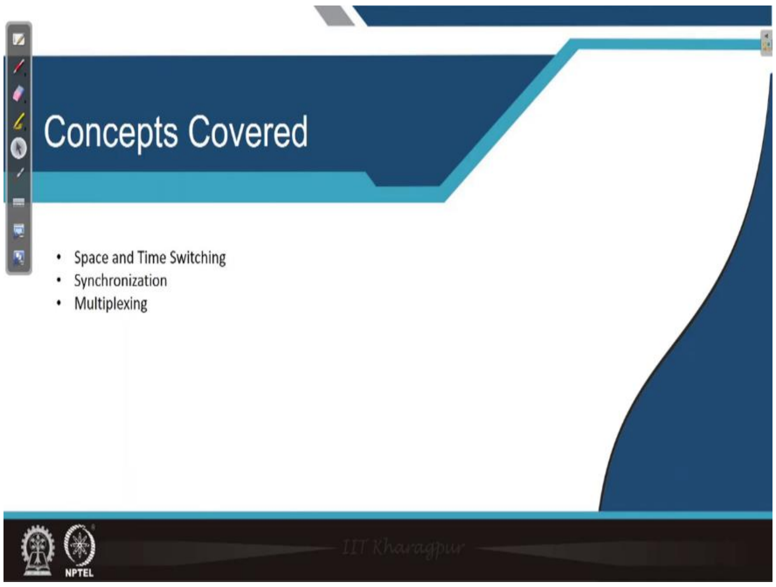
Communication Networks
Prof. Goutam Das
G. S. Sanyal School of Telecommunication
Indian Institute of Technology, Kharagpur

Module - 02
Circuit Switched Networks
Lecture - 07
Space Switching Architecture cont'd

Ok, so far, we have started our journey of circuit switch network. So, we have seen some of the components of circuit switch networks, like the multiplexer and switches. This is something we have discussed. And then we started discussing Switch Architecture which is the most important part of a circuit switch network because that is the element that makes the switching possible ok. So, we have already seen that among the switching, there are two kinds of switching.

(Refer to Slide Time: 00:54)



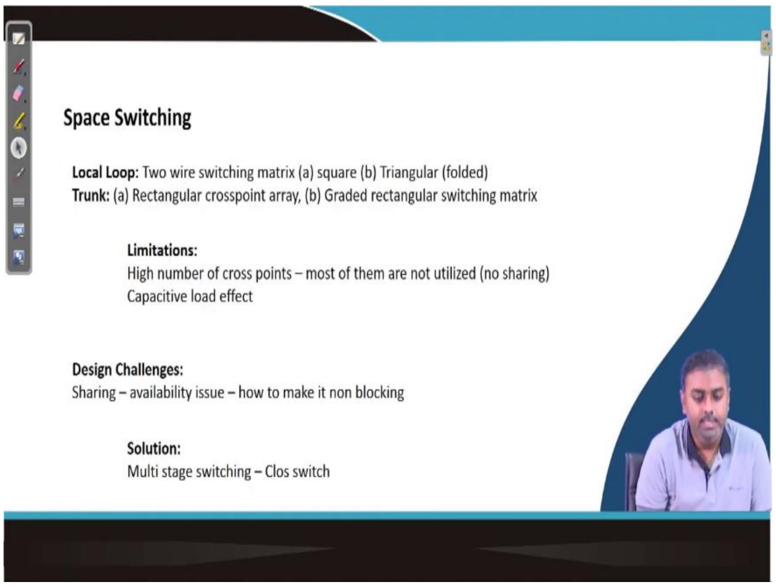
Concepts Covered

- Space and Time Switching
- Synchronization
- Multiplexing

IIT Kharagpur
NPTEL

So, one is called space switching one is called time switching. Of course, time switching requires time division multiplexing. So we will discuss that later, but initially, we started discussing space switching ok.

(Refer to Slide Time: 01:08)



Space Switching

Local Loop: Two wire switching matrix (a) square (b) Triangular (folded)
Trunk: (a) Rectangular crosspoint array, (b) Graded rectangular switching matrix

Limitations:
High number of cross points – most of them are not utilized (no sharing)
Capacitive load effect

Design Challenges:
Sharing – availability issue – how to make it non blocking

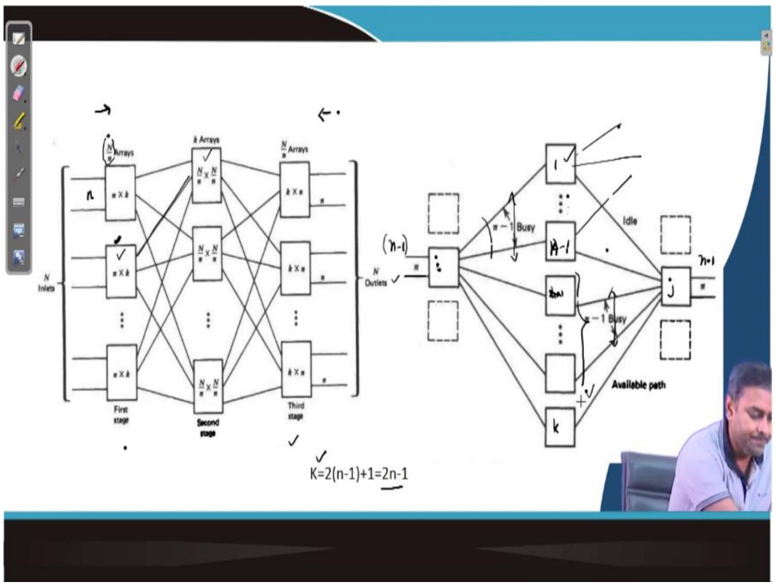
Solution:
Multi stage switching – Clos switch

So, in space switching, we have already been told that again, in space switching, there are two kinds of switches; one is for local loop ok, and the other one is for the trunk. So, these are the two switches we have already discussed. We also discussed their characteristics of them, and then we started discussing some simplistic switching architecture through cross-point ok.

We have also started seeing why this means the design of full-blown cross-point switches are not very efficient, how we can minimize the number of switching elements, and what is the advantage of these things we have already discussed. And then, we came to a particular point where we could actually share some of the switching elements so that we could reduce the number of switching elements. This sharing part is the most important part probably of the switch designed to reduce the number of switching elements.

So, let us try to first see how this sharing can be done. This is something we have started our discussion on, and we will further go into that deep into that discussion. So, we have already told that multi-stage switching which is given by Clos a brilliant design through which we can share some of the middle-stage switching elements, and with that,, we can see how we design more efficient switches.

(Refer to Slide Time: 02:25)



So, this is the structure we have already discussed. So, we have said that multi-stage, we will start with three. Of course, we will show that the same structure can be repeated for an even higher number of stages. So, initially, we will start with three-stage switches. So, this is something we have already demonstrated. That is what we do. We subdivide the input port into multiple small switch stages.

So, basically, as you can see out of this N port at the N port. So, out of this N port actually, we have actually subdivided it into N by n number of switch port switch switching elements. So, basically, we have N switching capital N switching input, and we have sub-divided into N by n, and each of these switching elements has N input ok.

This is something we have already discussed, and then from there, k outputs means ports are coming out. So, that is k, and then from that k, we are actually feeding it into a middle-stage switch. So, this structure we have already seen in this middle means three-stage switches, and it is a symmetric structure; that means input and output will look similar ok.

So, if you see it from this direction or this direction, they look almost similar. So, it is a symmetric structure, and then we started actually exploring how we make it a non-blocking switch. So, we have also discussed what we mean by non-blocking fully non-blocking switch. So, that is something we have discussed, but then we have also given the Clos argument to make it non-blocking.

And with that, we have come to this particular formula for the number of middle stages, which is the sharing stage we have talked about. So, this sharing stage has to be 2 n minus 1. This is something we have already discussed with some argument, ok. So, we will probably take this argument a little more forward, and we will also try to see why it is non-blocking and all those things we will do.

So, just the very basic argument of that which we have already discussed is that if we have an ith particular input stage to some jth output stage, we are trying to connect one user to another user. So, when it will be non-blocking, so, it will be non-blocking if somehow through these elements. So, basically, there are k number of middle stage elements as you can see 1 2 up to k. At least one clear path from input to this middle stage and middle stage to this output j I have availability ok.

So, what do I mean by availability? It might happen that some of the input ports are already blocked, so; that means I cannot get any of these input ports, and it might also happen that some of the output ports are blocked, then I cannot get the whole thing. So, basically, if I wish to get connectivity from the input to output, I need to input this port from the first stage to the middle stage, and then I also need a free path from the middle stage to the end stage.

And remember, I am trying to give connectivity from a user from ith switching element to jth. So, basically, the output of ith and the input of jth must be free through a particular switching element; this is what we need, actually. Now, let us try to appreciate the blocking. So, what might happen in the worst-case scenario? I still need to give non-blocked switching. So, what time might that happen?

So, it might happen that the worst case scenario might happen that already some n minus 1 are occupied ok. So, if n minus 1 is occupied, definitely some n minus 1 output port from this particular thing will be already occupied. So, that n minus 1 I cannot take. So, n minus 1, some middle k switching elements will be occupied by whom I have connected.

Because remember, according to my structure, from a particular input switching element to a particular middle switching element, only one connectivity is there if you see it carefully. So, once that one connectivity, that means at least one user is connected to a middle stage, that middle stage switch is no longer accessible for that input switch switching element right.

So, this switching element is not; if this is already this particular port is already taken, then I cannot use this particular switching element middle stage switching element, so if suppose that n minus 1 other than the 1 I have requested now. So, let us say the last 1 I have a request, and this wants to go to the last one of these, and these two are free. So, there is no blocking from the user's point of view; it is the switch blocking ok.

So, we think that input and output are available now whether the switching elements can give me a path from input to output. So, let us try to see if this n minus 1 definitely they will be taking some n minus 1 input port. So, immediately some n minus 1 up to n minus 1 will be already occupied, ok.

So, those many; so, sorry, this is not the n minus 1. This is the n minus 1, probably because of this dot dot dot. So, this n minus 1 is already occupied, ok, and they might now be going somewhere else, ok. Why I am saying somewhere else is if one of them goes to this j, then that will be the common one; we will talk about that later with an example also.

So, they must be getting switched to some other output elements, ok. So, this is the worst-case scenario again; remember, we will give an example. So, this is what is happening. So, this is going to some other this one this is going to some other element. So, that is how things are happening, ok. Now what might happen? Now here also, n minus 1 might be busy.

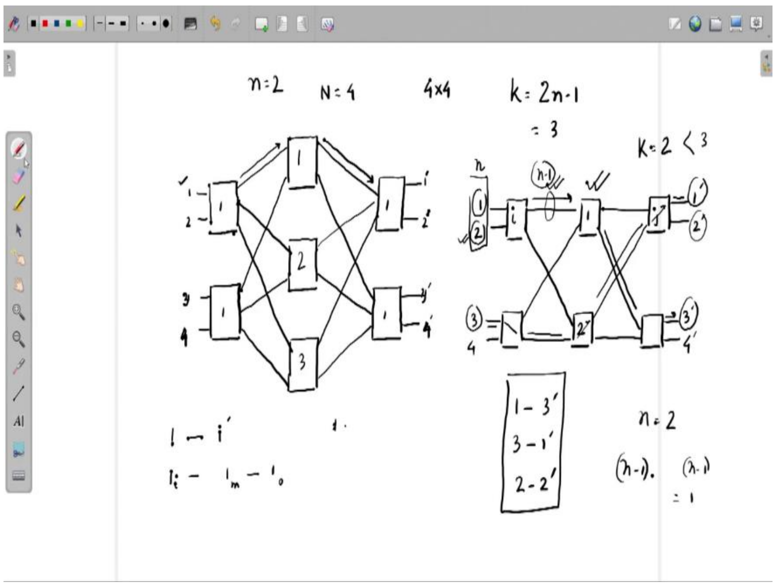
And in the worst-case scenario, what might happen this n minus 1 ports are coming from all other switching elements. If this happens, then what will happen that n minus 1 middle switching elements are blocked because inputs are blocked? n minus 1 other means, mutually exclusive other switching elements are middle stage switching elements are also blocked because the output ports are blocked.

Now if he still has to get connectivity, another extra middle stage has to be there. So therefore, in the worst-case scenario, I need to have this 2 into n minus 1 because n minus 1 plus n minus 1 over here. So, 2 into this n minus 1 and this n minus 1 over here, these two are required plus 1 more should be there to make this last connectivity possible. So therefore, 2 into n minus 1 plus 1, which is 2 n minus 1, should be the minimum middle stage.

So that I can get full connectivity, this is a minimal requirement that is something we should keep in mind. So, that is Clos argument, and with that, we have come to this magic figure of k equals 2 n minus 1 ok. So, let us now try to see with this how the switches are further can be classified. So, now, let us try to see if we can classify these switches further.

So, what we are trying to say is that the switch I have designed with Clos argument with a middle stage to n minus 1 is strictly non-blocking now; we will give that term a strictly non-blocking switch. So, will give an example that will give this particular concept clearly.

(Refer to Slide Time: 10:18)



So, let us try to see. Let us say I have to design a 4 cross 4 switch. So, capital N is equal to 4. I want to design a 4 cross 4 switch, ok. In the middle stage, I put if this is 4; I actually put my n as 2. So, n equals to now I have taken the design as n equals 2. So, if n equals 2, what will be my middle stage? So, that should be 2 n minus 1 which is 3. So, I must put 3 middle-stage switches.

Now, with this example, you will clearly understand why the Clos argument of the worst-case scenario with strictly non-blocking is valid. So, it is a symmetric architecture. So, this should also be 1, 2, and that is 1, 2. I have input ports 1 and 2 and here also 1 and 2. I have connectivity, of course; from everybody, there will be one connection to every other switching element. So, similarly over here and symmetric structure ok, this is the connectivity ok.

Now, let us say this first user 1. Let me just distinguish the output port. Let me put a dash over here. So, let us say I have a connection request which is 1 to 1' ok; without thinking anything, I can give connectivity. So, basically, I take the middle stage; I have a choice; I can take any middle stage, remember? 1 to 1' connectivity, I can give it like this. So, basically, 1 goes to the first one and then goes over here, and I can give the connectivity like this. I can even give the connectivity via 2.

So, I give from here 1. I connect to the second port. This 1 then goes to 2 Oks, and from there, it goes. So I can give any connectivity. So, let us say I reserve the connectivity like this, ok. So, I can do that. So, 1 to 1 connectivity I have given. So, basically, it goes from 1 via the middle stage 1 and then 2 1 ok. So, this stage is what I am putting. So, 1 is the input stage, this is the middle stage, and this is the output stage; ok, that is the connectivity. Now next to the connectivity, let us say I get 1 2 or let us say. So, 1, 2, I must put over here probably 3 and 4. Let us just discriminate 3 and 4, ok.

So, now I need connectivity from 3 to 2 there, ok. So, this is something that is allowed. So, I can take connectivity from anywhere to anywhere ok, So, 1 to 1; I have given connectivity now, and I can give connectivity from another port ok to another one. So, this connectivity also I can give.

So, if I wish to give that connectivity, so, let us now try to see similar to this thing. I will parallelly draw another one where k will be less, and then I will try to show how things become blocking ok. So, the similar structure I will put with another one where middle stage; I will have only 2. And then again, the connectivity because there are two stages, so both will be connected.

Now I will demonstrate what is happening. So, k is 2 over here. So, only. So, basically, I have deliberately put k equals 2, which is less than my requirement, to n minus 1, which is 3. So, I have put a less number of this 1, and then I want to demonstrate that there will be blocking so when this blocking will be happening. So, what might happen? Suppose I want connectivity, ok. So, which is like this from this 1, I need to connect to this 3' let us take that ok.

So, 1 to 3' I need connectivity, ok, no problem, I can do that. So, I will take this one and this one. So, this switch will be configured first port to the next port, and this will be connected ok. So, this is what I have done. Now let us say I have another connectivity which is 3 to 1 dash. So, first 1 to 3 dash, now 3 to 1 dash, I have another connectivity.

So, 3 is looking for connectivity to 1’ ok 3 to 1’. I can give connectivity like this. This is possible because I go first stage switch, I give a connectivity like this, then next stage, I give a connectivity like this, and then this stage, I give a connectivity like this. So, 3 to 1 dash is also possible. This is what I have taken.

Now let us try to see if I can take connectivity from this switch. This is my ith element and jth element earlier Clos argument,t whatever we are trying to see. So, this is that element. Now I give I wish to give connectivity from 2 to 2 dash. So, 2 to 2 dash, let us try to see if it is possible 2 to 2 dash I wish to give connectivity.

So, this input port is blocked already; as you can see, it’s already being used for 1 to this 3 dash connectivity. So, this is blocked now; if I 2 to 2’, have only other option is going to go over here. So, if I go over here again, you can see this output port is blocked. So, I cannot give connectivity from 2 to 2 dash via this switch. So, this switch is also blocked in the input, this switch is blocked in the output, and I am completely blocked over here. I cannot provide connectivity.

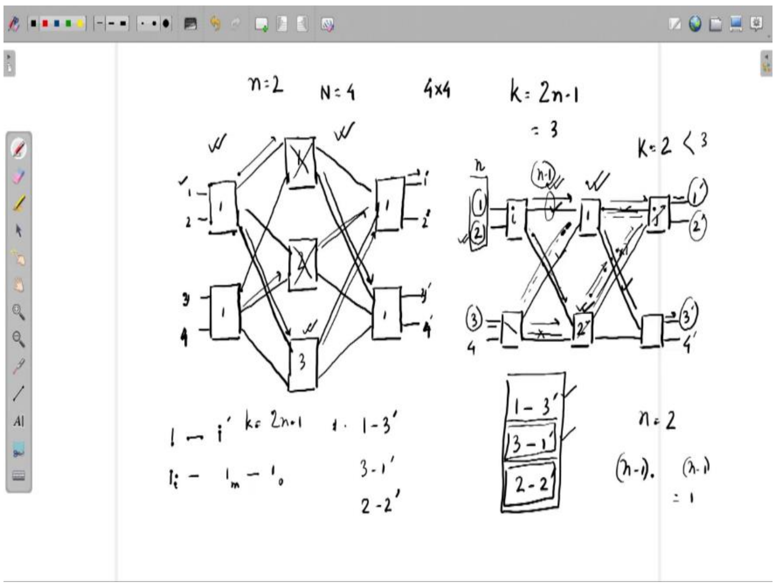
Can you see this is the problem; this is what has happened because of my choice of configuration. So, that is when even though 2 and 2’ are free, I am not capable of providing connectivity between these two. What is happening? As you can see that whenever I am trying to give connectivity from 1 to 2 to these guys.

So, remember Clos’s argument if there are n number of users at the input. So, my n is equal to 2. So, n minus 1 if they have occupied; that means 1 of them. So, my targeted user is 2. 2 to 2 dash connectivity I want to give what is the n minus 1 user that 1 user. So, user 1 has already taken it. So, this is one. So, this is the n minus 1 inputs that are already occupied.

And the output is another n minus 1, that is n minus 1 because it’s 2 that is equal to 1. So, that n minus 1 is coming from another switch. If this is what is happening, then basically, your n minus 1 and n minus 1 are already occupied. You at least need another switch to give this connectivity.

If you see it over here, over here, if I try to do that same connectivity, let us see again, we will do the same connectivity.

(Refer to Slide Time: 19:07)



So, let us try to do that same experiment. So, I need this first connectivity 1 to 3’ ok. So, from 1 to 3’, I do not really think about where I give connectivity I can give like this 1 to 3’. So, it goes like this is blocked now means this is already taken. Now I have I want to give connectivity 3 to 1’ ok. 3 to 1 dash, suppose I want to give connectivity. So, I can give it like this 3 I go like this, I go like this and give connectivity to 1’ now 2 is also occupied.

Now I want to give 2 to 2’ connectivity. Now 2 to 2’ still, I have another switching element 3 free through which I can give connectivity. So, I can provide the connectivity like this, and it is still possible. So, I have a free path. So, as you can see, as long as I go by this rule of k middle stage equals 2 n minus 1, that is sufficient for me to give non-blocking connectivity. This is possibly ok. So, this is absolutely possible.

Now, the thing that is required is whenever we design a switch like this with middle stage 2 n minus 1, that is called strictly non-blocking, but this switch is completely hopeless; that means we cannot provide non-blocked connectivity if you see there is redundancy over here whenever I have put 1 to 3’ connectivity I have taken it like this 1 to 3 dash that is fine this 1 I have taken.

Now, when the next request came 3 to 1’, instead of taking this connectivity ok, I could have taken I will put dot dot dot. I could have taken 3 to 1 dash this connectivity. If I could have taken that then 2 to 2 dash, I could have actually given that connectivity via 2. Because then this will not be there. So, this output port will be free. I could have taken that connectivity via this one.

So, that means, suppose 1 to 3 dash and 3 to 1 dash, I initially have given connectivity like the first time I have discussed involving both 1 and 2. Now whenever 2 to 2’ comes, I can see that it’s blocked because both the middle stages are already occupied. What can I do? I can now rearrange myself. So, or the switching configuration. I can see that this 3 to 1 dash connectivity can be provided in a different way. So, that is when I am rearranging the connection.

It actually requires a kind of disruption in the already existing connection. So, some call is going on; I actually disrupt for a few. It will take a few milliseconds, but I have to do that. So, I take this connection which was earlier via 2 ok. So, like this, I take that, and I make that connection via 1. I can do that. Because this port and this port are free, these two ports are free.

So, I can do this connectivity via this one. So, once I do this particular connectivity from 3 to this 1, I do rearrangement, then 2 to 2’ connectivity, my second switch switching element gets free 2 to 2 dash connectivity. I can do it like this. This is becoming a possibility. So, that is called a rearrangeable non-blocking switch. So, I can initially have a strictly non-blocking switch ok, which is this one for that I need 2 n minus 1 stage. I can further reduce it if I allow this rearrangeability.

So, I initially give connectivity as the connection arrives I initially give connectivity according to their configuration. So, I can just randomly pick whatever middle stages are free I can give connectivity between them. And then, as the new connection comes and they get blocked, then I try to see which are the earlier connection that I can actually rearrange. So that this new connectivity can still be provided, so this is called rearrangeably non-blocking switches.

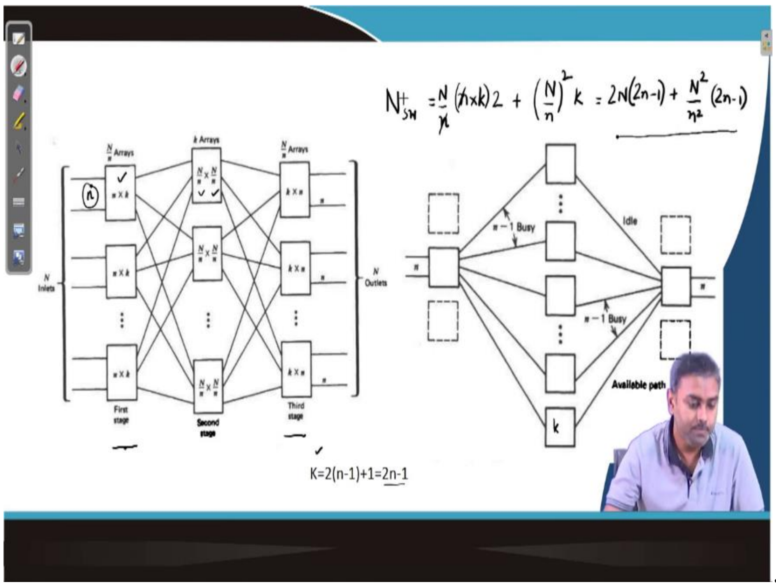
So, according to my switch design, there can be two kinds of switches. So I can have strictly non-blocking. Most probably, my target will be strictly non-blocking because that does not degrade the quality of connection which is already established over here whenever I am trying to rearrange what is happening. Even for some milliseconds also, I am trying to rearrange at that time whatever communication is going on that will be lost ok.

So, if the end user is capable of taking that or they are allowing these things, then I can probably go for a switch design which will require an even lesser number of switching elements ok.

So, basically, what we have now seen is that whenever we do switching, we have a strictly non-blocking that, according to Clos's argument. Now you have probably understood that part. So, it should be 2 n minus 1 middle stage. We can even go for less number of middle stages, but then we have to rearrange them, and that is how the switch gets classified into two classes; one is strictly non-blocking, and one is rearrangeably non-blocking ok.

So, if we now go back to our earlier discussion. So, basically, what we have seen is that we have got this many means switching elements, and from there, we need to see our earlier discussion was what is the overall number of switching elements that are required right. So, over here, if you try to calculate how many switching elements are required.

(Refer to Slide Time: 25:09)



$$N^{+}_{SW} = \frac{N}{\cancel{n}}(\cancel{n} \times k)2 + \left(\frac{N}{n}\right)^2 k = 2N(2n-1) + \frac{N^2}{n^2}(2n-1)$$

$$K = 2(n-1)+1 = 2n-1$$

.

So, let us try to see how many numbers of switching elements are required N switch. Let us say I have designed a strictly non-blocking switch; that means k equals 2 n minus 1 middle stage I have taken. First of all, let us try to understand. How many switching elements will be required for each one of these small elements? We assume that it is a full cross-point rectangular cross-point kind of connectivity.

So, initially, n crosses a or n into k number of switching elements required in each of these blocks. How many such blocks are there? Capital N by n these many blocks and. So, that is the input stage, actually. So, at this stage.

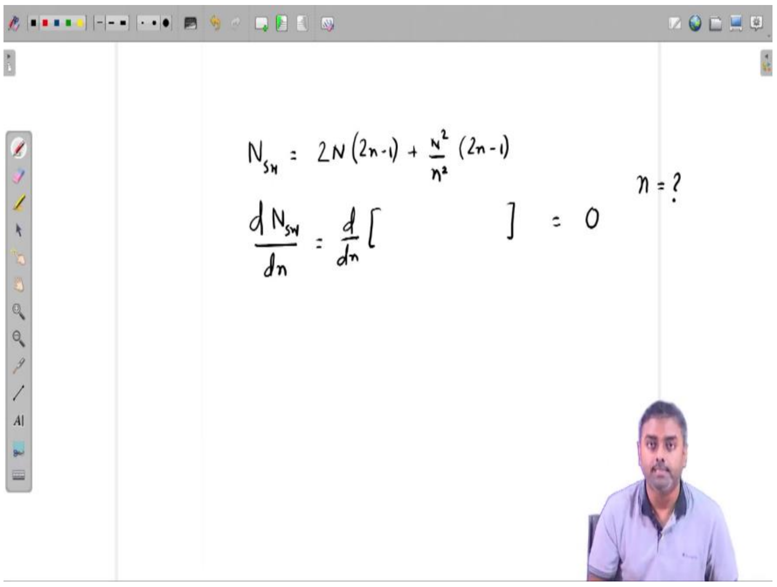
At the output similar kind of thing. So, this can be multiplied by 2 plus middle stage N by n and cross N by n right. So, N by n whole square; how many such switches are there? k ok. So, replace k so that will be this n n gets canceled. So, 2 n into k, which is 2 n minus 1 for strictly non-blocking plus N square divided by n square into 2 n minus 1. So, these many switching elements will be required.

Now, as you can see, after I made it strictly non-blocking and took the minimum number to make it strictly non-blocking. Remember this k I can even take more than 2 n minus 1, but that is unnecessary because I have already achieved strictly non-blocking. I should not take anything more because 2 n minus 1 gives me fully non-blocking; why should I require any other sharing middle stage? I do not. So, this should be the minimum 1 that is required for strictly making the switch strictly non-blocking. So, if I take that is already minimum ok.

So, N switching elements is the minimum I require for a strictly non-blocking switch; this number now, can we further optimize? Yes, I have. I still have a variable which is in my hand, which is the small n; how do I design that? So, I have to choose a small n. So, for the same N cross N switch, I will be getting a non strictly non-blocking switching matrix ok.

So, on that, I can now design small n; what should be my targeted small n? So, I now should take a small n which minimizes n switching elements. How do I do that? So, if I just go back over there.

(Refer to Slide Time: 27:46)



$$N_{SW} = 2N(2n-1) + \frac{N^2}{n^2}(2n-1)$$

$$\frac{dN_{SW}}{dn} = \frac{d}{dn}\left[\qquad \right] = 0 \qquad n = ?$$

So, basically, my N switching element that we know, what was that? That was actually if I just go back to my presentation. So, that was 2 N 2 n minus 1 N square by n square 2 n minus 1; ok, sorry, ok. All we have to do is now we have to differentiate it because we are minimizing it, and we have to make it 0 ok.

So, from there, we will be getting a value of n, and that should be the because if we are minimizing with respect to some variable and if we take that to be continuous. So, n we are right now taking to be continuous that is all right initially we will try to do that we will get a particular optimal n where these n switching elements becomes minimum. And then we get a closer integer to that, and then we put it accordingly.

So, with that, we will be able to design what should be my initial stage, the smaller switch dimension that we will be able to design from there automatically what should be the number of middle stages that also we will be designing and overall, we get a switching element which is having the best design ok. Or best sharing design where it is still a strictly non-blocking switch. I can also go for rearrangeably non-blocking that will have less number of middle stages.

So, this is something we will continue our discussion on this and try to see what will be the minimum number of switching elements that are required in the next class.

Thank you.