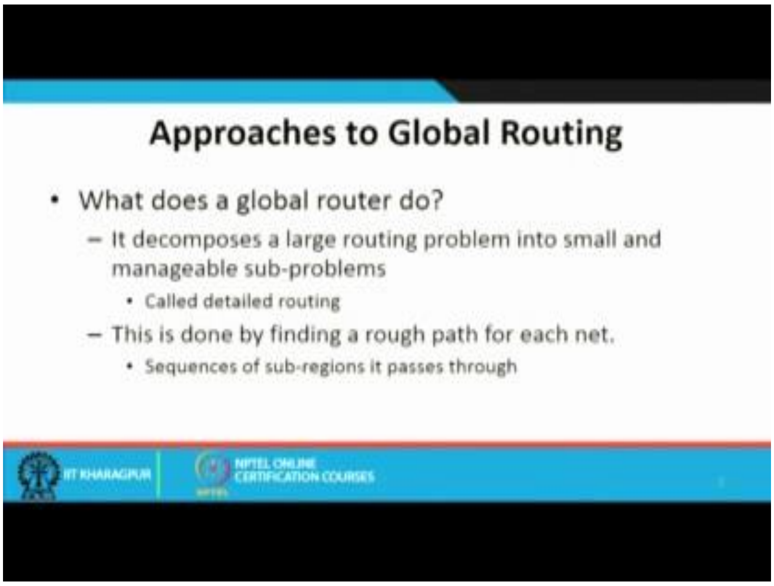
VLSI Physical Design
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 19
Global Routing (Part II)

So, in the previous lecture, we were looking at some of the graph based data structures that can be used to solve the global routing problems. Now in this lecture in this present lecture we shall looking at some other approaches namely I mean in set of bottom up and drop down hierarchical approaches. And also some approaches which are based on means optimizing some kind of an objective or course function. We shall be looking at some of those approaches which are used in practice.

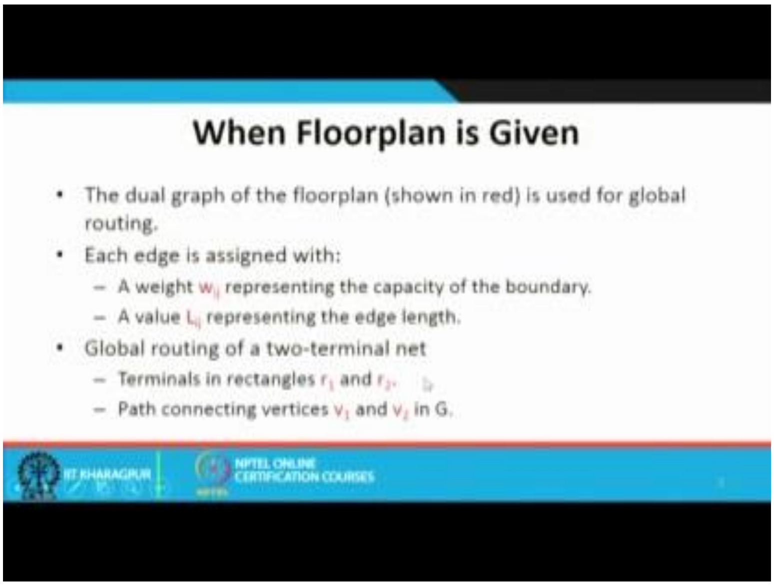
(Refer Slide Time: 01:00)



So, broadly speaking, this already we have discussed earlier in detail global routing, what does the global router do. It tries to manage the whole routing problem using divide and conquer kind of an approach. It tries to decompose a large routing problem into smaller sub problems, which can be solved or handled one at a time so global router takes the whole routing problem in its entirety, and it breaks it up into smaller sub problems which are then fed to the detailed router one sub problem at a time. For example, a channel routing sub problem so the channel router will look at only that smaller sub problem and there are very good algorithms available for channel routing it tries to solve it and then it moves to the next one.

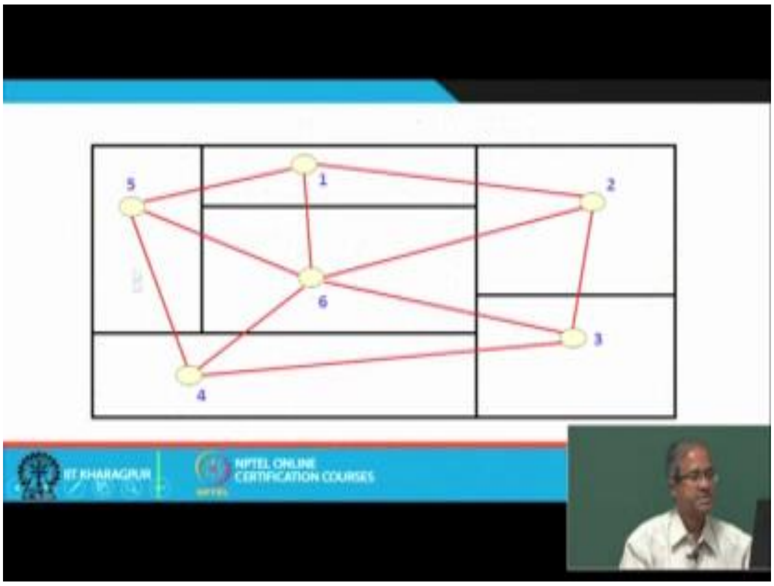
So, this makes sense because instead of considering the whole global problem where there can be thousands and thousands of nets and so large areas so many blocks. So it would be extremely complex to handle. So as I had shown using an example in the last lecture this can be done by determining rough paths for each net that needs to be connected, which actually means this sequence of the channel or the sub regions it needs to go through.

(Refer Slide Time: 02:44)



Now, let us come to discussion on how we can go for global routing in scenarios where our floor plan is given or when our placement is given. Suppose my floor plan is given there are some design styles they are said where floor planning and placement does not make much difference. So let us say I have a placement graph available with me or let us say to start with the floor plan graph. From the floor plan graph how I can handle the global routing problem. So let us take an example for this case I shall show it.

(Refer Slide Time: 03:31)



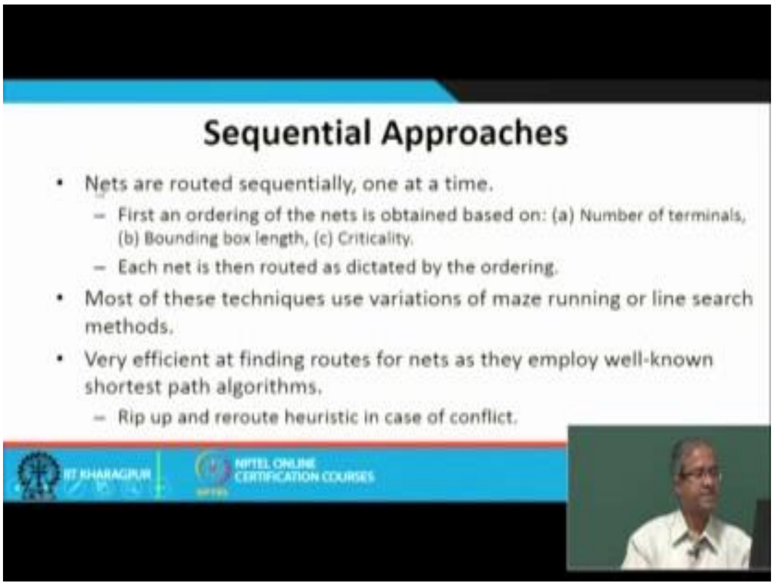
So, I am showing the duel graph of the floor planning like this. Let us say so here I have a floor plan where there are 6 blocks which are placed and super imposed on the floor plan I am showing the duel graph. So you recall the definition of a dual graph. Each rectangular region corresponds to a vertex. There will be an edge between 2 vertices if there is a common boundary between the regions. Between 5 and 1 there is a common boundary 5 and 6 there is a common boundary and so on. So this duel graph can be used for global routing the idea is as follows for every for every edge you can define a weight that represents the capacity of the boundary, like you see if the blocks are of shape like this is a quite long boundary which means the number of connections are possible between these two. So I can have a higher weight for this edge. Similarly, between 1 and 5, the width of this boundary is quite narrow. So I will have a smaller weight for this boundary. So the weight of this graph will give you the capacity of the corresponding channel or the boundary.

And of course, the edge length will be represented by this L ij, so both of these are important. Now when I talk about global routing, so we have 2 terminals located in 2 different blocks, so you have to find out a path connecting them; like you see in this graph suppose I have a pin located in block 5, I have to connect it to a pin which is located in block 3. Now this graph will be having some weights in each of the edges. Now higher weight means that is a more preferable path to follow because that boundary has higher capacity, we are trying to find out a path which has the maximum weight, there are many graph algorithms which are existing for that maximizing the path weight.

For example, for this problem maybe you see that your path weight will be this. This will be your best path 5 to 4, 4 to 3. So once you have it, then you have a global problem global routing problem defined. For interconnection we will have to first go through this channel, let us call this c 1, then you have to go through this channel will call it c 2 and here. So here every edge essentially indicates a channel. So a path between 2 vertices which is essentially indicates a sequence of channel it is going or passing through. So if you have the duel graph corresponding to floor plan then directly the path will give you the sequence of channels that you are traversing in order to get the corresponding path.

Now, suppose I mean we have the placement given to us. So when we have the placement given to us then the routing region space is already defined. This is like the channel intersection graph. So this is quite similar to the channel intersection graph which we have already said. So we can use this so means if we have the placement given to us, but not the router if we have the floor plan given, but not the placement then you can use this approach, but if you have the placement given then you have a routing graph which is nothing, but the channel intersection graph that we have discussed where the vertices will represent regions it will correspond to channel edges represent adjacency between channels or the duel of this either way. The dual graph is a channel intersection graph.

(Refer Slide Time: 07:54)



Now, you can follow a sequential approach like this. Given a set of nets I will assume that the nets are all 2 terminals in nature. In every net you are trying to connect 2 points right. 2 points you want to connect that is a net. So you can order the nets and handle them sequentially one at a time. The ordering can be based on a number of heuristics like number of terminals. So a net which contains largest number of terminals can be given higher priority.

Bounding box lengths, the nets which are furthest away from each other that I mean the nets which connect points which are furthest from each other that can also be given priority and criticality. Some nets which are already identified as critical by the user can be given in a higher priority for interconnections.

So, you see there are 3 things. Nets that connect more than 2 points larger number of points can be given priority, nets which connect points which are longer distances from each other that can be given priority, or the nets which the user has mentioned or specified as critical which would determine the overall performance of the system that can also be given priority, so that the best path can be found for them. For the others may be you will be founding a roundabout path not so good that can be accepted.

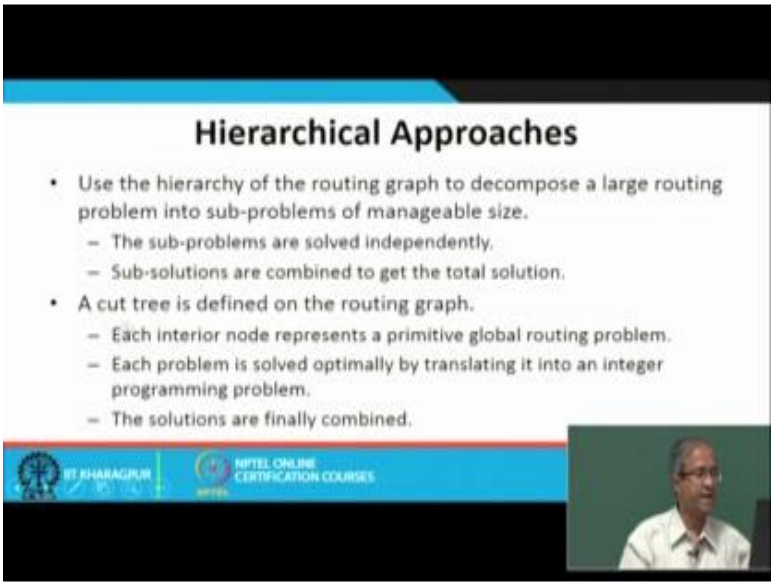
So, this sequential approach is used typically variations of maze running or line search methods. And there is another you see means I will just mention this there is a very commonly used method for rip up and reroute if something goes wrong. You see now what will happen.

Typically, this situation is like that say you take a routing problem try to solve it using the methods that we have discussed global routing followed by detail routing. At the end you may find that some nets, set of nets which you are still not able to route. So for those as the last resource you can use the maze or area routing algorithm to find a path if from anywhere it is existing. Because when you are using or into the detail routing phase you are only considering one channel at a time. You are not looking at the global scenario anymore, but if you see that the there are 2 points which are still to be connected you first try any of the maze routing algorithm to see that whether you are able to find a path. If so sine, otherwise people typically use, well it is a last resort it is called rip up and reroute.

Rip up means you see many of the routing decisions that we take are based on some heuristics. Which means if you run it again may be you will be getting a different solution. Rip up means some regions where you find that there is some congestion you withdraw means all the nets which have been routed and again redo it. Maybe the next time you do it you will be coming up with the better solution you will be coming out with different solution which can be better, which may be accommodate which may be able to accommodate with this new connection as well.

So, rip up and heuristic works pretty well typically, and it is a favorite for many designers when they are into the last phrases of the physical design.

(Refer Slide Time: 11:49)

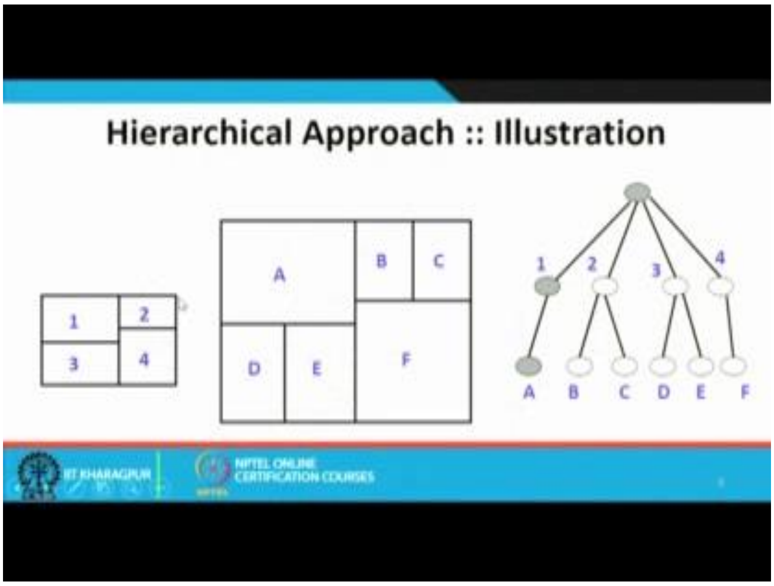


Well of course, you can go for the hierarchical approaches. So here what you do? Here we use the hierarchic of the routing graph. So again this is a divide and conquer kind of an approach it decomposes a large routing problem into smaller sub problems, which can be solved independently. And once you solve the smaller sub problems independently, you will have to combine them together. Again so you combine them together to get the overall solution right so I will show you an example. So you define something like a cut tree which is quite similar to the tree that we have discussed when we were talking about the partitioning problem. That we define some kind of a partitioning tree and you can cut it get some partition something similar to that.

So, the cut tree is defined in such a way that every interian node of the cut tree represents a global routing problem. And each of these global routing problems can be solved independently. Now this smaller node indicates global routing problem which is smaller in size. You say we shall see later that how the routing problem can be translated into integer programming problem. See integer programming problem the advantage is that you are mapping it into an optimization problem which you are solving using some available tool solver tool.

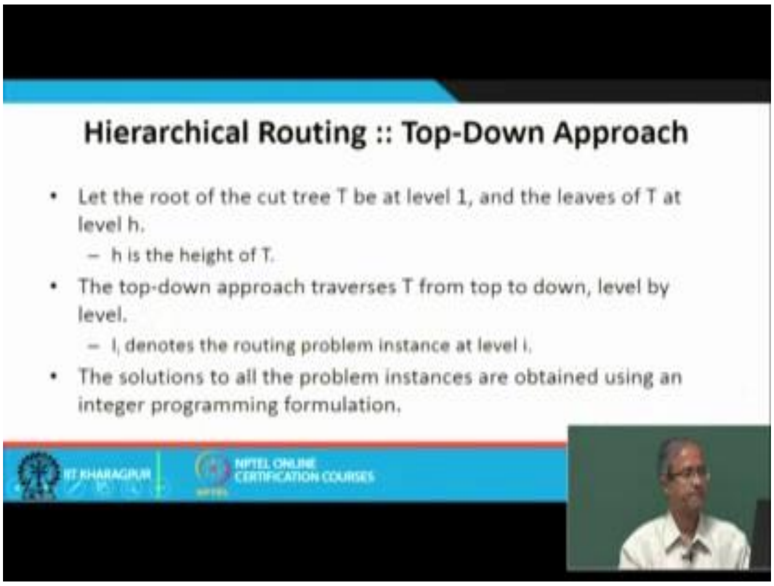
Now the advantage is that once you get a solution, you are guaranteed to get the best solution, but the problem is that you cannot use it to solve large problem instances. You can use it only for smaller sub problems, so that whenever you are using a divide and conquer kind of an approach it means it actually makes sense whenever smaller sub problem you can use integer programming to solve it optimally. And when you have many such sub problem to sort you combine them together to get the solution of the total problem.

(Refer Slide Time: 14:15)



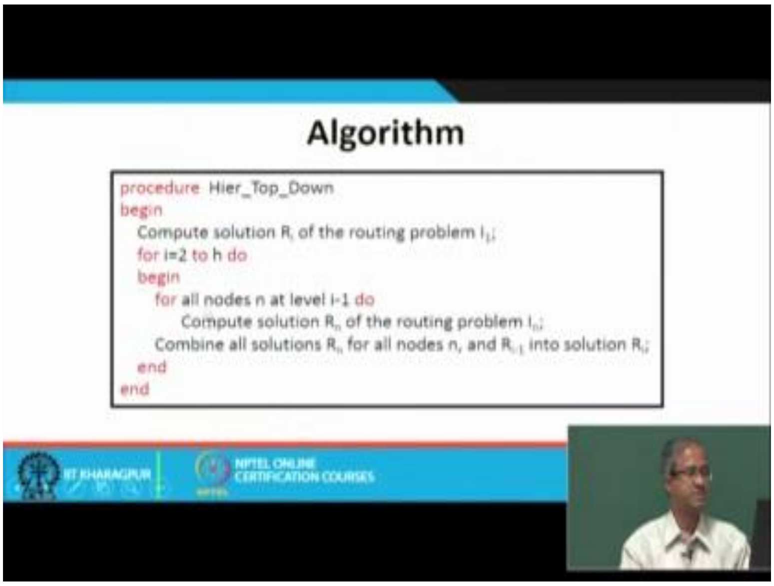
So, very rough idea behind it; let us say initially the total routing problem they can be divided into 4 sub problems with respect to the regions on the layout surface. So this is similar to partitioning. So in this method as if we are using means partitioning routing in hand in hand way so 1, 3, 2, 4. Then again 1 in the next level can be mapped to a block A, 3 can be mapped to 2 blocks D and E. 2 can be mapped to B and C and 4 can be mapped to F. So this can be represented by a graph like this where each of these nodes represents individual routing problems. Like for example, this B is a region containing many blocks. You solve the routing problem here, sort the routing problem here, combine this B and C, you have solved 2. Similarly, you combine D and E, you have solved 3. In this way you go in a hierarchical fashion and you can solve the sub problems.

(Refer Slide Time: 15:24)



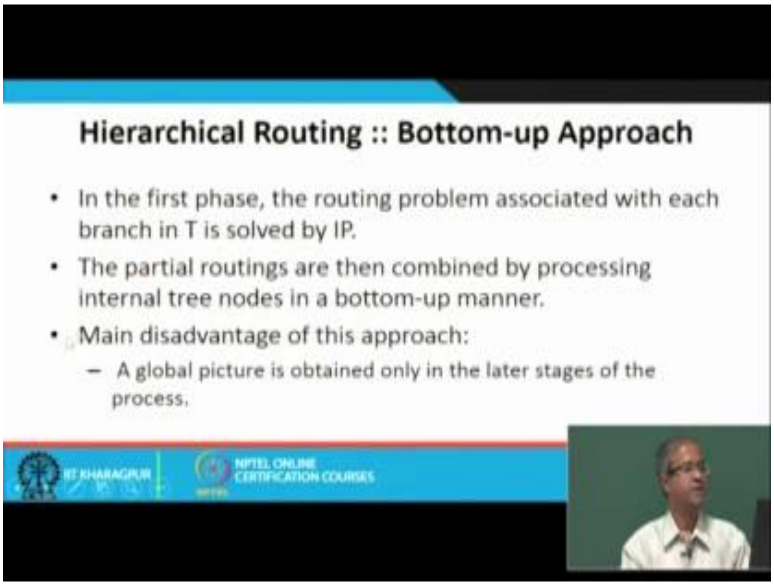
So, the top down approach say, so this is the cut tree which is called so you can cut it at any places. So the leaves I am assuming that the level h and root is at level 1. So you traverse the T from top to down level by level, so it is like this. This is a top level view you go down you get a more detailed view. If you want to further break this, you go to a more detailed level. This is top down. So the problem instances at level i at every stage you can solve using an integer programming problem, because they will be manageable and smaller in size.

(Refer Slide Time: 16:09)



So, without going into the detail; roughly these are this is the step wise procedure that you compute solution for the routing problem i 1 for the tree starting from the route 2 to h the leaf nodes, for all nodes at level i plus 1, you compute the smaller sub problems using some methods like integer programming, then combine all the solutions. So you do it for all the level by level. You go on being this, this is simple. So conceptually this is simple.

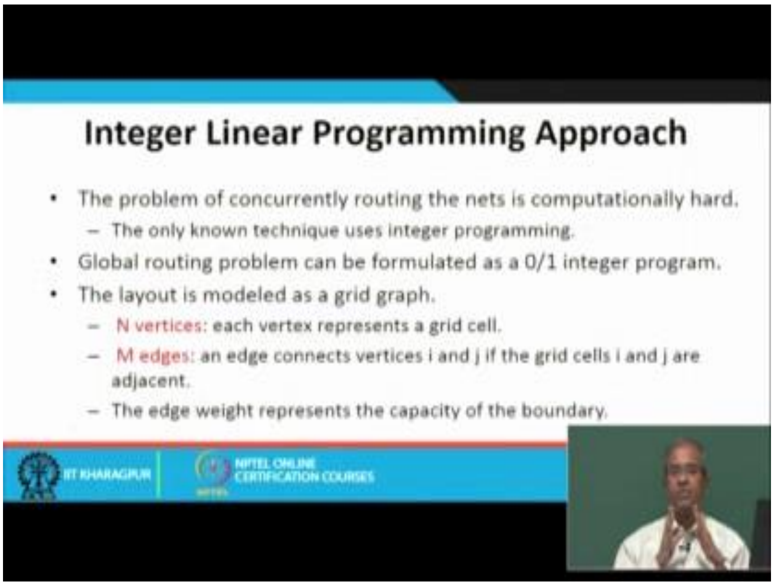
(Refer Slide Time: 16:52)



Bottom up is different. Bottom up you are going from the bottom to the top; that means, the routing problem associated with each branch of the cut tree is solved. Then you combine them to form a node at the higher level. Then you move to the higher level, but the problem is; that means, as compared to the previous approach top down where you started with a global view of the problem you try to divide it up into partitions and try to solve it, but here we are looking at the smaller sub problems and try to combine them together, but you do not know what your final routing problem will look like, so you will get the global picture only much later actually. So will practice it is done some mixture of top down and bottom up can be used for handling large problems.

So, again this is the algorithm for bottom up, so for all nodes that the bottom level you compute the solutions by combining the solution of the children nodes. You go on repeating this from the leaf node down to the root node, bottom up fashion this is just the conceptual steps.

(Refer Slide Time: 18:15)

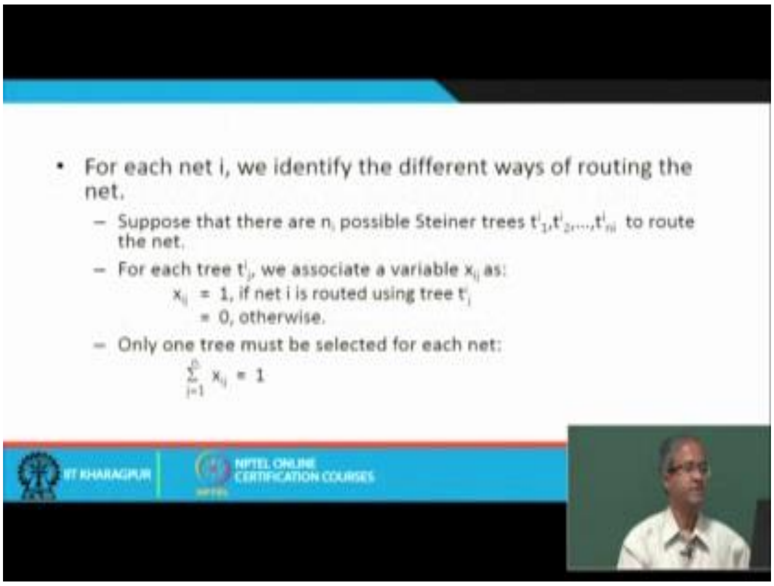


Now, in this kind of hierarchical approaches one thing I mentioned is that, at every stage you have a routing sub problem and you are trying to optimally solve it using some kind of integer linear program. You map it to a integer programming problem and use a integer programming solver so many such solvers are available, you can use it to get a good solution and as I said that you can use this kind of solvers to get solution for small to moderate sized problems, but not for large problems because the time complexity increases quite rapidly.

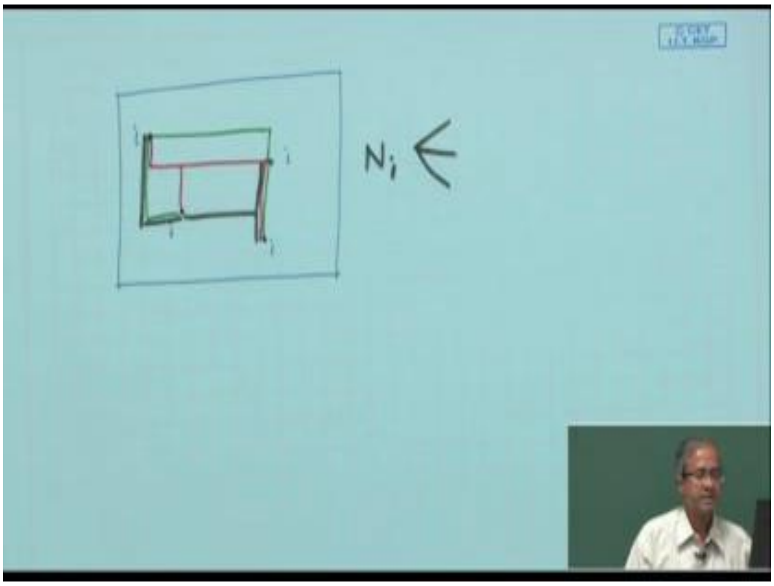
So, the concurrent routing of the nets, here what you are saying is that you are not considering one net at a time, but you are considering several nets together. Now for this kind of a general routing problem like you see I am just repeating so when I am considering a small problem instance, so that problem instance will not contain one net, but maybe there are 10 nets. 10 nets to be connected so I consider this whole problem of connecting 10 nets together, and use integer programming to get a solution. But our other solutions which we have discussed earlier, they typically take one net at a time and try to handle the routing problem and solve it incrementally one at a time, but here we have one method where we can take a problem where multiple nets the routing issue of them can be handled in a concurrent fashion.

So, global routing problem in this case can be formulated as a 0 1 integer programming problem. 0 1 means they variables take on the value 0 and 1 and this solver has to solve for all the variables. Now the idea is as follows the layout is modeled as a grid graph this we can possibly afford because our sub problem is not very large. There are N number of vertices which represent the grid cells. There are M number of edges which indicates the adjacencies between the vertices. And the edge weights as I had mentioned earlier and the data structures represent the capacity of the boundary. Now let us see how integer linear programming formulation can be done.

(Refer Slide Time: 20:57)



(Refer Slide Time: 21:12)

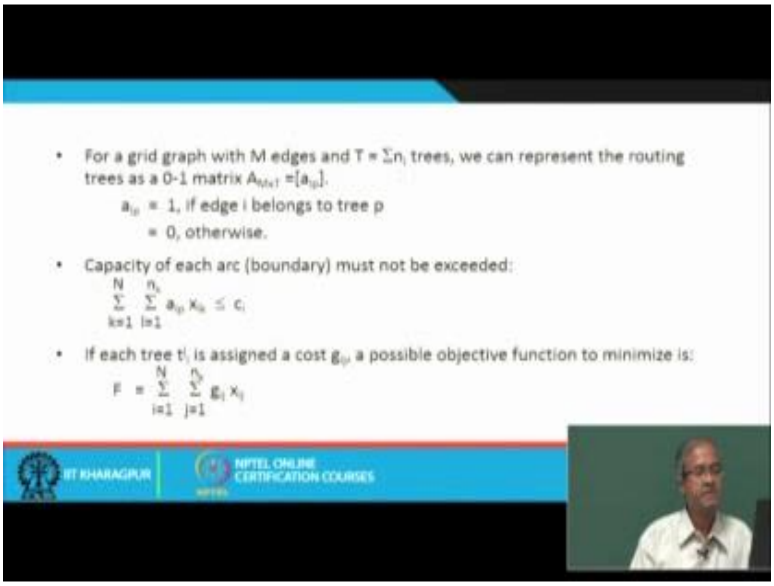


So, it is in fact, it is quite straight forward we do it in this way. Suppose I have a net i let us say let us take an example. Suppose in a problem instance so I have a net i where there are several points. So now, we look at several different ways of connecting them. Well I am just showing several alternatives. So one way of connecting them may be like this. This may be one possible connection. Say another way of connecting may be like this. Say another way of connection maybe like this. So for any net in i there can be several alternatives.

Typically, you know when your layout the words you just model them as Steiner trees. So you can say there can be many Steiner trees corresponding to every net that you are going to layout. This is something you have to evaluate. So here we are assuming that for each net N i there are some N i possible Steiner trees. So there are algorithms to approximately determine the possible Steiner trees for nets. So you use one of those algorithms and let this Steiner trees be denoted by t 1 i, t 2 i up to t n ii.

Well now out of these N i Steiner trees you will have to select one of them to route the net. For a particular net you have to use only one way of connecting them not more than one way. So this you specify like this. You use a variable x ij where x ij will be 1; if net i is routed using tree t ji and 0 for all other j values. So out of the possible N i values or N i trees, only one of them will be having the value one for all others it means 0. This means that I am using the jth Steiner trees for connecting net n. This is one and just to make sure that I have selected this 1 for only one j value, for all others I have made 0. I also add a constraint that sigma of x ij should be 1; that means a single one followed by all other zero; that means, one Steiner tree should be selected for a net. So we can represent the constraint like this.

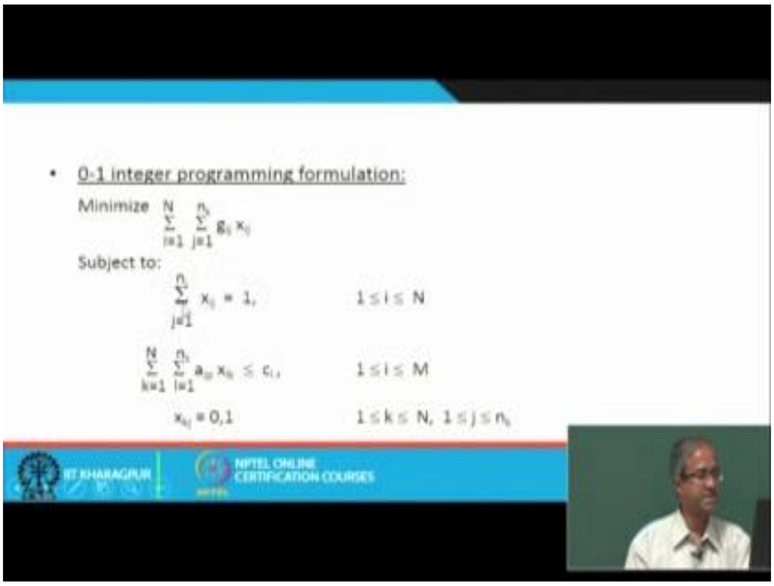
(Refer Slide Time: 24:28)



Now, suppose the grid graph has M edges and there are capital T trees for all the nets. So n i is the number of trees for net I, so there are so many nets I am saying that we want to handle concurrently. So sigma of n i will be the set of Steiner trees for all the nets under consideration. So capital T will indicate the total number of such trees. So the routing problem we can define as follows. We define a matrix capital A which will have dimension M cross T. So M is the number of edges that many rows and number of trees as the column. And the coefficient will be 1 if a edge belongs to a particular tree and 0 otherwise. So this will tell you that which tree is using which edges. So you appropriately set the entries of this matrix to 1. And each boundary arc will be having a capacity c i. You ensure that the capacity does not exceed c i. What you are doing? This a ip multiplied by x, I mean x is which you recall, x is this matrix which net you are using. So only for one of them it will be one for rest it will be zeroes. So you see total number of edges that we are taking that should not be crossing the capacity. This is actually for all the nets.

Not for one, for all the nets. You see that the total capacity of the link i should not get crossed. And if each t i, you assign a cost g ij. Then the objective functions maybe to minimize the sum of the weighted cost for all the tress. So this will be overall integer programming formulation which can be summarized as follows.

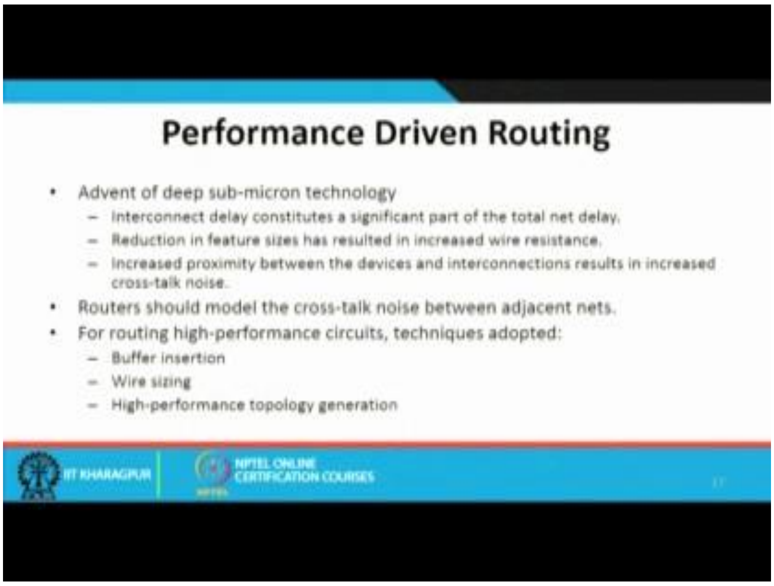
(Refer Slide Time: 26:54)



Minimize this objective function, subject to this condition which says only one of the Steiner trees for each net we are choosing. This says at for each boundary ci each boundary i the number of nets crossing should not exceed c i, and x k j these are all binary values. So if you solve these problem you will get values for the variables x ij, a ip and so on. This will be the output of the integer programming ip solver, ip solver will give you the values of the variables that minimizes the objective function, which is nothing, but the solution that you want.

You will get the solution that for every net which Steiner tree you are selecting and for each Steiner tree which edges you are using. So the matrix is a and that x will give you that information. So this is just one thing that using integer programming, you can get the best solution for routing sub problems, if provided number of nets and the size of the nets is not very large. Then you can use this to get an optimum solution.

(Refer Slide Time: 28:18)



Now, of course, there is another important issue which will be dealing with in detail later when we talked about performance driven issues like timing and delays. See with the advent of the deep sub-micron technology as the device or the transistors are getting smaller and smaller, the delay of the interconnection is starting to dominate. So earlier the gate delays were higher, the interconnection delays were less small, but now it is becoming the other way round. The interconnection delay is becoming as important as the gates sometimes it is also dominating.

So, this cross talk noise issues we should also be handling this. So, increasing proximity between the devices because of the small sizes and increased density, so this has also resulted in increase in cross talk noise issues so modern day routers should also consider the cross talk noise. So we shall see later that we can introduce buffers to reduce noise across long segments. Wire sizing some wires can be made more wider as compared to the other and topology generation is something that also we shall be looking at.

With this we come to the end of this lecture. Now in the next lecture we shall be starting next week. We shall be first looking at the detailed routing sub problems and some of the techniques which are used there. Followed by we shall be moving into the timing related issues. We shall be starting with some discussions on clocking and how we can route the clock nets and so on. After which we shall be going into some other timing related issues like static timing analysis cross talks etcetera.

Thank you.