

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

**Lecture No #23**  
**Backend Design Part-IX**

In our last class we were discussing some of the grid routing algorithms. If you recall, we had started our discussion on the maze running or maze routing algorithms. We had talked about 1 of the very popular in classical algorithms namely the lee's algorithm. And if you recall how the classical lee's algorithm operated it started from a given source and it proceeded with labeling the neighborhood grid cells in a breadth first fashion. So since it is proceeding in a breadth first fashion the idea behind lee's algorithm is that if a path exists from a source to the destination you will always find it. But 1 drawback of lee's algorithm was that it was more like a blind search. Well there we were not giving any emphasis on approximately in which direction the target cell is located. So the cells we are labeled in all possible directions but finally when we touch the destination we have found out a path.

So we have also looked at some of the improvements of lee's algorithm in terms of reducing the memory well. Number of cells you need to label like the order in which you start the process, which is the source which is the destination or you can start from both directions or you can or you can have an imaginary bounding box and limit also the numbering process. Instead of numbering they must 1, 2, 3, 4, I had talked about a scheme where you can alternately label the cells as 0, 0, 1, 1 and so on. Ok. So today we would be talking about an algorithm which is similar to lee's algorithm in the sense that it also works on a grid it also labels the cells. But it is more of a goal directed method it tries to take advantage of the fact that which direction the final target is located.

(Refer Slide Time: 03:12)

**Hadlock's Algorithm** © CET I.I.T. KGP

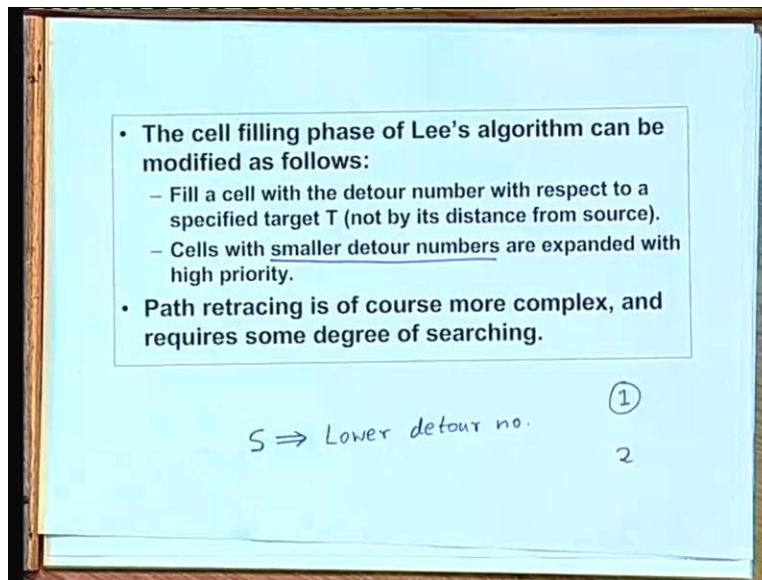
- Uses a new method for cell labeling called detour numbers.
  - A goal directed search method.
  - The detour number  $d(P)$  of a path  $P$  connecting two cells  $S$  and  $T$  is defined as the number of grid cells directed away from its target  $T$ .
  - The length of the path  $P$  is given by  $\text{len}(P) = \text{MD}(S,T) + 2 d(P)$  where  $\text{MD}(S,T)$  is the Manhattan distance between  $S$  and  $T$ .

Now this algorithm is called hadlocks algorithm due to the person who initially proposed it. Now in hadlocks algorithm there is a concept of detour numbers. And we actually label the cells not by its distance from the source but by the so called detour numbers. Now as I just mentioned this is a goal directed search method, from a source we somehow try to find out which direction the target is located. Now the detour number the definition the way the detour number is defined it is defined with respect to a path. Suppose I have a source here, I have a target here, suppose I have a path like this. It is going towards the target then moving in this direction like this. Detour number is defined as the number of grid cells which are directed away from its target. Because if you take this simple; example in the best case, we could have connected  $S$  and  $T$  by a straight line that is the best case.

But may be some obstacles are here in between ok. Suppose to overcome this obstacle we have to move away in this direction say 3 grid cells. So when you come back here we again have to come back 3 grid cells to reach the target. So detour number is an estimate of how far away we are moving away from the target. So it is the number of grid cells directed away from the target and it is defined with respect to a given path. Now if  $d(P)$  denote the detour number which is the number of cells are moving away then the total length of this path will obviously be this. The

best possible path which is the Manhattan distance plus twice of the detour number the number of cells you move away will be the number of cells you will again have to come back ok. So this is how you can estimate the length and this is how you can use the detour number ok.

(Refer Slide Time: 05:34)

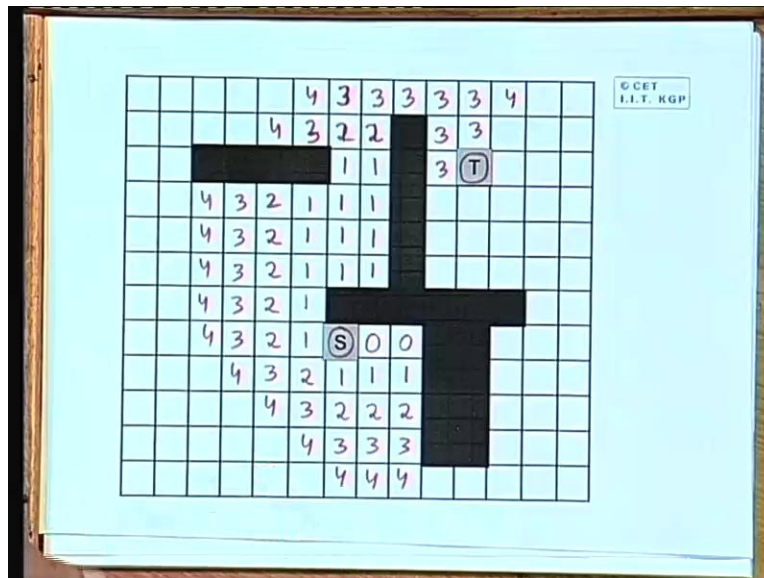


Now using detour number we can modify the cell filling phase of the lee's algorithm like this. Say in case of lee's algorithm we start with a number 1 then 1, 2, 3, 4, in that way we proceeded in a breadth first fashion. But in this algorithm what we will try to do starting from a given source, we will always try to move towards a direction which will give us a lower value of detour number. The cells which I labeled with a lower value of detour number those are the more desirable paths to the destination possibly. So here also we are doing some kind of a breadth first search. But it is a goal directed search we are not doing a blind depth first search.

We are leveling the cells by detour numbers not by the distances from the source ok. So the cells are filled with the detour number with respect to the given target. And the way we expand the cells are label the cell's we always take this smaller detour numbers as the cells with higher priority. So if a cell is labeled with a smaller detour number say 1 and another cell is labeled with 2, we will be expanding this cell first. But as we will see, I will give an example that retracing of

the path will be slightly more complex as compared to the lee's algorithm and you need some kind of a search to do that. So let us take an example to illustrate the detours algorithm.

(Refer Slide Time: 07:19)



Ok. So this, an example grid we take this is the given source and this is the target to which we need to connect. First is that we observe that the target is to the right and to the top of S. So the desirable paths will be either the path which are going to the right or going to the top. But the top paths are block to the obstacles. So only we have this 1 cell which will have a detour number of zero. Detour number zero means we are not moving away from the target. The rest all will be having a detour number 1 because we will have to move away from the target. Similarly this is the cell we labeled in the first step in the next step again you see the neighbors of zero. There is another cell find out that is also will be having a detour number of zero because still we are moving in the same direction. But now the problem arises we cannot move in any other direction with the same detour number. So we will have to move out. So now the different choices are 1, 1 and 1.

These are the detour number 1 because they are moving away from the target. Now the next step will be expanding all the remaining cells 1s. So if we expand in this direction, this will be again

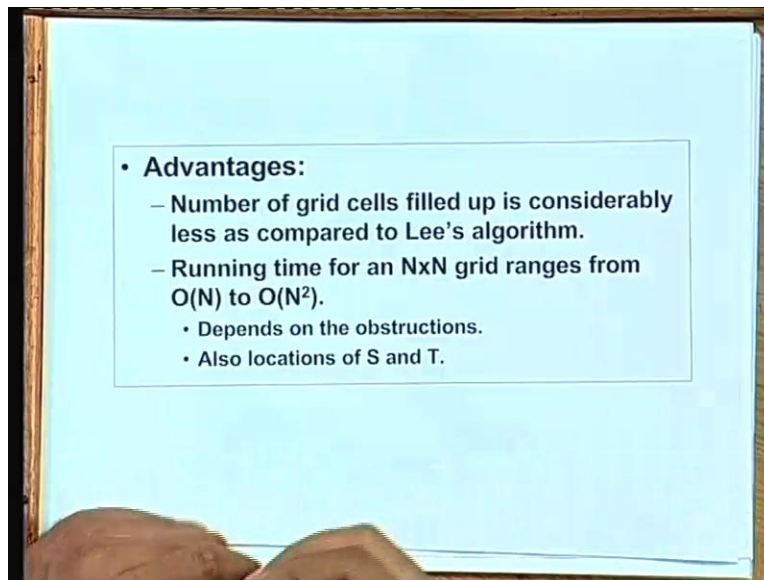
be moving away. This will become 2 this will also become 2, this will become 2, this will also become 2. For this 1 the left cell is also you are moving away 2. But when you move up you are not moving away, so this will be 1. So now this 2s will be the cells of lower priority you take the cells with a lower number this 1. So try to expand this first. So you are moving away label it as 2 this will be 1 you are moving in the direction of the target. So you continue this process. For this 1 here you are moving away here you are moving in the right direction. Here also you are moving in the right direction.

Similarly for these 1s here you are moving away here in the right direction here in the right direction above 1s. Similarly you continue this process. This will be 2, this will be 1, this will be 1, in the next step again you expand this. This will be 1, this will be 1, this you continue. This will be 1, this will be 1. In the next step this will be 1, this will be 1, this will be 2. Oh sorry, sorry, sorry. This was 2 right. This will be 2 because here you are moving away right, right you are right. This will be 3 fine. So now you are moving away from the target because you have reached the row of the target you moving up. So from ok now you have so many cells which I labeled 2. So this all this cells are candidate for expansion. So for this 2 you labeled it with you make it 3 for this 2. This will be 3, 3, 3, this all will be 3 because you are moving away. So now the lowest number 1s a 3. So now you will have to expand 3s. So if you move here, this will become 4, this will become 4, this will be 3 this will all be 4s. This will be 3, this will be 3, this will be 3, this will be 4, this will be 3, this will be 3, you finally reach the target. So this is the essential idea behind hadlocks algorithm. Well you just label the cells by the detour numbers and you always take the lower detour numbers to explode in that direction. So you are always trying to explode in the direction of the target this is a very simple example I have taken.

But for but for more complex examples number of cells you need to expand is typically much less as compare to lee's algorithm. But of course in this example you cannot appreciate that really but for more complex examples you will see that the number of cells that are getting expanded are much less. But if you look at this example we will find that retracing of the path is not that easy. Well 1 thing you know that if the neighboring cells of the target is marked as 3 then you will have to retrace a path where the labels will be in the descending order. First some cells marked 3 then some cells marked 2 then some 1s finally to the source. This will be more

like some kind of searching is required for this process and this searching is not trivial. I suggest that you try to find out find out a scheme or an algorithm to retrace a path from the target back to the source retracing is not as trivial as in case of lee's algorithm.

(Refer Slide Time: 12:48)



Now as I said that the number of grid cells filled up is considerably less as compared to lee's. But running time though it is less as compare to lee's algorithm in general but in the worst case this can also go up to N square for N by N grid. So the worst case complexity is not reducing it really depends on the obstructions and also the locations of S and T ok. Now these maze running algorithms have the advantage that if you have a path existing between the source and the destination this will find the path. But the problem is that the memory requirement and the computational term overheads are pretty high. So there are some alternate algorithms which have been suggested where the information is not kept or maintained in the form of a grid of cells. Rather has some geometric shapes and some lines these are called they so called line search algorithms.

(Refer Slide Time: 13:55)

**Line Search Algorithm** UCET  
I.I.T. KGP

- In maze running algorithms, the time and space complexities are too high.
- An alternative approach is called line searching, which overcomes this drawback.
- **Basic idea:**
  - Assume no obstacles for the time being.
  - A vertical line drawn through S and a horizontal line passing through T will intersect.
    - Manhattan path between S and T.
  - In the presence of obstacles, several such lines need to be drawn.

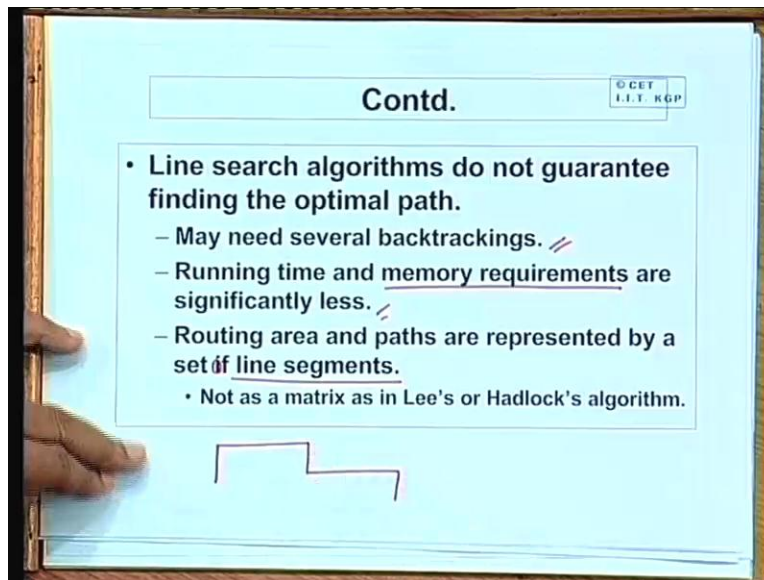
The diagram below the text shows a source 'S' (a circle with a vertical line through it) and a target 'T' (a circle with a horizontal line through it). A vertical line is drawn through S and a horizontal line is drawn through T. They intersect at a point. A square obstacle is placed between S and T, and the intersection point is to the right of the obstacle.

Now here let me tell you in the line search algorithms we are not trying to reduce the time and space complexity. In fact we will see that we can reduce this space complexity drastically. But the time complexity really does not increase much. Now in case of line search algorithm the basic idea is that well to start with assumes there are no obstacles. Suppose you have a source out here you have a target out here. The idea is that if you draw a vertical line through S and imaginary vertical line through S and an imaginary horizontal line through T they will intersect at some point. So if you trace back from the point of intersection back towards T and S you have found out the Manhattan path. Now in the absence of obstacles only 1 such horizontal and vertical line will be able to find the path.

But if there are obstacles in between you may have to trace a number of such straight line paths and we will have to keep track number of such line intersections. And you will have to trace those intersections to get a set of straight line segments. Now the idea is that we are not maintaining the information about the grids and the obstacles as an array like in lee's or hadlocks algorithm. Rather we are maintaining the exact coordinates and the shapes of the obstacles and also the coordinates of the source and the destination. Now the basic data structure will be storing will be using here is some kind of you can say straight lines will be storing information

about straight lines. And the processes were involved that whether 2 straight lines are intersecting, whether a straight line and a given geometrical obstacle or rectangular block typically they are intersecting and so on.

(Refer Slide Time: 16:07)



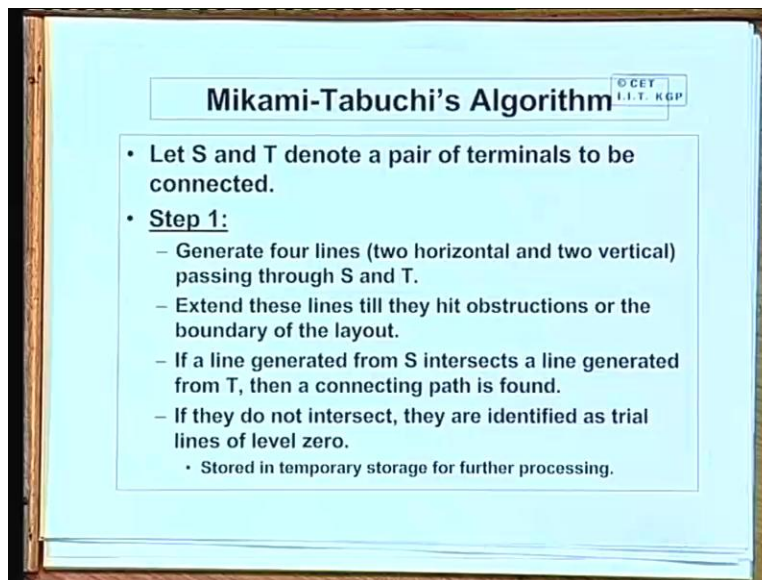
But a point to note is that these line search algorithm always do not guarantee the optimal path. Well you may find a path but that may not be the best possible one because you are always trying to find out the longest straight line segments and intersecting them and finding out. But there can be some shorter paths with a with a lot number of bends. So here the numbers of bends in the wires are typically less ok. So may need several backtrackings in the presence of obstacles. Well the running time and memory requirements are significantly less in fact memory requirements is drastically low as we will see.

And the routing path which you finally find out is represented by a set of line segments. So from source may be you go you will get a path like this some straight line segments. But you can compare this with lee's or hadlocks algorithm where finally the routing information you get in the form of a matrix. From the matrix you will have to deduce back and determine the path ok.



Now under the line search algorithm we would be discussing 2 different algorithms first is an algorithm due to mikami and tabuchi, then we will talk about an improvement of this.

(Refer Slide Time: 17:32)



Mikami tabuchi is the algorithm is fairly simple in concept. This says that S and T are the pair of terminals you want to connect. Step 1 is generate 4 lines, 2 horizontal and 2 vertical passing through S and T. Suppose this is S and this is T to start with, you pass horizontal and vertical line through S and a horizontal and vertical line through T. These are the 4 lines. Now these lines will extend in all directions till they hit the floor plan boundary or an obstacle. So this will be extending till they hit obstructions or the boundary.

Now if a line which is starting from T intersects with a line starting from S we have found out the path. But if not these set of lines we have started with they are identified as trial lines well initially they are of level zero. Then there level 1 2 3 will go on. These are the trial lines now the trial lines we keep in some kind of a list or a queue for further processing. This is the data structure we are maintaining we are keeping information about these lines right.

(Refer Slide Time: 19:00)

**Contd.**

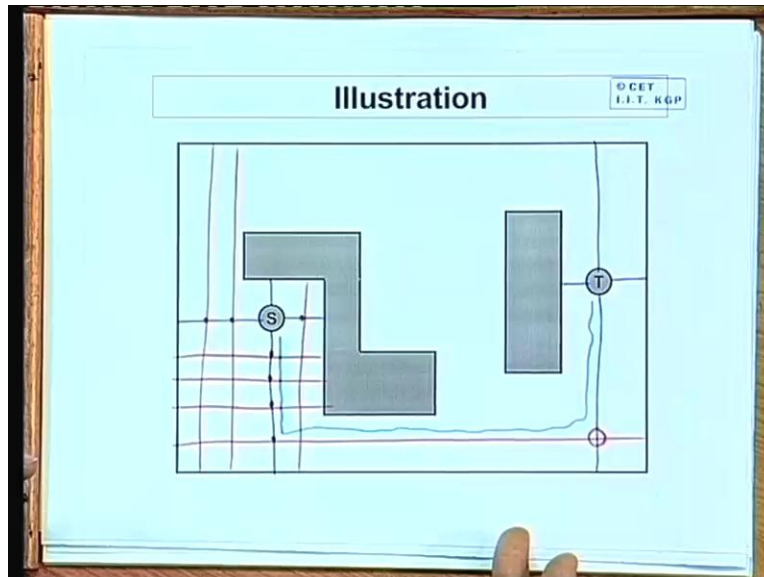
**Step  $i$  of Iteration:**

- Pick up trial lines of level  $i$ , one at a time.
  - Along the trial line, all its grid points are traced.
  - Starting from these grid points, new trial lines (of level  $i+1$ ) are generated perpendicular to the trial line of level  $i$ .
- If a trial line of level  $i+1$  intersects a trial line (of any level) from the other terminal point, the connecting path can be found.
  - By backtracing from the intersection point to  $S$  and  $T$ .
  - Otherwise, all trial lines of level  $(i+1)$  are added to temporary storage, and the procedure repeated.
- The algorithm guarantees to find a path if it exists.

Now in general in step  $I$  of the iteration we pick up the trial lines of level  $i$ . See at step  $I$  the trial lines of level  $I$  minus 1 have all been taken care of. So we pick up the trial lines from the list or queue whatever you call. Now along the trial line all the grid points are traced for example this is a trial line. Now here although we do not have explicitly the grid structure but we can identify the grid locations along this line. Suppose these are the grid locations. So along the trial lines all the grid points are traced and from these grid points new trial lines are generated which are perpendicular to this line.

So trial lines like these will be generated which will be perpendicular to the trial line from where you are exploring. Now you continue this process if a trial line of some particular level  $I$  plus 1 intersects a trial line from the other terminal point may be at level  $I$  plus 1 or less then we have found out a path. So by tracing back the intersections we can find out path. Now this algorithm will guarantee to find a path if it exists, but not necessarily the minimum 1 as I mentioned. So let us take some example and illustrate it so that it will be clear that how it works.

(Refer Slide Time: 20:41)



So let us take an example like this. This is the boundary, these are the 2 obstacles this is the source and this is the target. So to start with there is a horizontal trial line like this and a vertical trial line like this we start from S. Similarly from the target these are the 2 trial lines. Well at the next iteration we will have to these are the trial lines of level zero. So we will have to identify the grid point suppose this is a grid point, this is a grid point, this is a grid point and these are all grid point suppose. So we will have to trace trial lines along all this grid points in the next iteration. Now in fact in this example here itself you will get an intersection. So from S trial line of level 1 is intersecting a trial line of level zero from T. So if you get that from here you can systematically backtrack and we will get the path like this this will be your path.

But if there are more obstacles you may have to generate some more search tracing paths. Now this is the essential idea behind the mikami tabuchi algorithm. But 1 thing you just try to understand that in this algorithm the number of trial lines we generate may be pretty large. Because, along each line there are so many grid points. But intuitively you see that all this lines are not really useful lines because say with respect to these we will see many of them are anyway they are hitting this obstacle. So just along this line you can you cannot go much further. But rather if you can identify those lines which can just extend beyond the obstacles those will be the

useful ones ok. So the next improvement of this algorithm it tries to utilize this fact to make it more efficient.

(Refer Slide Time: 23:12)

**Hightower's Algorithm** © CEF  
I.I.T RGP

- **Similar to Mikami-Tabuchi's algorithm.**
  - Instead of generating all line segments perpendicular to a trial line, consider only those lines that can be extended beyond the obstacle which blocked the preceding trial line.
- **Steps of the algorithm:**
  - Pass a horizontal and a vertical line through source and target points (called first-level probes).
  - If the source and the target lines meet, a path is found.
  - Otherwise, pass a perpendicular line to the previous probe whenever it intersects an obstacle.
    - Concept of escape point and escape line.

The diagram shows a rectangular obstacle with diagonal hatching. A horizontal line and a vertical line intersect at a point to the right of the obstacle, representing a path found. A red pen is pointing to the bottom of the slide.

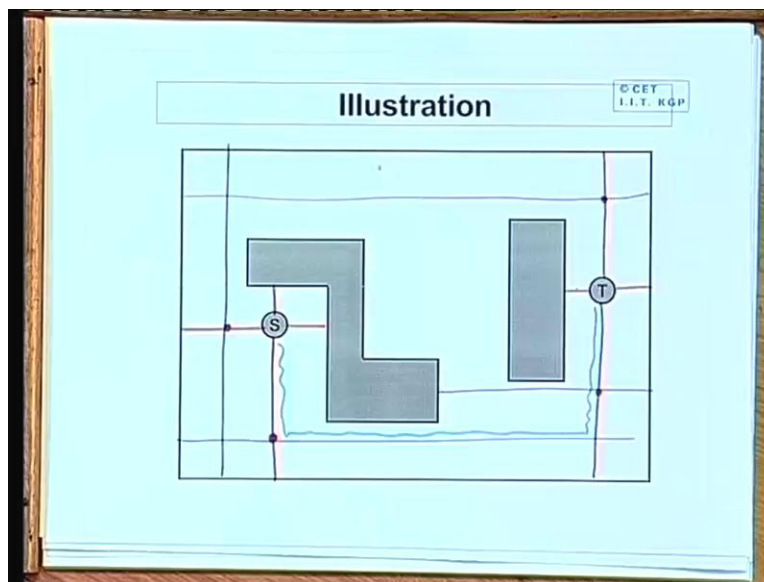
This is called the Hightower's algorithm. (Student Noise Time: 23:14). This will not guarantee optimal path just as I told you it tries to just trace this straight lines and try to find out the longest straight line intersections. But there can be a shorter path with lot of bends that will find in more number of iterations. Say here number of iterations is equal to the number of bend you can see. So there can be a path with less number of bends but the path length may be long number of bends may be yeah. But the shortest path may have large number of bends so the number of iterations will need to detect that path will be more. Ok so in case of Hightower's algorithm well as I told you this is similar in concept of mikami tabuchi in the sense that here also you are throwing lines along horizontal and vertical directions.

But you are not generating all line segments perpendicular to a trial line with respect to the grid positions. But rather we are only generating a few of the interesting lines. Given a trial line we find out only certain interesting grid positions and we generate only trial lines perpendicular at those positions not in all grid positions like in Mikami tabuchi's algorithm. So the idea behind

this interesting grid points is that the new lines should be able to by pass the obstacles which are lying in between. Now by passing this obstacles these there is a concept of escape point and escape line. You are trying to find out suppose here we had an obstacle. So you try to find out that which is the best point you should choose so that you can escape out of this obstacle.

This point will be called an escape point and the corresponding trace line trial line will be called an escape line ok. So steps of the algorithm are similar to mikami tabuchi's you pass horizontal vertical line through source and target to start with. So if they meet you have found out a path well otherwise you pass a perpendicular line to the previous probe whenever it intersect an obstacle. But here you use the concept of escape point and escape line not means here you are not considering all possible grid points and the lines emanating from them ok this is the main difference. So just if you tell me, yes. (Student Noise Time: 26:02) There can be more also lets just take the example and try to find out. Let us take the same example.

(Refer Slide Time: 26:17)



These are the first level probes; the horizontal and the vertical lines ok. Now the horizontal vertical lines will be hitting the boundary or the obstacles. Well the boundaries we are not concern for the timing we are more concern with the obstacles. So you look at this line. This line

is hitting the obstacle. So you trace along this line and find out exactly at which point on this line you can come out of the obstacle it is a point out here. It is a point out here where you can escape out like this. Similarly for this trial line the escape point will be here but you can just come out of the obstacle. Similarly with respect to these for this obstacle the escape for this line, the escape points will be here and here. So it will be 1 trial line like this, another trial line like this.

So in this is how you are trying to find out the you can say the escape points you will have to search the obstacle. Reach the obstacle boundary just the first point beyond that that will be your escape point. Now in this way you go and tracing and find out the first intersection of a line from here and the line from here in fact we have already got the path which we had earlier obtained this 1 this will be the first path you will get. (Student Noise Time: 28:17). See we only expand those lines which hit the obstacle because other lines mean what the other lines are already extended along its length if there is an intersection they will intersect anyway. So we need not worry about the other lines they have been explode to the maximum possible extent. But the other lines will hitting the obstacle may be if you go away just means outside the obstacle you will find a path a new path. (Student Noise Time: 28:48). On T yes. (Student Noise Time: 28:53)

This is not hitting an obstacle. But we are identifying the escape point because when we take these are the trial line will have to find out the escape point when you are trying to draw say lines perpendicular to this. Then we will come to know that there is an obstacle and we find out the escape points. (Student Noise Time: 29:14) Yes, yes, yes. Here whether hits or not we have to search and find out the escape points. Now the main idea being this algorithm is that the memory requirement is drastically reduced. But computational time is not really reduced much. Because we have to make a lot of searching in order to find this escape points and keep track of this escape lines and intersections. So the running time does not decrease appreciably. But the memory requirements get reduced drastically ok. So this is 1 advantage of this kind of line search methods.

(Refer Slide Time: 30:02)

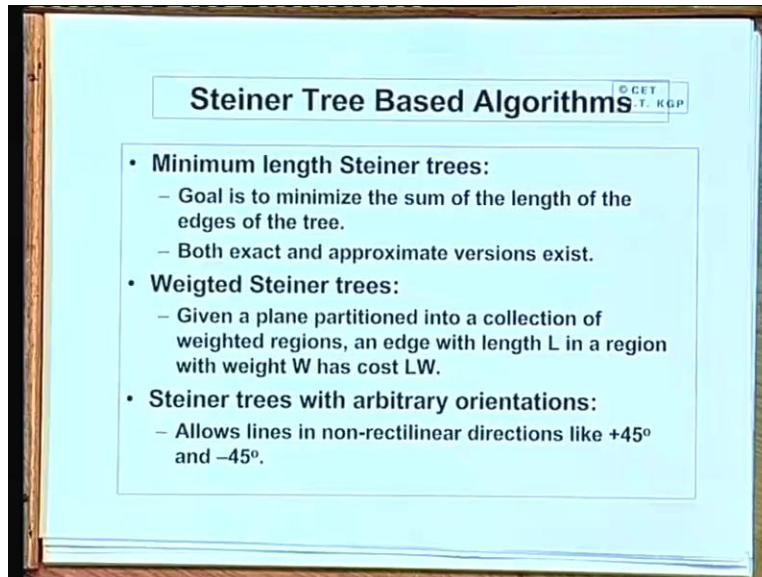
**Steiner Trees** © CET  
I.I.T. KGP

- A tree interconnecting a set  $P = \{P_1, \dots, P_n\}$  of specified points in the rectilinear plane and some arbitrary points is called a (rectilinear) Steiner tree of  $P$ .

- A Steiner tree with minimum total cost is called a Steiner minimal tree (SMT).
  - The general SMT problem is NP-hard.

Ok, just to mention there are a number of algorithms which specifically talk about Steiner trees. Because ultimately when you lay out a net on a surface it is a Steiner tree we are trying to draw because a Steiner tree if you recall they will have the connection segments as horizontal and vertical lines and there can be some intersection point like here and here which are not 1 of the terminal points. So some algorithms try to find out well determining the best or minimum cost Steiner tree is an NP complete problem. So there are algorithms to find out a good Steiner tree connecting a set of points. See the maze running or the line search algorithm that we have discussed they are basically used to connect a pair of points at a time. But a Steiner tree kind of algorithm they can take a set of points a net list a net and they will try to construct a Steiner tree con actually connecting all this points so they look at the problem slightly more globally. So the minimum Steiner tree as I had mentioned it is a NP it's a NP hard problem. So you never try and go for the minimum possible solution.

(Refer Slide Time: 31:30)



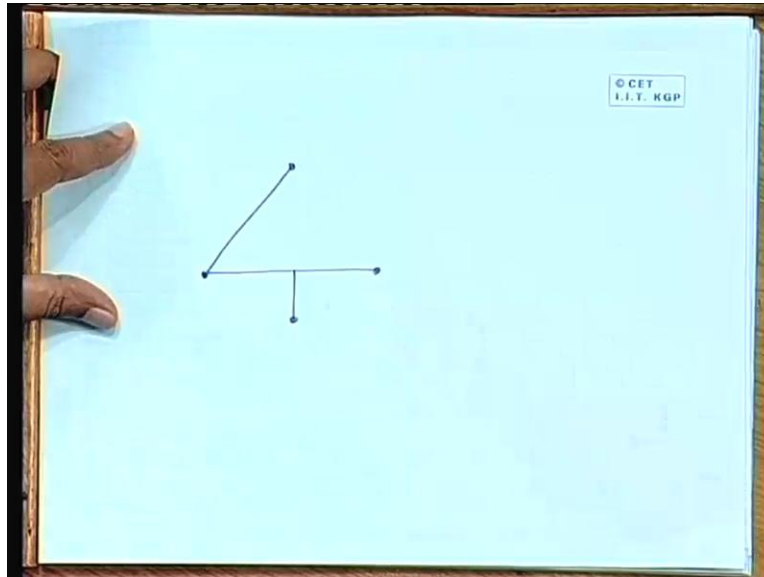
Now with respect to Steiner tree based algorithms I am not going into detail. Just I am just mentioning that there are several classes of such algorithms which have been proposed well. One is the minimum length Steiner tree of course this is an NP hard problem I have just mentioned. The goal is to minimize the sum of the lengths of the edges of the tree. Exact versions exist but the running times are large there are approximate versions using some heuristics which give fairly good results. Many commercial routers use them in fact. Well this Steiner tree may have some weights like say a tree when you have a say when you have a number of terminals to be connected suppose this Steiner tree is like this this is the best Steiner tree. But with respect to the layout area this segments may have to pass through different routing regions with respect to the floor plan.

Now these different routing regions may be having different costs with respect to the area available for routing. So accordingly you can define some weights of the different edges of this Steiner tree. So in that respects you can even talk about weighted Steiner tree and try to minimize the cost of the tree. The length is L and the weight is W so the cost of that edge will be L by L into W. So in weighted Steiner tree you also take into account the congestion in the individual segments of the line through which routing regions you are actually taking those lines through. In



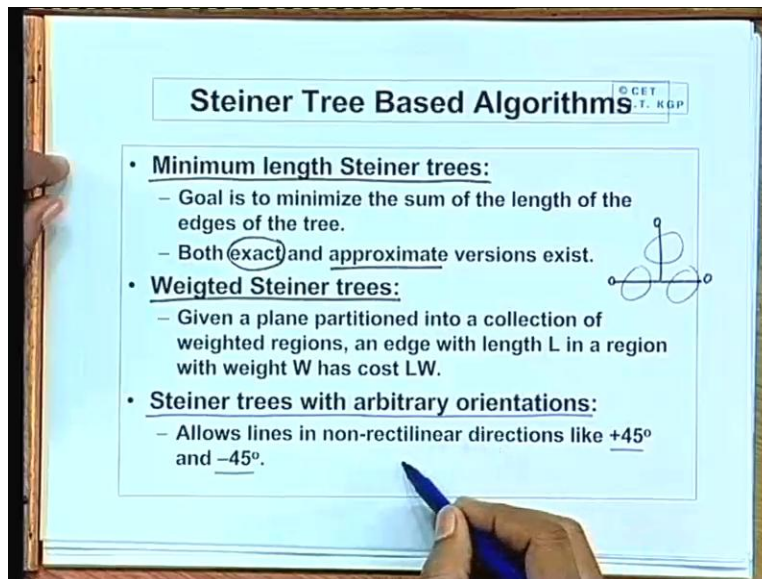
fact some of the modern tools also considered such Steiner trees with non rectilinear edges which are angle at forty 4 degree or minus forty 4 degree.

(Refer Slide Time: 33:37)



So there you can have some interconnecting lines like this. You can have a line connecting like this another line connecting like this another line connecting like this. So lines at an angle of 45 degree are also allowed there.

(Refer Slide Time: 33:52)



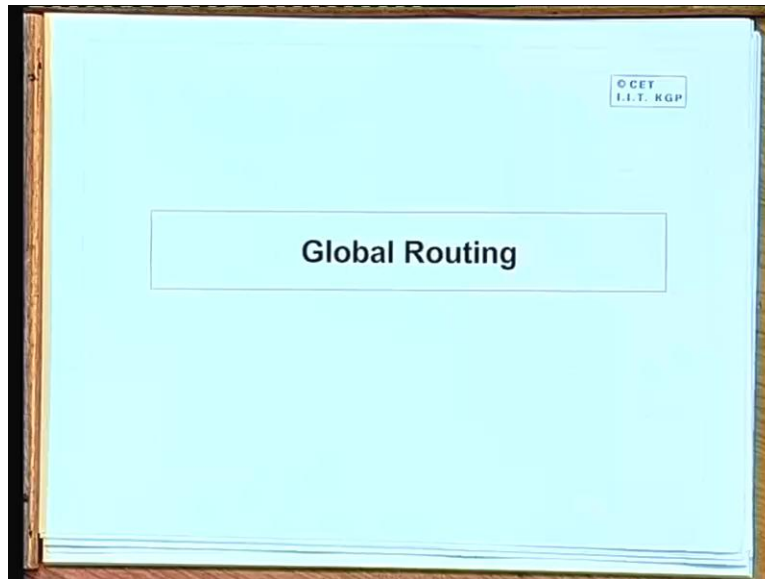
**Steiner Tree Based Algorithms** ©CET S.T. KGP

- **Minimum length Steiner trees:**
  - Goal is to minimize the sum of the length of the edges of the tree.
  - Both exact and approximate versions exist.
- **Weighted Steiner trees:**
  - Given a plane partitioned into a collection of weighted regions, an edge with length  $L$  in a region with weight  $W$  has cost  $LW$ .
- **Steiner trees with arbitrary orientations:**
  - Allows lines in non-rectilinear directions like  $+45^\circ$  and  $-45^\circ$ .

A small diagram of a Steiner tree is shown to the right of the text, consisting of three points connected by lines that meet at a central point at 120-degree angles.

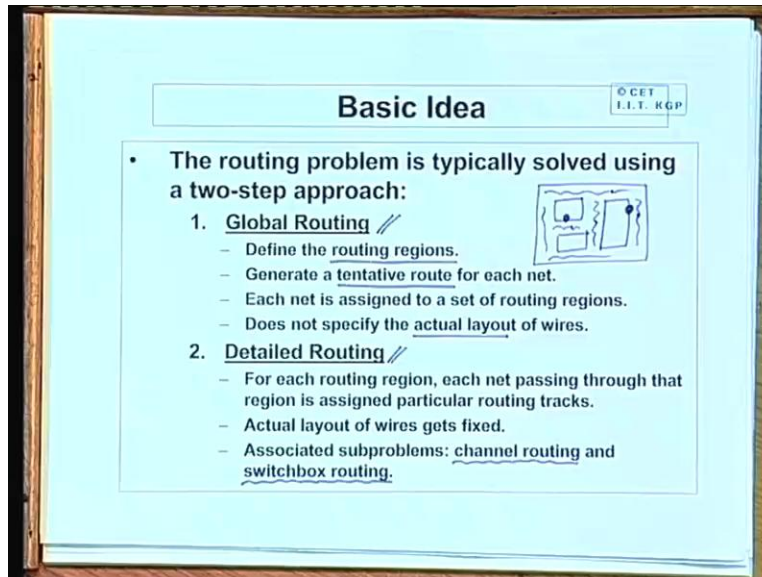
But as you can understand the problem becomes much more complex here. But some CAD tools do use them. So actually we have looked at some of the grid routing algorithms basically where the layout the obstacles and they already routed nets they are expressed in the form of a grid. Well the grids you can represent either by a matrix or by some geometric information like lines and some rectangles. Now we have looked at a small sub problem of routing this grid routing whatever we have looked at that is just a set of algorithms which we use to connect a set of points a pair of points typically. But now let us look at the routing problem from a slightly more global perspective then we will see later how this algorithms have just discussed they fit into the total picture ok. So we now look at something called global routing.

(Refer Slide Time: 35:11)



Global routing means we are looking at the routing problem globally we have the whole chip surface we have all the obstacles we have all the nets which are to be routed. So now we have to make a plan that how we will be routing these nets there can be thousand nets. For example so how will be routing this thousand nets in what order through which areas we will have to route the nets net because we cannot use a maze running algorithm to route all this nets. Because the chip area may be quite huge there may be nets which may be running from 1 corner of the chip to the other corner. So just using those algorithms in a chip may be too costly we will have to do some kind of a divide and conquer strategy ok.

(Refer Slide Time: 36:06)



**Basic Idea** © CET  
I.I.T. KGP

- The routing problem is typically solved using a two-step approach:
  1. **Global Routing** //
    - Define the routing regions.
    - Generate a tentative route for each net.
    - Each net is assigned to a set of routing regions.
    - Does not specify the actual layout of wires.
  2. **Detailed Routing** //
    - For each routing region, each net passing through that region is assigned particular routing tracks.
    - Actual layout of wires gets fixed.
    - Associated subproblems: channel routing and switchbox routing.

The diagram shows a chip layout with several rectangular blocks. A path is highlighted, showing a tentative route that passes through various routing regions defined by the spaces between the blocks.

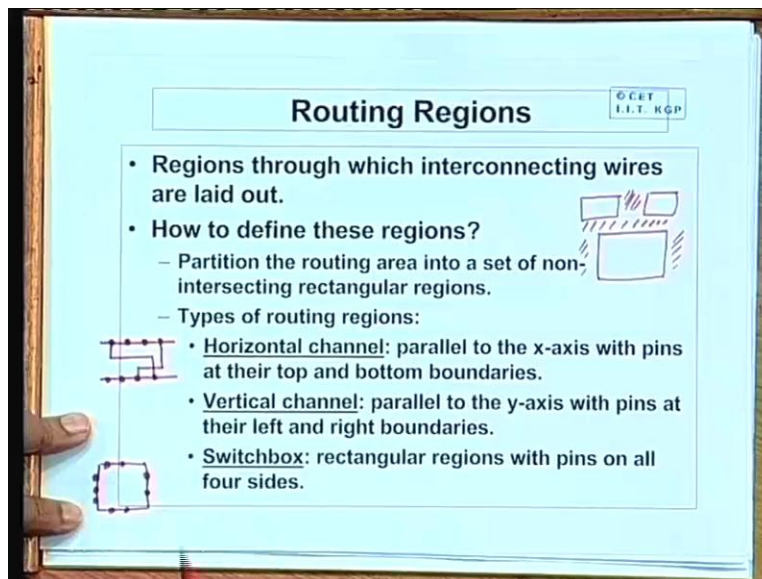
So in practice the routing problem is solved in 2 steps. The first step which we were discussing now that is called global routing which will be followed by another step called detailed routing. Now in global routing what we do we define the so called routing regions? Because ultimately in the chip after placement we have the blocks placed in certain location of course there is some space left between the blocks. So we will have to define the routing regions and there, relationship among themselves. With respect to the routing regions we have defined we will have to generate a tentative route for each net this is tentative this is not the exact. For instance on the silicon floor suppose I have a block like this I have a block like this I have a block like this so these will be the routing region ok this is 1 routing region this is 1 routing region the bou these are all routing regions.

So if we have to connect a pin say from here to a pin say out here then we will have to find out the sequence of routing regions through which this net has to be routed. For example I can say you first come here then come here then come here then come here. So we specify a sequence of routing regions for the net. But we do not specify the exact geometric layout of the wires as yet. We just give a tentative route that this is the path you will take. Now each net as I said is assigned to a set of routing regions like this. But this global routing does not specify the actual

layout of the wires that is done only in the second step. After this rough routing information is generated then you consider one region at a time and try to complete the exact layout of that region.

For each routing region each net which are passing through it is assigned particular routing tracks which means you are assigning exact geometric locations for those nets. So after detailed routing the actual layout of the wires will get fixed ok. Now we will see later that for this detailed routing there are 2 sub problems which come up one is called channel routing and the other is called switch box routing. And in fact all the most common detailed routing sub problems can be mapped into either channel routing or switch box routing. Ok first let us look at the problem of global routing.

(Refer Slide Time: 39:23)

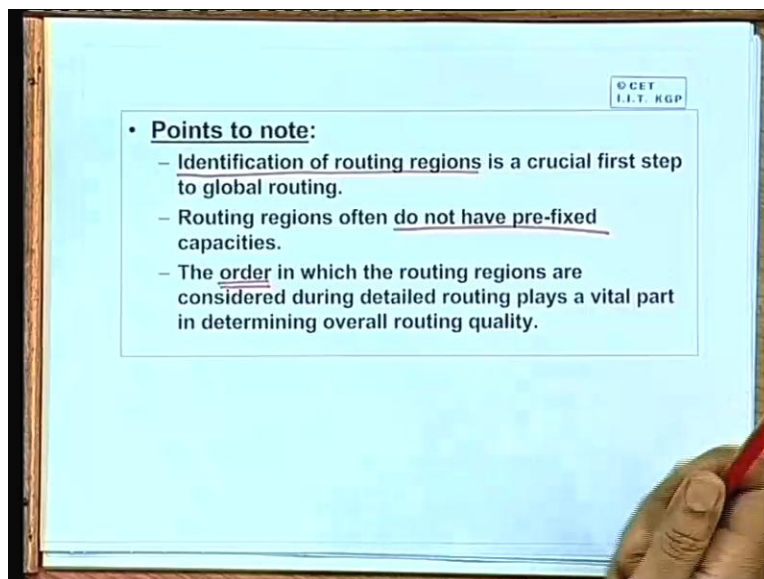


Ok so first let us talk about the so called routing regions I just mention the routing regions at the areas which I left between the blocks during placement. They are the regions through which the interconnection wires have to be passed. Now the problem is that how do we define this regions well it is not difficult you can partition the total routing area available to you into a set of non-intersecting rectangular regions. Like suppose you have blocks laid out like this then you can say

that well this will be 1 region this will be 1 region this will be 1 region this will be one. All will be rectangular in shapes ok. Now with respect to the geometry of this blocks and the associated regions routing regions can appear in 1 of this 3 types. Well horizontal channel means well the name horizontal means it is parallel to the x axis it says that the channel is like this with some points which you want to connect on top and some points on the bottom boundary.

We will have to connect some of the points from the top to some of the points from the bottom may be you will have to connect a wire like this connect another wire like this and so on. This is horizontal channel and if you rotate it by ninety degree it becomes a vertical channel. So that the pins are located to the left and right and it is parallel to the y axis. But in some cases the problem becomes more complex and we get something called a switch boxes switch box is a channel where pins are located on all 4 sides and we will have to interconnect them. So the routing area is of course rectangular in shape but pins are available on all 4 sides. This is called a switch box. So a routing region can be either some kind of a channel or a switchbox.

(Refer Slide Time: 41:45)

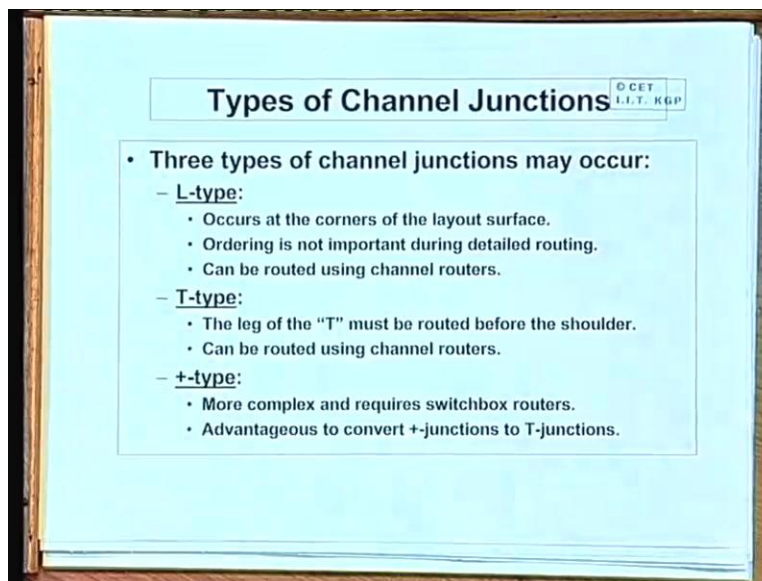


Now before we actually start the process of global routing this identification of routing regions is extremely important and in most design style of course we are not talking about FPGA's and

gate arrays well other than FPGA's and gate arrays. Routing regions do not have pre fixed capacities because if you feel if you see later that well the space becomes a little less for routing I can always shift the blocks up to make a little more space. Like in standard cell there are some rows of cells with some space in between some routing.

So if I find that the space is not sufficient I can always shift the rows up to make more space. So the routing regions they do not have pre fix capacities. But it is important to determine some kind of an order in which the routing regions have to be considered during detailed routing. Because the order in which you take up the regions for detailed routing that will play a significant role finally in determining the quality of the layout ok this we will take some examples and show you later ok. So this are related to the routing regions. Now also let us talk about something called channel junctions.

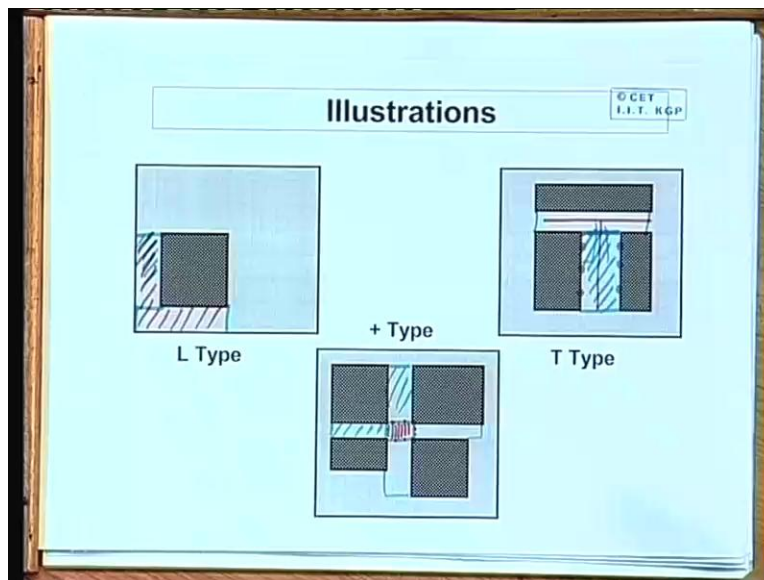
(Refer Slide Time: 43:13)



Well. In a typical layout switch boxes are rear but there will be switch boxes in some places you cannot avoid but channels are much more frequent. So we will be getting mostly the channels. So when you have channels you will have inter intersection or connection between channels broadly

you can have 3 kinds of channel junctions. One is the L type channel junctions which can occur near the corners of the layout surface like I have an example illustrative examples slide say here.

(Refer Slide Time: 43:58)



Suppose this is a layout and there is a block near the corner this is a layout channel from here you may want to route some nets this has a shape of a L right. So this occurs near the corners well for L type channels ordering is not important this you can take up in any order because in a L type channel the other channels don't interfere much with this channel this is near the boundary. And you can use the conventional channel routers to handle L type channels also how. Suppose you have an L type channel here you can break it up into 2 channels like this you can have 1 channel like this you can have another channel like this. So you can treat each of them as individual channels and you can route them using channel routers individually. So L type channels do not create much of a problem in terms of ordering. T type channel is the second type where the channel between several blocks has the shape of a T like the example here. There are 3 blocks and you have a T out here this space you have in between has the shape of a T ok.

Now if we have a T type ok here we are basically calling it a channel but our definition of a tan channel is the pins are only at the top of the bottom. So here we are using channel for to express

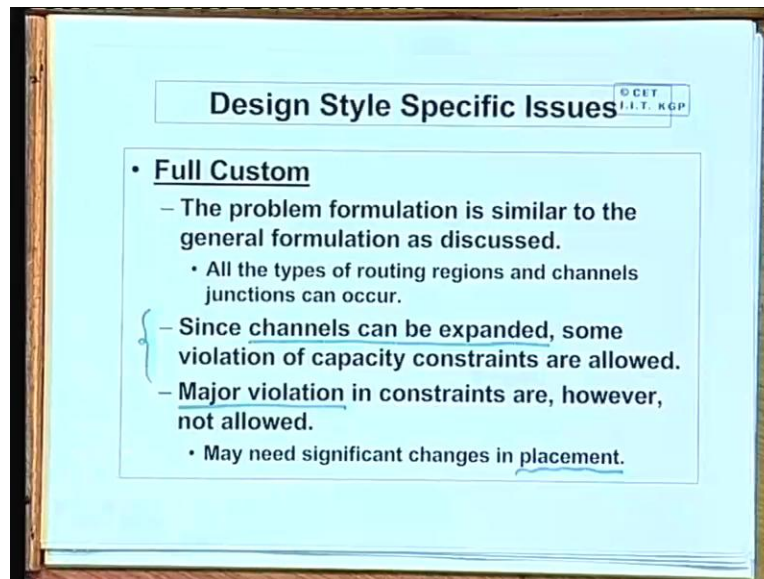


that it is it is a routing area ok. But this T type channel also can be routed using conventional channel routers how see if we have a channel like this then we impose an ordering what we say is that we will be routing this particular channel first. The leg of the T we handle first because I will tell you if you handle this leg first there will be some pins on this side, there will be some pins on the other side so you will have to interconnect them well moreover there can be some pins out here also.

But on this boundary the position of the pins are not fixed those are so called floating terminals. So as this channel routing is completed those nets which are going out of this they will get assigned some fixed points on this boundaries. Now after these points have been fixed up this becomes more like a conventional channel. But if you do it the other way around you will be in problem you cannot treat this as whole channel with a hole in between it becomes more difficult to handle. So the leg of the T is handled first then the shoulder ok. But the more complex kind of channel have a shape like a plus where we will have switch boxes you will you will require switch box routers which are typically much more complex as compare to channel routers.

So an example like this so if you have something like this then well you can have a channel out here you can have a channel out here similarly a channel here and a channel here. But the space in between this will be a switch box this will be a switch box because here pins will be coming from all 4 sides right. So here you cannot do away with switch box routing so a heuristics says that during placement you place them in such a way that this kind of plus type junctions are avoided. Ok so you better move the blocks around so that you get T type junctions as find convert plus type into T types wherever possible ok fine.

(Refer Slide Time: 48:27)



Now finally some design style specific issues well if you have a full customs design style then all the problems you have just mentioned all of them apply. There all kind of channel junctions as I have mentioned can apply blocks can be of all possible shapes and moreover here the channel capacities are not fixed the channels can be expanded and some violation of capacity constraints are allowed because we can expand the channels. But 1 thing you remember well all the way saying that we can handle expansion of channels if required. But if there is a major of violation we see that this space we have kept we have to double that space then basically what we are talking about is that we are making a drastic change in the placement by shifting the blocks up considerable may be if we go back to the placement and redo it will be better. So normally during routing well we allow some violation of capacity constraints but not drastically. So if there is a major violation then typically we may have to go back to the placement just address this problem again come back ok.

(Refer Slide Time: 49:49)

**Standard Cell**

- At the end of the placement phase
  - Location of each cell in a row is fixed.
  - Capacity and location of each feed-through is fixed.
  - Feed-throughs have predetermined capacity.
- Only horizontal channels exist.
  - Channel heights are not fixed.
- Insufficient feed-throughs may lead to failure.
- Over-the-cell routing can reduce channel height, and change the global routing problem.

FT Cells

A cannot be connected to B

But if you have standard cells for example standard cells if you recall there the cells are all of the same height they are all placed along some of the rows. Now I had mentioned earlier also that in standard cells there are some special cells available in the library these are called feed through cells. Feed through cells do not do any computation they just connect a pin on the top to a pin on the bottom. For example in here this represents a feed through cell this is another feed through cell. So a feed through cell is used to connect a cross channels. So when you are doing a placement then the placement tool will know that how many nets are crossing the channels. So accordingly the feed through cells are inserted the placement tool will know that how suppose it says that there are 2 wires that are crossing these 2 channels so there will be 2 feed through which are inserted ok.

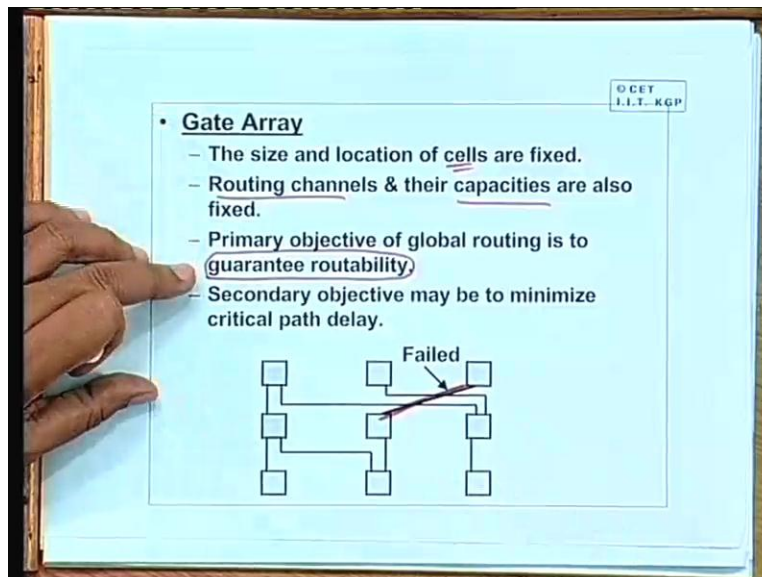
So in standard cell design style after the placement location of each cell gets fixed. Not only that feed throughs also get fixed the capacities and also there exact location. But if during routing you find that due to some reason the placement tool made miscalculation the number of feed. For example here if these 2 feed through have already been occupied by some nets and you need to connect A to B. So here you do not have any path of connecting. So we will have to insert a new feed through cell. So here if you have in sufficient number of feed throughs this may be lead to

failure in the interconnection. So the placement tool must be careful enough to insert sufficient number of feed throughs to of course it is not a very difficult problem you can just analyze the net list and you can find out. Yes (Student Noise Time: 52:23)

See the placement stage it is easy because at the routing stage see you are carrying out routing and some at some point you find that the feed through is less. So if you try to insert a new feed through here then everything on the right will have to be shifted. So what you can do you will have to undo all the routing you have done and then insert it and then again redo it. Because if already some routing you have done if you shift a part of it well it will also carry all the wires along with it. So the topology of interconnection may means it may become very inefficient if you just drag it like that. So well you can insert but you will have to redo the routing of the corresponding channel after you do that. And you will see that there is another new technique which has come up since the last few this is called over the cell routing this says that you not you utilize not only the space between channel but also this space above the cells for interconnecting some of the pins.

So if we utilize this space above the pins then the amount of space you require between them becomes much less. So you can virtually get a channel less design so the channel size becomes insignificantly less. (Student Noise Time: 53:55) Channel heights (Student Noise Time: 54:00) No in standard cell there is no reason why they are fixed that you can move around. Only thing you are specifying that which cell you are placing along which row. Because in standard cell there is nothing which you are already pre-fabricated everything you are fabricating those you can move around.

(Refer Slide Time: 54:24)



OCET  
J.I.T.-KGP

- **Gate Array**
  - The size and location of cells are fixed.
  - Routing channels & their capacities are also fixed.
  - Primary objective of global routing is to guarantee routability.
  - Secondary objective may be to minimize critical path delay.

Failed

The diagram shows a grid of cells with a routing path that is blocked, labeled 'Failed'.

And finally in case of gate array this size and locations of the cells are fixed the routing channel and there capacities are also fixed how much space you have. So placement and routing must go hand in hand suppose there are 2 routing rows between the cells and you have already connected like this. Now you need a connection like this this we cannot do. So you will have to change the placement and you will have to again redo it. So for a gate array the main criteria is typically to guarantee routability whether it can be routed within the available limited resource or not. Well there are issues like critical paths and other things they also will have to address. But routability is most important there. Ok so we had looked at some of the basic problems in global routing. Now in our next class we will be looking into the global routing problem in some more detail. And we will talk about exactly how the global router tries to find out the regions for each individual net. Ok. Thank you.