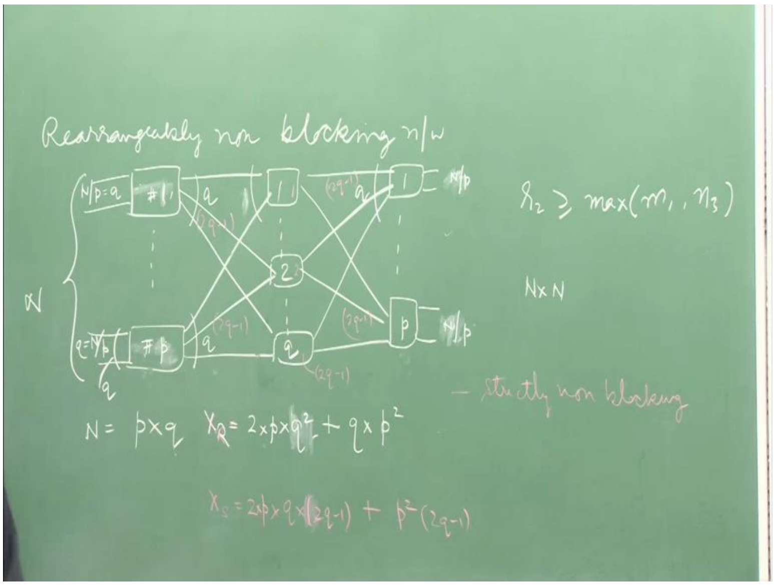
Digital Switching
Prof. Y. N. Singh
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Lecture - 10

So, we will start with now the issue of cross point complexity of rearrangeable as well as step in on blocking cross network, but this time what we will do is, we will not actually compute, this thing for a simple cross network. Actually I have done this thing for a simple sticking on blocking cross network earlier, but now we will do what we call recursive construction of these switches. We can actually further optimize, and then we have to find out what is the most optimal configuration that, and actually it is very difficult to find out an optimal configuration, but since we are looking for a bound, any value is going to be fine. So, I am going to look at, whatever minimum I can achieve, but there can still be better bounds, that room is still there, but whatever has been so far in the literature that is what we are going to look into.

(Refer Slide Time: 01:15)

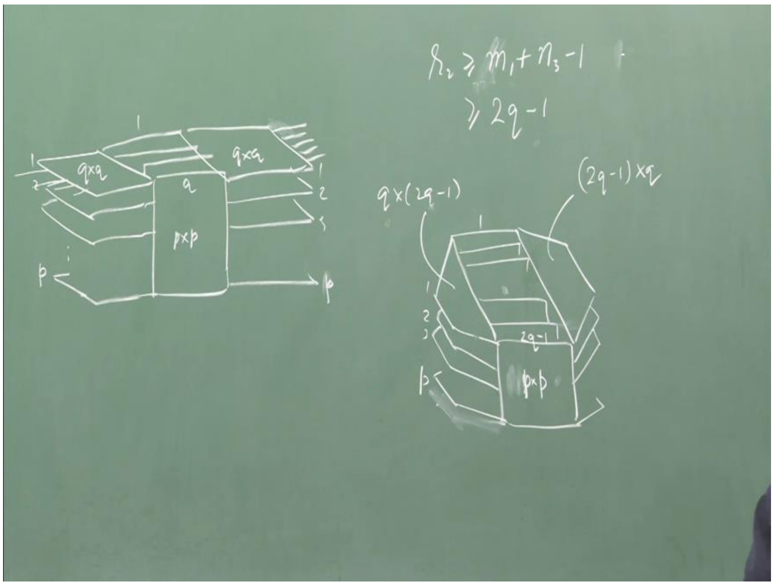


So, if you look at cross point complexity, this recursive construction. So, first of all let us take rearrangeable network. So, there will m 1 by n 1, and we will have some middle stage which is, and we already know the definition that by so far, is in happen to be theorem that your r 2 has to be greater than equal to maximum of m 1 or n 3; that is the condition which has to be satisfied. What do we look, is in this case. I am just going to change my variable, so that they have become consistence with notes, but I think you should be able to understand the swap. Now instead of using m by N I am going to use p by q. The idea here is that I am now worried about total number of ports, and I want an optimal configuration for this ,and this N can be now broken into 2 numbers; which is p by q. Once I do this; so, for N by N switch, because this dimension has to be kept fixed. I have to build up a rearrangeable non blocking switch, and find out what is the cross point complex, there is the idea. So, in this case, what we will do is, I am also going to make an assumption. Let me change the dimension here, this p by q. So, this actually means number of switches which will be present, will be q, 1 to q. Now each of the inputs will be N by p sorry N by q N by p sorry. This has to be p, this has to be q, and I am talking N by N switch symmetric calculation.

I am not worried about a symmetric state as of now, because I am looking at a bound for N by N. For N by N cross bar it is o N square. So, these are typical configuration. So, this is again q number, this number is going to be p, so this will also be N by p. So, q number ports which are coming out. So, number of cross point here will be pretty simple, it will be p into q will be number of cross point here, and total p number switches are there; its p, that state this is actually similar kind of structure gets repeated here.

So, you multiply it by 2 plus get q switch is here, each of them will have p by t p square, is q into q, no; p cross N upon p. How many inputs are coming, N cross p cross N cross p should be there; p square q. I think I have to restructure it, I have actually reversed, this value I have made a mistake here. This is q by q, so this has to be q square actually. This N by p is nothing, but q inputs, so q inputs q outputs, so these are q square complexity. And there p number of switches multiplied by p and 2 times, each switches p by p switch here. So, its p square q times. In my notes, it was actually reverse; that is why the confusion actually happened.

(Refer Slide Time: 06:43)



And in checked I can draw this something like this. So, this is q by q each one of them, 1 2 and so on, and you have total p switches like just put boards like this. So, there are q inputs and q ports going on, and then I can create at cross connection and so on. So, it is 1 2 q each 1 of them is p by p. So, these are q ports coming out, and here q ports going in. So, you can actually look into this kind of structure.

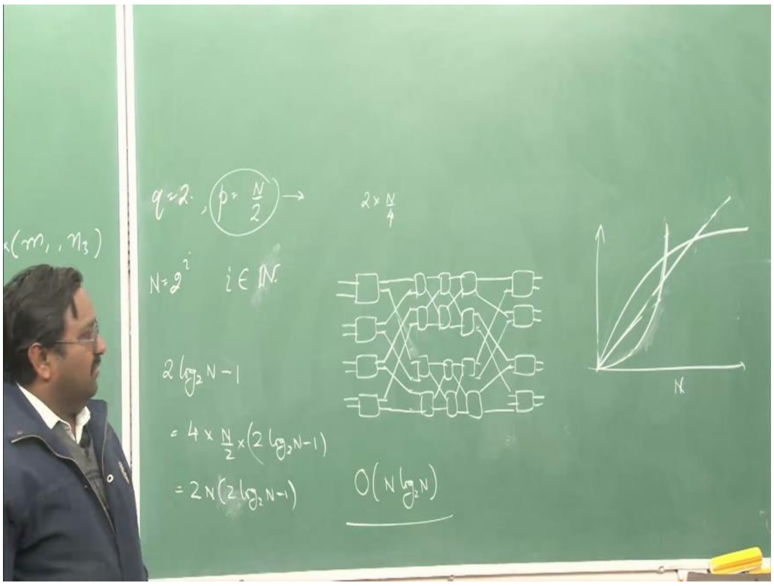
So, I am doing exactly this, in 3 dimensional spaces. This crisscross is you need not to remember. And for stick sense non blocking I can also draw the same diagram, expect in this case. Now what will happen is, the condition is r 2, has to be greater than or equal to, m 1 plus N 3 minus 1, and since I am looking into that p by q kind of split. This value will be how much; numbers of ports are q; so 2 q minus 1. So, if you make equivalent figure, just for memory sake, you are expanding actually. So, 1 2 3 and you will have, total p switches going here. Number of switches which I staking here 1 2 and so on, and the last one will be 2 q minus 1, and this will be a p by p switch, and this will be an expansion of q inputs and 2 q minus 1 outputs. Same as going to true here on this side; 2 q minus 1 inputs q outputs. So, that is state non blocking switch.

This one, this switch is a N by N switch. I am now factorizing into p by q. Some numbers, see I have to factorize then only I can create state non blocking switch. Then of course, once I do this p by q, I have to find out what is the cross point complex. I am trying to indentify getting hint, how I should divide b and q, what value of p I should. I can take they are many options of factorizing, which option I should take which will give me the least value. Minimum values of this cross point number of cross point in this switch, whole switch matrix. So, if I take N by p is equal actually after this. So, it is a q by q switches, I only require q out q by q here for rearrangeable non blocking configuration. If it is non blocking configuration, this will be 2 into q minus 1. So, this switch is equivalent to this sum. When you want to draw equivalent to this one, you will make a change, and in that case. So, this will be 1 this will be 2 and so on, this will be 2 q minus 1. So, I am drawing with a different color. So, this color corresponds to, strictly non blocking configuration. So, this value instead of this will be 2 q minus 1, this value also will be 2 q minus 1. This still remains same p p; only this will change 2 q minus 1, rest everything remains same.

So, in that case also you can find out the cross point complexity if you wish. These are cross for rearrangeable, these are cross for strictly non blocking. So, for this what will be the value. So, there cross point complexity q into 2 q minus 1. This happens p time, there are 2 stages symmetric, and you have p square 2 q minus 1, but I have still not recursive construction. In fact, each of these blocks can be further reconstructed by, again using a 3 stage network, and we always know if I go to a 3 stage network from a single cross bar, I reduce some cross point. So, number of cross point will get further reduce the each one of these stages, each 1 of these switching is further broken down into fundamental; that is a recursively I will keep on doing it.

So, I have to find out a formula and do the recursive solution, it will form a series and we will get a solution from there. So, now coming back to first of all strictly non blocking. Here you can see what will be the form. So, maximum contribution comes from this thing. So, I would like to have q as small as possible. If I keep q as small as possible my cross point will be less, this is clear. And when I recursively take each element and further do further break it, it will, because I am anyway taking it, so every time I will keep on doing it to be minimizing. So, I know what is going to be a minimum value. Here it is not where is state forward, what would be the minimum value, here we will come later on, because if I try to minimize on this other side actually expense. So, number cross point optimum does not happen when q is minimum. Here it is going to happen that way actually. So, that is the hint, and then from there we can further solve it.

(Refer Slide Time: 14:56)



So, in this case, you can always take q as 2 . So, once q as taken as a 2, what will be value of p; N by 2. 2 is the minimum value possible, and I am taking N as 2 raise power i. Well that is usually is the simpler thing to do that N will always assume to be 2 raise to the power i, where i is an integer. In fact, i should be a natural number i should not even call it as an integer. So, that is the condition which usually will be taken. So, in this case. So, what will be the cross point complexity. Now this p can be further broken down, and I will get 2 into N by 4 configurations. So, I can keep on doing it actually this way. So, for example, say lets I want to build up a 8 by 8 switch, how I will do it actually. So, 8 by 8 is. If I take q is equal to 2 and p is equal to 8 by 2 which is 4, which actually means the switch will be of this kind, and I have got now 4 by 4 switches here.

Can you remember something from this, I have talked about this earlier in one of the lecture. We are essentially going to that. I can now further do this 4 by 4 can be broken, this is real (( )) non blocking. So, if I represent. I also make 4 by 4 by rearrangeable non blocking, which will remain rearrangeable non blocking. In earlier construction what I have done was, I did something I never told you that, I just only stated rearrangeable non blocking, but now you know why it is going to rearrangeable non blocking, what is the reason, because that condition is always going to satisfied; that r 2 has to be greater than or equal to maximum of m 1 and N 3. So, this 4 by 4 further can be actually done, or I can replace this by angle 3 stage network. So, same thing will go here, same structure. These are recursively recursive construction. So, every time in the end how many stages will be there, if I keep on doing it, and from there I can find out cross point complexity. So, in this case you will find out, you will require 2 log 2 N minus 1 these many stages, if you count; each 1 is a 2 by 2. I am assuming N is equal to 2 raise power i. The reason for this is, when I did first division, I got first stage and last stage, middle stage was the larger switch. So, I got 2 stages plus first middle one. I further break it 2 more stages will come out the middle one.

Ultimately in the end the middle one will be nothing, but a 2 by 2 in the end. So, first time 2 stages came out, next time 2 stages came out and the middle one, the last stage. This will keep on happening till log 2 N times. Then ultimately you go to 2, when it is 2 by 2 you simp. So, it has the actually 2 log 2 N, because here I am not getting 2, but only 1 so I am subtracting 1 actually. In the end I will get 2 by 2 which is equal to one, so that has to be subtracted, I am not getting 2 in the last, so I am subtracting. And now I know that how many switches will be there. These are going to be N by 2 switches of 2 by 2. So, cross point complexity in this case, will be. Whatever is the cross point complexity of it 2 by 2 switch, this basically meant, this cannot be further broken it is not possible. It has to either cross or bar states. So, it require 4 cross points. So, I can write that cross point number directly here.

So, I can write this 4, is for a 2 by 2 switch. Number of switches are N by 2, N number of stages are 2 log 2 N minus 1. So, which gives me cross point complexity as 2 n, but this exact number of cross point, there is not complexity. So, lower order terms can be removed actually. So, there are 2 terms here; one is going to be 4 N log 2 N minus 2 n. So, 2 N is having lower than the first term, which is being subtracted. So, I am looking at a bound. So, this is going to be order of N log 2 N. So, as N grows how my number of cross point grows, in this case, what you can estimate, from here for this recursive construction. 2 log 2 N it a complexity at each stage. I think this is the number of pressure. 2 log 2 N minus 1 is number of stages. But states required to break down the. Matrix and what?

Student: In each state we have.

Right, each state will be unlike 2 of this. The recursively constructed rearrangeable non blocking switches far simpler.

Student: So, why you multiplied by 4.

Because each 2 by 2 element this cannot be broken further, this requires 4 cross points. 4 is the complexity of basically. It is not, complexity always means, if I am changing my dimension or size, complexity essentially deals with that, it does not deal with the exact number. So, for example, if my class; simple example I am taking if my class size is N n number of students number of questions which will be ask in 1 hour, how it grows as my N grows. So, I can do kind of a theoretical basically an experiment observation based on the figure out if my N grows; N is here, how my number of question grows. If this grows like this there is a problem actually, complexity is to 1 if it goes like this there is a difference scenario. It can grow linearly; that is also a problem, but usually by experience we know this is always going to be, initially it grows like this then it goes down, it is not even like this. Slope is all usually less than 1, because ten students will never ask ten questions per hour, it usually less than that, or slope is always less than 1, and it is saturate. So, because what happens most of students will actually have, common questions.

So, the first guy who asked he gets the answer, other people also gets the answer. So, intuitively I know this is going to be of this kind. So, idea is how the growth happens. So, essentially what I am not worried about is, not the exact value. Exact value can always be computed, but what I am worried about, as N grows what happens, so it is not linear. So, remember what, we had N square; N square is a curve of this kind and there. In fact, this has drawn wrongly if going to be same scale, it will be something like this, N is equal to 1 will be something like this. N log N will be something like this, which will crossover; that is what your. This is a cross bar, this is going to be your, this will be your rearrangeable non blocking switch. A shape is look going to be like this, but I am not sure whether this N log to N 1.

A slow piping also has to be less than, it has to be something like this N log N, not upper one. This has to be like this, but remember rearrangeable non blocking, this is strictly this is O N complexity usually is not there; O N log 2 N. No I think o N log 2 N has to be higher, this cannot be lower, this was correct, this 1 strictly non blocking. these are not up to scale, if you put in on same scale this always, at larger dimension this is not good actually, strictly non blocking, lower dimension strictly non blocking is fine. But I am not using integers, because switching means, all values are discrete number of inputs ports changes 1 2 3 4, you cannot have 1.5 input ports. So, these are just for notation purpose actually, and question is, this kind of complexity is better N log N, because it is not exploding. The number is not exploding actually large.

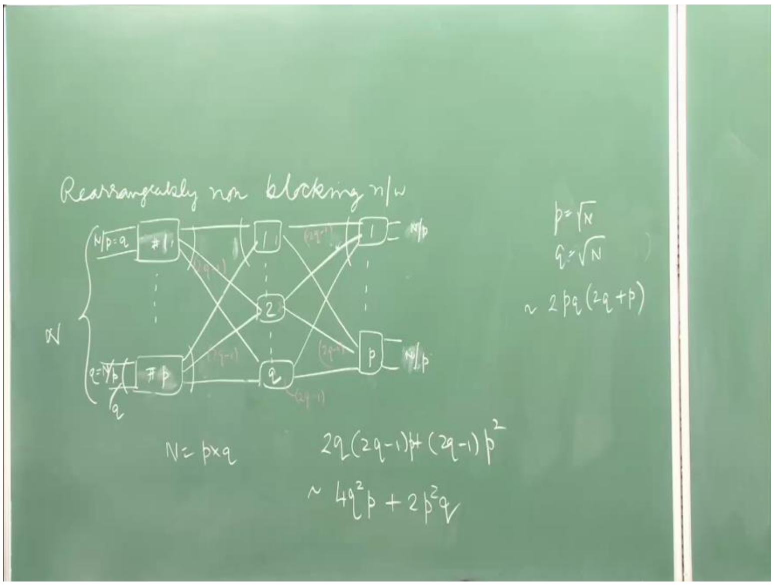
Student: 3 by 2 also.

Yeah.

O N 1.5. So, which is somewhere in between these two. So, I have not plotted exactly I am just intuition drawing it in, these need to be verified. So, let us go for strictly non blocking switch how we will compute for that, and of course, as I stated earlier there is a baseline, this is bench network. If you break this; these are the output taken as output port they, say baseline network, and if I take these are the input ports, and these are output ports, these inverse baseline networks. So, they actually come in the category of banyan; tree of banyan networks. Banyan networks has a property where, if there is exactly 1 path between each input and output port. These are required for self routing switches actually, perfect at switching system. So, coming to now strictly non blocking configuration, how that will be achieved.

Network N is to 4 3 by 2 so that was not recursive construction, that was a, we just did the break up, that was not for recursive construction. Now we will go to the recursive construction. So, idea is this, you remember this red ones now; that is the configuration, and with these red ones, this has to be now recursively constructed. The question is, what should be p and q; first question always comes this. They are infected does not matter of what, way I do the break up. I can always do to raise power k into 2 raise power j kind of break up, and keep on breaking further; you will always get 2 log 2 N minus 1 stages. You may not end up in getting the same bench network kind of this.

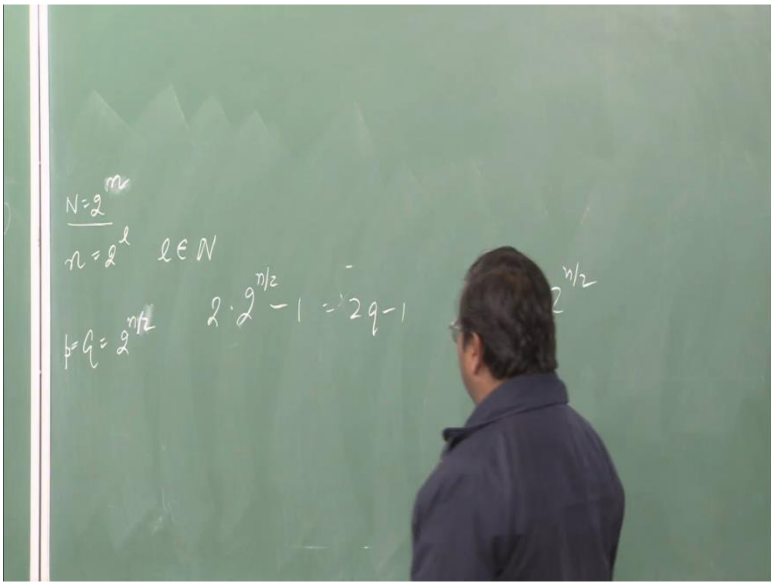
(Refer Slide Time: 28:38)



But here this choose is very crucial, because in this case, your cross point complexity actually is, 2 into q 2 q minus 1 plus this, and this can be now approximated as 4 q square. I can forget for example, this minus 2 q is very small element I can add that, I am only increasing my cross points, because I am as a worried about that is actually bounds, not about the exact estimates. So, this can always be done this kind of approximation. So, I can estimate this thing as 4 q square. 1 c was there 1 c was there 2 p q 2 p q. You are right, I will put that p here over q square plus p, and this will be 2, and this can be written as, and what we can do is. In fact, over 3 by 2 when we computed, N 3 by 2 complex 3, we have to break this down. Now question is this value whatever I do, I do not know the optimum. For optimum it is going to be difficult, I have to first of all optimize take the derivative of this, and then of course, try to compute. We have got some value actually earlier, it was I think route n; under root N by 2.

So, what do we want is, just create a bound. So, if I actually keep it p and q both equal, and make it root of N, it is going to be simpler to compute, and whatever I compute will be a bound. Actual estimate will always lower than this. Can we made lower than, if you can carefully build up that. So, we will do with that p is equal to root of N, q is equal to root of N. So, from here also I think you can try it out. I think it will same root of N by 2 will actually come; N by 2 square root will come actually from here also, but I am not going to do with that, I am going to take, p is equal to root of N and q is equal to root of n. That will place a number of restriction number of ports. Now incoming ports is N, this is only breakup which I am taking care, N is fixed. N p N q absolutely integers for sets. you have to choose. There is a restriction on N that sense, but I am not worried about now exactly those restrictions. I am just trying to create an expression. This is kind of not exact thing, but it gives an idea.

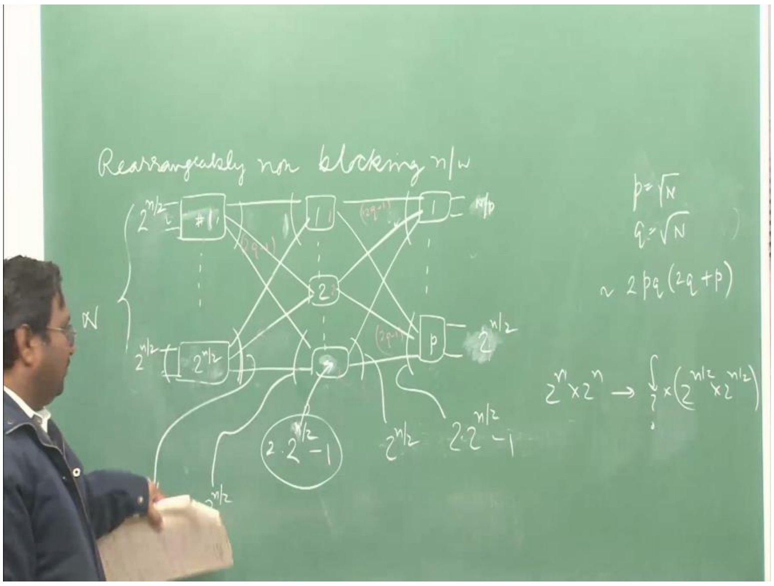
(Refer Slide Time: 32:06)



So, again for convenience sake, I can take N is equal to 2 raise power N, and this N is nothing, but 2 raise to power l, where l is a natural number, or l is an integer actually, positive integer. So, I am further restricting this, because that is how I can compute, otherwise it is going to be difficult to compute. If I take for example, this q is equal to 3, this becomes a different situation. So, I am usually restricting to only 2 raise to the power i kind of situation, switch are built that way. The reason being, because you optimally use your addresses space. We have often mostly with binary systems, and easily. There is a wire if it is on and on, it gives 0 and 1, and for through multiples buses, multiple wires in a bus I am trying to generate addresses, each port has to be address for controlling.

So, you will build up a 3 by 3 switch usually, you will always make 4 by 4. 3 by 3 also requires 2 control lines, and 4 by 4 also require 2 control lines. 5 6 7 does not matter, next is 8 actually. It requires 3 control lines. So, control space you are always trying to optimize. So, that is the reason why 2 raise to the power i 2 raise power N actually is used, but I am further putting up these restriction in this case. Because I have to do a square root, because of that this has been done. So, what will happen is now, the first element I can, write this is 2 raise to the power. So, number of this switches or q. So, q will be now, N is this. You take the square root, 2 raise to the power N minus 1. No, N by 2. So, how many middle stage switches are there; 2 into N by 2 multiplied by 2 minus 1. This is also equal to p. number of middle stages are q, which is going to be, because 2 q this has to be 2 q minus 1 for a non blocking configuration. And what is the size of middle stage switch; p by p. So, switch is of size to the power N by 2 by 2 raise to the power N by 2. So, I know that the middle stage configuration. Here I am just going to replace by this number.

(Refer Slide Time: 35:40)



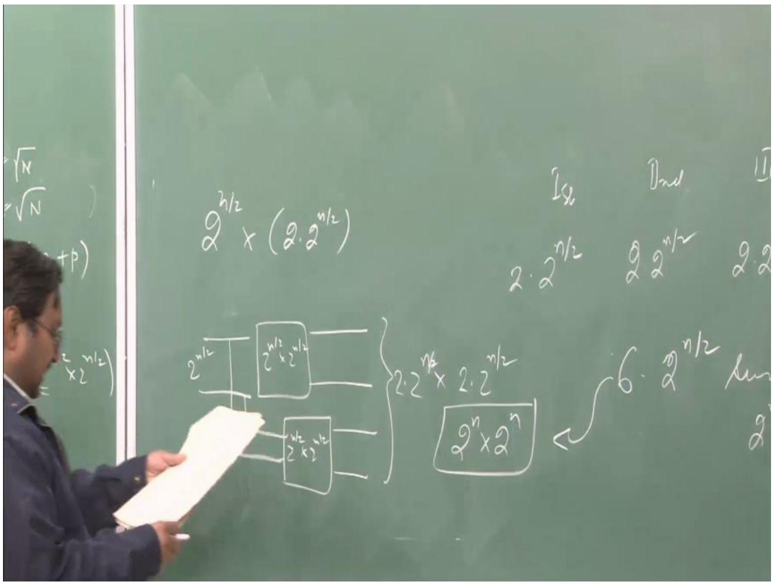
So, there are this number of switches 2 2 raise to the power N, these many number of switches, 2 raise to the power N by 2 by 2 raise to the power N by 2 is the switch dimension in the middle stage, input stage you have these many 2 raise to the power N by 2 as the input ports. Number of switches is also 2 raise to the power N by 2; outgoing ports there has to be this much. Sir here this 2 to the power N by 2 this is the optimum value I can take. This is not an optimum value, but I am estimating amount I can take anything. Actually estimate will always be lower than this, in terms of cross point. There is a maximum possible, but this is very close to optimum. Under root N y because we are. We can easily compute.

Student: Compute or recursively constructed.

Recursive construction is also possible. See I have not a still told how it will be recursively constructed, there is a problem here, there is minus 1 setting, how you will construct a switch. This switch how you will construct recursively. So, I will make an assumption here, because I am estimating amount I will estimates slightly higher, this minus 1 I will forget actually, in the number of switches here, here also I will forget, then I will prove that each 1 of these switching element can be done by 2 raise to the power N by 2 by 2 raise to the power N by 2. So, I know that 2 raise to the power N by 2 raise to the power N switch. So, idea is 2 raise to the power N switch, can be made with some number of 2 raise to the power N by 2 by 2 raise to the power N by 2 switches. How many number, this is what we need to find out.

So, I have gone from this switch this switch, and this is recursion. So, I can put these thing here I can find out 2 raise to the power N by 4 by 2 raise to the power N by 4. I can keep on doing it till I get 2 by 2, and 2 by 2 I know the complexities is 4, or number of cross point is 4. Let us put all those values do the solution, and you will get the complexity expression. That O N log 2 N 2 is for 2.58, that value will come from there actually. So, similarly on this side, this value is 2 raise to the power N by 2. So, for this it is clear that how many switches are there, these many switches are there, 2 raise to the power N by 2 by 2 raise to power N by 2 switches. Here, I do not know, so what we will do is, we have to make an. I can forget this minus 1 actually, let my outputs be larger how does it matter.

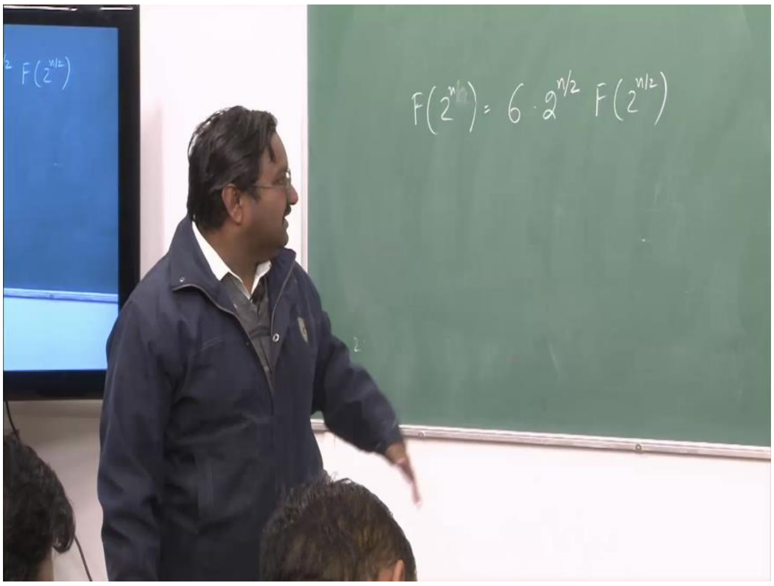
(Refer Slide Time: 39:17)



I can use 2 raise to the power N by 2 by, I can use this switch also, does not matter and so far it is still not blocking, and it is fine with me. Now how to build up this kind of strictly non blocking switch. Usually you will try to see that, let me make a cross bar whose outputs ports are more. I can do something smarter actually the number of inputs is 2 raise to the power N by 2. I can just duplicate each 1 of these wires, use 1 switching to recover N by 2 by 2 raise to the power N by 2, take another switching element of same size. Is this strictly non blocking switch, I can take any input. These strictly non blocking. So, for an output port is free, input port is free, I can always set up the connection, each element is strictly non blocking and that is a parallel configuration. So, each one these blocks, can be implemented by this configuration.

This actually means how many switches will be required in the first state, how many 2 raise to the power N by 2 by 2 raise to the power N by 2 switches will be required in first stage; 2 for implementing each switch. So, you required 2 into first stage. These many switches will be required. Second stage; this minus 1 again I am doing away with it, I am using 1 higher number actually; that is what I did here, so this also minus 1 has to go. So, second stage also requires 2 raise to the power N by 2 into 2. Whatever I did for first stage, I do inversion of that. Third stage also requires the same. So, how many require is, 6 into 2 raise to the power N by 2 switches of size 2 raise to the power N by 2, and this can be used to create, this much size of a strictly non blocking switch. So, I can now write recursive relation actually from here. So, I call it F.

(Refer Slide Time: 42:30)



So, F which is the cross point for this is nothing, but 6 into 2 raise to power N by 2. sorry this is 2 raise to the power N. f is the function which gives you the cross points, number of cross points. So, number of cross points is a bound on the cross point, number of cross point actually. So, I can recursively now solve it, I want recursive relation. So, the cross point which are required for 2 raise to the power N. So, it is more than the minimum required, we have added one more. Let us say it is not 2 into q minus 1 it is actually 2 q, which I have been using.

So, I can go with the recursion with this, and then solve it. So, I think I will stop now, because you have to actually go for another class. So, I will take up from here, you can try on your own. Anyway it is also given in the notes, but I still feel that you should try on your own, and try to get the expression. So, next lecture will actually solve it, and then will go for still better switch, because this is not the exact value I can get, even better bound. I can still get a strictly non blocking switch, with a different configuration, which is going to have lower cross point complexity; that is a canter network.

So, this is where I am leaving now. This cannot be rearrangeable. There is no rearrangement required, it is a strictly non blocking, this is a strictly non blocking configuration. Only thing is that important thing is to get this recursive relation. You should appreciate and understand this. Say it is important concept to actually to understand how you arrive at this expression, solving it after this is all, then it is a kind of normal routine thing. Can you explain this point this, how do you get the duplication of.

We have to create a switch which is having 2 raise to the power N by 2 inputs q into 2 q minus 1. So, instead of 2 q minus 1 I put 2 q. That is sir. But then how to create it was the issue. I want to convert everything to 2 raise to the power N by 2 by 2 raise to the power N by 2, everything I am trying to break down. So, if I take this configuration, I am able to create this matrix; 2 of them I put together. Any input you take, I should be able to connect to every output. This whole box, is giving you this simulation inspecting like this. This is basic building block, but this is a trick usually people will get the struck only at this point. Once you know this you can do it. This what you are getting fourth time not 2 time and. No what is this output; number of ports here, 2 raise to the power N by 2 2 raise to the power N by 2 the 2 times 1 and 2.

So, I can be in general I can actually create a caster, I can take all these parallel lines, and I can create 2 N by 2 by some number, i into 2 raise to the power N by 2 is possible. You refer i such switches, all inputs are parallely connect to all of them, only bus size or wires required will be hire. Basically we are answering that we have to implement 2 into 2 q. That is why we are doing a duplication. So, that is why you required 2 q elements in the second stage. You also require 2 q elements also in the first stage, as well as in third stage also. So, it becomes 6 times because of that, and because number of switching elements is this 6 into this, and that is what actually now follows further form here.

Thank you.