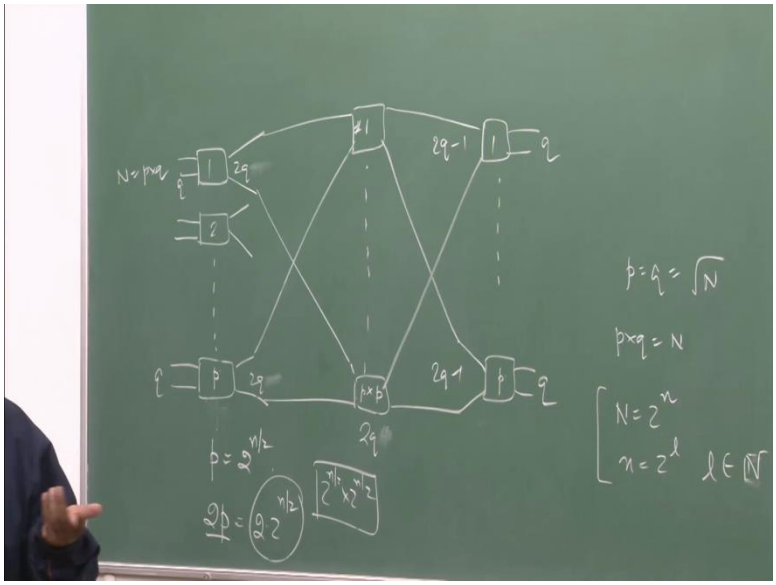
Digital Switching
Prof. Y. N. Singh
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Lecture - 11

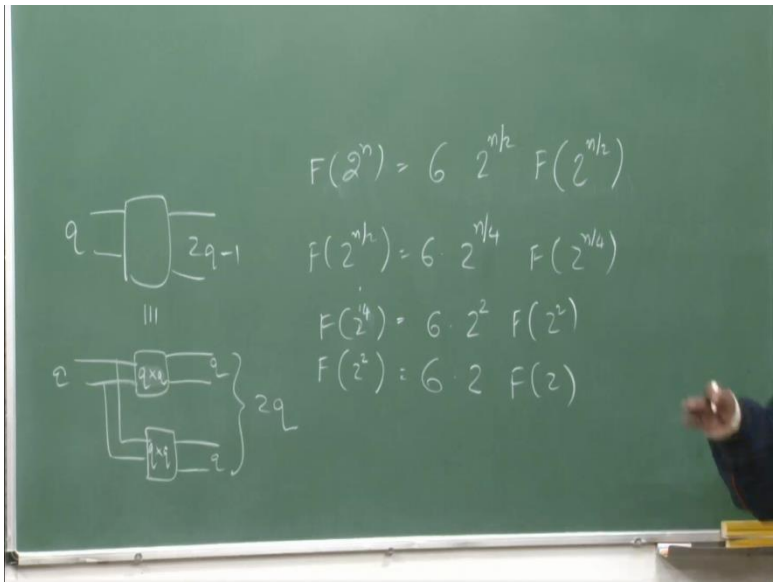
So in the previous lecture we were looking at the cross point complexity of strictly non blocking network.

(Refer Slide Time: 00:25)



So, what we did was, we actually have taken N is equal to p by q kind of expression, and we have got I think q and 2 q minus 1, that was the dimension in which every inputs switch actually was design in this fashion, and total p number of switches were there. And there were middle state switches each one of these switches is a p by p configuration. And there were 1 and 2 q minus 1 switches in the middle stage, and symmetrically there was a output side switches were also there. So, there were q outputs of q ports coming from each port, and their number was again a p. And what we did was, the way it has been derived that; p is equal to is q taken as root of N, and p into q will be N; that was the condition I think with which, we were trying to build up the cross point, number of cross points which are going to be there, in the strictly non blocking switch, and we were trying to build up the recursive construction.

(Refer Slide Time: 02:19)



Next step what I did was, I took the switch, which was having 2 inputs 2 q minus 1 outputs. I said this can be equivalently represented by, a q by q switches. So, number of outputs will be now q plus q if you put them together these are 2 q; so not 2 q minus 1, but 2 q. I can equivalently represent this thing by this. This also means, the middle stage switch. Now there will be modification, you will get 2 q here. We will also get 2 q switching element switching; boxes of switching elements are also in their middle stage. Each one of them now can be equivalently represented by 2 q by q switches. So, intern what is total number of switches, which will be requiring, for a total number of cross point if I want to compute. So, this is number of the way I said a cross point in 2 raise to power N; that is the way N was defined, and I had also taken that N is 2 raise to power l if l belongs to a integer, or I can called it initial number 1 2 3 4 5, it cannot be negative. So, I can very well write it as integral thing, with these conditions.

Sir why we calculating F raise to power, m of 2 raise to power, when we going to be realized, this started with the under root and cross N block switch why you want to realize N cross n. Because for this I will be able to solve the cross point this expression, for the number of cross points. I am just building up a recursive relation; recursive relationship with the basic building block that will be terminating point. Any recursive will be solved, some is basic condition and for the general purpose N, any function of building block. See, ultimately I should be able to reach to F 2, F 2 cannot be spread any further, idea is that. So, at the 2 by 2 switching block has to be, what to called basic

building block. I am just creating the whole switch, it is nothing but 2 by 2 switches, which is a basic element, which we have also done for rearrangeable non block. So, in that case now number of cross points required for a 2 by 2 is 4. So, for example, what we did for rearrangeable non block. I said 2 log 2 N minus 1; my number of stages will be there. Each stage will contain N by 2 2 by 2 switching elements.

So, these are total number of switching elements required for a strictly rearrangeable non blocking switch, and each switch will require only four. So, this total number of cross points required in a rearrangeable non blocking switch, what we get from slepen duds theorem, but this was on the condition that N is 2 raise to power something, otherwise I could not have come to this condition there will be some F 3 or something. There will be some constants, which are ultimately come, but this will be more or less true for all cases, just log 2 comes because of that, but we know it is going to be logger flip, what will be base that it will be different. I can do also 3 by 3, I need not always look 2 by 2.

So, while creating reconstruct what we called recursive construction of rearrangeable non blocking switch. I was every time dividing by 2. I started with 2 by 2 basically, I think I can do 3 by 3 also. So, maybe you will have log of base 3 something N in that case, and you will have F of 3 which will be 9, which is a constant; N by 3 does not. This constant does not matter this will become N by 3 this will become 9 log 3 N. This is splay constant minus 1 you can forget. Ultimately it will be N log N, that will be the complexity. So, it does not matter.

So, what log base log you take does not matter actually. So, I think same is also true here, I am anyway solving for 2 by 2 N, but expressions are come out to be same, but this way it is simpler. So, what we will have is, we I think re-state this particular point that F of n. So, N is 2 raise to power n actually, small n will be. I will be requiring p number of elements and p is 2 the square N by 2. So, in total 2 p q by 2 q q is switching elements, and q has been taken as a root of n. So, I will require.

Student: Sir how is it 2 of p is familiar, because here the impose number of switches is p.

Ha this one.

Student: No in the ending input stage.

Input stage.

Student: It is.

Yes p switches. So, 1 is coming from in this switch, 1 output port coming from each one of them. So, there will be p inputs.

Student: Sir, that is sir in that in calculations.

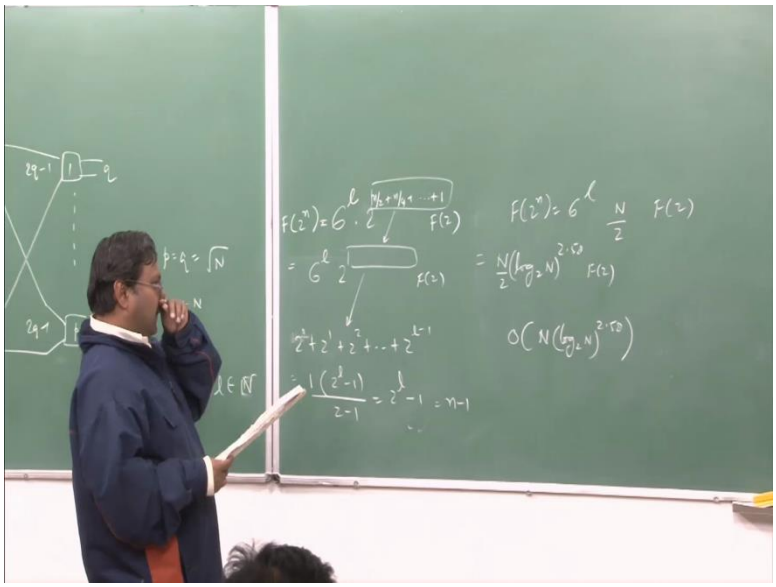
This 2 into 2 power N by 2, how is that 2 factor coming in from. I am coming to that. So, if there are how many 2 raise to power N by 2 switching elements. Now each q by 2 q switch, is implemented in this fashion, I require twice of that. So, total number of elements which I require is, 2 p here, which is nothing but these many elements of size q by q actually, q by q is this basically mean, q also happen to be 2 raise to power N by 2. So, you require these many number of elements, basic elements I just 2 raise to power N by 2 raise to power N by 2. Here how many elements will be required, you require 2 q elements; each one of them is t by p types, this is nothing, but p by t only. So, again you require 2 into 2 raise to power N by 2 even in this stage. Similarly, on this stage, because this, you just take the mirror image of this 2 q inputs and 2 q outputs, I can also implement the same switch.

Sir is there have been particular advantages of implementing the switch like that. This one, actually I required this much, I require q input ports and 2 q minus 1 outgoing ports. Let us to be strictly non blocking, that is the only condition. I am trying to do this, I am trying to remove 1, I am trying to duplicate, so that everything, what we require is basic building block is this. Just for using this I can create a strictly non blocking switch of N by N 2 raise to power N by 2 raise to power n. So, I can build up a recursion formula actually, it was done for that purpose. So, in turn 2 here, 2 into 2 raise to power N by 2 2 into 2 raise to power by 2, same here, I am requiring total 6, and the switch which I require is 2 raise to power N by 2 by 2 2 raise to power N by 2, this box which I have drawn. So, I can write the cross point complexity of this as this.

So, I have got the recursive relation now. But this relation is true, I can very well write, this also same recursion one. I can keep on doing it iteratively a large number of times, I am just completing it so that. now this is going to 4 N by 2 N every time, the power itself is also going that is why I have written 2 raise to power l; l is equal 2 raise to power l,

where l goes as integer, that is the reason why it has been done that way. I can keep on putting up the values, taking from the lower one to the upper one and so on. Ultimately what you will get is a series.

(Refer Slide Time: 12:36)

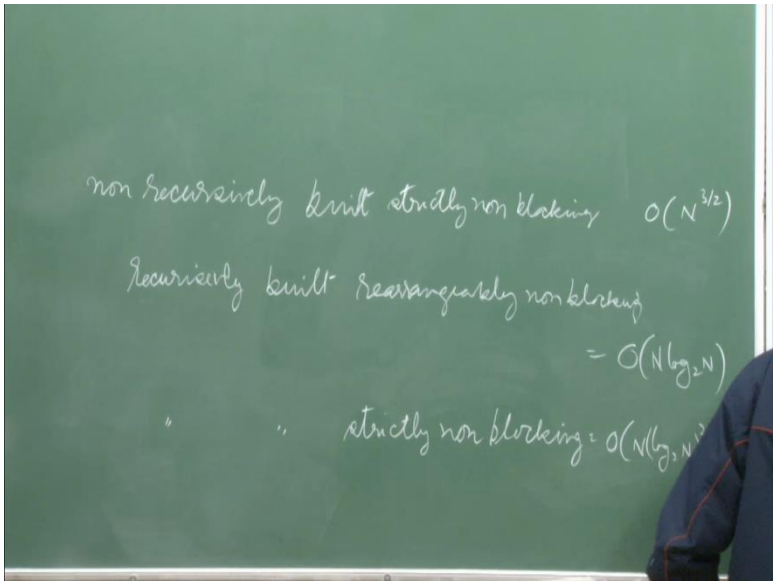


6 power l So, how much value, this is 2 raise to power 0, this is 2 raise to power l minus 1 2 raise to power l is n. So, 6 raise to power l. see N by 2 is this, 1 is 2 raise to power 0 total l terms have to be there. So, I am counting that terms based on this index 1 2 something. So, there has to be l term that is why 6 raise to power l has to be there. So, that is the way you can remember how many terms, otherwise usually it becomes difficult for some people. So, need not remember it, you can derive it from here itself. N F of 2 F of 2 is 4, does not matter whatever be these are constant, some constant k. So, 6 raise to power l, whatever is the series what is the value of this. So, it is 1 which is 2 raise to power 0 2 raise to power 1 2 square. So, I am talking about whatever is there, this particular series, and this is nothing, but geometric progression. You can very well find out the sum.

So, first value; whatever is the ratio, 2 total number of terms minus 1 2 minus 1 2 raise to power l minus 1 2 raise to power l is, nothing but m N minus 1. So, this will be. This value is nothing, but N 2 raise to power N is nothing, but capital N the size. 6 raise to power l has to be now sorted out. So, this is how much; 6. So, everybody knows by basic elementary mathematics. So, 6 raise to power l is l log to 6. I can very well write this is

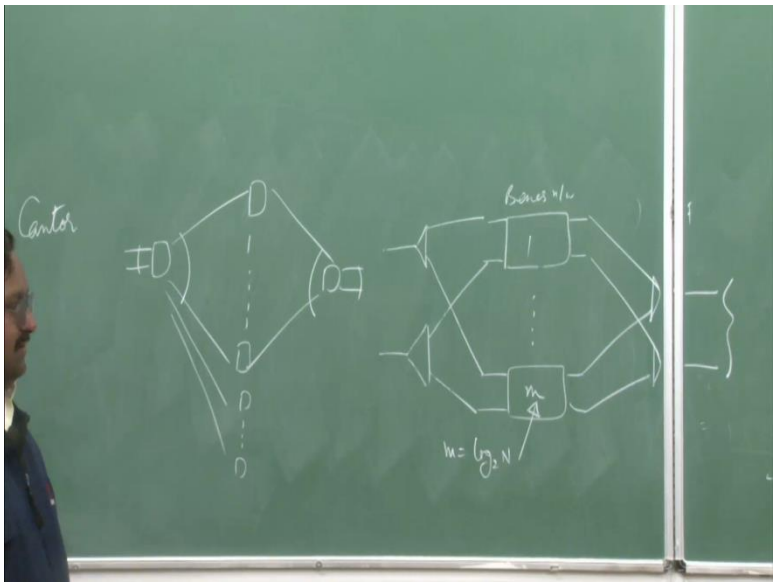
this 2 raise to power l whole raise to power log 2 of 6 2 raise to power l is how much, this N. and from here N is nothing, but log 2 of capital n. So, I can write here. So, intern now log 2 of 6 is 2.58. So, you will get N by 2 log 2 of N 2.58, whatever is F 2 is a constant. So, your cross point complexity is this, for a recursively non blocking switch. So, now, let us go back and recall what we did so far.

(Refer Slide Time: 17:28)



So, there is a non recursively build strictly non blocking. What was the cross point complexity. When we went for recursively built rearrangeable non blocking, I got this. And when it is recursively built, strictly non blocking 2.58. And of course, now the question arises can I have still something better than this or not, because I just made an estimate, whatever I am going to compute cross point complexity cannot be worse than this. I can still find out something, which is better than this at least, it gives us a bound. So, can I built up something which is strictly non blocking, and have better complexity than this. So, that is a next question, and of course, this again was unsort by cantor, and he came up with idea of cantor network. Now, this also leads us to a fundamental method by which, you can make almost any switch as a non blocking or rearrangeable non blocking. I have actually mentioned this fact many times. So, I will just state the same fact here, and then we will move ahead.

(Refer Slide Time: 19:55)



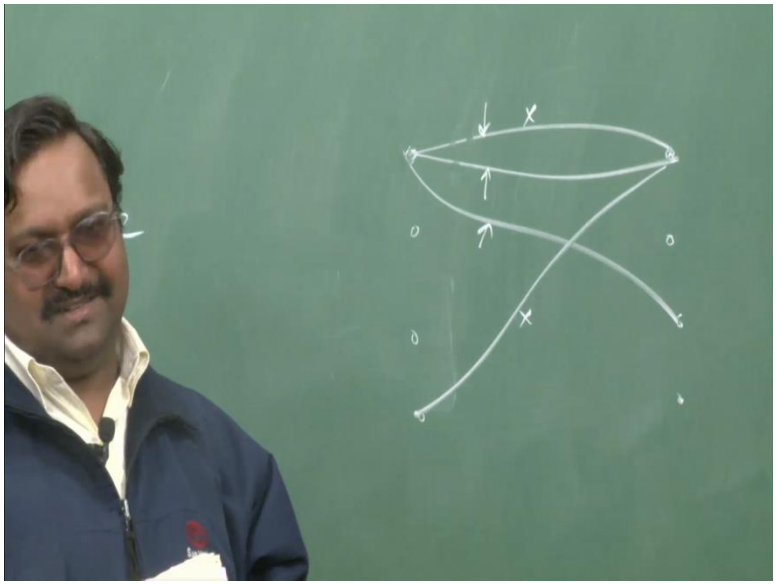
So, idea is this, I have a switch, it is now connecting too many other switches actually in the middle stage. So, far these ports are higher than these, or these ports are higher than these, both these conditions have to be satisfied, blocking can only happen, because of the interconnections. So, it is possible if I keep on expanding on these intermediate stuffs, keep on adding more and more, at some time I will become strictly non blocking. So, cantors actually take up the same idea ultimately, but he could build up the proof that was the thing. What he said, I can actually have, what did it he took a rearrangeable non blocking network, but he was not using any rearrangements. He used something called bench networks, and many of them, and for any every input he started using something called demultiply, actually it is a switch 1 by N switch.

So, one port is connecting here, and one port connecting here for every input port, he does that. So, what is that actually happening is, earlier if it is only a rearrangeable non blocking switch, I required only 1 single bench network. This is again basically 2 by 2 basic elements 2 log 2 N minus 1 stages, and connection is, the way we build up a recursive, this rearrangeable non blocking switches same block. So, if I do not use this demerge, all input ports are only directed to here, and they are only again directed to in same fashion here. So, forget everything else. So, all these inputs are directed here, and directed here it is a rearrangeable non blocking switch, because this can be rearrange always to setup the connection, and that is bench network if basic element is 2 by 2, and with 2 log 2 N minus 1 switches stages actually.

So, what did it is he kept on using these multiplexers. Now remember they are complexity of this multiplexers, is clearly decide by number of ports here and number of these switching elements, multiplication of these 2, nothing more than that. He said if I keep on increasing this number at some point of time, when this number is going to reach to a certain threshold. I should be able to setup a path without disturbing any earlier connections; it should be possible all the time. So, the whole theory is, whatever he did was he just find out what is the value of this m; minimum value of m requires so that you can always set up the path, without disturbing earlier one. So, there are no rearrangements. And once you know you can find out the complexity of all these switching elements.

So, 1 by m switch 1 by m switch m by 1 m by 1 and total N such switches actually, and used that the complexity of these will be lower, and complexity of these will be higher. For this already complexity is o N log 2 N, for this the complexity is N into some this particular factor, multiplied by 2 is to take this one also consideration. Complexities of these switch elements will be always lower than this. So, this is what we will decide, and ultimately what you he did he proved where that m went m will take a value of log 2 N, that time this particular network will become strictly non blocking, and how this log 2 m comes I will tell, but when m reaches log 2 N, the switch will be strictly non blocking, that was the cantor's contribution.

(Refer Slide Time: 24:26)



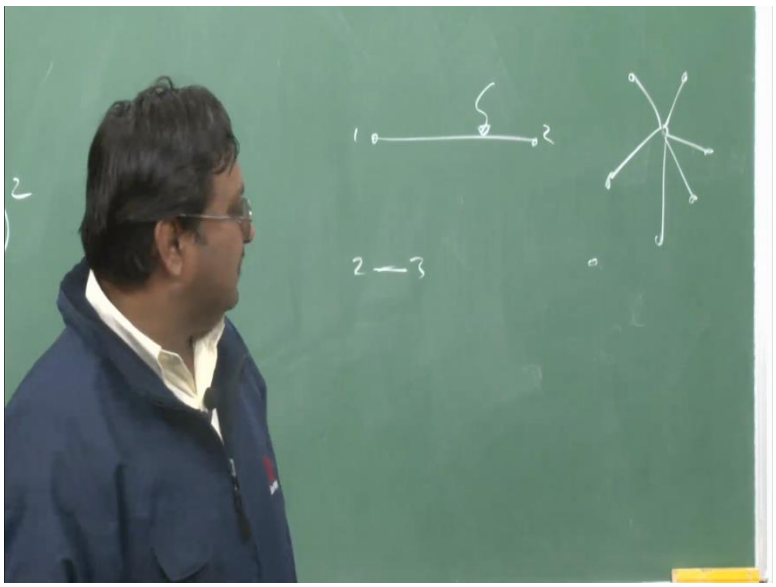
And was this condition is there your complexity is N log 2 N for this one. this is a complexity, because of the middle stage elements, complexity because of this N log 2 N for these elements, and this being 2 times, this lower order than this, so I can neglect whenever I am computing complexity. So, ultimately you will end up in getting a complexity which is this. How does this wave thing came, because the bench network locates N log n. Bench network is N log 2 N and number of file switching elements, number of bench network m is log 2 n, so multiply by that. So, it is not log 2 N and this is how 2.58, but log 2 is N square; that is the bound now for strictly non blocking switch. We still do not know whether the bound can be still further reduced, can it be 1.8 maybe, but nobody knows, but the bound is certainly not more than 2, it is currently 2 N log 2 and N square. N for rearrangement narmits N log 2 N, where power is 1 actually log 2 N power one. So, somewhere in between actually the best possible value will exist, but currently none best possible value is 2.

For unique casting. it is for unique casting, not for multicasting that is, it its only for unique casting. I think that it is true for whatever we have done so far. We have not taken any case for multicasting. Multicasting the problem is a tricky actually, and as I mentioned once, that you can actually use graph theory basic principles to do that. So, one way for example, if I am looking for claus network for the pulse matrix. I was trying to mention that if you look at that the input switch has the node, output switch has the node. So, these are bipartite type graph. So, these are the input nodes, these are the output nodes, I can setup a connection between them, and creating a link. So, link can be there between these, I can create a link here, I can create a link here, but total number of links switch can emanate, from a switch that has to be controlled. So, each one of them has to follow the different middle stage switch.

So, what I have to ensure is, if I start coloring these, links itself. I have to ensure that at any edge at any node. All the links which are emanating from a common node, has to have different colors. They have to choose different middle stage switches; same as to be true here, and total number of colors which are required to do a coloring of edges. What is the minimum value required actually. So, if you can prove that, that number will tell you what how many middle stage switches are required. If I can put say color x here and same color can be used here, I can use same middle stage switch that is what it means actually, but between these 2 I cannot use same color, because they actually have a

common node. We cannot have same color, they have to have different color, they have to go through 2 different middle stage switches. So, find out the minimum number of colors, and that I think theorem is known as Hall's theorem, in case of graph theory, but what the way we have done it is slependau theorem, but technically results are same, for rearrangeable non blocking switch. So, coloring problem actually you can convert it to edge coloring, this is not a node coloring, this is a edge coloring problem, and from there you can actually solve, find out the connections. But anyway, since we have already done through slepen duds I am fine with this, even for multicasting this will be true; that is multicasting cannot be handled by this edge coloring, this is done through a node coloring problem.

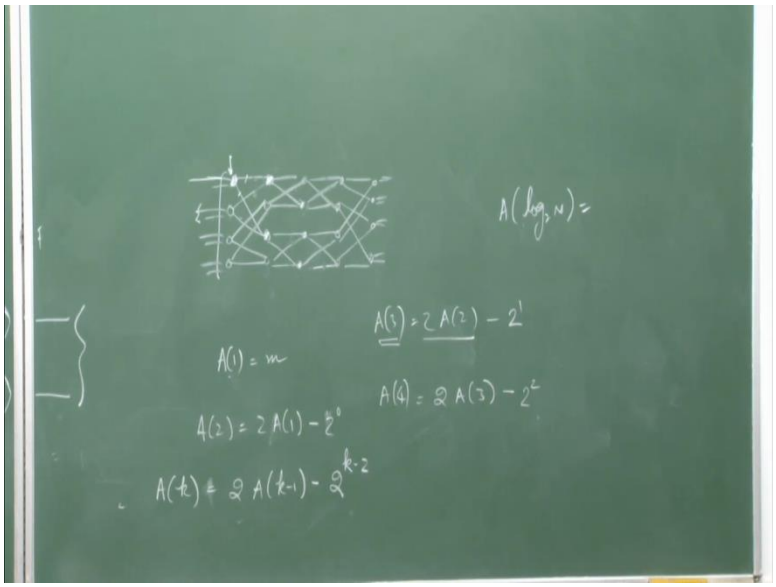
(Refer Slide Time: 28:59)



So, what happens is, a connection itself; say connection between this node and this node this connection is counted as a node, and if 2 nodes 1 particular connection 1 2 and then the similar connection is that 2 3 connection is there, if they have the conflict. So, I represent this thing by a node, and this connection has a conflict. So, I create a link. So, for 1 particular connection, all the connections which can have conflict with this, will be the neighbor actually. Any node which is not the neighbor, this does not have a conflict with this, and then I can design what you called node coloring problem. node 2 never in node should have the same color, because they are having conflict with each other when a connection is going to be setup. So, this is any different domain; that is what actually is going to be used for multicasting resolution, how to setup the paths for multicasting. So,

coming back to the cantor network how to prove this. So, as given result now let us start proving it. So, we have to understand what to bench network first. I will just draw that one of us to know.

(Refer Slide Time: 30:20)



So, I have taken a 8 by 8 bench network. So, same thing is repeat I am assuming that, you can visualize the same drawing repeated multiple times; that is what is the bench network. So, this blocks is that. So, this guy can connect to. So, we call this as first stage all across, all the bench networks. So, how many nodes it can reach, compute that. So, there is total N nodes, N switching stages. It can reach one here, second here, third here, nth here. So, in the first stage, the node switch can be reach by an input, is m. It does not matter whether the connection has been set up or not, but I can always setup a path till this particular switching element or this switching element, this switching element, because of this is not connected, all these lines are free, because these are free I can always setup a path till that particular road at least, from first to second stage now we will come.

Now, if this you look this particular switch, this also has second input here. In worst case, now all these second inputs and all these switching elements 1 to m, all of them will be connecting to the second input port here. Worst case scenario, suppose that already path has been setup for this second guy. So, this path will be either connecting this first one here, or first one here, or first one here; and this has to be routed to the

second stage. So, it can either take this, or this, or in the next one the same second element lines, or in the last one first and second. Only one of them can be taken, all of them cannot be. So, this input should be able to reach how many nodes in the second stage; twice of this number, because I am able to go to m first stage switching elements, twice of that minus 1 has just been occupied by the second incoming port. So, A 2 which I can reach is 2 of A 1 minus 1. Similarly now it reaches to A 2 number of switching elements. All these switching elements are able to further now reach to how many, twice of A 2. but all those elements, this second stage here for example, is able to reach from here, is able to come even from here, sorry from 2 more sorry this one and this one. So, these two also, where every one of them, will able to reach these. In worst case scenario, these also have been set up already.

So, actually which is the number of elements in the third stage to whom it can be reached is, 2 A 2 minus 2. If you go to A 4 similarly, unfortunately I have not drawn that bigger diagram, because those one, two and third stage is this, and that is the large stage. So, I need to only go to log 2 N stage actually, not any further stage. Say if it is a larger dimensional A 4 will exist. So, A 4 will be twice of A 3 minus how many, 2 square. It is 2 1, it is 2 raise to power 0. In general our expression will be; A k twice of k raise to power k minus 1 minus 2 k minus 2 this k cannot be larger than log 2 N. this k cannot be larger than log 2 N; that is the maximum you can have from the input side. So, depend on whatever with the value of m, these many nodes switching elements I should be able to reach. So, once you understand we actually have, cleared the big hurdle. This is only the tricky part, one of the trickiest parts.

I tell you what is the notation A 1 A 2. So, let me do it here, I am able to reach here; second element and m third element. I am not drawing any extra new thing, only what wherever I can reach. So, m nodes I am able to reach that are what I have written A 1. Now this can reach here, and here sorry some middle one, in the second stage twice of these. So, I have written 2 into A 1, but this input, this is also there coming second port. I am assuming first case the second particular input port is already connected. So, if it is connected it cannot be routed to in any other fashion. it has to use one of these switches has to be consumed in that, either it is connected from here to here, here to here, or maybe through this one to this, anyone of these. So, suppose this is gone, is already

consumed, because this path has already taken. So, how many are left; 2 A 1 minus 1. Next stage.

Student: Sir, same thing can happen with other switch also which one.

Same thing can happen with other block also.

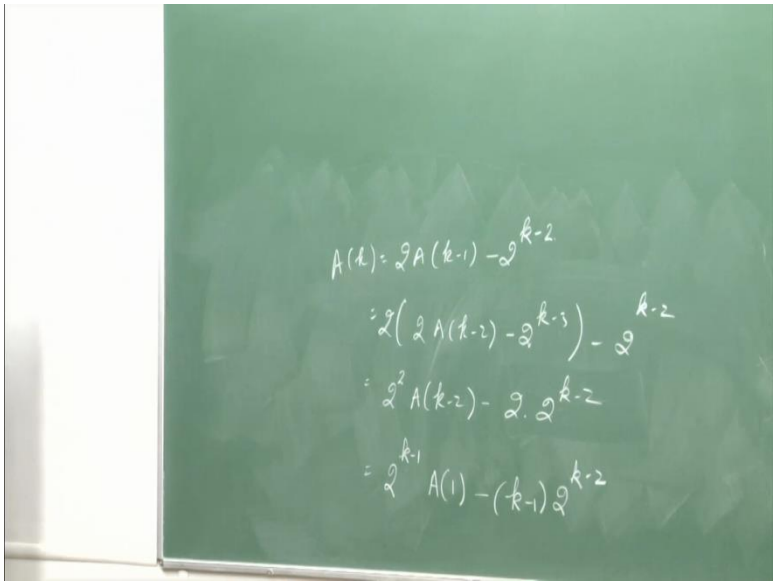
Student: This one also.

But you can only set up one path, this is a switch. So, I have already taken this path, sorry for the first one, I have already taken this particular, this is from second one actually. If this is already been taken, I will not be connecting through this switch, that can be only one path, unique cast. Either I can set it through this, through this, or through this. There are many alternative paths, only one of them can be consumed. If I am already path has been set up through this route, only this will get consumed. So, only one node can get consumed here. So, next one again you will do, this will not expand, this has already been consumed remember. So, in the third stage is 2 times the A 2, these many will be coming, but these nodes can also be reached by some other route also, something like this will be happened. So, these elements are there which can also get, can reach to this particular node. In worst case those 2 ports also occupied, so 2 nodes will be gone in that case. So, for example, this is gone, and this is gone, this is possible.

Sir, but any if they get other nodes, then we could have consider those nodes also while considering A 2. No, these two other ports cannot reach any other port. See you have to understand; this kind in first stage can only reach here. In second stage this input port can only reach here and here, cannot reach at these two. In a third stage only they will reach to everywhere, and I have already consumed for example, this particular switch for a path, I cannot reach here, those are anyway excluded. So, every time there is a fresh input port which will come, and this will ensure that there is no conflict with earlier allotments. So, I am just doing doubling and minus this. So, the general expression is going to be this, or you have to carefully sit down and think about it actually. So, that is the only way you can understand, because unless you have a very big diagram and you start actually doing calculation, start counting, you would not be able to appreciate. You actually can draw a 32 by 32 bench network, then you can very well understand that.

So, try at doing larger bench network and try to study them, then you will appreciate what I am saying. And the only thing which is very important is, this port through this switch can go to here, here, and here, but only of the path will get consumed. I am assuming whatever worst case scenario that path has been set up by somebody already. So, what is node there available to me I am trying to look at that, and everybody every path has been set up except the one, which I am trying to. Only one input is free and one output is free, that is the case, which we are considering. So, in worst case scenario, the middle stages switch A log 2 N. So, how many nodes I should be able to reached in the middle stage, that is the question, so that will be A log 2 N. So, put the values here this is a recursive relation again. So, if you start solving this recursive relation, I can do that actually from here.

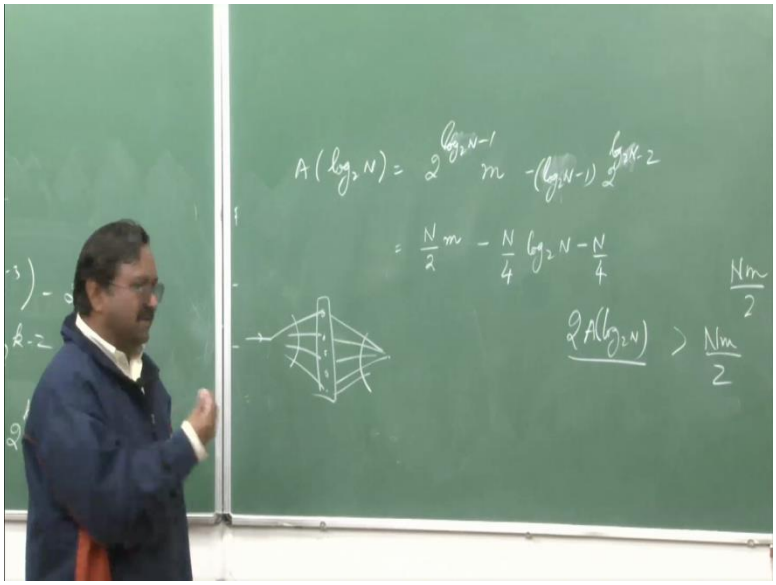
(Refer Slide Time: 41:57)



So, A k twice of A k minus 1. So, I can now put 2, whatever is a value of A k minus 1. this will be. This intern will be, this. If you keep on doing recursively keep on solving, you will end up in getting 2 raise to power k minus 1 A 1. Some of these 2 already equal to k you have to understand, I am doing that same thing, from where you can verify. This is just every time I am increasing the number, so this will be. This is counting basically, it is going to count keep on counting, because 2 multiples and normalize is the exponent. So, it always 2 raise to power k minus 2. So, this is what should be your relation at the end. So using this, what is A log 2 N A 1 is m, I have already given that constant; A log 2 N is, this is m minus and this is nothing, but k. So, log 2 N minus 1 log 2 N minus 1

log 2 N minus 2, I just replace k by log 2 N, and this value is; N by 2 m minus, this is fine.

(Refer Slide Time: 44:10)



So, these many nodes I can reach from the left side, or the inputs side. My switch is completely symmetric construction wise. So, if I look at the input port or sorry the outgoing ports from there I go backward to the middle stage. So, from each outgoing port I should be able to reach to these many same numbers of elements in the middle stage. Free, it is free actually elements in the middle stage. These are the free elements, which I am counting in worst case, and how many middle elements are there. There are total m bench network. Each bench network will have N by 2 middle stage elements. Total number of middle stage elements is N m by 2.

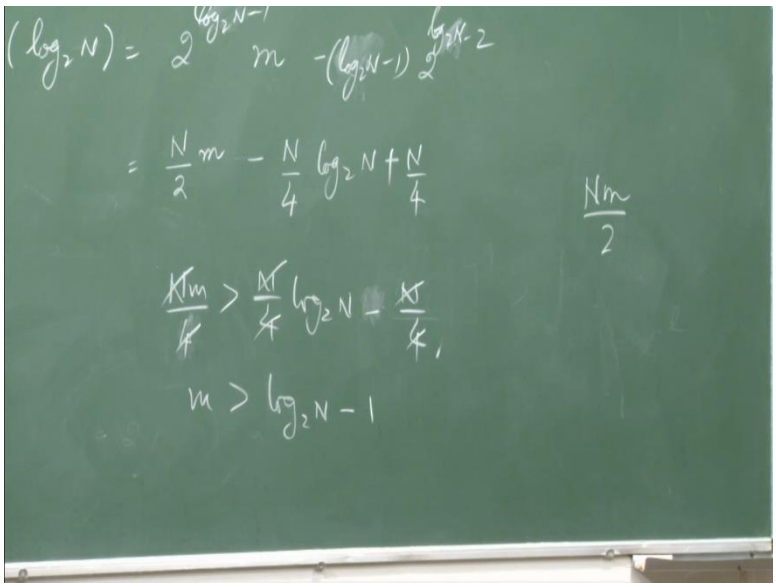
So, which actually implies, if I said 2 of A log 2 n. So, number of elements which I can reach from the input port, number of elements which I can reach from output port, I sum all these elements, free elements, which is nothing, but 2 a log 2 N. if this value is, greater than N m by 2; means what. This is like; from this input I am able to reach to these particular ports. From these I am able to reach these particular free ports, total number of ports is already known to me, which is N m by 2.

So, total number of free reachable ports and total number of free reachable ports if it is more, there are ports available in middle, which are common, which are reachable from left side as well as from the right side. There is some common free port, free middle

stage switching elements which are available, that is the meaning. So, for example, I can take a very simple thing, and explain this. For example, I have only 4 middle stage elements, in general sense. I am able to reach only one free, and from here also one free. So, these 3 input lines are not free, these 3 outputs are not free I cannot setup the connection.

So, 1 plus 1 is less than 4, I make it 2 for example now. I can reach here, I can reach here, it is like a not set up the connection. I can reach 2 from left side 2 from right side, which is equal to middle stage elements; it is a worst case scenario. But suppose I can reach 3 from this side and 3 from this side, and there are only 4 middle stage elements then. You try all combinations you will find out, there is going to be some middle stage elements, which are reachable from both sides, for all possible scenarios, and if I can find these I can use these 2 to set up the path. So, my switch is shift be non blocking. I am able to set up the connection without disturbing any earlier connection.

(Refer Slide Time: 48:51)

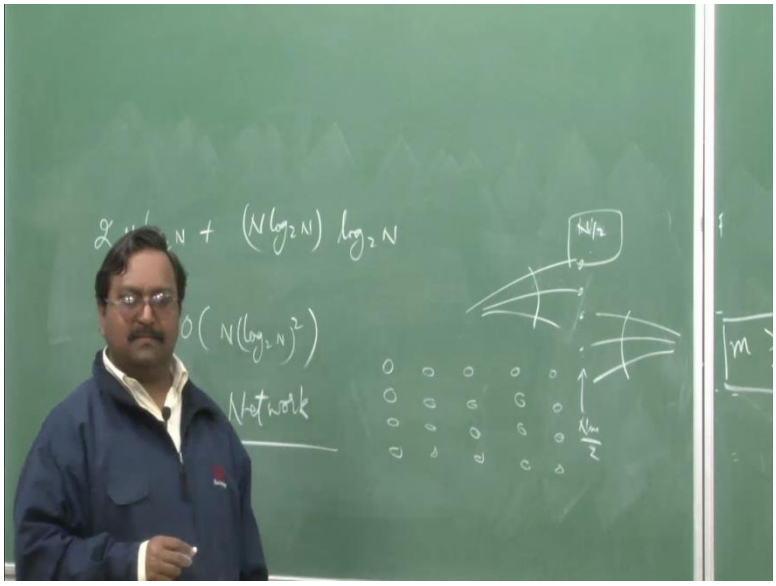


So, once you solved it you will get, this place whatever I have done here. In this case this will be N m by 2 minus N by 4 log 2 N minus N by 4 multiplied by 2. Yes that I have not done. So, these 2 I can take, this will be N m. So, this is N m by 2 can be subtracted. So, I can remove this thing and this will be.

Student: Sir, why did you cancel out, sir it will not cancel out.

It cannot cancel out. Whole multiplied by this A 2 1 in generator left hand side and right hand side in denominator how did it cancel out. It will become 4, N m by 4 that is the way it has be. So, I can remove this particular thing, and this will become 4, and this whole thing I can take on the right side. So, here N will cancel, 4 will cancel. So, m has to be greater than. I think I have missed out something. So, this has to be minus A. So, m has to be greater than log 2 N minus 1, which technically implies that m is greater than or equal to log 2 N, if this condition is satisfied. I can always find out some free elements in the middle stage, which can be reached from both sides, and this can be used to set up the connection, and thus switch is strictly non blocking.

(Refer Slide Time: 51:29)



So, cross point complexity in this case will be, you have N de-multiplexes switches, and they are 1 by log 2 N each 1 of them. So, they require log 2 N cross bars. There N log 2 N complexity in the de-multiplexer side, multiplexor side also same twice plus the middle stage is going to be N log 2 N log 2 N such bench network. So, all with your cross point complexity is, o N log 2 N square, and that is what is the cantor network. So, with that I think we will finish today, and next class tomorrow we will be looking into one of the examples of, what we called wide sense non blocking system, where you actually start using an algorithm. If you use a proper algorithm, switch is always remaining non blocking state, it will never being in the blocking state, but if your (( )) not careful you just choose anything haphazardly, you might end up in blocking state. In this case irrespective algorithm you can always, you are always in non blocking state. So,

far that is what we have done. So, wide sense non blocking example we will do tomorrow.

N named by 2 by something why should be greater than N m N m y 2 sir. The same example, the middle stage how many elements are there, middle stages N m by 2. So, what I can reach from this side, and what I can reach from this side some of these.

Number of switches number of 2 by 2 will be N m.

Number 2 by 2 elements. Switching blocks.

Each bench network will have N by 2, and there are such m bench element.

Can we do it otherwise, I mean only half is less than N m by 2.

In the sense k maximum number of networks what you told earlier, from maximum number of middle stage switches which I can either from the inputs side or from the outputs side, has to be symmetrical.

So, I just double that. In case you do not double it, will we get the same relation.

No without doubling you cannot. See reachability from this side, and reachability from this side; total sum if it is greater than whatever is available, there is something which has been counted more than one time, which is common. I am trying to just prove that there is some common element in the middle stage which is free, and effectively to that point from here this side I can reach from here to also this side, there is a path available, and I can always use that to set up the connection, is a reachability analysis technically. I am trying to analyze the reachability.

This N by 2 coming from.

Bench network, remember that. So, these are 8 by 8 network.

Basically, it has what type of strategy to do it recursive assertion of.

There is no recursion here.

Student: Sir in middle stage you have done recursive.

Middle stage is a recursively constructed rearrangeable non blocking. It is a bench network, and that a bench network in the middle stage is coming from that rearrangeable non blocking recursive assertion, and we are reading (( )). Bench gave the configuration network is known as bench network. You can also do it through recursive construction of using 2 by 2 elements, not 3 by 2 or any other, it is only done through 2 by 2, for a specific case it is a bench network.