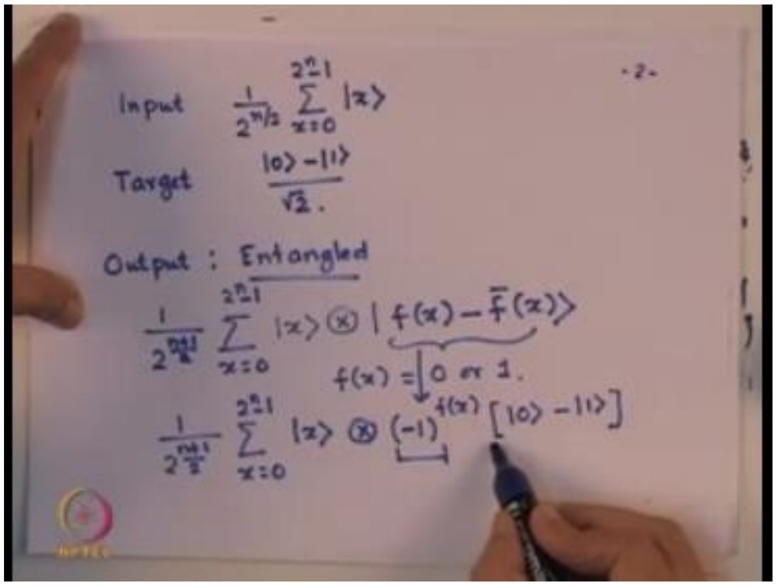
Deutsch Jozsa Algorithm

- Input : $|0\rangle^{\otimes n} \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{x=0}^{(2^n-1)} |x\rangle$
- Target : $|1\rangle \xrightarrow{H} \frac{|0\rangle - |1\rangle}{\sqrt{2}}$
- Output (Entangled) :

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{(2^n-1)} |x\rangle \otimes |f(x) - \overline{f(x)}\rangle$$

Now let us look at what am I getting, so we have said already my input is uniform linear combination.

(Refer Slide Time: 07:59)



-2-

Input $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle$

Target $\frac{|0\rangle - |1\rangle}{\sqrt{2}}.$

Output : Entangled

$$\frac{1}{2^{\frac{n+1}{2}}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |\underbrace{f(x) - \bar{f}(x)}\rangle$$

$f(x) = 0$ or 1.

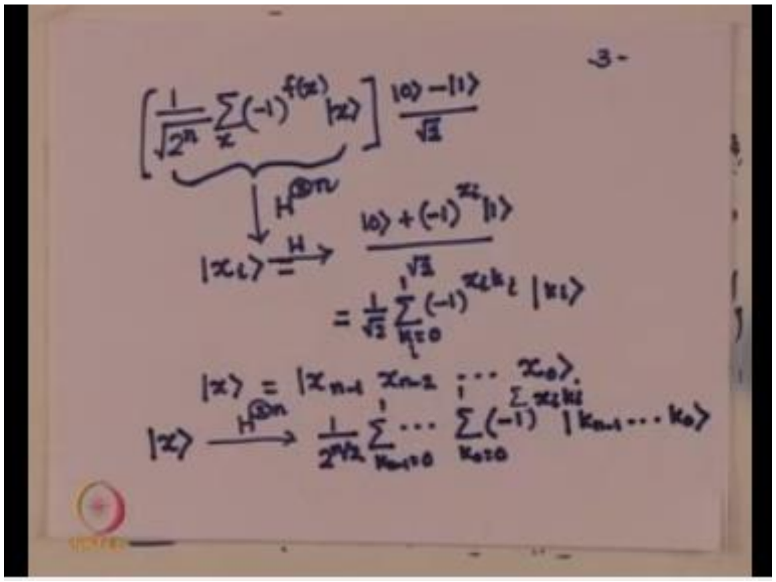
$$\frac{1}{2^{\frac{n+1}{2}}} \sum_{x=0}^{2^n-1} |x\rangle \otimes \underbrace{(-1)}^{f(x)} [|0\rangle - |1\rangle]$$

$1/2^{n/2}$ $\Sigma_x = 0, 2^{n-1}$ there are total 2^n number of states x, the target is one pass through Hadamard gate which gives me |0> - 1/√2. So what is my output, now this output is an entangled output, it gets entangled because the function that it calculates is f(x) corresponding to every X that is here but when that gets XOR with 0 I get f(x) but when you XOR it with one I get n bar of x, so as a result I will be writing my output as $2^{n+1/2}$.

This is product of this with this Σ_x = |0> 2^{n-1} X because that part is not changed, now what happens here is I get f(x) because the oracle calculates f(x) XOR sit with zero and I get an F(x) one when it XOR it I get a bar of x. So this is what I got, now we have already stated there are two types of function f(x) is equal to either 0 or 1, since f(x) is equal to 0 or 1 this quantity is either 0-1 or 1-0.

Depending upon what the value of f(x) is, so you can write this then as everything else remains the same, - $1^{f(x)}$ [|0> - |1>] So this is that phase factor that comes up because of the fact that I could either get 0-1 or 1-0, let us write it in a slightly different way because you notice that I can associate this phase factor would be the first n qubit itself because there is a -$1^{f(x)}$ this is independent of X. So therefore my output can be written as.

(Refer Slide Time: 11:21)



-3-

$$\Big[\underbrace{\frac{1}{\sqrt{2^n}}\sum_x (-1)^{f(x)}|x\rangle}\Big]\frac{|0\rangle-|1\rangle}{\sqrt{2}}$$

$$\Big\downarrow H^{\otimes n}$$

$$|x_i\rangle \xrightarrow{H} \frac{|0\rangle+(-1)^{x_i}|1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\sum_{k_i=0}^{1}(-1)^{x_ik_i}|k_i\rangle$$

$$|x\rangle = |x_{n-1}\,x_{n-2}\,\cdots\,x_0\rangle$$

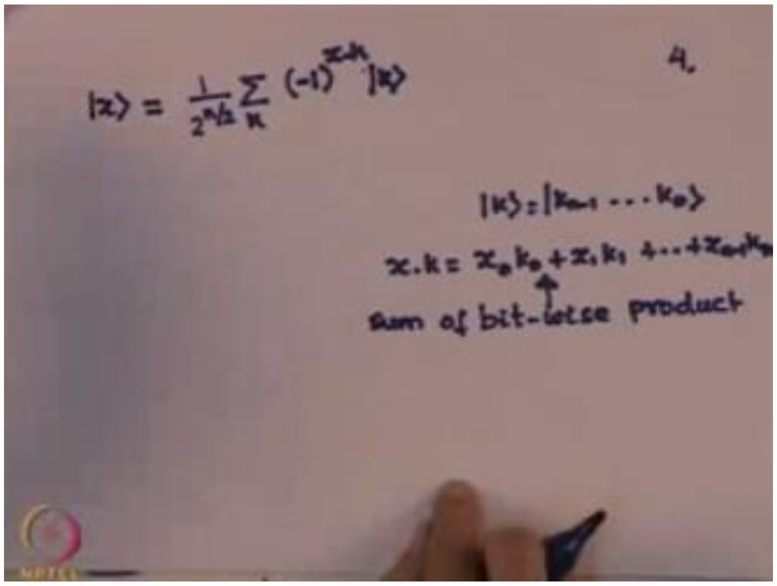
$$|x\rangle \xrightarrow{H^{\otimes n}} \frac{1}{2^{n/2}}\sum_{k_{n-1}=0}^{1}\cdots\sum_{k_0=0}(-1)^{\sum x_ik_i}|k_{n-1}\cdots k_0\rangle$$

1/2 $^{n/2}$ or 1/√2 ^{m} Σ $_{x}$ limits I have written ones, -1$^{f(x)}$ |x> and the second the target qubit is simply |0> - |1>√2. Now let us look at this that supposing now this first n qubits are passed through a Hadamard gate to show that all of them are passed to Hadamard gate that is the notation I use, now what will happen? Now firstly I know this is a little bit of an algebra, so firstly I know that when I have x_i I pointed it out earlier.

X_i pass through Hadamard gate gives me |0> + (-1) ^{xi} |1>. Now there is a quicker way of writing this and that is equal to1/√2 Σ $_{k}$ = 0 to 1, (-1) ^{xik} or if you want it to be a little more symmetrical let us call it k_i x_i k_i then ki, now you can see that this is the same because if ki is equal to 0 I get (-1) ^{0} and 0 if ki is equal to1 I get (-1) ^{xi} this way. So if I then expand my X is equal to |x $_{n-1}$, x_{n-2} etc up to X0.

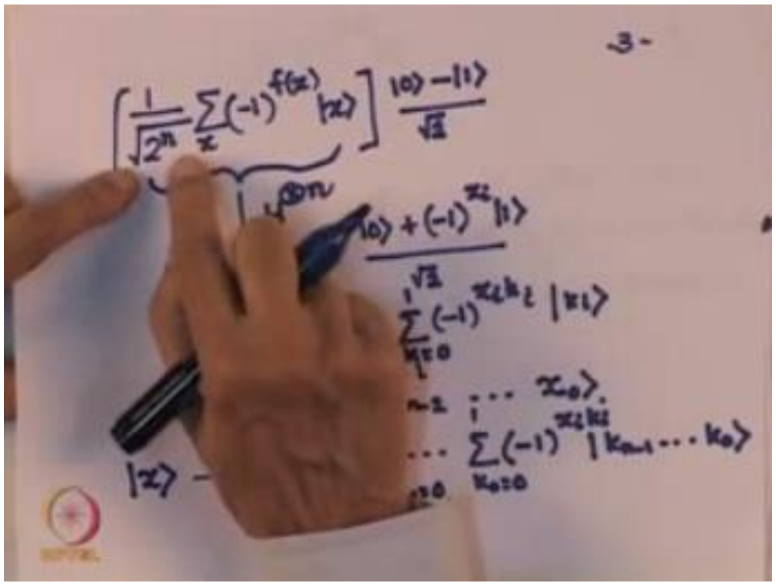
Then when the input to the first n cubic inputs are passed through the Hadamard gate what I get can be written as 1/2 $^{n/2}$ so each one of these terms that was there, I will write it in this form. So therefore I will have sum over K $_{n-1}$ = 0 to 1 and like that up to k_0 equal to 0 to 1, I get -1$^{\Sigma x_i k_i}$ and of course K_{n-1} K_j so one phase factor from each of the terms. Now I can compactly write this.

(Refer Slide Time: 14:23)


$|x\rangle = \frac{1}{2^{n/2}} \sum_k (-1)^{x.k} |k\rangle$
4.
$|k\rangle = |k_{n-1} \ldots k_0\rangle$
$x.k = x_0 k_0 + x_1 k_1 + \ldots + x_{n-1} k_{n-1}$
sum of bit-wise product

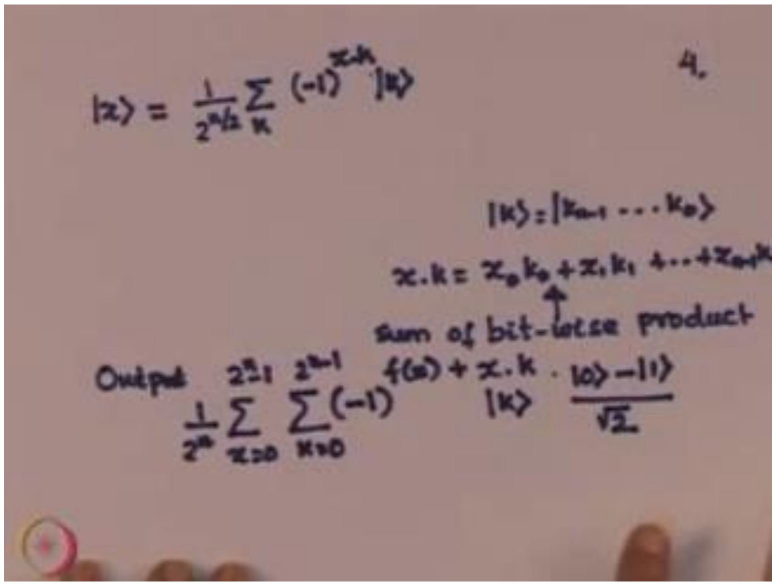
So $1/2^{n/2}$I will take care of the remaining factors later some over K, so K is simply the state so K is my state $|K_{n-1}$ up to $K_0>$ and this 1 I will write at $-1^{x.K}$ now what is x.k, x.k is basically x_0k_0 + x_1k_1 up to x_{n-1} k_{n-1}, so this is if you like bitwise product sum. So let me write it sum of brig bit y product x.k, something like a scalar product the way we write it |k>.

(Refer Slide Time: 15:33)


-3-
$\left[\frac{1}{\sqrt{2^n}}\sum_x(-1)^{f(x)}|x\rangle\right]\frac{|0\rangle-|1\rangle}{\sqrt{2}}$
$\frac{|0\rangle+(-1)^{x_i}|1\rangle}{\sqrt{2}}$
$\sum_{k_1=0}^{1}(-1)^{x_1k_1}|x_1\rangle$
$\sum_{k_0=0}^{1}(-1)^{x_ik_i}|k_{n-1}\dots k_0\rangle$
$|x\rangle$

So now going back to my original input before I had put it into a Hadamard gate there was another $1/\sqrt{2^n}$ factor there and a $-1^{(fx)}$ there and of course there is a $\sum/x$ there.

(Refer Slide Time: 15:48)



4.

$$|x\rangle = \frac{1}{2^{n/2}} \sum_{k} (-1)^{x.k} |k\rangle$$

$$|k\rangle = |k_{n-1} \ldots k_0\rangle$$

$$x.k = x_0 k_0 + x_1 k_1 + \ldots + x_{n-1} k$$

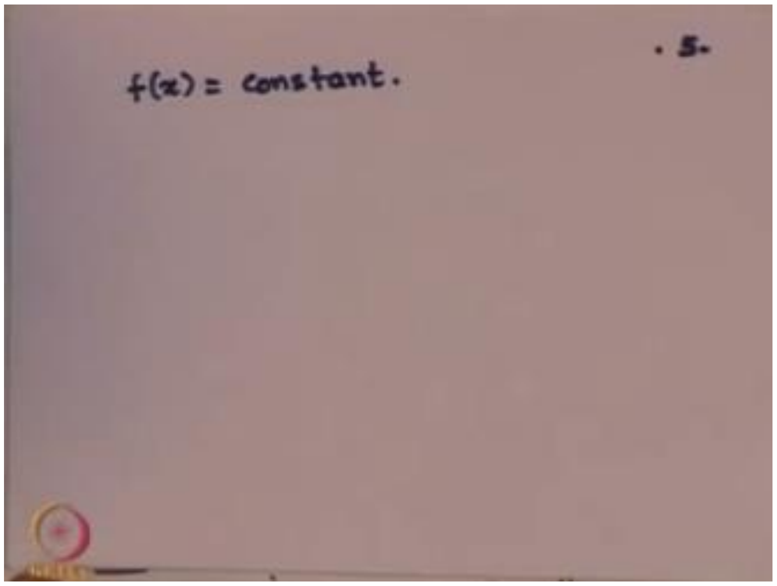
$\uparrow$ sum of bit-wise product

Output

$$\frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{k=0}^{2^n-1} (-1)^{f(x) + x.k} |k\rangle \cdot \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

So when I combine all this I get my output to be given by 1/ 2^n ∑ over x=0 2,2^{n-1} k=0 2^{n-1} $(-1)^{f(x)}$ which was already there, + x.k which we just now calculated state k and of course |0>-|1>/√2, so this is my general expression. Now what I will do now is to evaluate this expression for the case for which f(x) is constant and also for the case for which it is balanced and we will see what comes out of it. Now let us suppose f(x) is constant.

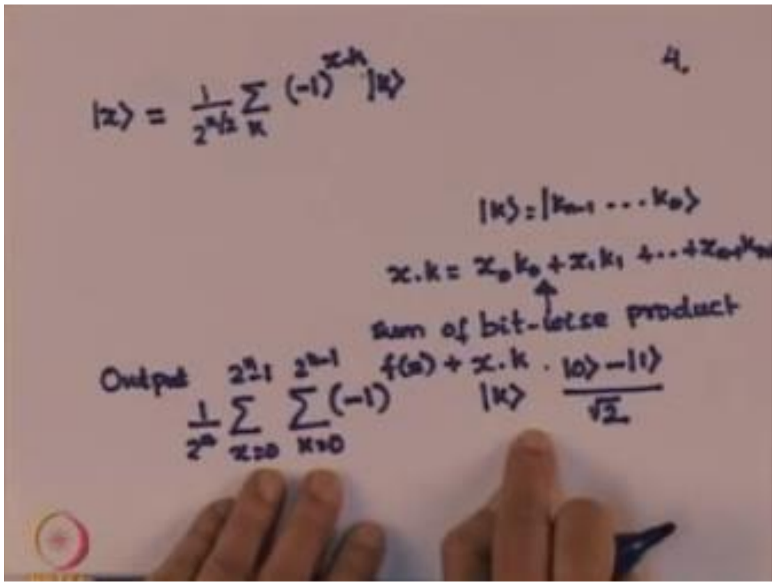
(Refer Slide Time: 16:47)



. 5.

$f(x) =$ constant.

(Refer Slide Time: 16:57)



4.

$$|x\rangle = \frac{1}{2^{n/2}} \sum_k (-1)^{x.k} |k\rangle$$

$$|k\rangle = |k_{n-1} \ldots k_0\rangle$$

$$x.k = x_0 k_0 + x_1 k_1 + \ldots + x_{n-1} k_{n-1}$$

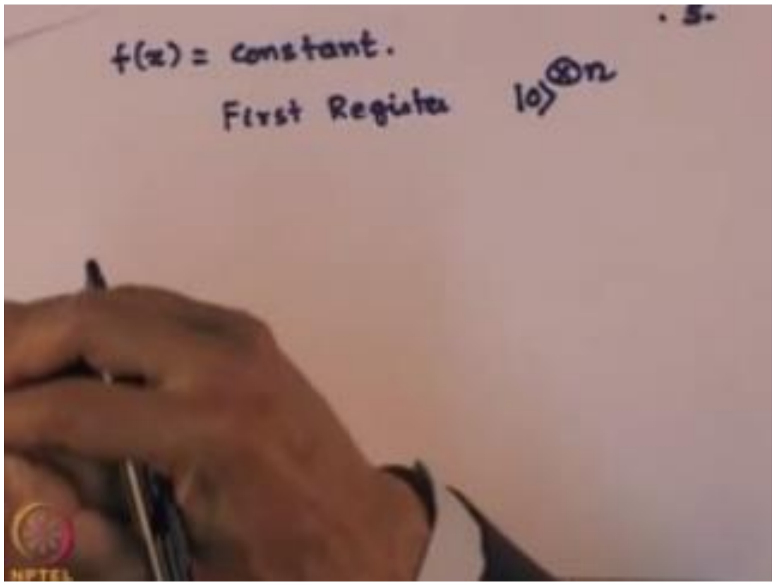
sum of bit-wise product

Output

$$\frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{k=0}^{2^n-1} (-1)^{f(x)+x.k} |k\rangle \cdot \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Now if f(x) is constant then look at this one, so I get a constant phase factor will come out of it and I will refer it $-1^{x.k}$ so the phase factor will have equal number of positive and negative terms because of $-1^{x.k}$ and this will cancel out, excepting for the case for which my k happens to equal to 0. So therefore, for the case for which k = 0 if f(x) is constant I get -1^0 all of them are adding up okay, and so therefore my first register will have the state 0, so f(x) is equal to constant.

(Refer Slide Time: 18:02)

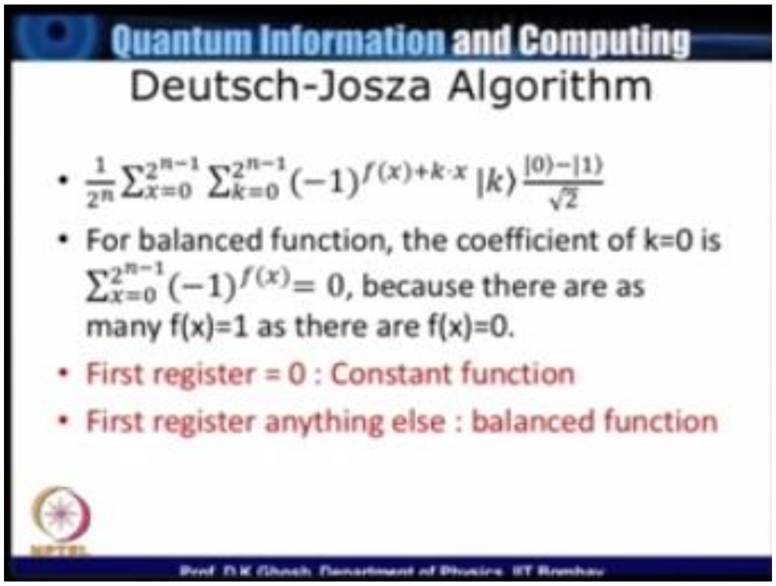


$f(x) = $ constant.

First Register $|0\rangle^{\otimes n}$

The first register will be in state |0> of course in n qubits.

(Refer Slide Time: 18:21)



Quantum Information and Computing

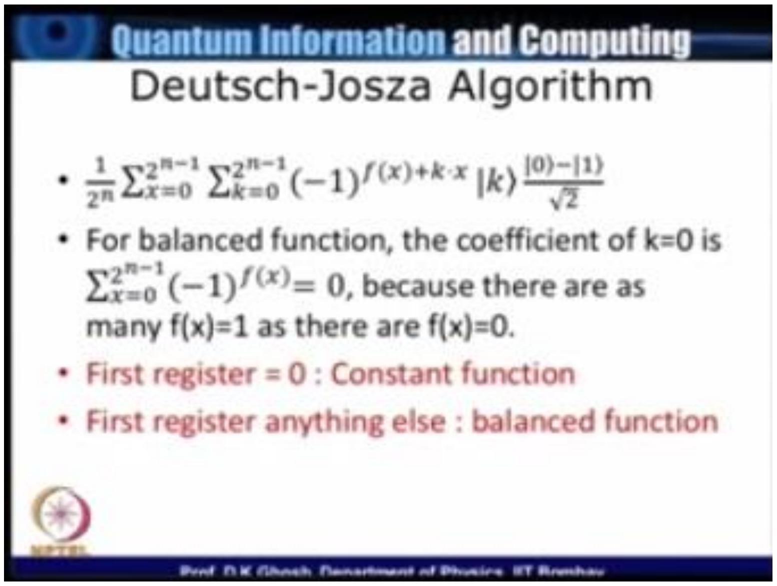
Deutsch-Josza Algorithm

- $\frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{k=0}^{2^n-1}(-1)^{f(x)+k\cdot x}|k\rangle\frac{|0\rangle-|1\rangle}{\sqrt{2}}$
- For balanced function, the coefficient of k=0 is $\sum_{x=0}^{2^n-1}(-1)^{f(x)}=0$, because there are as many f(x)=1 as there are f(x)=0.
- First register = 0 : Constant function
- First register anything else : balanced function

Prof. D.K Ghosh, Department of Physics, IIT Bombay

Now let us look at the same structure again for the case where the function is balanced, but this time I cannot actually have that simplification because there are too many terms there, but I do something else. Let me find out what is the coefficient that comes out with k = 0 term. If you refer to the slide again.

(Refer Slide Time: 18:50)



Quantum Information and Computing

Deutsch-Josza Algorithm

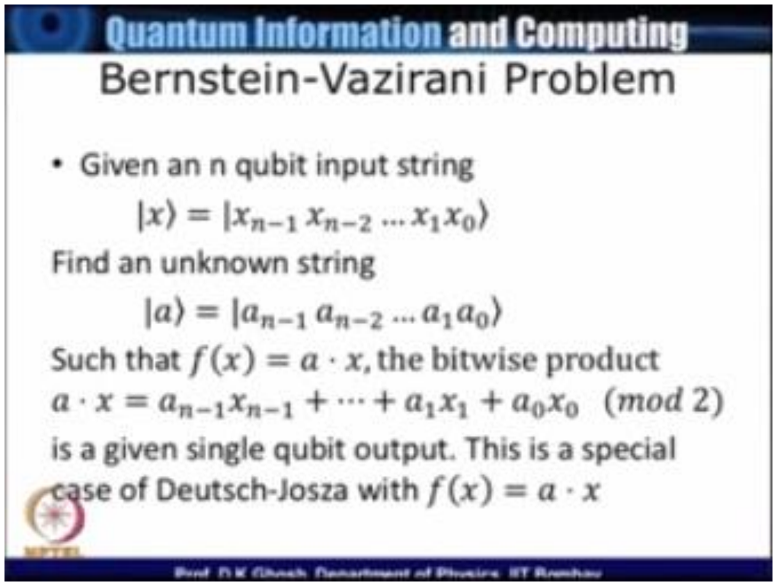
- $\frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{k=0}^{2^n-1}(-1)^{f(x)+k\cdot x}|k\rangle\frac{|0\rangle-|1\rangle}{\sqrt{2}}$
- For balanced function, the coefficient of k=0 is $\sum_{x=0}^{2^n-1}(-1)^{f(x)}= 0$, because there are as many f(x)=1 as there are f(x)=0.
- First register = 0 : Constant function
- First register anything else : balanced function

Prof. D.K.Ghosh, Department of Physics, IIT Bombay

You notice that if I take k = 0 then this k.x of course goes away and I am left with only $-1^{f(x)}$ and ∑/x=0/-1^{n}, now obviously in since the function is balanced there are as many f(x) equal to one term as there are f(x) equal to 0 terms, so therefore this ∑ over X then notice is that there is no x also here, the k.x term has gone away. So that would be equal to 0, so notice is immediately that there is a big difference between the two cases.

In the first case when the function is constant my first register will contain only the state |0> but if the first register measures anything else then the function must be balanced. So once again a single query determines whether my function was balanced or not, and that is obviously a great improvement over the corresponding classical algorithm. This has a very trivial extension.

(Refer Slide Time: 20:20)



Quantum Information and Computing

Bernstein-Vazirani Problem

- Given an n qubit input string

$$|x\rangle = |x_{n-1}\, x_{n-2} \dots x_1 x_0\rangle$$

Find an unknown string

$$|a\rangle = |a_{n-1}\, a_{n-2} \dots a_1 a_0\rangle$$

Such that $f(x) = a \cdot x$, the bitwise product

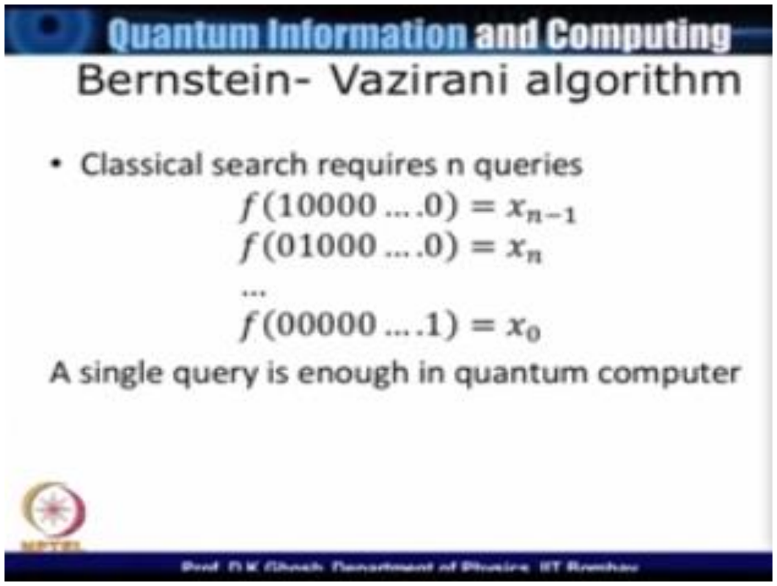
$a \cdot x = a_{n-1}x_{n-1} + \cdots + a_1 x_1 + a_0 x_0 \pmod 2$

is a given single qubit output. This is a special case of Deutsch-Josza with $f(x) = a \cdot x$

Prof. D K Ghosh, Department of Physics, IIT Bombay

To a problem which is known as Bernstein-Vazirani problem, now in the Bernstein-Vazirani problem we have an input string x = x and -1 x 0 and there is an unknown string a this unknown strings information is known to your oracle. And what we do is that the oracle calculate for a given xf(x)=a.x where a.x is the sum of the bitwise product the dimension just now, that is a.x is equal to this, and this is single qubit output. So therefore this is essentially a Deutsch-Josza problem with the change that your f(x) is a function which is equal to a.x the problem in Bernstein-Vazirani problem is to find out what is this unknown string.

(Refer Slide Time: 21:27)



Quantum Information and Computing

Bernstein- Vazirani algorithm

- Classical search requires n queries

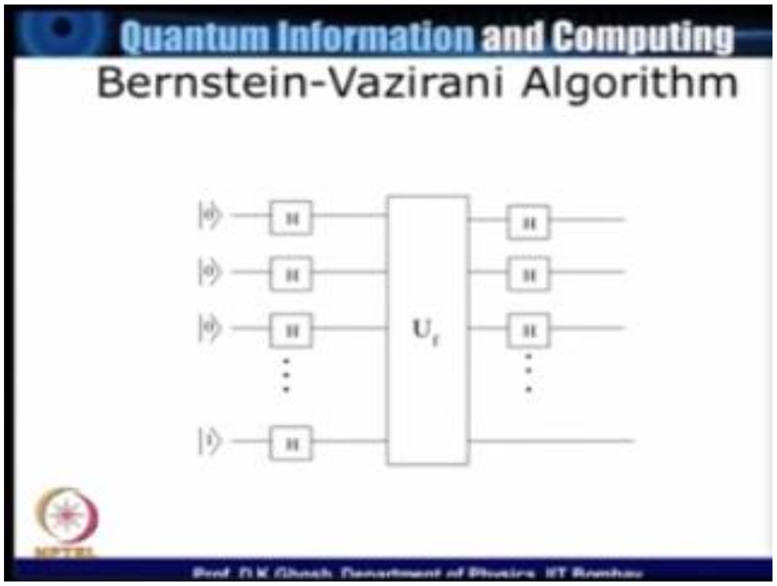
$$f(10000\ldots.0) = x_{n-1}$$
$$f(01000\ldots.0) = x_n$$
$$\ldots$$
$$f(00000\ldots.1) = x_0$$

A single query is enough in quantum computer

Prof. D.K Ghosh, Department of Physics, IIT Bombay

Now what about classical searches the classical search requires n number of queries because I can evaluate what is for instance f (100), f(010) and find out immediately what are these things should be a n -1 a and up to a zero, so therefore it requires n number of queries.

(Refer Slide Time: 21:53)

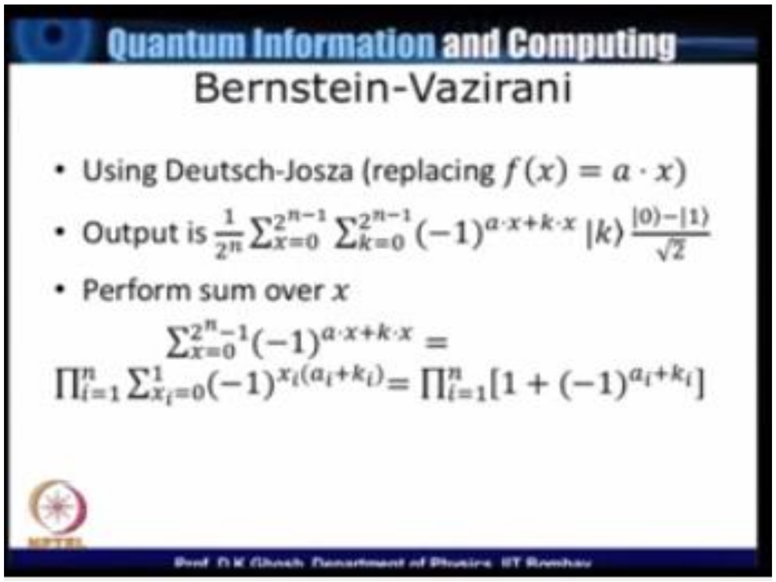

Quantum Information and Computing

Bernstein-Vazirani Algorithm

Prof. D.K. Ghosh, Department of Physics, IIT Bombay

Now basically what I have drawn here is the same picture as that of the Deutsch- Josza with the oracle being that oracle which computes for a given X a. X.

(Refer Slide Time: 22:09)



Quantum Information and Computing

Bernstein-Vazirani

- Using Deutsch-Josza (replacing $f(x) = a \cdot x$)
- Output is $\frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{k=0}^{2^n-1}(-1)^{a\cdot x+k\cdot x}|k\rangle\frac{|0\rangle-|1\rangle}{\sqrt{2}}$
- Perform sum over x

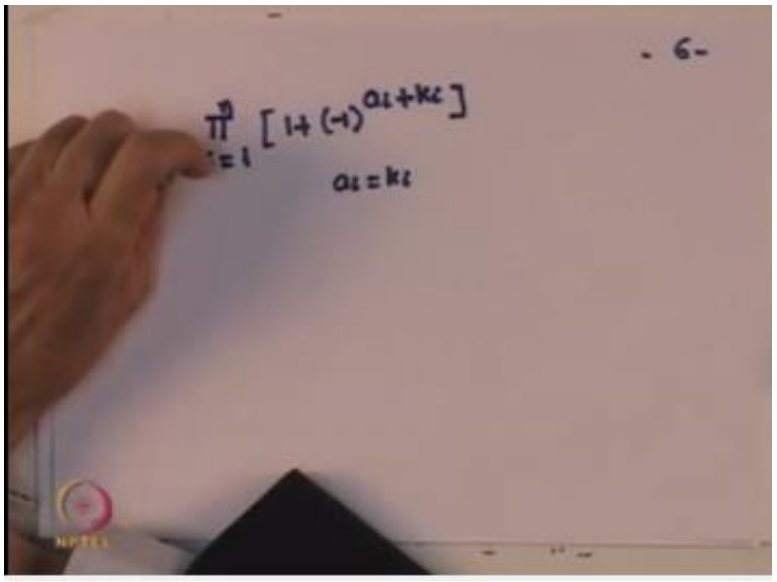
$$\sum_{x=0}^{2^n-1}(-1)^{a\cdot x+k\cdot x} = \prod_{i=1}^{n}\sum_{x_i=0}^{1}(-1)^{x_i(a_i+k_i)} = \prod_{i=1}^{n}[1+(-1)^{a_i+k_i}]$$

Prof. D.K Ghosh, Department of Physics, IIT Bombay

So let us look at that the what I have done in the second line of the slide is to rewrite that Deutsch- Josza expression for the output accepting that in place of the f(x), I have written a.x. So this is a.x + k.x now what we can do is to remember that the sum over X is only related to this phase dot. So we say that we are going to find out what this sum over is, now we know already that this is an n qubit state and this is simply -1 to the power this.

So therefore this sum can be written as product over i = 1 to n and x_i which can take the value 0 or1, so $-1^{xi\ ai\ +\ ki}$ because that was my definition of the bit wise. So if you perform this sum over x_i = 0 to 1 just two terms here, when x_i = 0 it is one when x_i = 1 it is -1^{ai+1}, so that this sum over X contains this factor.

(Refer Slide Time: 23:45)



Let me just repeat this i = 1 to n, 1 + (-1) $^{ai + ki}$. So this is what some over x gave the other things of course remains the same. Now let us examine, let us examine what this term is like. Now notice there is a product of i = 1 to n, so accepting that $a_i = k_i$ for each i there would be one term at least for which this term which is being in the product one of the terms at least will be equal to zero. So therefore, the $a_i = k_i$ that is what will come out of it in the sum, so if I measure the first register then I determine what is this plain a_i with certainty.

So once again we have determined a classically I would not say difficult problem they are all reasonably simple problem in classical computing also, but what we have done is to find in a single query something which took many queries in classical computing. These three simple algorithms where illustrated to tell you about the inherent or an intrinsic power of quantum computation we will be later doing two major algorithms where the classical the quantum computer has a great advantage over classical computation.