

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

Lecture No #27
Backend Design: Part – XIII

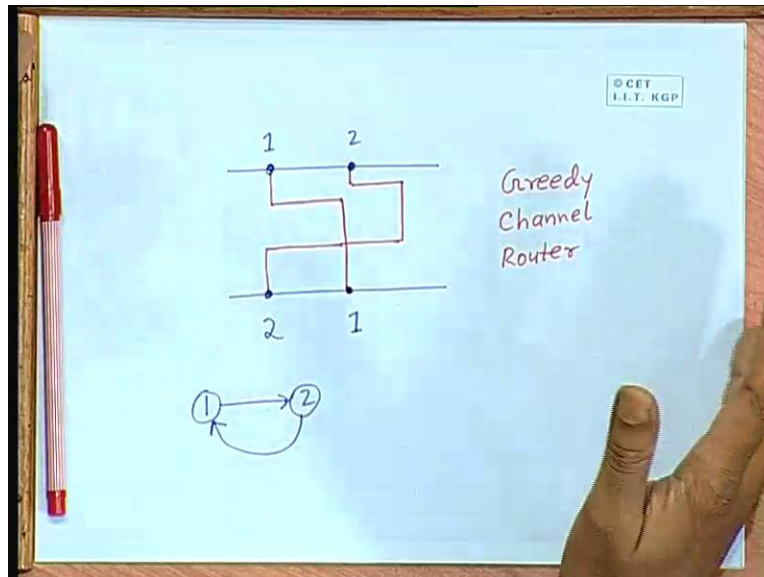
We have looked at so far with regards to channel routing. There we had seen that we did not allow any cycles in the vertical constraint graph. Now since we did not allow any cycles in the vertical constraint graph obviously doglegging was not a mandatory issue. Of course we had looked at 1 algorithm the dogleg router which can place doglegs. But not with an eye to tackling or handling cycles in VCG. But rather to minimize the number of tracks required obtaining a solution. So today first we would be looking at an algorithm which can handle cycles in the VCG in fact which can handle any kind of arbitrary constraints. Now it is called greedy channel router.

(Refer Slide Time: 02:03)



Now before going into the algorithm let me first take a very simple example to illustrate the basic concept.

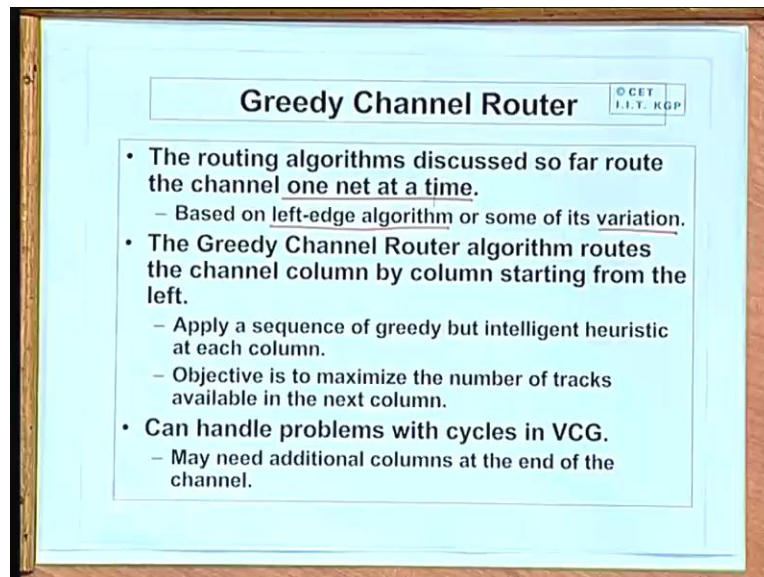
(Refer Slide Time: 02:15)



So let us take a very simple example where there is a cycle in the vertical constraint graph. So as you can see from net 1 to net 2 there is a vertical constraint edge similarly from 2 to 1 there is an edge. Now in this method what we do we systematically proceed column wise and go on. Routing the nets 1 by 1 like from here we start like this from here we start like this. Now in the second column since there is a constraint we cannot connect both of them together. So what do we do we connect 1 of them and from the other point and let this proceed to the right in a from the other point let another net proceed to the right. Now, when you arrive at the next column position these 2 net segments which belong to the same net 2 will finally get merged.

So we will be getting a solution like this. Now the greedy channel router which we would be discussing now proceeds in a manner similar to this. Well this router is greedy in the sense that it takes only local decision it does not look at the problem globally unlike the other algorithms. If you look at the algorithms like the left edge algorithm dogleg algorithm and also Yoshimura and Kuh net merge algorithms there we had looked at the whole problem in its entirety. There we had talked about the horizontal constraint graph we had talked about the vertical constraint graph and then we had combined them to get some solution. But the algorithm we would be considering now namely the greedy channel router this will not be looking at the whole problem at a time.

(Refer Slide Time: 04:50)



This is the main difference of the greedy channel router as compared to the algorithm that we have seen so far. See the algorithms we have discussed so far 1 characteristic was that well whatever way we represent the information in terms of some graph. The channels were routed 1 net at a time this was the characteristic. But the order in which the nets are routed there actually some way we have selecting that. There is a left edge algorithm we are selecting that by the x coordinate of the left edge and so on. And all this algorithms were based on the basic left edge algorithm or some kind of its variation. Dogleg Yoshimura Kuh all of them were finally using the left edge algorithm. So 1 main characteristic of these algorithms were that 1 net was taken at a time routed and then the next net was taken. But in contrast the greedy channel router that we would be considering now this does not look at the problem globally.

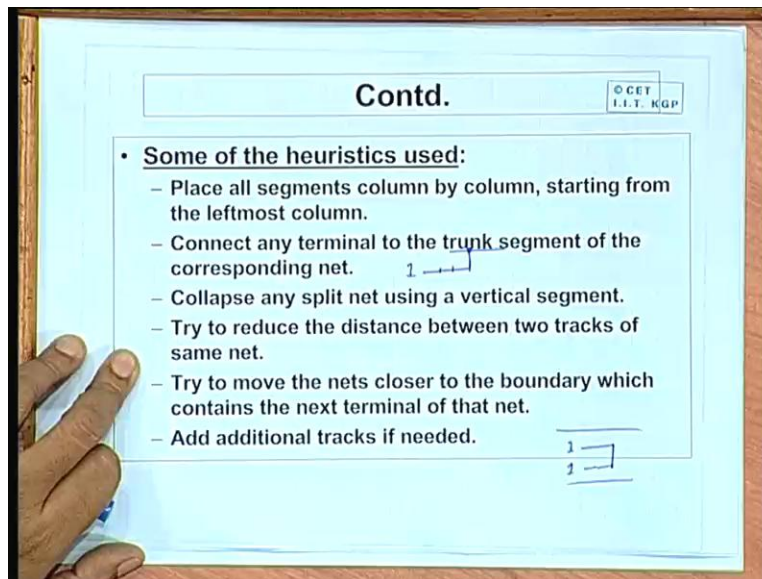
It looks at the local sub problem and make some greedy you can say decisions so as to try and make an optimum decision at the current column position. So this routes the channel column by column not net by net it routes multiple nets at the same time. It scans the channel column wise and starts routing the nets all together whichever nets are encounter at the columns they are routed all together column by column routed means not entirely 1 is not connected to so each net advances 1 column at a time ok. Now at each column position we apply a sequence of greedy

heuristics of course these heuristics are based on experience so they are also intelligent. The objective of this heuristic is that in the next column position the number of tracks that are kept free that is maximized. So if we take some greedy decisions at the current column position so that the number of free tracks in the next column position gets maximized possibly.

We will be getting a good solution subsequently. Objective is to maximize the number of tracks available in the next column. For example we are currently what we must find tracks. For example in the present column position say we have seen that 3 tracks are used up 2 are occupied. But if we straight away extend them extend them to the next track it is possible that the next track will be saying that we require 2 more tracks. But using some heuristic you try to push the tracks on 1 side with respect to the next terminal that is encounter we will see the heuristics so that the number of additional tracks that maybe required subsequently maybe reduced. But of course these are greedy approaches this is not necessarily always true but for most of the cases this is true. And since we are not looking at the problem globally well you can handle any kind of constraints even cycles in VCG.

This is the first algorithm we are looking at which can tackle cycles in the vertical constraint graph. But if we have cycles in the vertical constraint graph you will require additional column positions to complete it as we had seen an example. In our last class that if we have cyclic constraints you need additional tracks with dogleg to complete it. So either in the middle of the channel or at 1 end of the channel beyond the boundary, so you may need additional columns to complete the routing if we have cycles in VCG right. Now let us see that what kind of greedy heuristics are applied in the column positions. Then we will be working out with an example so that we will understand how it works well.

(Refer Slide Time: 09:27)



Some of the heuristics that are used are like this ok. The first 1 is of course it is not really heuristics this is what I had said that you place this segments column by column starting from the left most. This says that if we have a channel which suppose in the first column position you have 1 and 2 in the second column position say you have 3. These have no connection here you have say 4 and you have one. Then what it says that you proceed column by column. In the first step you start net 1 you start net two. Then the second column you see that well you have starting a new net here 3. So well, if we have, if we do not have space you can make a jog out here, let 1 come here or if a space. If we have a space you can start 3 from here.

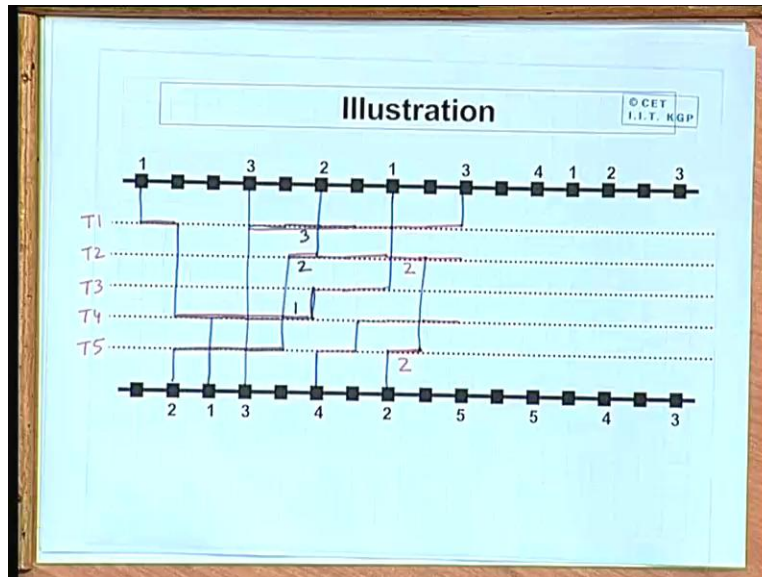
This we will see for an example similarly 2 will be extended. Here 4 will be continued and this 1 net which was there that will be connected here. So at each column position you are doing something. So first let us look at the heuristic then we will just work out a complete example. You place the nets segments column wise not net wise unlike the previous algorithms starting from left. Connect any terminal to the trunk segment as they are encountered. This means suppose you are go you are extending column wise a net say corresponding to net number 1 suddenly in 1 column position, you find that well you have a terminal corresponding to net number 1 in a straight way of connected to it, connected provided there are no constraints or

conflicts ok. Sometimes you may see the 2 segments for the same nets are proceeding from left to right. So at the first opportunity you collapse them you connect them with a dogleg merge them together.

This says that in the channel suppose there were 2 nets, 2 means segments for the same net which we are proceeding due to some conflict before you could if you were unable to connect them earlier. So at the first opportunity you connect them like this. This says that you collapse this split net using vertical segments. Try to reduce the distance between 2 tracks of the same net. There is some heuristic for that I am not going into the absolute detail of this. But whenever you have 2 nets corresponding to the same track try to bring them as close together as possible. Because finally you will have to merge them with a vertical line and longer that vertical lines more is the chance of conflict with the other nets. So just you try to bring them as close together as possible. And the next heuristic says that say these 2 we have merged fine but you have another terminal for net 1 sometime later.

So you have a choice you can either proceed with the net from here or from here. So which track we will be using to extend net one. This says you try to move the nets closer to the boundary which contains the next terminal. Since your next terminal is on the top of the boundary you try to follow a track which is towards the trunk as far as possible these are the heuristics to follow and sometimes well you normally start with a given number of tracks. But sometimes you may see you may need additional tracks you may add additional tracks if may necessary. So if you can provide additional tracks if you can provide additional columns at the end if required then this algorithm can always find out a solution it can always complete the route. So now let us take a compressive example and try to work out on this, I think the concepts will be clear. Let us take a channel routing problem like this.

(Refer Slide Time: 14:23)



Where we have taken 5 tracks to start. In fact this problem can be completed in 5 tracks. Track 4, track 4 and track 5 fine. So we start it column wise. So let us use 2 colors for the horizontal and vertical. Well in the first column you have only net number one. Well I am not showing the zeros the unnumbered terminals are the zeros are no connections. So in the first column we have only one. So you use the first track to extend net number one. Now we encounter the second column now in the second column you see that a net number 2 is starting so let 2 proceed like this there is another consideration for net one. Instead of proceeding a straight away like this see the first time you start a net you normally try to start it from the nearest available track.

Now the next consideration is that the next terminal for net 1 is on the lower boundary bottom boundary. So instead of extending it out here make a jog a dogleg and bring it down here. So what we are saying is that let net 1 come down to here ok. In the third column in the top there is no connection here there is a connection. So 1 you connect it to the trunk segment for 1 this gets connected. Now 1 is still not finished there are some other terminals for 1 also. So 1 you extend well this 2 you cannot make a jog because there will be a conflict although the next terminal of 2 is on top. You cannot make a dogleg at this column itself because already the vertical segment

for 1 has been connected. So you will have to extend further like this. Now you come to column four. Column 4 you have 2 terminal 3 and 3 right 2 on top of the other nothing else.

So you straight away connect this 3 to 3 by a vertical segment and now these 2 you will have to extend right. So since you have already connected this again there are constraints. You do not have any other choice rather than to extend this like this. Now you have a choice out here. See this is the net for 2 the next terminal for net 2 is here. And this is the 1 and 1 is also top but 1 is later. So 2 come first so 2 will be given priority. So net number 2 will make a ok. Sorry, sorry I have missed 1 point. Now this 3 although you have connected there is another 3 terminal on the right and that is towards the top. So you also start a segment from here like this ok. So from here net number 2 you will bring closer to this 2 as far as possible. Now since already this is occupied you can bring it to the second track. So net number 2 you can bring it up to the second track. Net number 3 you extend this also you extend.

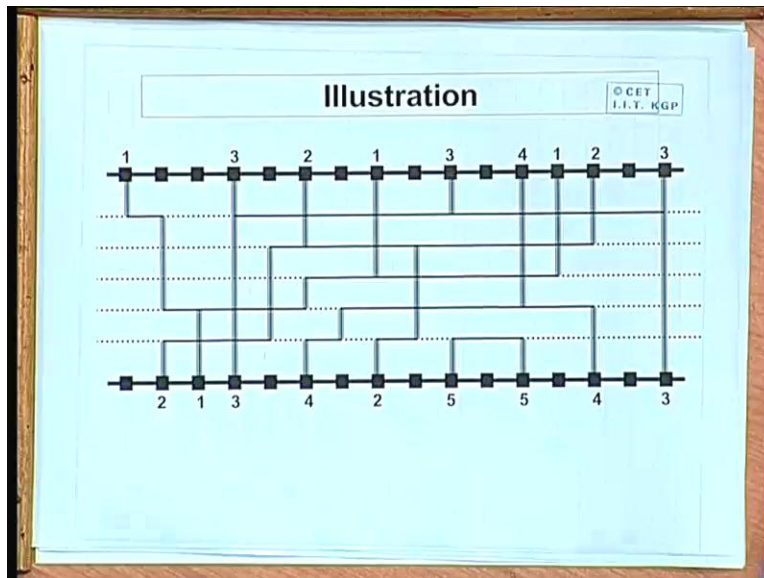
Net number 1 ok this also has a terminal on the top this also you ok this you cannot do a jog right now. You have already made it you extend it ri like this. So we have arrived at this column where net numbers two. Well I am just writing for our convenience. This is net number 2 this is net number 1 and this is net number 3 these are still proceeding. So at this column position you have a terminal 2 you connect it to this. 3, there is a terminal in the right this will be proceeding 2 also has a terminal on the right. Yes, [Students Noise Time: 19:28] because this colors you are unable to see ok fine. So instead of this black let me use red ok. So now at this column position see this 3 is proceeding like this. This 2 will also proceed because 2 also have a connection down. But it cannot go down well it cannot well this 1 will take a jog 1 has a next terminal here.

So this 1 will get up rightly of ok that is blue going up like this so 1 will go up and this 2 will also continue because 2 has a terminal like this so in this way it will go on. Say in the next column position this will be extended simply this will come here this will come here, this will come here. So now we have terminals 2 and one. So 2 cannot this 1 suppose you connect 1 first this 1 you connect to this. Since we have connected 1 2 cannot be connected to 2 in this column position. So this net 2 will have to be extended further on the right and this net also which was supposed to be connected you cannot connect here you also extend to the right. So now in the

next column position there are 2 split segments of the same nets which are proceeding. [Students Noise Time: 21:17] Ok. 4 I missed. Now ok. 4, 4, 4 will be proceeding like this. So 4 4 will be proceeding. Now here the next terminal of 4 is towards the top so this will make a jog like this. So 4 will proceed like this. 2 and 3 also has a terminal on top 3 will be proceed like this.

So similarly the next column 3 will proceed and a connection will be made 2 2 also will proceed 2 has a terminal on top 2 will proceed. Net 4, 4 will also proceed and 2 you can split you can merge the split nets here itself there is no conflict. So 2 and 2 gets connected here. So in this way it will continue. So 3 the next terminal of 3 is both on top and bottom it does not matter you just let it continue. 2 also it is on top let so let 2 continue so now a new net of 5 starts. 4 also 4 it is on the bottom 5 sorry 5 will continue and this 4 will also continue. 4 will go up [Students Noise Time: 22:50] ok 4 will go up right. 4 will go up like this. So in this way it will go on again. So 3 will again continue 2 will continue 4 will go up and get connected here. This is 5, 5 will proceed and 5 will get connected here. In the next step well I think 1 thing I missed I made a mistake here this net 1 also should go on because there is another 1 terminal here.

(Refer Slide Time: 23:50)



So this net 1 should be extended like this. So anyway, so this is how we are doing in step by step and I am showing you the final solution you can just have a look at it so the final solution looks like this. So the net 4 which we are saying it cannot go up since net 1 is going along this net 1 I did not show there. So net 1 will finally reach here and means you can do this comp completion. So we have seen you see here that in this solution the way we have proceeded there are doglegs. There are so many nets which are split across multiple tracks. In fact in this example of course there are no cycles in the VCG but there can be cycles in the VCG also which can be taken care of also. But this greedy algorithm is a fairly powerful algorithm. You can use it to route even very difficult routing problems which otherwise the other routers are algorithms is unable to complete it usually.

But grid router works pretty well. So there is some other routing algorithms also I am not going into the details of those there is another method which is also well which has also been showed to work well that is called hierarchical routing. It says instead of looking at the whole problem in its entirety you try to divide it up into sub problems and try to solve this sub problems 1 at a time some kind of a hierarchy. But I am not going into detail of the algorithm if you are interested these algorithms are available in the text books. But in fact the hierarchical method your algorithm also can handle cycles in the VCG it can handle such problems in doglegs.

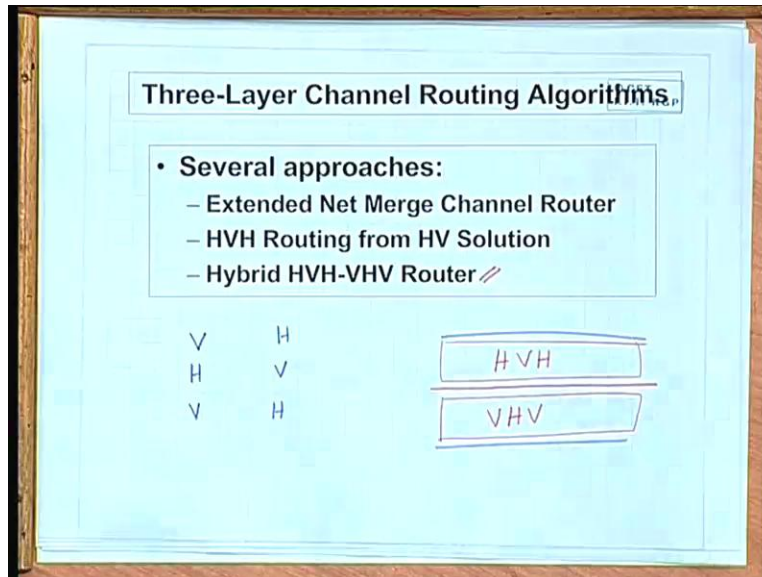
(Refer Slide Time: 25:44)

	LEA	Dogleg	Net Merge	Greedy	Hierarchical
Model	Grid-based	Grid-based	Grid-based	Grid-based	Grid-based
Dogleg	No	Yes	Yes _x	Yes	Yes
Vertical constraint	No / Yes	Yes	Yes	Yes	Yes
Cyclic constraint	No	No	No	Yes _✓	Yes _✓

Just to summarize the different routing algorithm that we have looked at and this hierarchical which you have not looked at but if you compare them all these models are grid based. Well I have not talked about any algorithms which are not grid based. In fact we will see later the power and ground routing they have some consideration regarding that where the width of the nets maybe different. But the Yoshimura Kuh algorithm has been modified I am not again going into the detail of that, so that you can have some kind of grid less routing where you can have the via from any positions the widths can vary there are some widths which are assigned to the graphs using them. So all the models we have seen so far they are grid based. Doglegs the do not use doglegs but the other methods may have doglegs. Of course the net merge in the version which we have looked at this does not have doglegs. But you can extend it there is an extension which we have not looked at there you can have doglegs. Because, in the net merge we are assigning a full net to our track.

Vertical constraints the basic left edge algorithm does not have but the modified version have and all this methods can handle. But cycles in the VCG only the greedy and the hierarchical routers can handle right ok. Now in fact dogleg net merge greedy these are all good algorithm. It depends whether you have cyclic constraints in a VCG. If there are no cyclic constraints you can use dogleg or net merge routers. If we have cyclic constraints you can use greedy or you can use hierarchal also it depends on your problem. Now these are all 2 layer channel routers which have been pretty well explored and the algorithms are very efficient. But this scenario today is that for routing in VLSI chips we typically today have more than 2 layers available. Well of course most of the channels can be routed efficiently on 2 layers we do not need more than 2 layers. But since more than 2 layers are available there are many algorithms which have been reported which consider more than 2 layers.

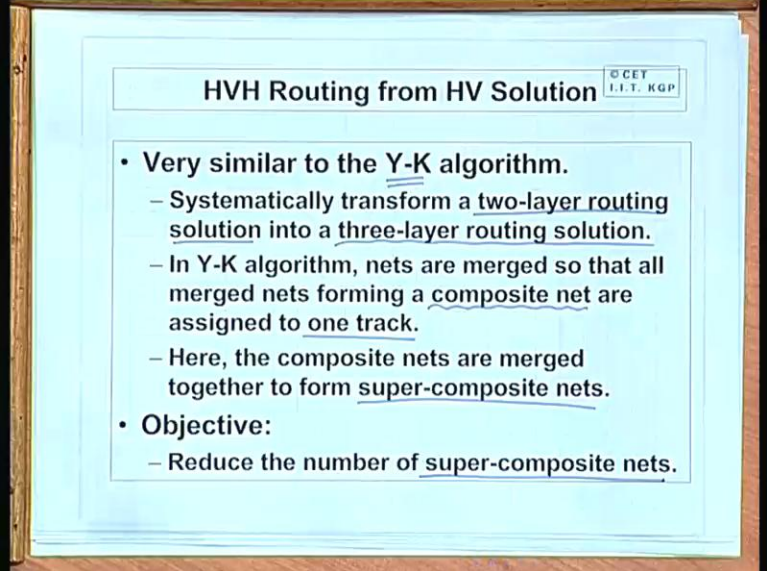
(Refer Slide Time: 28:21)



In particular some work has been done on 3 layer channel routing. 3 layer means you have 3 routing layer for example on 1 layer you do the vertical segments route second layer horizontal third layer vertical or the other way around VHV or HVH. Now here we look at 1 such algorithm. There are actually several such approaches we have been proposed the net merge channel router the basic Yoshimura Kuh had been extended to 3 layers. But the algorithm that we would be looking at is that you start with a 2 layer solution. You try to map it to a 3 layer solution HVH routing from HV solution and there is another approach which says that we use both HVH and VHV.

Here the idea is that you have the channel you divide the channel into 2 regions 1 on top 1 on bottom. The top region of the channel it uses HVH routing the bottom region uses VHV routing and there is 1 track in between which is used to connect between the top segment top zone and the bottom zone. There is some work which have been reported on this also that we found that the number of tracks you require this is less as compared to the others in general. So let us look at this the second method to start with a 2 layer solution to arrive at a 3 layer solution ok.

(Refer Slide Time: 30:14)



HVH Routing from HV Solution © CET I.I.T. KGP

- **Very similar to the Y-K algorithm.**
 - Systematically transform a two-layer routing solution into a three-layer routing solution.
 - In Y-K algorithm, nets are merged so that all merged nets forming a composite net are assigned to one track.
 - Here, the composite nets are merged together to form super-composite nets.
- **Objective:**
 - Reduce the number of super-composite nets.

So this method again is conceptually similar to the Yoshimura and Kuh algorithm. Well, let us try to recall for the Yoshimura Kuh algorithm did. We have a number of nets if 2 nets are not in constraint either horizontally or vertically then we can assign them to the same track. So Yoshimura and Kuh merged a couple of such nets into a composite net. Now a composite net was treated as a whole it was assigned to the same track. Now in this case you can go 1 step further 1 step further means you can define something called super composite nets. Say a super composite net say for example it may consist of 4 different nets n_1 n_2 and n_3 and n_4 . Well out of these n_1 and n_2 can be placed on track 1 of the first layer n_3 and n_4 can be placed on track 1 of this of the say third layer. So I am using the same track but maybe across layers.

So when you say a composite net they can be mapped into the same track in the same layer. Now a super composite net can be mapped into same track in several layers so this is the basic idea. So let us see. So in this algorithm we start with a 2 layer routing solution starting from there we try to systematically transform it into a 3 layer solution. Now the 3 layer solution we look at is HVH not VHV we are assuming there are 2 horizontal layers and 1 vertical layer so that the tracks can be placed on 2 layers. So just what I had mentioned in Yoshimura Kuh algorithm nets are merged so as they can be treated as a composite net and assigned to a single track. But here

the composite nets are further merged into super composite nets and our objective here is not to minimize the composite nets. But rather to minimize the number of super composite nets ok fine.

(Refer Slide Time: 33:15)

Contd.

- Two composite nets in a super-composite net can be assigned to different layers on the same track.
- A track-ordering graph is used to find the optimal pair of composite nets to be merged.
 - Vertices represent the composite (tracks) in a given two-layer solution.
 - The directed edges represent the ordering restrictions on pairs of tracks.
 - Composite interval t_i must be routed above composite interval t_j , if there exists a net $N_p \in t_i$ and $N_q \in t_j$, such that N_p and N_q have a vertical constraint.

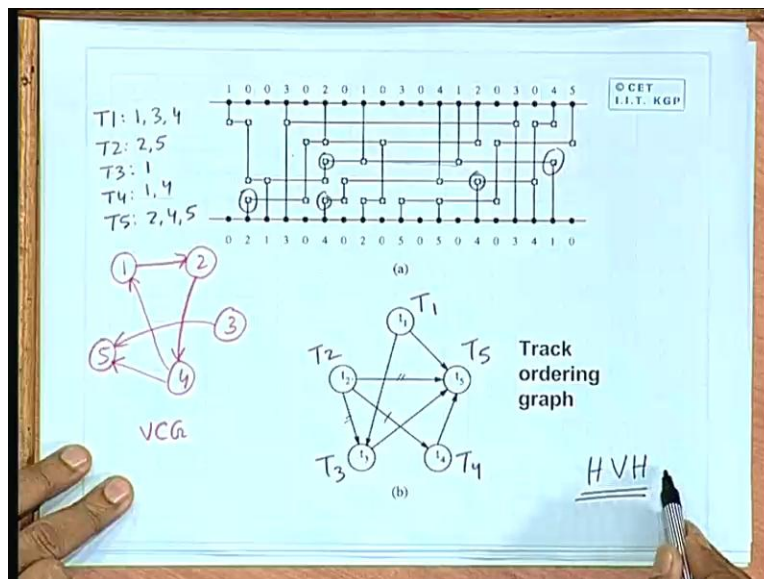
$v_i \rightarrow v_j$

So just continuing on it with what I had said that 2 composite nets in a super composite net they can be assigned to different layers. But on the same track just what I had said. They will be assigned to the same track but on the 2 horizontal layers. Now we use a data structure to keep track of these kinds of constraints. There is something called a track ordering graph. See we have a set of composite nets to start with. The track ordering graph has the vertices which represent the composite nets. So the track ordering graph can be the output of the conventional Yoshimura Kuh algorithm. So after we have done conventional net margining in in 2 layer routing we have got the composite nets. So in the track ordering graph the vertices will represent the composite nets. There will be directed edges between some of these vertices which will represent some ordering restriction on this pairs of tracks.

Now there will be some restriction as we will see these restrictions will tell you that some of say if there is a edge from say v_i to v_j . This will tell you that the composite net represented by v_i must be placed above the composite net corresponding to v_j . There are some vertical constraints

which will require v_i to be placed above v_j ok. So these edges in the track ordering graph will give you some idea regarding the ordering in which the tracks can be assigned that is why it called the track ordering graph. So composite interval t_i must be routed above composite interval t_j if there exists any particular net N_p belongs to t_i and N_q belongs to t_j such that in the original VCG there was an edge from N_p to N_q because t_i is a composite net it can consists of 3 nets, t_j is also composite net it can consist of 2 or 3 nets. If any 1 net from t_i has a vertical constraint with any 1 net of t_j then t_i has to be placed above t_j ok. This is the idea. So now let us see to an example how this works.

(Refer Slide Time: 36:25)



Well here is a complete example which has which had been routed already using Yoshimura and Kuh algorithm. Well here the font sizes are is slightly small but I am telling you. Here you have a 5 track solution on track number 1 you have nets 1, 3 and 4. Track number 2 has 2 and 5 track number 3 has 1 only 1 net number one. Track number 4 has well there is a dogleg well here we are considering a solution a 2 layer solution. So track 1 is also placed on t 4 track 1, track 4 ok. So this is a solution which uses dogleg so this is not necessarily the output of Yoshimura Kuh it maybe output of any algorithm any routing algorithm there are some doglegs here so no problem. T 5 the fifth 1, 2, 4 this is again 2, 5, 2, 4, 5 ok. Now this is the track ordering graph where this

represents track 1, track 2, track 3, track 4 and track 5. Now we have seen these are the 5 nets. Now with respect to the original problem if we also shown the VCG side by side.

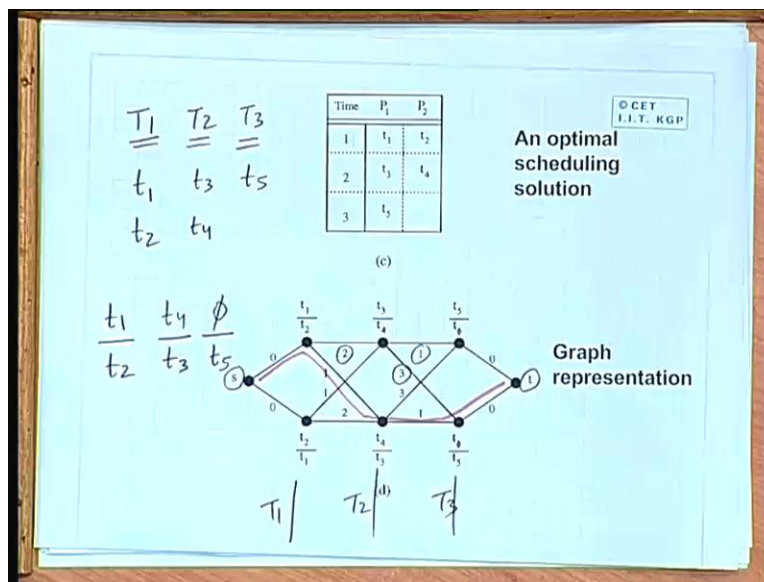
These are the 5 vertices 1, 2, 3, 4, 5, I am showing the vertical constraint graph. So 2 on top of 4 so in the edge from 2 to 4 1 on top of 2 3 on top of 5 4 on top of 5 2 on top of 4 output there 4 on top of 1 ok. So this was the original VCG you see that in the VCG there was a cycle also. That is why we required dogleg to break the cycle. Well now the point to note is that with respect to the VCG if you inspect the VCG and this list you can identify this edges. Say from track 1 which consists of nets 1 3 and 4 well you look at the edges where it is going 1 3 and 4 edges are going well just other than 1 it is going to 2, 5 . So 2 and 5 corresponding to the nets t 2 and t 5 so there will be an edge from 2 and 5. T ok it just you can inspect the solution and find out the, you need not have to look at the VCG originally. See from t 1 this is your track 1. So track 1 say for ex in first column position there is nothing in the second column position there is no constraint as such well there is net number 2 in the fifth track. So there will be an edge from t 1 to t 5.

Then here first track out here there is no constraint here then first track next position is here, here 4 out here, 4 is in track four. So there will be an edge to track 3, 3, 4. That is how this is this is no constraint sorry this is no constraint here constraint is here. So this must be placed above this this is track 3 so t 1 to t 3 is in so t 1 t 5 and t 1 t 3 are the 2 edges corresponding to here and here. Similarly you can look at the other nets and trying to find out which net has to be placed above which for example from track 2 t 2 t 2 there is 1 constraint here t 2 has to be placed above t 3 this edge similarly here t 2 has to be placed above t 5 this edge. Similarly t 2 to t 4 there is a constraint here t 2 to t 4 so this 3 edges are there. So in this way you can construct this track ordering graph. Track ordering graph will tell you that in which order or what are the relative ordering of the tracks which has to be placed above which well this well if you are looking at an HVH routing problem.

So you are trying to place or keep these tracks within 2 horizontal layers. But the composite net you have already defined that has already been mapped to 1 of the tracks out here. Well you were not going to break that composite net anymore. You are trying to see whether the 2 different composite nets can be placed on the same track on the 2 edge layers out of these five.

For this you can look at this graph this graph is more like a precedence graph. If you treat these nodes as some jobs or processes to get executed and this arrows indicate the precedence constraints so from top to bottom as if we are considering the tracks. So you see what are the things which can be done in parallel suppose we have 2 processors which correspond to the 2 horizontal layers. So if they can be done in parallel you put them on the same track on the 2 layers. So 1 possible schedule from this graph can be obtained like this.

(Refer Slide Time: 43:31)



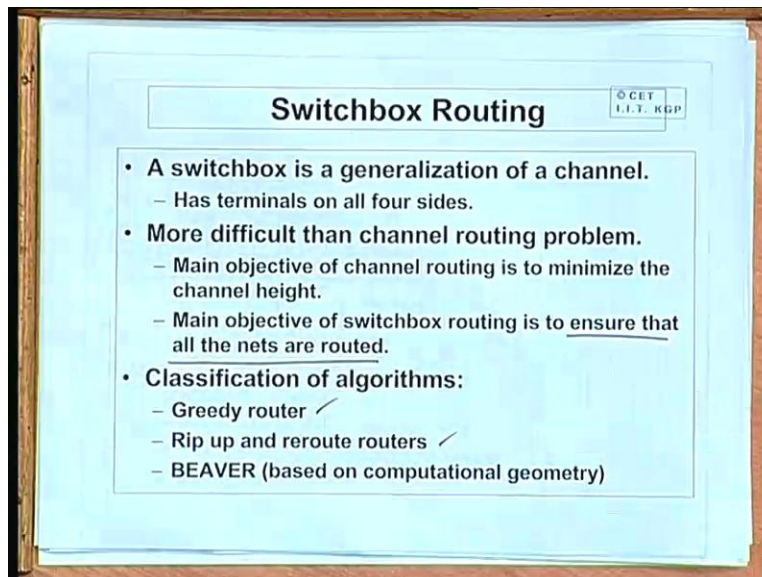
This in the first step you can assign t_1 and t_2 to P_1 and P_2 there is no edge between them. So this can be assigned to the first track well in terms of the processor scheduling analogy we are considering it the first time unit. These are the 2 processors assume this is an analogy you are taking. So after these 2 are done so these edges are also done with. So now well t_3 t_4 are independent now so now you can take t_3 and t_4 together in the second step in the last step you are left with t_5 . So starting from the track ordering graph you can get a schedule like this. The schedule tells that you which are the tracks which can be put as a super composite net and assign to the same track on the 2 layers. Now once you have this schedule you have several alternative so now you know you would be requiring finally 3 tracks t_1 , t_2 , and t_3 , these are the super composite tracks now in the first track well if I represent them by lower case t_1 and t_2 you can

place t_1 and t_2 in the second track you can place t_3 and t_4 in the third track you can place t_5 . But now we have a choice there are 2 layers 1 top of the other.

So either I can put t_1 on top and t_2 below t_2 top t_1 below. There are 2 layers even on the same track. I can put this above this or this above this and similarly t_5 other is empty so I can put t_5 on top or on bottom. So I construct another graph where you can say each level in the graph these are the levels ok 1 level 2 level 3 level this levels indicate these super composite tracks t_1 t_2 t_3 . Based on these 2 possibilities you have 2 vertices in each level of this graph. The top vertices say t_1 is on top of t_2 the second 1 says t_2 is on top of t_1 . This says t_3 is on top of t_4 t_4 on top of t_3 . Similarly t_5 null, null, t_5 , so this graph captures all the possibilities you can have in placing these 2 tracks across the 2 layers. Now once you have identified the vertices you also connect them with edges across successive tracks. These edges will be having some weights. Well of course I am not showing you how the weights are coming. But weights are indicating that how many via you require bet between the adjacent layers if we do the assignment like this. Say some of the edges are labeled as 2 some as 1 some as 3.

3 mean you require 3 via connections if you follow this followed by this. At the beginning you add a dummy source node s and a dummy target node t and you create this problem as a minimum flow problem and you find a path which will be minimizing the total weight. So in this example for instance the minimum weight solution will be this. So this says that in the first 1 you place t_1 on top of t_2 . In the second 1 you place t_4 on top of t_3 . In the third 1 you place t_5 below on top it is empty. So if you assign the tracks in this way to the 2 layers this graph helps in minimizing the number of via connections. Because anyway any path you select that will be 1 feasible solution with the same number of tracks 3. But the via connections may vary. So if you find a minimum cost path in this graph that will give you a solution with minimum number of adjacent via connections ok fine. Such a low, we have looked at a number of channel routing problems. In fact 1 thing I will not go into the detail.

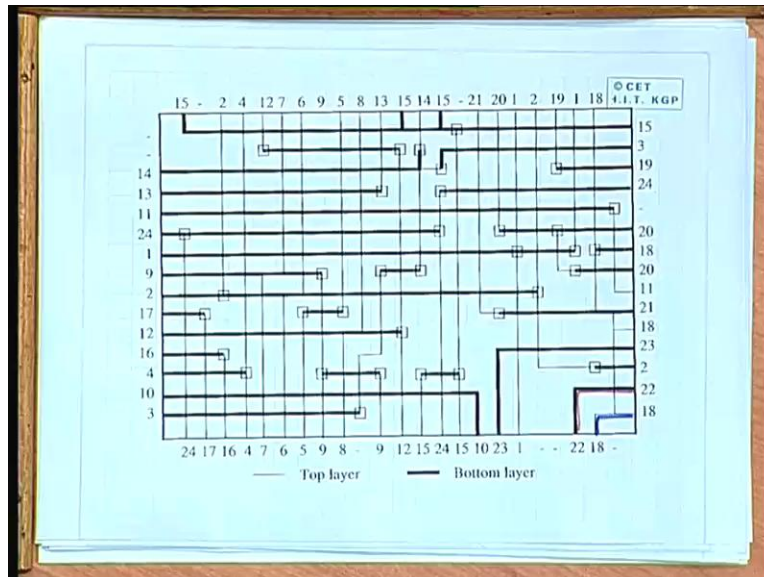
(Refer Slide Time: 48:45)



There is I had mentioned that there is another kind of a problem called switch box routing where there is a routing region with terminals on all 4 sides. Well I will not going into the detail because there is no good and deterministic algorithm for switch box routing. All the algorithms available are heuristic in nature and there is no well defined layer as in channel routing vertical horizontal nothing like that. The main objective of switch routing is somehow or the other you have to complete the routing. So this is certainly much more difficult problem as compared to channel routing. Here the main objective is to ensure that all the nets are routed because the channel size is fixed your switch box area is fixed.

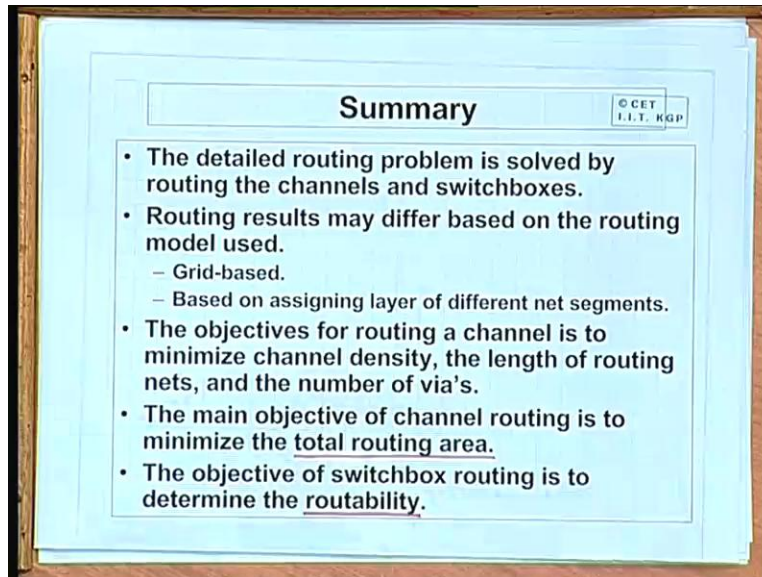
But there is no reserved layering or horizontal vertical layers you can put unconstraint anywhere you want. There are several algorithms which have been proposed greedy approach rip. See in fact in switch box router in some cases when you are unable to find a path you can use some version of the lees algorithm or line search algorithm also to find out a path. So it is more like some of the other you try to find out the path ok. And even there was an interesting method also called beaver which is based on some concepts of computational geometry this method is all work pretty well.

(Refer Slide Time: 50:18)



So I am showing you the solution of 1 reasonable size in switch box this is taken as example in many papers and many because I bench marks. You will see that there are some nets like this. This entire connection is put on the same track. Similarly this on a different layer but on the same track same means I mean layer. So there is no concept of separate horizontal and vertical regarding here. So although this is a 2 layer solution but the nets can be laid out arbitrarily with any kind of horizontal and vertical segments on any layer ok. So there is no better method available for switch box routing as compare to this.

(Refer Slide Time: 51:18)



Summary

- The detailed routing problem is solved by routing the channels and switchboxes.
- Routing results may differ based on the routing model used.
 - Grid-based.
 - Based on assigning layer of different net segments.
- The objectives for routing a channel is to minimize channel density, the length of routing nets, and the number of via's.
- The main objective of channel routing is to minimize the total routing area.
- The objective of switchbox routing is to determine the routability.

So to summarize, so we have seen that there are 2 kinds of problem channels and switch boxes. Grid based model is the most popular which we have seen so far layers to different net segment. That is what we do different net segments they are assigned to the different layers and tracks. So main objective of the channel routing is to minimize the channel height the total routing area but the switch box routing the main concern is routability to complete the routing ok. So in our next lecture we shall be starting our discussion on some specialized routing problems like 1 is I told you clock and power routing and second is that over the cell routing where in addition to the channel we can route the cells over some other available areas ok. So this we shall be considering in our next class. Thank you.