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

Lecture – 2

So, in the previous lecture I had discussed about what is basically is meant by the switching system. And I explicitly stated that it is the information switching about which we are going to discuss in the course. And then I actually gave the history that, how the manual telephony actually started manual telephony basically means the manual switches.

Here by the human operators were used for doing the switching function at the exchanges. And we required those exchanges because we wanted to reduce the cost of laying the cables, because we never wanted to have full mesh connectivity, where every user is going to be connected to every other user.

So, and then of course, there was a all kind of a problems which where there with the manual system. And I had discussed about the protocol or the complete sequence of procedure which was used to set up a call or teardown a call a call, and how the call tills were recorded in the register, which was the paper register in the earlier days. And then which was then later on used for to generate the bills by the companies.

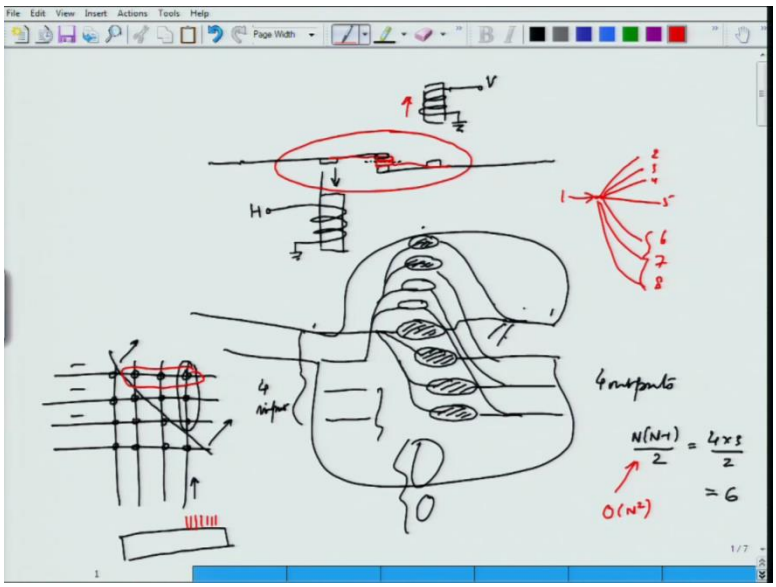
And then of course, I mention that a gentleman inventor name A B Strowger actually came up with the idea of having an automatic exchange. And it was purely electromechanical it was using mechanical component basically rotary a arms which will change the connection point, they connect to different out going ports from an incoming port. And these were are all essentially being controlled through electromagnets, and using electromagnetic electrical pulses. So, these magnets the arm electromagnets will cause the arm to actually move the connection arm, and this there by changing the connection between input and output port.

And I actually discussed about pictorially how a uniselctor an how a 2 motion selector will use to look like, but these are all old stuff now, because I think none of the exchange all across the world are working on this thing except probably in some museum perhaps. And but you can still find pictures on the net, and then I had come up with I discuss the

concept of cross point. A cross point was invented because of the maintenance issues which was involved in case of the electromechanical switch or Strowger exchanges. And which is basically gears being worn out with time, the connections are basically wherever the 2 contexts 2 points metallic points are coming in to contact, those getting wearied out. And there is dust oxidation all kind of issue because these are all expose to air.

So, ultimately people came up with an idea of having 2 metallic contacts, which actually get either in touch with each other or they are separated in a glass bulb, which is sealed glass container and having some inert gas. That is basically is the cross point then I also mention there is going to be a 2 electromagnets, which are required. So, that both of them are energized, you will have the cross point snapped in. Cross point will have a connection you can actually work even with one, but with 1 you cannot build a switch. So, let me start with having only 1 electromagnet doing this job.

(Refer Slide Time: 03:54)



So, you will actually have a glass bulb, and then there will be 1 cross point there will be another cross point. This there actually is 1 cross point, but they 2 metallic contacts. So, I can actually push an electromagnet like this. And whenever you are going to energies this at this point at this control input, this will armature will actually will move down and in turn the connection will be made.

So, the input port will be connected to the output port. Now, if I build up this kind of thing and I want to create a cross bar basically a switch this is going to be slightly tricky it is not that easy. So, let me take up a 4 point, and I want to connected to say 4 outgoing ports, and now I am going to create a switch. So, I require a point here. So, this is essentially cross point I am showing it through a bulb, and this getting connected here.

So, whenever this snapped in the connection will met here. Now, I can create another bulb here another bulb here. Now, how these will be actually controlled that is usually is the issue. So, similarly I can make 4 of these kind in the second branch, and then connect them to only thing I have to ensure is that for any output line, either this bulb or this bulb, and there will be actually similarly there will be more bulbs here.

So, only 1 of them has to be operated. So, there will be four lines only signal should come from only 1 of the incoming lines, not any not more than 1 because, then formation cannot be separated out, it will get mixed up actually. So, this technically is nothing, but is a crossed point this I mentioned in my earlier lecture also. So, this is 1 cross point this is equivalent to this is the next 1, which is equivalent to this one third one which is this 1 and 4^{th}, and similarly, this 1 corresponds to this.

This one corresponds to this and so on. So, all these cross points and which is; obviously, I will require 16 cross points of this kind for connecting 4 inputs to 4 outputs. And of course, as I mentioned I did not I did not I need to only maintain about this upper half of the context because, 1 will usually will never like to connect to 1. Because, this 1 and this one are connected to the same user actually in this 2 is connected to second user you will need technically require N into N minus 1 by 2.

So, in this case this will turn out to be 6 4 into 3 by 2 which is 6. So, I have 6 cross point 1, 2, 3, 4, 5 and 6. Now, the problem is I need to have how many control lines to actually set up the connection. Because, I will have some control circuitry because that is required additionally which will control each 1 of those cross points, then I have to also some logic which I have for example, maintaining that only 1 signal should come from to any output port.

So, this output port for example, this one can get a signal from this input this input and this input. So, only one of these three should be activated. So, one in each column only one of the cross point should be activated not more than 1. That is one rule it is possible

to activate more than 1 in a row. So, it is possible that I can activate this scenario this usually is known as multi casting, a multi casting technically means from user 1 I can send it to a group of user the same information.

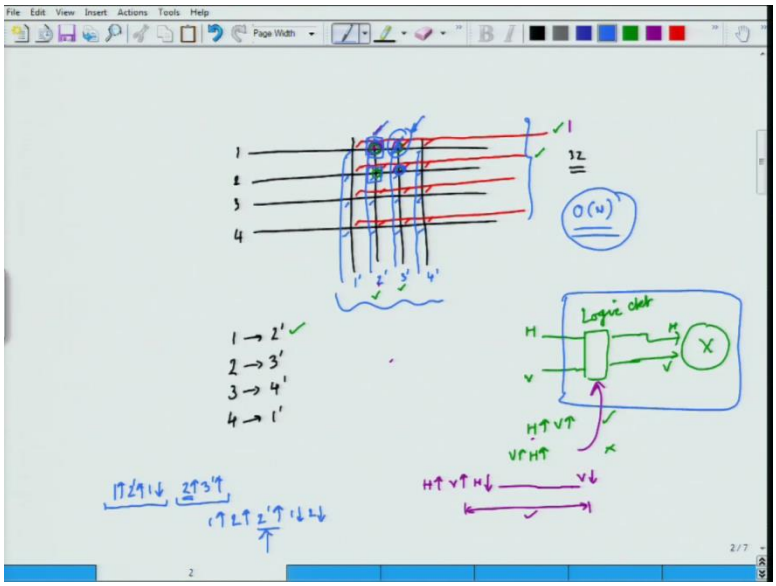
So, 1 can send it to 2, 3, 4, and 5. It is not sending to 6 7 8 for example. So, this multicasting if it is sending it to everybody, this is broad casting. So, that is a different between difference between multicast and broad cast. So, only for multi cast imple[ment]- implement whenever multicast need to be implemented by a switch more than 1 will be active a cross points will be activated within a row.

So, this actually means I require 6 actually control signals. So, there will be 6 wires going out from here, and as you said this actually complexity is O N square. So, has my switch size increases, I require large and larger number of control wires being going out from the control from the controller unit.

Now, how to actually handle this scenario can I make it better. So, if you remember in the earlier lecture actually, I have drawn modified the picture there was a modified version this was not the 1. I had also something called an electromagnets sitting in air. There were 2 control lines. So, 1 is a horizontal I call it other 1 we call it vertical control line, when this is snapped in this actually there is a half way somewhere.

So, you will end up in getting this is will be the new position, when this electromagnet will get activated this will further move, and you will get this position. So, then the snap will happen. So, unless you activate both the horizontal and vertical the connection will not be made between the 2 points. So, cross point will only operative both the electromagnets are operated with 1 it will not. Now, how to solve this whole problem actually is an issue now, let me try to set up a certain connection if I do it this way.

(Refer Slide Time: 10:22)



So, I am drawing a 4 by 4 cross bar. So, that technically means again I can actually have four lines, but that actually means for all the 16 cross points I required 16 into 2 32 actually control wires. But, this is actually being doubled, but may be this is possible let me try it out. I can build up a control 1 control for the complete horizontal line, and this is controlling all the electromagnets here.

Then for the 2nd row again I may have 1 single control line controlling all the electromagnets. So, this is the way actually this can be done, and I can use similarly the vertical lines for controlling the vertical component of the electromagnet. So, number of control wires are required are here is 4 into 2 which is 8 which is O N the complexity is linear here it is not N square.

Now, can be actually work with this kind of situation. So, if for example, I want to set up a connection between say 1 to 2 prime, 2 to 3 prime. So, may be probably let me check if I can make all connections are not. So, and remember these are not symmetric usually whenever you make connection from 1 to 2 the other side of connection will be from 2 to 1. Because, our voice is always bidirectional and I am assuming this connection to be unidirectional.

But I am actually take it has generalized switch I am not putting the symmetric conditions which are usually 2 for most of the voice circuits. So, voice circuit is always

set up in both the directions. So, that when any 1 of the 2 users are talking, other person can listen to it. So, it is actually a bidirectional communication between 2 end points.

So, if I want to set up 1 to 2, what I will do is; I will excite. So, I am going to actually put excite 2, I will excite 1. So, this will be snap in, and this connection will be done. I want to set up 2 and 3. So, I will excite 2 here, and 3 here, and this will snap in, but there is going to be a problem here. So, the problem here will be when I am going to excite 2. So, this will also excite this thing because column is already excited. And whenever I am going to excite 3 this is also going to get excited.

Now, this is a complication. So, maybe we need to put up some other rule. So, what we will do is; for any cross point. So, I am going to have 2 control wires, but I can put up a logic circuit this will then create I am now going to create a logic circuit here in between. And one of them will be creating controlling horizontal line other one will be controlling a vertical line of the cross point.

Now, this logic circuit will ensure that, if the horizontal line is then activated first, and vertical line activated second. Then only H and V both will be activated on the outgoing line if only H if V is done first, and H is done later on then this will not be activated.So, this will cause the activation this will not cause the activation. So, once you do this what will happen to the same switch.

So, let me see what is going to be happen. So, if 1 2 connection I am going to set up 1 is this, I am now doing horizontal first. And 2 I will do second. So, horizontal vertical the connection should snap in.

So, connection will snap in this will work 2 and 3. If I do 2 here 2 been activated. So, this line gets activated. And then what I will find is here the vertical this 2 was activated first. So, this cannot be activated actually. So, this vertical connection will not snap in, only this will snap in, but when I am going to go for 3, when I will activate 3 know be[cause]- for this particular connection, I will find that now the condition H V because 1 has been activated first and three has been later.

So, this connection as well as this connection will snap in. So, from 4, I am able to reduce it to 3. So, I have to do further innovation if I want this thing actually work otherwise this system will not work. So, I need to modify the logic circuit, and may be

what I can do is; one of the options is I actually activate a horizontal line first then vertical first then would put down the horizontal line, connection will remain connection set up and horizontal and vertical both are up. And once, you put horizontal line down connection will be maintained.

And when the vertical line will be now put down then the connection will be closed. So, this is the period, when the connection will snap in. So, I can use this particular logic, and this can be programmed in this logic circuit actually. So, once you do this let me see what is going to happen. So, when 1 is going to go up which is horizontal, 2 is going to go up and then of course, is like 1 is going up, 2 is going up, 2 prime, and then 1 is going down.

So, once this happen this connection will snap in. Now, when you are going to activate 2 after this there has to be all these lines are they are active. And now only depending on which line I am going to now go will go up, it will depend on that. Now, I am going to actually put 3 prime will go up, but 1 is already down.

So, 1 up and 3 up condition will not happen. So, 1 up and 3 up condition does not happen. So, this will not snap in this will not snap in, only this particular part will snap in, when 2 3 will happen. The other condition is for this 2 prime is up, but 2 prime is not going up after 2, this 2 is going up later than 2 prime. So, this cannot this connection also cannot snap in only these 2 connection will snap in, and they will remain till the time 1 at time 2 prime or 3 prime which are of. So, these will actually control the connections.

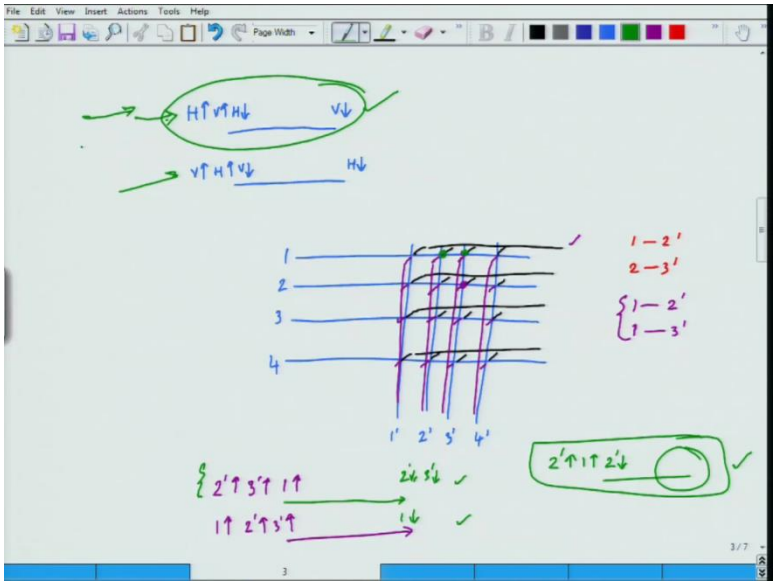
So, from many other input you will not able to set up the connection to these outgoing ports. So, only one of these columns can remain activate any point of time.

Only thing which you have to ensure is you should never do 1, 1 up we should never do then 2 up, and then if you are going 2 prime up, then there is a problem actually. So, if you do 1 up, and 2 up, and then 2 prime up, even if you then actually laid down 1 you go down 2. In this case these both these particular points and these point both of them will remain snapped in because, that is what they will be happening at this point.

So, every cross point will be snapped 1 at a time, that is a condition if you maintain this algorithm can work very well to set up the connection in case of a cross bar matrix. And you require only 4 horizontal control line, and four vertical control lines, and complexity

is going to be O N as for as the control lines are concerned. Only thing is that with the every cross point you require some additional logic circuit which is nothing but technically logic circuit is a sorry it is a sequential logic circuit. And this can be built using some flip flop and other thing. So, I think all fundamentals of basic electronics this electronics can be used here to build up the system.

(Refer Slide Time: 19:43)



So, that is how is the cross bar used to actually work. But, so far what I have done actually very smartly, what I have done is I have always said horizontal goes up vertical goes up, and horizontal goes down. And the connection remains on when the vertical goes down, the connection is closed. You should ask a question, why it cannot happen this way? That vertical goes up horizontal goes up vertical goes down connection remains, and when horizontal goes down connection gets closed.

Let us see, what will happen if I am going to use this particular strategy I am going to take same 4 by 4 same connection sequence, and let us see if I can do that I can handle this connection. So, I am going to now do the vertical thing first. So, my connection set up pattern in 1 to 2, 2 prime and 2 to 3 prime. So, that is a pair which I am trying to set up.

So, 1 to 2 prime if I want to set up as per rule that I have to activate 2 first. So, if I do activation of 2 first, 2 prime goes up, 1 goes up, 2 prime goes down, connection should remain on till 1 goes down. Now, at this point I am going to now, put when this

connection is up that time let me try out 3 prime goes up, 2 goes up, 3 prime goes down in the next connection set up.

Let see what happens in this scenario. So, when 2 is going to 2 prime is going to be activated, 2 prime gets activated, 1 prime will remain activated at this point of time in this row. So, this is control lines. So, you will do 2 prime first as you can observe from here, and then 1. So, this connection will snap in, and this 1 line is now active, this is active at this point of time 2 prime then goes down, when 3 prime goes up.

And so this, and this all thing get activated, but 3 prime has not been activated first. So, this line will have nothing to do. This will not get activated, and once 3 prime is not activated 3 prime is activated 1 prime has 1 has not been activated first, it has to be activated later on if this point has cross point has to be activated, whenever you will excite 2. So, this point will snap in, nothing else can actually snap in this line is not active only 1 is active here. Now, 2 is got activated 3 prime you will put down, and this line will remain on hold. So, both these connections will work. So, this actually does work. There is no issue in this case, but only problem in this case is you cannot implement a multi task thing. Suppose, I want to set up 1 to 2 prime, and 1 to 3 prime kind of scenario, I want to set up both these connections now let see whether this is possible or not possible.

So, in this case I need to activate remember, I need to activate vertical first. So, I will activate 2 prime. I have to activate 3 prime I have to activate 1 prime, and then of course, 2 and 3 will be activated first when 1 prime is activated these both will snap in. Let me show it with a different color, and then 2 prime can go down 3 prime can go down. And so for 1 is their being on hold the connection will both of these will remain snapped in.

But, these both have to be done simultaneously now, if I am going to use horizontal vertical horizontal then what will happen? The same scenario I have to now activate 1 first 2 prime, 3 prime and do a control. So, both of them will snap in, and then 2 and the 1 has to go down. And then these 2 points will remain you snapped in till the 2 vertical lines which are going to be there.

You need to actually hold down to 2 vertical lines which are there. So, I think these both of these systems do work without any problem, except in certain scenarios. But usually what is preferred is this horizontal vertical horizontal, and then output line is what is

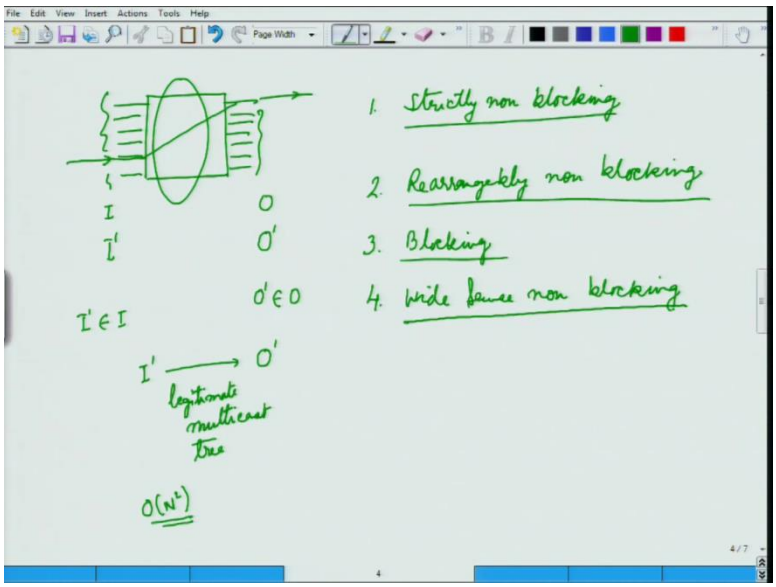
going to be used for controlling in because once you snap the connection, you are not going to set up any connection to the vertical line.

Now, remember in this scenario of multi cast, which I had actually used earlier here vertical has been used first. I have been setting both these multicast simultaneously. Suppose the first point has been already set up wherever; I have done 2 I have done 1 and 2 prime has been gone down connection is up. And now I want to set up from 1 to three also another multicast connection.

Now, that will not be visible in this scenario, but that will be visible if I am going to use this particular case. So, usually most of the implementations were actually have been built, with this kind of thing. Of course, the way to handle this particular situation is if you are using lower kind of control system is that if you want to set up if already set up from, 1 to 2 prime already a connection. And now you want to also set up 1 to 3 prime when this connection is active you dismantle both the connections and now do a set of simultaneously. So, you have to break up an earlier connection that was the only way this could have been done here.

While in the second case or with the first, this particular control strategy this can be done without any issue without breaking the earlier connection. So, this was the most popular variant which was actually used. So, this is how the cross point or cross bars usually used to work, and this forms the basic unit of the switch.

(Refer Slide Time: 27:02)



So, usually the switch are defined in this way there is a input port there is a output port. And important property of this particular switch is it is a strictly non blocking. Now, I have actually now define these terms because in the whole this lecture series you will be actually listening to these terms very frequently.

So, before going on further. So, we define something called a strictly non blocking that is a first thing. So, meaning of this is if you are having input port, which is free. If you are having an output port which is free. An irrespective of whatever other connections which have been set up between other input output pair.

I will be able to always set up a connection or a path between this free input port free output port without disturbing any other connection. So, a formal definition will come once we will define a clause theorem, which is for a speed blocking thing is basically says that, if there is a set of free input share and set of free output share.

So, it is talking terms of set. So, there is a I prime if I is the total num[ber]- total set which is their total number of inputs O is the output. So, O prime is free outputs, I prime is free inputs. So, I prime is a subset of I, O prime is going to be subset of O. So, I should be able to create a legitimate multicast tree, from any input which belongs to I prime to any subset of O prime, any subset of O prime. So, there is a legitimate multicast tree which can be created.

So, that is a most general definition of a strictly non blocking, and this has to be done without disturbing any existing connections, that is what the strictly non blocking means. Now, second variety or say kind of switch or property which is known as rearrangebly non blocking. So, this says actually the same thing can be done, but the condition that I will not disturb the existing connections cannot be maintain.

So, existing connections might have to be disturbed, but will may set up the same connection there will be broken for some time. Then I will again set up the connections and once, the new connection is set up the input to output mapping will be still maintained same. Only there will be disturbance is which will happen. So, I might I might be rearranging the structure in between, and then this will become rearrangeble non blocking. So, in that sense if you see when I was looking at this particular case.

If I you are going to use a strategy V H V, V up, H up, V down, to set up a connection this is actually rearrangably non blocking switch, because I am going to disconnect, and then connect for such kind for creating certain multicast tree. If from 1 input you are connected to 1 output connection is already on functional, and then you want to create the same input to another output free output, then you have to disturb the earlier connection and then you have to connect again.

So, in that sense you sets a rearrangably non blocking. So, usually whenever I am going to use a block, I am going to use a call it a cross bar, I am actually us[ing]- assuming that. I am going to use a strategy of horizontal control lines first sorry horizontal control line first, vertical then and then horizontal down then connection is set up till the vertical line also [pulls/pushdown] pushdown actually.

So, that is means we will be making the strictly non blocking. So, rearrangably non blocking also I think now is clear, we have to do the rearrangements, but any legitimate multicast tree between input and output pairs can always be created, but it quite disturbance of existing connections.

But existing connection mapping from input output will be a still can made, but connection will broken for some time. But, not necessary all the time, but some of the time it will be done it cannot be guaranteed. In strictly non blocking it will be guaranteed now that thing which is going to be blocking probability blocking switches. So, blocking switches means there is input is free output is free, but as something is not there in between the switches not possible to set up the path.

In cross bar of course, you would not see this, but cross bars we do not use I will come to the reason why we do not want to use this. Because, cross point complexity here is O N square and of course, next question comes is can I make up a switch which is strictly non blocking like cross bar, and it still have a cross point complexity less than O N square.

So, I will give a hint line later on we will actually prove through clause theorem yes it is actually thus possible. So, blocking switch are those switch is where even if input and output is free sometimes, you will not be able to set up the connection. And of course, the 4^{th} category is wide sense non blocking, we will see an example of this kind of switch also. And you will formally prove that yes this is done through actually building a what

we call is state diagram, and say state transition diagram and saying figuring out that if I can avoid certain states.

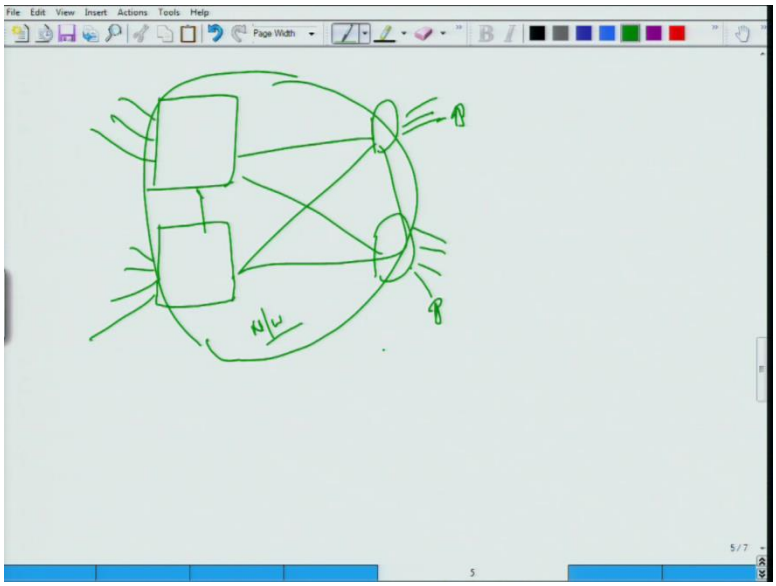
This switch will always remain non blocking switch. So, what happens is basically the meaning is if while setting up the connection I follow certain algorithm. And that algorithm is well defined, and which I ensure that I do not get in to blocking states. I always avoid the blocking state; I will be able to set up all possible connection patterns between free input and output ports.

This usually happens because, free input and output port they can be connected together, in more than 1 possible ways. You have to choose judiciously only if those ways by which switch will always go in to a state, which will always a non blocking state. It does not create a blocking structure.

So, those kind of switching instructions are wide sense non blocking switches, we will also see as in an example of this kind of thing. Now, I have build up the switch a cross bar which is O N square complexity number of cross points which are required are N C 2 if it is only a upper half triangle of the full square. There it is going to be O N square, and it is a strictly non blocking switch.

So, theoretically now the question is can I actually have something better than this or not. Yes it is possible to build up something better clause network provide O N 3 y 2 O nth power 1.5 is the complexity which will come. So, anyway I am not going in to that as of now the only way that O N square now can be further reduced, will have the clause network comes, I am now coming to that thing.

(Refer Slide Time: 34:37)



That if I build up only 1 switch which is sitting on blocking may be it may not work. So, idea is can I actually create and of course, if N becomes say 10000 or 20000 or you say 1 lakh or 1 million or 1 crore how you will are going to set up a switch. So, in fact, here also one should understand the concept of network.

So, what you do is you never use a very large size switch as such you then will you start always going to use multiplally smaller switches, and then create interconnection between them. So, you can actually put multiple switches like this, and create subscriber to that. So, these are subscribers attached and I can inter create an interconnection between these switches. So, this what forms the network.

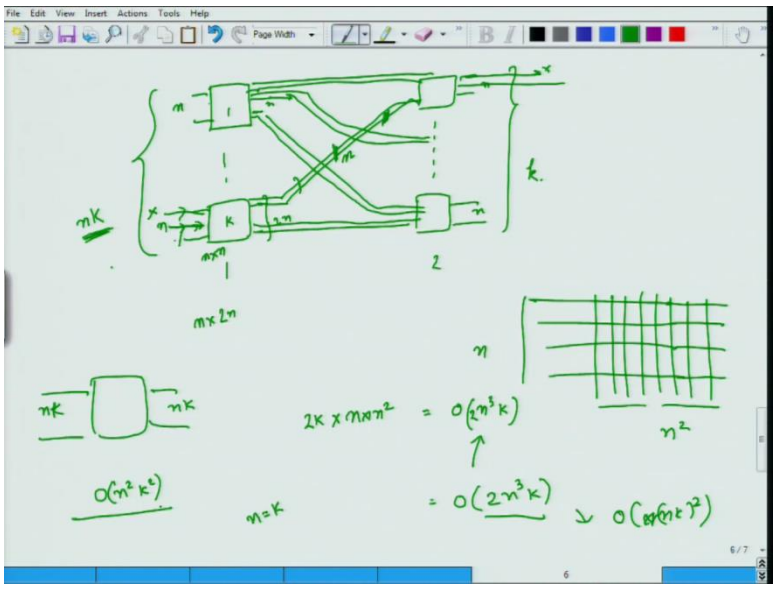
So, all these switching matrices can be there is 1 building or they can be distribute all across the country, all across the globe. And that what forms that actually network. But as far as user is concerned whether you are in a Kanpur here, and you are connecting to somebody in Bombay you see that whole network is nothing, but simple gigantic switch. So, internal network is usually is not visible to the subscriber.

So, in fact, network is technically is nothing, but single gigantic distributed switch which is built in redundancy mechanism and built in maintenance mechanism. So, part of it can fail, but the reaming part will always work even if some part your connections can be routed from other roots.

So, the larger distribution structures are going to be random usually it will be mesh, but what we can do is we can within a small dimension from within a building if I want to create within a organization. I want larger dimensions switch, maybe it is a good idea that I can take a smaller switching elements, switching cross bars, and then join them in certain to create a bigger gigantic switch. And that is a regular connection pattern we call it a switching network.

So, next step is then that we have to see that how the switching is structured is going to be built. So, now, this using small switching elements switching matrices I need to create a larger switch that is basically is the problem first. And I need retain first certain property. So, I think natural thing is that I would like to create a very similar property like we have with the cross bar a strictly non blocking property, and still create a larger switch using smaller cross bars. So, let see if that can actually be done or not done.

(Refer Slide Time: 37:25)



So, may be commonsense will tell I can going to get a cross bar, some value is n by n and I can use multiple of them. And I am going to try with regular restriction, how this will be done. So, simple idea is that there will be n input and there are 1 2 k. So, now, total size which is going to be available is n into k now, but if I actually make this some input here cannot be connected to this output there is no path between them. So, I need to do something. And secondly, I also want is the path length from here to here input to output has to be exactly same almost.

So, one possible way is I can now, because I want n k inputs and n k outputs. So, let me also have similarly the other cross bar. So, which also have N outputs, and they k such things which are there. So, there is 1 stage 1 which is input is stage 2 which is output and I want to create a connection between them. So, that everybody can talk to everybody else, but I want to keep them strictly non blocking, that is basically is the idea. So, this is a kind of switching network which we are going to create, but it is a structured identity.

Now, each of this input in this particular switch should be able to connect to any 1 of the outgoing port, which usually is the condition. So, what I can do is I can create it as n by n. So, there n outputs, and their n such switch. So, I can guess distribute them. So, 1 wire going to each 1 of them. So, I can do same thing here. So, perfect I have done this, and their all free ports I have used as input port and output port. Now, this switch in the question is whether it is a blocking switch or strictly non blocking switch whether it is a rearrangably non blocking switch.

So, we can actually observe if somebody has already set up a connection from here to here this path is busy. This switch per say the element itself is strictly non blocking, but this path gets occupied next free input port, and next free outgoing port they can there connection between them cannot be made because, this path is not available this path is already occupied.

So, this makes a blocking switch. So, may be than idea can be instead of using n by n here, can I use n by 2 n. So, let there be 2 lines n. So, I can put 2 lines all the way coming to everybody, I can do it. Now, this is a different kind of cross bar cross bar earlier time I have told you was only having 4 incoming and 4 outgoing. So, I can actually have now, 8 outgoing and 4 incoming.

So, I can put that cross bar now, I can set up 2 connections, but when I try setting up 3rd connection oh it is not possible actually. So, ultimately what you will do is if I want all the n connection you want to go to this end connection I require, how many lines here actually that is very important. So, in worst case if all these ports are connecting to all these outgoing ports from switch 1, from switch k, I require here N connections.

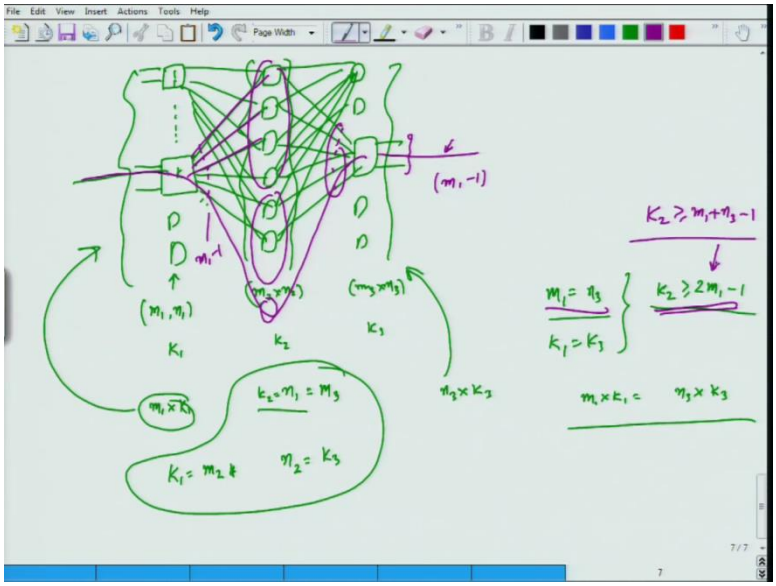
So, you will be building up n, n square here because you require n for this n for the next switch n for this total n square outputs. So, you will end up in n into n square switch, and

you require 2 k such switches. So, number of cross points will be required is n cube 2 k. Now, the whole thing can be equivalently represented by a switch off n k by n k.

So, which will have a complexity of O n square k square. In this case my switch complexity is 2 n cube k, and of course, if it turns out to be that n is equal to k both of them are going to be technically same, and both will be strictly non blocking switches. So, even if 2 stage actually it is not possible because, there is no other way you can make the interconnection. And this is the only way you can build up a strictly non blocking switch. So, complexity is going to remain same it does not change.

So, because if you put k is equal to n this will turn out to be Oo of n k square here also it is O and n k whole square. So, maybe you have to go to a still better mechanism. So, let see how this actually be done. So, may be with 2 stage this is not possible, but third stage is certainly is possible, and that is where the clause network actually.

(Refer Slide Time: 42:57)



It has come in. So, idea is this that I will have a switch I will have multiple of them, and then I can create many such switches. Now, this switch can route the calls in this way there is exactly 1 path to each of 1 of the switches, and I can connect similarly this thing to each 1 of them and so on.

So, idea is that you will actually have m 1 by there will be I will define actually m 1 by n 1 switch here, k 1 of them k 2 switch is and k 3 switches. So, because there n 1 outputs.

So, k 2 should be equal to n 1 that is how you can give, no input port or no output port here should be left free only input ports are free here, and output ports are free only here.

So, that is a condition and of course, total number of input ports will be m 1 into k 1. These are the number of incoming ports, outgoing ports are n 3 into k 3. So, these switches are m 2 by N 2 these switches are m 3 by n 3. And since we are talking about a symmetric switches these both terms m 1 in to k 1 should be equal to m 3 into k 3. So, these switches should be same and you can see that k 2 has to be equal to n 1. So, that each 1 of these out port is connected to 1 switch in next stage. Similarly, you can see that m 2 should be equal to the number of incoming port has to be equal because 1 line is coming from each one of them. So, k 1 is equal to m 2 similarly, there should be equal to n 2 should be equal to k 3 and of course, k 2 is equal to n 1 is equal to m 3. So, this condition should be satisfied.

So, this kind of restriction is known clause network. And of course, intuitively or we will actually formerly prove it this switch will be non blocking, if I am taking a case where m 1 is equal to m 3 its symmetric thing. And when m 1 is equal to n 3 and k 1 is equal to k 3. So, in that case when you will have k 2 greater than or equal to 2 into m 1 minus 1.

If this condition is satisfied this switch will be strictly non blocking switch the logic is pretty simple that there is a free input port here which I want to connect I will show it with a different color. There is a free input port here, which wants to get connected to this one. So, remember there are only these many links which are there, which will be nothing, but so if all output ports are occupied except this 1 which I want to connect you already have because, I have not taken m 1 is equal to n 3.

So, m 1 minus 1 these links are occupied. Similarly, on this side also m 1 minus 1 links are occupied in worst case these might be choosing a set of nodes here, these might be choosing another set of nodes here. And there is no overlap between them. So, I need only 1 more extra which can be used to set up this particular free path to this, and this will always guarantee that in worst case I am able to set up the path, it becomes strictly non blocking. That is basically is the philosophy which will actually mean is m 1 minus 1 plus m 1 minus plus 1, k 2 has to be greater than or equal to that.

That will give me the, a strictly non blocking condition these also what is the clause theorem technically is. In fact, there is a generalized form for that it says that k 2 has to

be greater than or equal to m 1 plus m 3 minus 1 actually. So, this is a more refined form. So, I have take it has taken a very simplified version of the same thing. Now, with this you can actually compute the compute the cross point complexity.

So, here I am actually now finishing and I think I urge all of you to try to make an estimate of what is going to be the cross point complexity for this configuration. This is now as clause network configuration and then you will actually figure out, and appreciate yes this is going to be have a smaller complexity then a cross bar. And I am still able to create a strictly non blocking switch. So, we will continue with the same thing this particular thing onward in the next lecture.