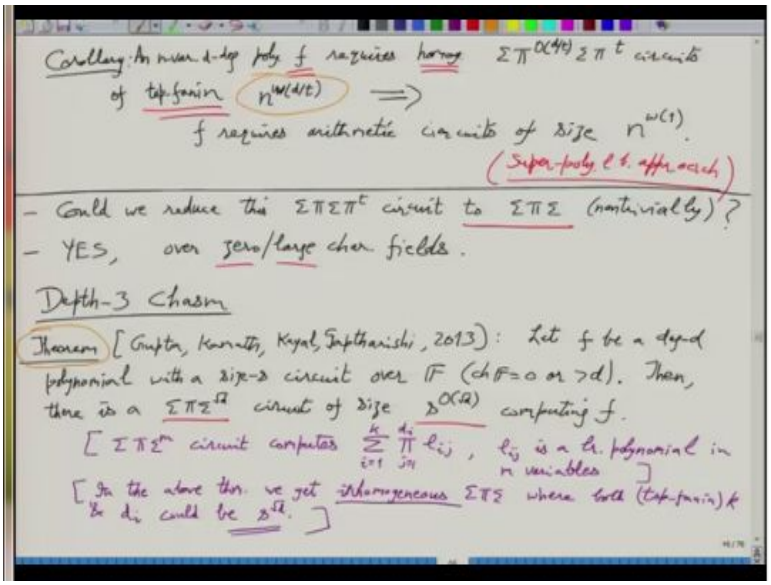
Arithmetic Circuit Complexity
Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture-11
Equivalence of Formulas and Width 3 ABP

So we are in the middle of this proof.

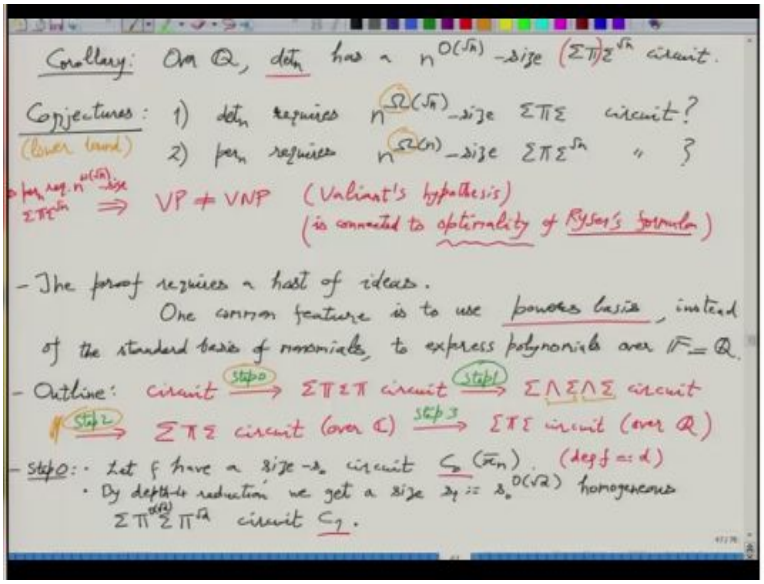
(Refer Slide Time: 00:16)



So we are trying to prove this theorem which says that if for a degree d polynomial you have a size s circuit, then you can squash that high depth circuit to depth 3 ok. So this will be a $\Sigma\Pi\Sigma$ circuit in the end, where the bottom Σ will only be adding $\sqrt{d}$ many things and overall size will be very large, it will be $s^{\sqrt{d}}$. But this is still interesting because it could have been s^{d}.

So this is much better than the trivial bound, so in a non trivial way, you can reduce depth all the way down to 3.

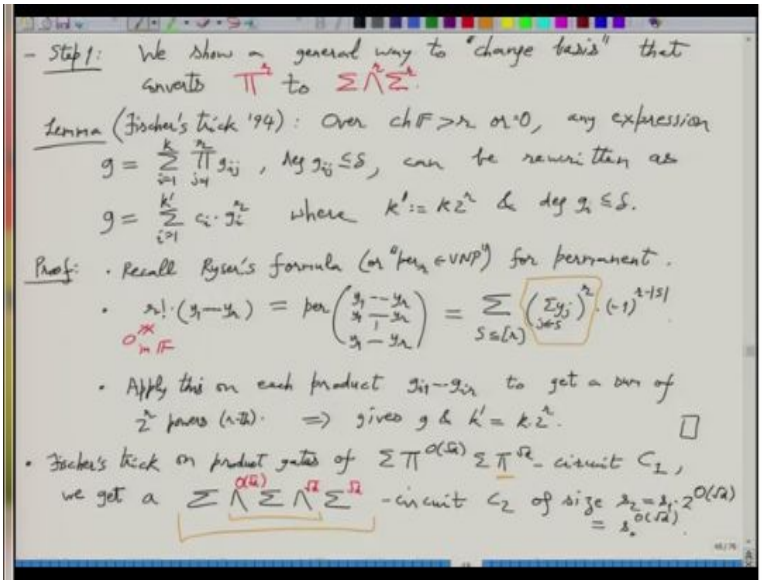
(Refer Slide Time: 01:10)



So this we will prove in kind of 4 steps, so the first step is step 0 is squash it down to depth 4 that we have already seen. Then step 1 is, replace these product gates by the sum of powers, it is called Fischer's trick. And this also we have seen, then step 2 is convert these powering gates or wedge gates to some other special form where it will be write a power as a sum of product of univariates.

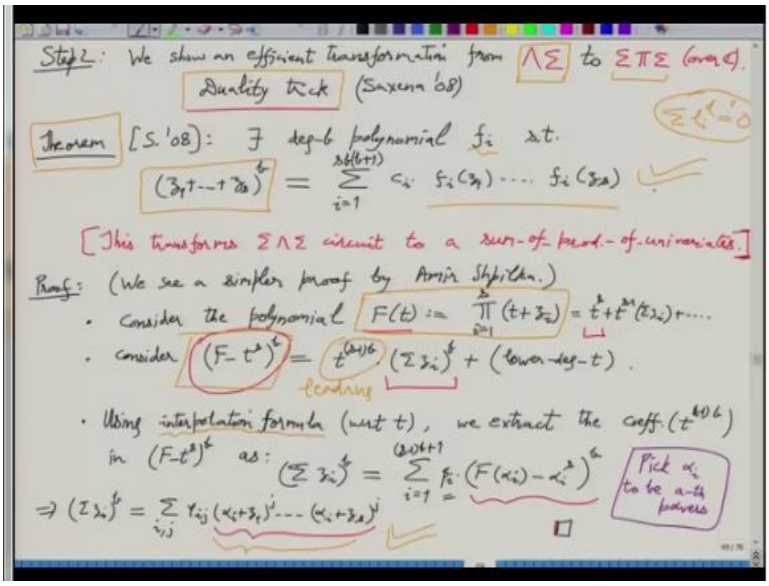
And since these are univariate polynomials, you can factor them over complex, so you can actually write power in this special form. Now it may not be clear why this is enough, but this will be enough, so let us go slowly through this step.

(Refer Slide Time: 02:22)



And then how this is used, so we want to convert Π^r to $\Sigma\wedge\Sigma$, well so this is already done that is clear and it gave as this $\Sigma\wedge\Sigma\wedge\Sigma$ depth-5 basically.

(Refer Slide Time: 02:44)



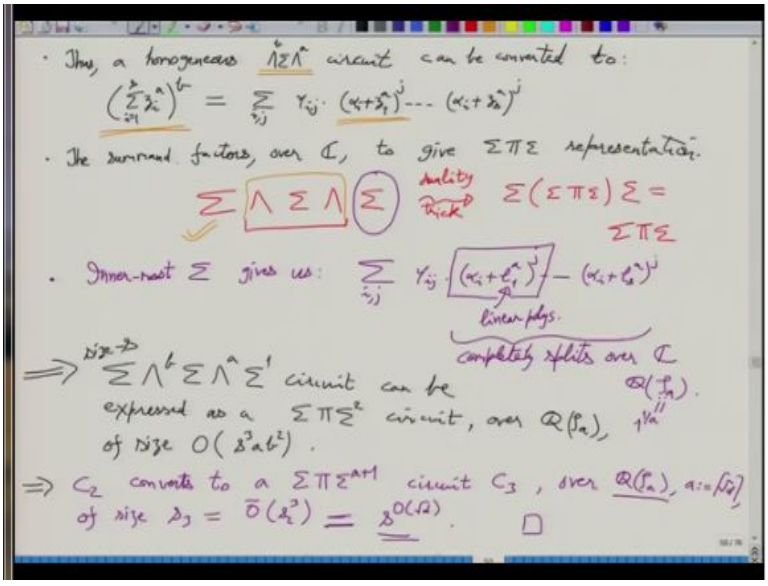
So, the step 2 actually I want to talk about step 2 which is this. So now $\Lambda\Sigma$ we can convert to $\Sigma\Pi\Sigma$ using this duality trick. So this duality trick is basically this identity that $(z_1 + \cdots + z_s)^b$ can be written as sum of product of univariates, where all these univariates are essentially the same, it is the same polynomial f_i, evaluated on these different s variables and the proof was just by looking at. So look at this auxiliary polynomial and from this extract the coefficient of t^{s-1} well

not exactly here, but here in this. So in this $(F - t^s)^b$ you extract the coefficient of t the basically the leading coefficient, extract the leading coefficient. And that you can do by interpolation. So if we have time in the end, we will work out the interpolation formula but at this point it is intuitive that if you have a polynomial of low degree.

Then very efficiently you can extract whatever coefficient you want, so that is exactly what is happening here. And that will be of this type , the dual expression here is what you will get, when you expand it out, so you exactly get this. So you actually can write $(\Sigma z_i)^b$ as sum of products of $(\alpha_i + z_k)^j$. This is the thing you should remember.

So this is a very simple identity but the point is that a priori it is not clear why we are doing this ,we are just reconverting powers, but what is the use of this.

(Refer Slide Time: 05:24)



So that will happen in the last step, this is where we stopped, so the beauty of this is here, now if you are given power of sum of powers. Let us say you are given this $(\Sigma z_i^a)^b$, then the previous identity will hold as well. So you will get these expressions, $(\alpha_i + z_1^a)^j$ and multiply these things and then add them, this small representation you will get.

And so that explains this much in the $\Sigma \wedge \Sigma \wedge \Sigma$ expression the middle part has been expanded this way. So, will it make any difference if you attach a Σ on top it would not. So this Σ is also fine because this just gives you more summands, that is fine. So all we have to now analyze is this Σ, the inner Σ. So what is the effect of the inner Σ is like replacing z_1 by a linear polynomial.

So you have now because of that, so this inner most Σ gives a continues to give a something easy. So this is just $\Sigma\ \gamma_{ij} \cdot (\alpha_i + l_1)^j \cdot\cdot (\alpha_i + l_s)^j$. So these are linear polynomials but this hardly changes anything because we can still factor this completely, $\alpha + l^a$ still factors completely over the complex field, why is that.

Well, whatever was the factorization with z_1 in those factors instead of z_1 just put l_1, so they remain linear factors. So this thing actually completely splits over complex or algebraically closed field, is that clear? So you get how many factors, so this summnd will factor into s times a many linear polynomials, it completely splits with. So do you need the full power of complex numbers here? what do you need from complex numbers. So that this thing completely factors, suppose $\alpha's$ and l_i's were all just integral or maybe rational, I mean, their coefficients are rational, then what is the number that you need in your fields, so that these polynomials completely split, for rational integer show, but even more special, so for this you will just need Q of let us say $Q(\zeta_{2a})$.

Because the point is, you also need those actually, no, so ok, what we can do, what these α, is let us go back, how did the α's come into the picture, we chose them. So here actually choose α's to be a-th powers because we just need in the interpolation formula we just need enough distinct elements in the field. So we can take them to be just a-th powers, let us say we take the first ath powers as many as we need.

So these are α's, so, which means that, so now let us come back here. So here α_i's are, in fact let us take $-\alpha_i$ to be a-th powers. So then here in this expression, since $-\alpha_i$ is an ath power, all

you require is ζ_a, ath root of unity to do this factorization. So ζ_a is just the a-th, a primitive ath root of unity which will be usually almost always it will be a complex number.

But these are enough. Is this clear? Since α_i's are negative alpha i's are ath powers to factorize this thing, you just need all possible ath roots of unity. And for that you just need 1 primitive root of unity that generates everything. So it is just a simple extension you have to go to $Q(\zeta_a)$. And there you have an expression, it is a $\Sigma\Pi\Sigma$ expression. So this discussion implies that more precisely $\Sigma\wedge^b\Sigma\wedge^a\Sigma$.

And in the bottoms Σ had just 1 variable. So let us call this 1 which is Σ^1 circuit can be expressed as a $\Sigma\Pi\Sigma$ circuit, where the bottom Σ has fan-in 2. If this can be expressed as a circuit of depth 3 over $Q(\zeta_a)$ of size, so who can mention the size after all these transformations? Think there is something like $s \cdot a \cdot b$ and slightly more than that, $O(s^3ab^2)$ size.

So, let us not worry too much about this exact bound, ultimately it will be polynomial in s and d and b given the original thing size s. So size s depth-5 circuit is brought down to depth 3 of similar size, do we agree? Any question here? And then implies that. So let us after this detour, let us go back so what was the circuit that we had developed in the previous step. So that was C_2, C_2 was this depth 5 circuit, where these powering fan in are $\sqrt{d}$, this is a and b.

And the bottom Σ is also $\sqrt{d}$, so let us now depth-3 using the reduction we have just seen and let us call that circuit C_3. So we can now write that C_2 converts to $\Sigma\Pi\,\Sigma^{a+1}$ circuit well a $\sqrt{d}$, let us keep $\sqrt{d}$, converts to $\Sigma\Pi\,\Sigma^{\sqrt{d}+1}$, C_3 over $Q(\zeta_a)$, I want a. So this is, a + 1, and a is $\sqrt{d}$, let us take the ceiling.

So depending on this bottom fan in you have to go to an extension of you have to go to a number field this number field and there you will get a circuit C_3 of depth 3 that much fan in bottom fan

in of size s_3 which is slightly cubic or slightly super cubic $O(s_2{}^3)$, s_2 was the size of C_2, is that clear. So this is what we wanted to show, in fact we have shown more we have shown something stronger that by going merely to $Q(\zeta_a)$.

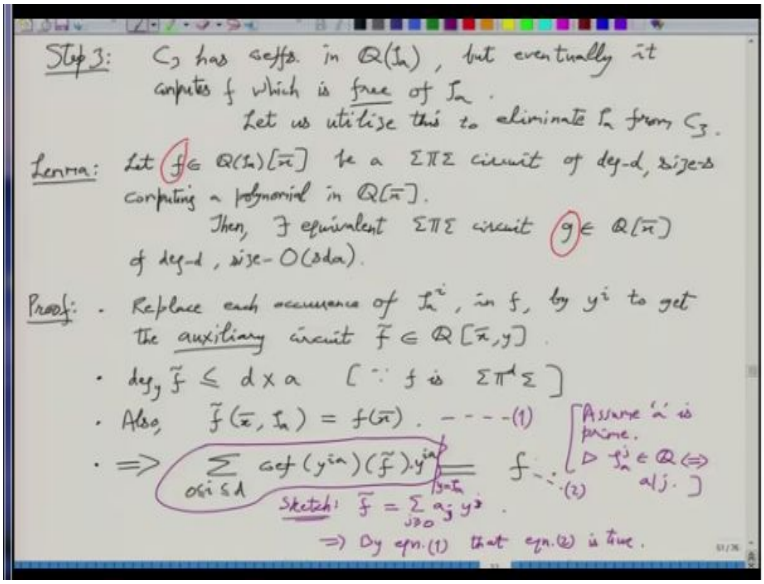
Any depth any general depth circuits C_1 or C_0 we can bring down to depth 4 and then depth 3 and even the fan in bottom fan in will be just $\sqrt{d}$, where d is the degree. But now it will have coefficients from the number field $Q(\zeta_a)$. And what was $s_2{}^3$ in terms of the original sizes, so the biggest price we paid was when we came to depth 4, so that is $s^{\sqrt{d}}$ so that is the result.

So size s degree d polynomial comes down to $\Sigma\Pi\Sigma^{\sqrt{d}}$ over $Q(\zeta_a)$ and size blows up to $s^{\sqrt{d}}$ which is also the fan in of the product gate here. So actually the depth 3 circuit is multiplying a lot of things, each of the factors have kind of small sparsity, so it is you are just adding around $\sqrt{d}$ many things then multiplying a lot of them and then adding a lot of them and that gives you a very low degree d polynomial f.

So this whole thing is very counterintuitive, but it is true, so you can come down to depth 3 in an non trivial way, any questions? So all that remains is a slight technicality that since your original polynomial was rational, I mean it was it had rational coefficients and this C_3 has irrational coefficients. So can you now convert can, you also make them rational, if you want to compute a polynomial that is rational.

Then why do you need to use these $\zeta's$, can we eliminate the $\zeta's$ also, so that will be the last step that will so that is for the final step 3.

(Refer Slide Time: 20:03)



So C_3 has coefficients in the number field $Q(\zeta_a)$, but eventually it computes f which is free of the ζ_a. So let us utilize this to eliminate ζ_a from C_3, how do you do this. A circuit that is using elements from a field extension to compute something in the base field, how do you eliminate the use of this field extension elements, why do you think it is unnecessary?

"Student-Professor Conversation Starts": Gauss Lemma. **Professor**: oh Gauss lemma? **Student**: To show that it is unnecessary,at least for this Q . **Professor**: oh Gauss lemma is very pedestrian. We have come much further ahead of Gauss. **"Student-Professor Conversation Ends"**

So, what we will do instead is to look at coefficients of the polynomial. So actually rip the wherever in the circuit you are using ζ_a which includes also powers of ζ_a, you replace ζ_a by just a formal new variable. So when you do then this will of course not compute the polynomial f now, it will computing something else because you have constant ζ_a by formal variable.

So actually the number of variables have now increased. It has become n + 1 now, so, you are now computing some auxiliary polynomial and from that auxiliary polynomial you can recover f ok. So we will do that you can recover it by interpolation and that auxiliary polynomial is free of

ζ_a essentially by definition, because wherever I mean you can see the circuit so you can see where ζ_a constant is being used and just replace them by y.

So now the circuit that you have is actually, it only uses rational numbers. Nowhere is it using anything up beyond Q.

"Student-Professor Conversation Starts": Then you could just set y to 0, right? Why do you have to interpolate? Professor: y to 0? That is a bad idea. You are saying that ζ_a is 0. You cannot say that, let us replace number 1 by y and then set y to 0, that is like killing all the constants.

"Student-Professor Conversation Ends"

So what you will do is you will have to extract y^a, y^{2a}, y^{3a} $\cdots$.

Because these are the powers of y that gives something rational y does not give anything rational because it is ζ_a, it is irrational and y^2 may not also. So the first time it will give something rational is y^a and then powers of that. So these terms you will extract and the sum of these is f. It has to be. So the sum is what you want to extract, so that becomes a lemma. So let f be a polynomial over this extension, irrational extension and variables be a $\Sigma\Pi\Sigma$ but it could have been any circuit that I think is not important.

This lemma has nothing to do with depth 3 of degree d size s, I am calling f as the circuit, let me do that. So f is the depth 3 circuit, computing a polynomial in $Q[\overline{x}]$, we are looking at a circuit that uses higher constants, but ultimately computes only lower constants as coefficients. So then these higher constants can be removed, so then there exists an equivalent $\Sigma\Pi\Sigma$ circuit g which uses only rational constants.

So all the intermediate computations are in Q x bar of the same degree size, so that is the conversion from f to g without much cost. So, the idea is simple, it is just that replace each occurrence $\zeta_a{}^i$ in f by y^i to get the auxiliary circuit $\tilde{f}$ which actually has only rational

constants, but has one more variable ok this extra variate rational polynomially. Now you have an expression for this $\Sigma\Pi\Sigma$.

The size has not changed, it is actually the same circuit essentially the same circuit, what can you say about the degree with respect to x's? So, that has not changed the intermediate degrees have not really changed or the upper bound remains the same what is the degree with respect to y, how bad can y get in terms of degree. So the original circuit degree bound was d, so it is d times what is the highest i in $\zeta_a{}^i$. The highest is a.

That is the bound d times a, so since f is $\Sigma\Pi^d\Sigma$, so the product that happens this product gives you degree of y not more than $d \times a$ which is small. So the degree in terms of y is very small. Also that is the important thing that from $\tilde{f}$ you can recover f by substituting y equal to , well the definition, y = ζ_a. So $\tilde{f}$ knows about f, how do you use that.

So basically use the fact that this $\zeta_a{}^j$, it becomes rational if and only if j is divisible by a, well this may not be true. They, by itself, is not true, but I do not need this. I want to say something else. I think my argument should not be based on this because this is wrong. Well, what I want to conclude is that if you look at the coefficients of y^{ia} in $\tilde{f}$ where i varies from 0 to d, this sum is equal to f:

$$\sum_{0 \le i \le d} coeff(y^{ia})(\tilde{f}) = f$$

Do you believe in this? I think, easiest argument would be to forget the model, just look at a simple polynomial, let us look at the polynomial $\sum_{i=0}^{a-1} b_i \zeta_a{}^i$. Can we assume that a is a prime number, then it will be easier. Otherwise we have to make we have to concoct other concepts, it will not help, let us just assume that a is prime. So when a is prime the properties that $\zeta_a{}^j$ this will be rational if and only if $a \mid j$ then it is true..

For example, if a is 3 then 1, ζ_3 and ζ_3^2. We look at this out of these 3 only 1 is rational, both the other 2 things are actually complex. So forget about Q, they would not even get real. So let us just continue with Q, so $\tilde{f}$ you can view as a polynomial in y, a_j these are the functions in x.

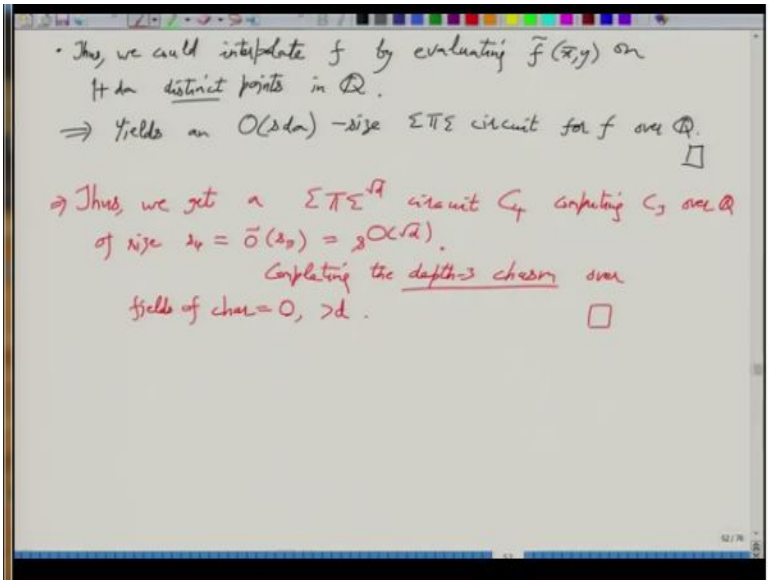
And in this polynomial when you substitute $y = \zeta_a$, the ζ_a^j's which will contribute, let us say real terms. They will exactly be the j's that are multiples of a, all the others will contribute complex, certainly irrational terms. So these we will ignore ok these we drop and the thing that remains is this sum after dropping and this has to be equal to f, this has to be equal to f because of equation 1.

So this implies by equation 1 that this second equation 2 is true. Do you agree? Because equation 1 RHS is real and in equation 2 when you look at the LHS, the LHS is the part that is contributing all these real coefficients. So actually if you want, if it helps, you can put also y not more, because it is already 1 and this is what you are evaluating at $y = \zeta_a$.

So it is actually this part of $\tilde{f}$ where you are substituting $y = \zeta_a$ and so the sum of these coefficients will become equal to f, all the other parts of $\tilde{f}$ do not contribute anything. All they can contribute is 0, if the other part contributes something non zero, then that complex thing cannot be cancelled by the real things and f is that real, is that clear. So that is the once you have this equation 2 what would these coefficients of f tilde that you do by interpolation.

Because you have a degree bound with respect to y, individual degree of y is low. So you can **so** just for every coefficient you will do interpolation. So that is the summand, of number of summands that you get is around d a and on you will do it d many times, so you get a d^2 blow up.

(Refer Slide Time: 37:12)



So all that is within our budget, thus we could interpolate by evaluating $\tilde{f}(\bar{x},\bar{y})$ on $1+da$ distinct points in Q. So this yields $O(sda)$ size ΣΠΣ circuit for f over Q with this finishes the proof. So you start with a circuit, you bring it down to depth 4, then you bring it down to then you take it up to depth 5, then you bring it down to depth 3, but you have gone to another, to field extension, to number fields.

So then you bring down the number field ok, so that is your final depth 3 circuit, so thus we get $\Sigma\Pi\Sigma^{\sqrt{d}}$ circuit C_4 computing C_3 and hence the original polynomial over Q. And it has size $s^{\sqrt{d}}$ so completing the depth 3 chasm over fields of characteristic. So 0 is clear because 0 is essentially rational proof, what other characteristics have we covered.

So the only place where we needed characteristic was this Fischer's stick. So they are the monomial that you convert the root of unity place also will. There is no root of unity it is expression over Q sorry change of characteristic, then at some point we needed a ath root of unity, it is an extension. We did we never had a ath root of unity we went up yes.

So the monomials that appear in your depth 4 representation those monomials should have degree less than the characteristic. So to put it simply, we can just say greater than d, then it will certainly work. So the depth 3 chasm is there, as long as the characteristic is either 0 or more

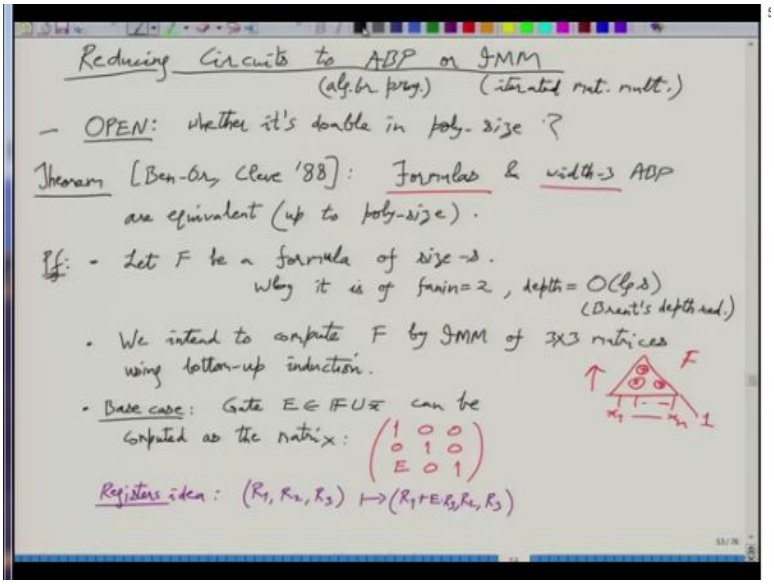
than the degree of the polynomial you intend to compute, more than this we cannot say because if the characteristic is even equal to d.

We do not really know what are we getting in the depth 4 representation maybe in the depth 4 representation we are getting some monomial of degree d and there is some product gate in fact which is computing that monomial. So when you try to do Fischer's trick it does not a provably it will not work, there are examples. So that is all we can say ok, so that is the limit of depth reductions all the way to depth 3 and decent characteristic fields.

Any question at this point? What is a is an upper bound just look around you find the nearest prime, there are lot of them that just an upper bound so here I will just say since I am on record that pick a prime. I do not want to give you other ideas ok, well. So then we will move to something quite different but still highly interesting, which is one way to reduce circuits is the depth reduction which is quite visual.

Question that you can if you wanted to convert a circuit into an algebraic branching programme or let us say iterated matrix multiplication. So we can try to convert a circuit into that as a product of matrices, so what would be your first answer.

(Refer Slide Time: 43:23)



So, reducing circuits to ABP's or IMM's weight algebraic branching programme which is the same as iterated matrix multiplication. So this actually is an open question, how to do this efficiently in poly size and the problem is that well in small circuits you can compute determinant of course that was the first major example. Determinant has a simple, has a small ABP that we know.

But what is hard to say or what we do not know is, are there other interesting polynomials or functions in that have small circuits but may not have small ABP's. And also when you look at the definition of circuit, it is not clear how can you convert it efficiently into an ABP. A circuit seems to be doing more than what an ABP can do , in this was, I mean this could have happened even for determinant, when we look at the circuit of determinant , it was definitely not clear whether the circuit can be converted into an ABP in an efficient way. It was by chance that we got a small ABP but completely different argument. So there might be polynomials which do not have small ABP's but have small circuits that we do not know. So we will not attempt this reduction, something weaker which is formula, let us only focus on formulas.

So this is an old result by Ben-Or and Cleve it says that formulas and width 3 ABP's are equivalent models always up to poly size poly blow up. So we instead of studying circuits general circuits we will study formulas, so fanout is 1 and we will show that will actually converted into width 3 ABP ok width 3 means that the matrix multiplication is for just 3 cross 3 matrices which should be a very surprising thing it is a very surprising thing.

So the in those days I think the motivation for them to do this was suppose in your computer you have only 3 registers. So 3 registers means that there is some information, let us say in register R_1 and R_2 and using that you can compute something and store in R_3. So your memory is restricted to only 3 registers, so using 3 registers, what kind of polynomials can you compute? It seems to be a very, very weak model.

So if you try to implement circuits, whatever a circuit is computing in 3 registers, it looks impossible I mean even assuming fan in too fan in and fan out you can assume to. But still doing it with just 3 registers looks difficult. So what they showed is that, if you have a formula that I have all of that can be implemented using just 3 registers and kind of I mean you can find the proof that it cannot be improved.

So you cannot do it with 2 registers, so width 2 is not possible but width 3 is enough, so let F be a formula of size s. And we can assume without loss of generality it is of fanin 2 fanout is 1 of course and what is the depth, depth is $\log s$, why can we assume that, that is why formula depth reduction ,Brent's proof basically. Let us now implement this using 3 registers or 3 cross 3 matrix product.

So whatever you will do in 1 step with 3 registers that you can see as a kind of matrix transformation. And when you do that thing then it becomes a product, so the sequence of this iterated matrix multiplication is really all these operations you are doing on 3 registers. And in the end you want to compute the polynomial, so we will do it by induction ok. We intend to compute F by iterated matrix multiplication of 3 by 3 matrices using botto- up induction.

Bottom up means that in the formula start with the leaves and then go to the root, so we are moving upwards, and the leaves are here ok. So what is the base case, base case is your gate is simply a variable or a constant. So you may also have a constant here, so gate E which is either a field constant or a variable. So this gate can be computed as the matrix, can you guess because I cannot guess. Yes it is basically it is not just the base case.

But it is the whole representation that we want to fix, so in the formula there are these gates addition, multiplication gates are there. And what we are saying is we will start with the lowest gate and then we will slowly move upwards covering the gates in a layer and then move upwards. So whatever a gate is computing, that should be written as some matrix , I mean it will be a matrix product.

But then what is it that the matrix product computes, it computes a 1 single 3×3 matrix and which entry in the matrix is your polynomial computed at that gate, so we actually want to fix that representation here, so to make the proof work will fix this representation ok. So it is basically the identity matrix with the your polynomial E sitting at the corner ok, there are 2 corners available, so we have take the bottom one ok.

So this will be the representation of a gate, in fact at any point of time in any intermediate step the invariant would be that when I look at the matrix corresponding to this gate. And this gate is computing polynomial E or expression E, in the matrix that E would be sitting here ok in the 3, 1 in the bottom left corner this E would be sitting and we will get a matrix products for that computes this.

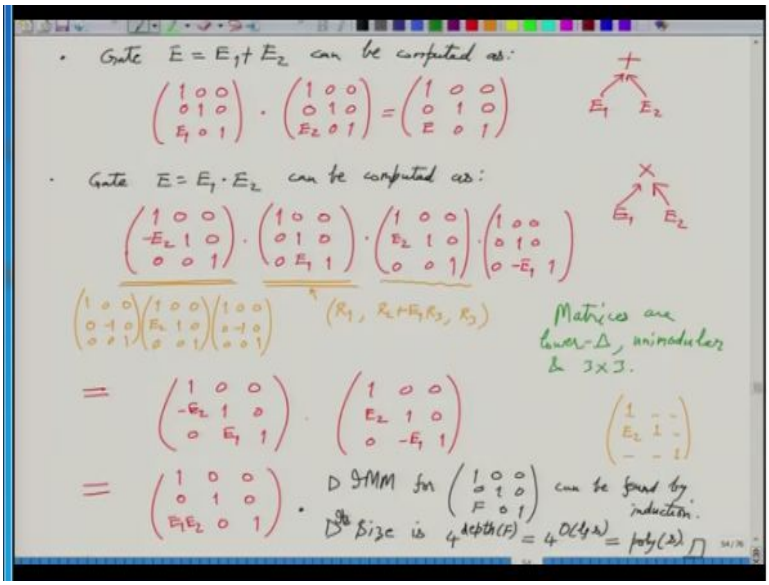
So that is the base case, so you can put variable or constant here whichever gate you are at in the leaf. Maybe I should also write the idea in terms of register. So register idea is, so this is kind of the transformation that if I have 3 registers $R_1,\ R_2,\ R_3$ what this is doing is. It will not change $R_2,\ R_3$ it will use them to come to change R_1 and what will R_1 become, will become this.

So this I mean if you want lower level implementation of this what it is doing to 3 registers is that it uses the last 2 registers and in fact it uses the last register R scale set up by E and then adds to the first register ok. This you get by multiplying this matrix on the right side of your register vector ok. So this is just right multiplication that gives you this, is that clear, I mean this is not very important.

But maybe gives you some kind of an intuition and as you are assuming that our register stores polynomial in this case. So it is actually storing a function, It is not storing on off or 01 this is the algebraic model. So here a register actually stores a function, so it starts, so you start in the register with simple functions like x_1 or 10 and then using transformations which are also very

simple, you will slowly build complicated functions in a register. So it is a very natural thing to do in this sense.

(Refer Slide Time: 56:30)



So now let us go above the base case, so first thing is addition, suppose you have an addition gate in the formula feeding in E_1, E_2. So this can be computed as, so R representation for E_1, E_2 or fixed. So E_1 will be 1 0 0 0 1 0 E_1 0 1 and E_2 would be computed as 100, so these are the 2 things for which you have transformation. So, you have these 2 matrix products and you want to sum these bottom corners.

So, **the** representation is so nice that the product will give you $E_1 + E_2$. And I have no justification except multiply and be convinced it indeed works. So $E_1 + E_2$ is E, so once you have matrix products for E_1 *and* E_2 then concatenation of that gives you the sum. So in the again in the intuition of registers, the registers will first work hard to compute E_1.

And once they have computed E_1 in the first register, then they will start working on E_1 to produce to add E_2 it is going from left to right, in terms of registers ok. And finally product that is even more complicated, so how do you compute $E_1 \cdot E_2$ using such matrices. So again E_1, E_2 is given to you as before and you have already done the only thing you could have done that is multiply.

So what else can you do to get $E_1 \cdot E_2$ now in the bottom left corner, sorry no all you can do is multiply and use kind of the symmetric operations. So one thing you have to remember is once you have this matrix corresponding to E_1, since you have 3 registers right you can shift your focus you can say that ok I rotate the registers. So R_2 becomes R_1, R_3 becomes R_2 and so on and then the matrix will also correspondingly change in a simple way sorry that is also this multiplication via different matrix.

So once you have this E_1 matrix you also have these kind of symmetric or analogues matrices are also there, where you change your point of view you call the ok, now this is the second register is my first register, and third is second and first is third and so on. So those matrices are also there, so what do you have to wonder at this point is well I can maybe use some of those and the corresponding once for E_2 and multiply them.

And if you think about this hard enough, it will work, so you will get this ok. So these are the 4 matrices. This one I mean you are basically interleaving the E_1 variants with the E_2 variants. And can you see that each of these variants is essentially of the same form as the base case. So for example, if you look at, so if you look at this one the second one you can ask the question, oh this is not in that representation.

So E_1 should have been in the bottom left corner, but why is it in the middle. So how will you explain that? Or let us just see at the register level, because that will be very easy to see. So what this is doing is, it is keeping R_1 the same, $R_2 + E_1 R_3$ and t. So instead of putting values in R_1, it is actually putting values in R_2. But they are of the same type it is actually multiplying something else which is R_3 with E_1 and then adding it to R_2.

So it is just a permutation of R_1, R_2, R_3 but the algebraic operation is the same, it is an analogue operation. And that the same thing holds true for the other 3 also, what about the – sign, how do I

explain the – sign. So if you have E_1, how do I get – E_1, this should be inversion, can you get 1 $\times$ 1. So how ok again probably do I say.

No it is not but at the register level, somebody tells me how to add $E \cdot R_3$, how do I learn $-$? I mean each of these operations is multiplying by to the left and right by the permutation matrix. So all of those are legal operation, you can just expand out this 4 way product as like an 8 or 9 way product, where you use these you use your representation matrix a few times and then the other matrix are the symmetric matrices.

So then your matrix multiplication, but from 4 ways is a many way, so then that way you can it is some work though, because to get to a - 1 you will have to do like at least 3 matrix multiplication setting. And one of those I mean a couple of those to be multiplication by something like - 1 1 1. So how are you achieving it by 3 matrix, so this sign still needs some work, so that I maybe I will mention it next time.

But these 2 we have covered, so willing to get now to say that it is like $R_1 - E_2 \cdot R_2$ something like that, no, no that it is but then given long matrix product for E, how do you get 1 for – E, oh so you can first multiply first column by - 1 then ok left last row - 1 and then the entire matrix where - 1 ok. So that is the solution, that is the solution then again the middle column by – 1.

But it you can do it like a ok, so fine, so let us do it, so this one is, how do I interpret this one. So I interpret it as follows: let us, I mean this you have already seen that I can get to this point from E_2 I can get to this I will just permute R_1, R_2, R_3, I will get to this but now I want to scale it. So what I will do is, I will first scale the second row, so how do I scale the second row by – 1.

I want to multiply the second row by - 1 so, first column and first row, so I will do this. So this has a scaling effect on the second row and now I will scale the second column ok oh it is many ways ok that is why you should do it. So second row and the second column when we scale simultaneously with minus 1, magically we see that it only changes E_2, so you get - E_2.

So that is the thing, so this actually is a slightly different calculation. It is not just register permutation but also these simple scalings fine. So from E_2 we can get to $-E_2$ and then we can put $-E_2$ wherever we want at least up to permuting the registers fine. So each of these can be produced each of these 4 and what is the product . So the product will be, let us multiply the first 2 and then the last 2 and this will be.

So believe it or not, you will get this, so E_1 it will you will get in the bottom left corner when you multiply these 2 why it is it takes 2 matrices like if you take E_2 to be a position like sorry E_2 into 1 let us suppose there is a matrix in E_2 is here ok diagonals are 1 1 1 diagonals are 1 ok. And this is 0 and the second matrix to be like 3 2 to be E_1, if you do those calculations.

You will get this intermediate one of the matrices I have written no are you saying that the signs are unnecessary yes why do we need those things, why did I put signs take the product of no, no but the sign is needed because look at the second row with first column multiplication. Second row with first column multiplication needs $-E_2 + E_2$ to cancel them.

And look at the third row with second column that also needs E_1, $-E_1$, so I mean you can consider all the ways. Now you know, what is the space of ways and trust me only this one will work, so fine. So now it is amazing that once you have a matrix product for E_1 and a matrix product for E_2 using some complicated transformations, you can also get a product for a matrix product for $E_1 \cdot E_2$, where $E_1 \cdot E_2$ sets in the bottom left ok so that is the hardness.

So this product case is actually harder than the addition case. Because it is very hard to get multiplication only in one entry of the matrix, the other entries remain untouched. It is only this 1 entry which is being yes . So that was, so these are the induction steps, you will do it again and again for every gate when you do this, is actually blowing up the size by 4 times. So, that should give you 4^s s which is bad.

So how do you analyze this properly, well do it by depth, so in 1 level, actually you it is giving you 4 times. So the previous level, the bottom level let us say you have produced the values of all these gates. And when you go to the next level, that size will blow up 4 times no, no, no, no, 3×3 , this is 3×3, we never use anything other than 3×3 matrix. So width is fixed by design, the only we are talking about the length how many matrices are being multiplied.

So when you go from 1 layer of the formula to the next, then the length of the matrix products becomes 4 times, basically because of this $E_1 \cdot E_2$,4 to the yes exactly. So what you get is iterated matrix multiplication for 1 0 0 0 1 0 and formula F 0 1, this can be factored can be found by induction and the second thing it size is $4^{depth(F)}$ which is $4^{\log s}$ which is poly(s).

So this matrix with the formulas in the bottom left corner, this can be factored into poly(s) many matrices they will be 3 by 3 matrices and give you exactly this matrix, so this is your width 3 ABP. So I mean once you have identified that F is sitting here in the bottom left corner, you can extract it because you can multiply by a row vector on the left and a column vector on the right.

So this vector multiplication on both ends will actually extract out F. So this is the definition of width 3 ABP, is that clear. And the practical kind of interpretation is physical interpretation is that using 3 registers, you can compute any function that has a formula, it just 3 registers are enough. It gives you enough flexibility to move things around in 3 register starting with trivial case base case and there are other beautiful things here.

So these matrices they are all lower triangular and they have determinant 1, so these are actually uni modular triangular matrices. So they are all matrices are lower triangular uni modular 3×3, so it is a very special product. So formulas have very special width 3 representation and yes that is one way we have converted formula to width 3 ABP. So tomorrow we will do the converse that will be the actually it is almost obvious how to do that. It is an easy solution and then we will talk more about width 2 ABP.