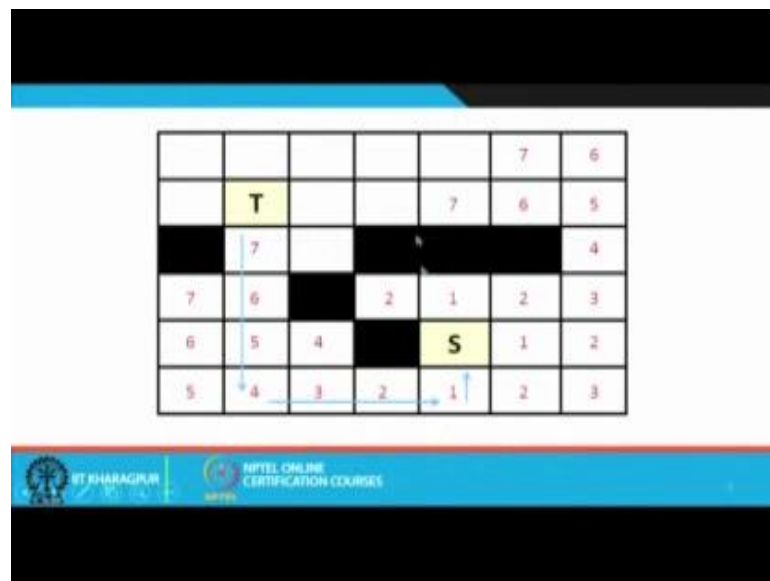


VLSI Physical Design
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 16
Grid Routing (Part II)

So, let us continue with our discussion so recall we were discussing about the Lee's algorithm in our last lecture so we had said during the 3 steps or the 3 phases of the Lee's algorithm how we fill up the cells, how we retrace the path and how we clear the mark cells as well as we mark the path as an obstacle for future runs of Lee's algorithm to join 2 other points so let us start this lecture by analyzing the space complexity a little bit of these approaches.

(Refer Slide Time: 01:06)



So, Lee's algorithm if you recall, we looked at a problem like this where our path was retraced. Now one thing that you just try to understand from this diagram suppose I have a source at one corner let us say here, and my target is the other corner so what will be the length of the path if it exists, let us say if there are m number of rows and N number of columns so you have to follow a path like this, there can be bends, but the total number of cells you have to label will be m plus N minus 1 in this case it will be 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 and 12 so it will be 6 plus 7 minus 1, m plus N minus 1 that will be the maximum length of a path between any pair of s and t. That is the worst case alright.

(Refer Slide Time: 02:14)

- Memory Requirement
 - Each cell needs to store a number between 1 and L , where L is some bound on the maximum path length.
 - For $M \times N$ grid, L can be at most $M+N-1$.
 - One bit combination to denote empty cell.
 - One bit combination to denote obstacles.

$\lceil \log_2(L+2) \rceil$ bits per cell

So, let us make the calculation of memory requirement with respect to that. So what we were saying is that, each cell needs to store a number between 1 and L where L denotes the maximum path length that is possible so I had said for an m by N grid I am just showing it again.

(Refer Slide Time: 02:38)

M

N

$M+N-1$

Maximum Label

$L=3$

$\lceil \log_2(L+2) \rceil = \lceil \log_2(5) \rceil = \lceil 2.15 \rceil = 3$

Suppose I have a 2 dimensional grid, with m number of rows and N number of columns. And if my source is located at one corner and the target is located at the other corner then the length of this longest path will be like this. And this length will be equal to as I had

said $m + N - 1$ which means so as we are labeling these cells, this $m + N - 1$ will indicate the maximum label that I may need to use. As of number this says 1, 2, 3, 4, 5 like that the maximum number can be $m + N - 1$. Not more than, that for our problem of m by N grid right. So $m + N - 1$, so many numbers I have to represent not only that I have to represent one unique bit combination which will denote the empty cells and another bit combination that will denote the obstacles.

So, a total of $1 + 2$ so many different things has to be distinguished. Now in the binary system if we have $1 + 2$ distinct things you have to distinguish and you want to represent it in binary you need \log to the base 2 $1 + 2$ ceiling of that. That many bits. So you need so many bits to represent the state of each cell.

The state of the itself can be either empty or it can be an obstacle or it can be a label which is numbered between 1 and this maximum 1 , ceiling is what ceiling is the smallest number or the small integer which is greater than or equal to this thing. Like a \log to the base 2 $1 + 2$ ceiling means, let us say let us take an example, let us say 1 is equal to 3 well 1 equal to 3 what this value will be $1 + 2$ will be 5 \log to 5 means, sorry it is \log to 4 means 2 \log to 5 means 2 point something. This will be 2 point something so this smallest integer greater than or equal to this will be 3. So this ceiling of this number 2 point something will be 3, but if this number is exactly 4 that that is 1 equal to 2 then this will be exactly 2 because 2 is an integer and smallest number greater than or equal to 2 is 2 itself, so if we say 2.15 let us say 2.15 will become ceiling of that will become 3.

So this is the memory requirement for every cell in the Lee's algorithm with this kind of numbering.

(Refer Slide Time: 06:09)

• Examples:

1. 2000 x 2000 grid
 - $B = \log_2 4001 = 12$
 - Memory required = $2000 \times 2000 \times 12$ bits = 6 Mbytes
2. 3000 x 3000 grid
 - $B = \log_2 6001 = 13$
 - Memory required = $3000 \times 3000 \times 13$ bits = 14.6 Mbytes
3. 4000 x 4000 grid
 - $B = \log_2 8001 = 13$
 - Memory required = $4000 \times 4000 \times 13$ bits = 26 Mbytes

Now, let us take a quick look at some sample grid sizes. So we have a 2000 by 2000 grid, b indicates the number of this l actually, l and b is the same m plus N minus 1, so m plus N this 12, this 12th is a number of bit, this sorry, this l plus 2. This we are calling as B capital B . Now it is m minus N minus 1 m plus N minus 1 plus 2. It becomes m plus N plus 1. So m is 2000, N is 2000, 2000 plus 2000 plus 1 is 4001. So I am not showing this sealing, sealing is there. If you take sealing off this, you will get 12. Because 2 to the power 12th is 4096. It is less than that. So the memory required will be so many cells each cell with 12 bits. If you convert it to bytes this becomes 6 megabytes.

Move to 3000 by 3000 grid. Where b becomes 3000 plus 3000 plus 1 log of that. 13 plus 2 to the power 13 is 8196 by 2. And so the memory required will be 3000 by 3000 into 13 which comes to 14.6 megabytes. Go to 4000 by 4000 grid. Your memory required will be b will be 13. Memory required will be 26 megabytes. You see that even for cells of these very nominal sizes the memory requirement is of the order megabytes, but if you look at practical grid sizes in the kind of various chips so which you have that can be million by million, close to that, so this number will be very huge. Memory requirement will be huge.

So, now we see that how you can reduce the storage requirement of the Lee's algorithm, so some improvements. Well before explaining these let me just try to explain to you the basic idea behind using this numbering scheme 1 2, 3, 4, 5.

(Refer Slide Time: 08:20)

- **Improvements:**
 - Instead of using the sequence 1,2,3,4,5,... for numbering the cells, the sequence 1,2,3,1,2,3,... is used.
 - For a cell, labels of predecessors and successors are different. So tracing back is easy.
 $\lceil \log_2(3+2) \rceil = 3$ bits per cell. 1.5 Mbytes for 2000 x 2000 grid
 - Use the sequence 0,0,1,1,0,0,1,1,....
 - Predecessors and successors are again different.
 $\lceil \log_2(2+2) \rceil = 2$ bits per cell. 1.0 Mbyte for 2000 x 2000 grid

(Refer Slide Time: 08:36)

1 2 3 4 5 6 7 8 ... L

a) Present & next cells have different labels
b) Present .. previous

↓

6 ← 7 → 8
R ← P → N

Distinct

1 2 3 | 2 3 | 2 3 | ...

You see in a Lee's algorithm; we are numbering the cells consecutively like this. So there are 2 things that you are just ensuring here. You are ensuring that number a the present and next cells have different levels and, secondly, for each level let us say I am in 7, so the cell which is just before that it will be just one less than that 6. So present and previous cells they also have default levels, but one thing is true, that you have a particular present cell let us call it P, the next cell let us call it N, and the previous cell let us call it R, they are all distinct they are labeled in a distinct way. Like if P is 7, example I took here if P is 7. Then your N will be 8 and R will be 6. So when you are carrying out

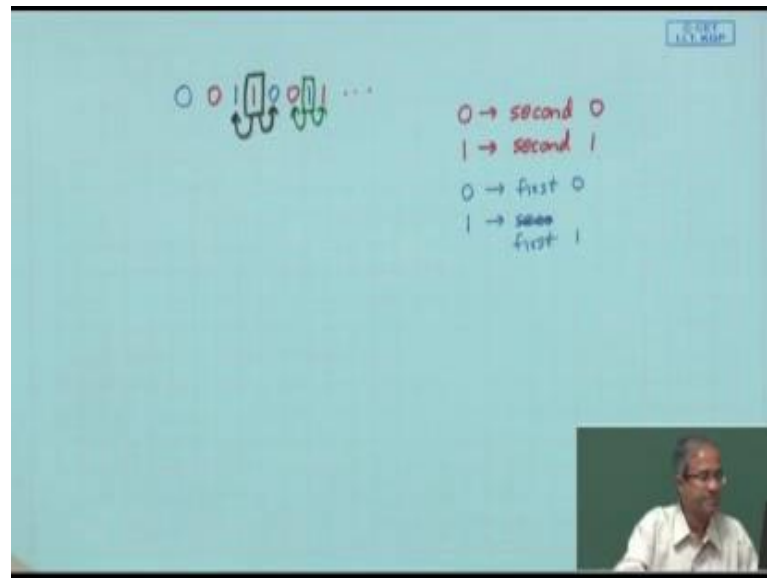
phase one of Lee's algorithm you know that which level to use next, and when you are retracing from 7 you are looking for a cell which is one less 6. So you know uniquely which direction you have to move right. This is one of the very important characteristics with this 1, 2, 3, 4, 5, this sequence holds.

Now, the first alternative that we talk about here is that well, even saying that instead 1, 2, 3, 4, 5, 6, let us say we use a numbering scheme like this 1 2, 3, again 1, 2, 3 again 1, 2, 3 and this order we go on numbering the cells. Here you look at some of the properties. You take any arbitrary number let us say this 2.

For this 2 the next label is 3, the previous label is 1. They are all distinct as it was here. Take any other let us say take 3, your previous level is 2 and next level will be 1 again which are all distinct. So the property of distinct numbering also holds for this so instead of using 1, 2, 3, 4, 5, 6 in this way I can use a numbering scheme which is more efficient 1, 2, 3, 1, 2, 3, 1, 2, 3. Because in the earlier case I was going up to a maximum of 1, but here I can go up to a number 3, not more than. That so if I go to a maximum of 3, then my number of bits per cell will be locked to only 3 plus 2, not 1 plus 2 and locked to 3 plus 2 means only 3, 3 bits per cell. So, if you look at the previous calculation here, here we are multiplying by 12th or 13, 13th, but in this new numbering scheme we have to multiply it only by 3. So it will become just 1.5 megabytes.

There is a more efficient scheme that we can about. Let us look at the other one. So here I am saying that we are not even going into 1, 2, 3 1, 2, 3 like that.

(Refer Slide Time: 12:44)



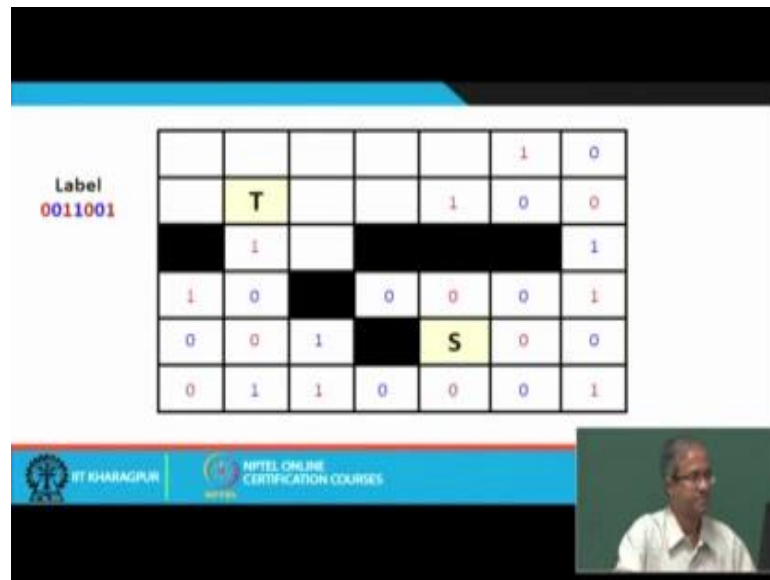
We are using only 2 levels. We are following this sequence, 0 then another 0, then a 1 then another 1 then a 0, another 0, 1 then another 1 and so on. This is the sequence in which you are going. Now you see here we are using 2 different colors to draw zeros and ones. A red 0 means this is the second 0 in sequence. This among the 2 consecutive 0 this is second 0; the red 1 means this is the second 1. Similarly, blue 0 means this is the first 0 and blue 1 means this is the first 1, sorry first 1.

So I can distinguish between the 2 zeros and 2 ones with respect to whether they are the first 0 or the first 1 I can keep track of that. When I am doing the numbering I can always keep track of whether the number 0 that I am using is the first 0 or the second 0 in the sequence. If I remember that then I can distinguish between the 2. Now if I am able to distinguish between the 2, now you see you take any label let us take I take this red 1, it is previous state is blue 1, next state is blue 0. You take another state let us say blue 1. For blue 1 the previous state is red 0 next state is red 1. So you check for zeros also you will find that for all the 4 cases the previous and the next neighbors will be distinct, and you can make a distinction, whether they are red 0 or red 1. You see red 0 and red 1 I am not explicitly storing, but for numbering I am keeping track of whether I am using it at the first 0 or the second 0.

So, this numbering scheme will also work. I shall show you with an example how it does. But if you can do this here this number 3 goes down to 2, so here we need only 2

bits per cell. It means in this case it is multiplied by 2, you need only just a 1 megabyte. This will be only 1 megabyte. In that case so you can reduce the storage speed, but let us say how 0, 0, 1, 1 labeling works. So this is what I have said 1.5 megabytes for this. This requires 1 megabyte.

(Refer Slide Time: 15:59)



Let us take that same example, but now we are labeling with 0, 0, 1, 1.

Let us start with a red 0 let us say. The source the immediate neighbors and coloring them with a 0. The second 0 as a matter of convention I am calling it a blue 0 the second 0 I label. Then I label 1, label 1. Then blue 1, then this way I proceed red 0 then blue 0, then red 1. So here I touch the target. Now you see the way I label suppose when I encounter a red 1, you see a red 1, the previous cell is a 0. And from that 0 the previous cell is another 0, from that the previous will be another one another one; that means, when you trace back from a red 1 you are expecting 0, 0, 1, 1, 0, 0, 1, 1 in that order the number should come.

So, you see you trace back in that same order 0, 0, 1, 1, 0, 0 you reach or you can also follow this path, 0, 0, 1, 1, 0, 0 like that. So that same thing you are distinguishing between the next label and the previous label. In this labeling scheme if the next label is 0 the previous label will be 1. If the next label is 1 the previous label will be 0. This condition is maintained. So you can uniquely trace back the path and this method

actually as you can this requires less storage as compared to the conventional least numbering scheme. So you get a path like this.

(Refer Slide Time: 17:57)

Retrace
0011001

					1	0
	T			1	0	0
	1					1
1	0		0	0	0	1
0	0	1		S	0	0
0	1	1	0	0	0	1

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

(Refer Slide Time: 18:00)

Reducing Running Time

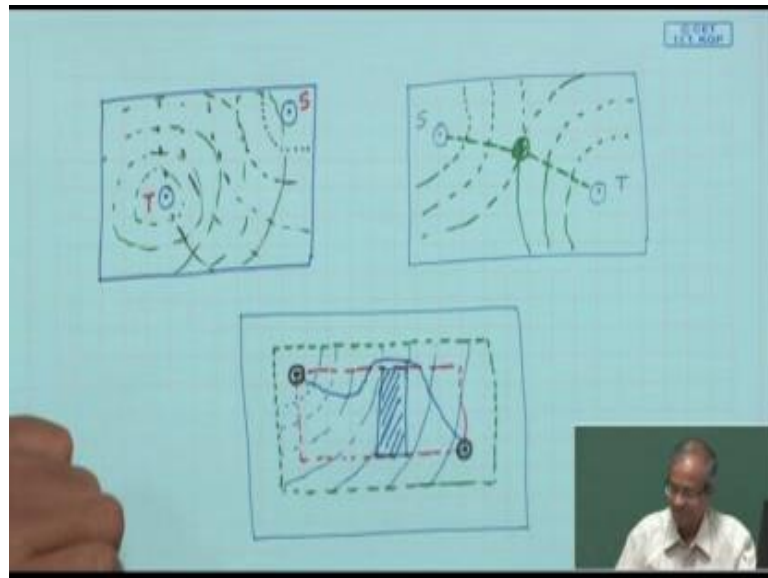
- Starting point selection
 - Choose the starting point as the one that is farthest from the center of the grid.
- Double fan-out
 - Propagate waves from both the source and the target cells.
 - Labeling continues until the wavefronts touch.
- Framing
 - An artificial boundary is considered outside the terminal pairs to be connected.
 - 10-20% larger than the smallest bounding box.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, there are several heuristics or you can say; that means, approximations that you can use in the Lee's algorithm to reduce the running time. There are 3 such strategies which are mentioned here. The first one says, so you have a pair of points source and target which you want to connect. Now which one you call S and which one you call T that is up to you right. So you call the first one S, second one T, or the first one T and the

second one S. So it does not matter which one you are calling a source and which one you calling as target. So this heuristic says that the point which is farthest from the center of the grid you choose it as the source of the starting point which means let me diagrammatically try to explain what does this mean.

(Refer Slide Time: 18:56)



Suppose in this grid, so I had one point which has near the corner and one point which has near the center let us say here. So what this heuristics says the point which is farthest away from the center you use it as the starting point, and the one which is closer to the center we use it as the target. So what is the motivation? So the motivation is simple, if you start from s well I am just showing the wave front like this.

The neighbors labeled as 1 then labeled as 2 labeled as 3 like this we proceed. Then you reach T, but if you start with T you will see that the wave fronts can move in all 4 directions. Like 1 then 2 then 3 then 4 like this, which means if you start with T you will have to label much larger number of cells as compared to the case when you what happens when you start with S. Because when you start with S you are in a corner you are only restricting yourself to these 90 degree the remaining 270 degrees you are not propagating the waveform. So number of cells you are labeling in this case will be much less. So the algorithm will be running much faster, right.

You see that the running time will be proportional to the number of cells you are changing or you are labeling right. Because more number of cells you label during

retrace you will also have to scan than many cells in the worst case. So this is the first one.

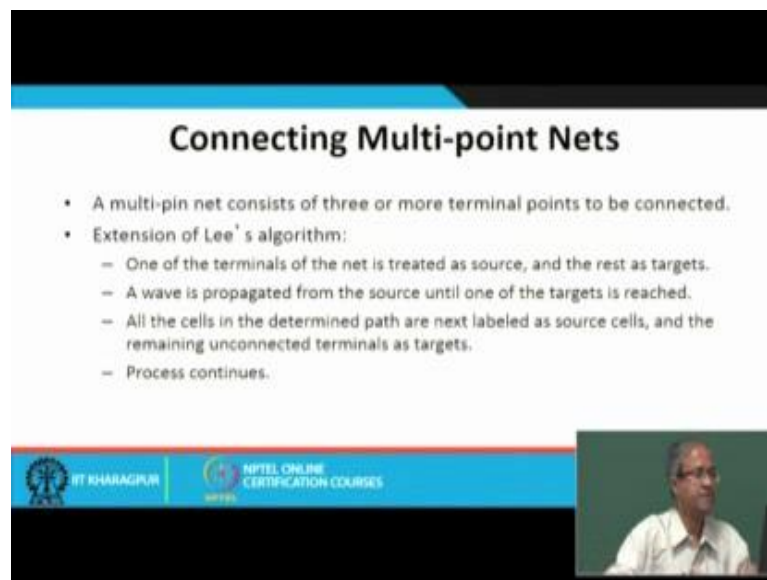
The second one says, it will you generate the wave fronts from both S and T this is called double fan out propagate the waves from both the source and target, labeling continues until the till the wave fronts touch. So let us explain this again with a proper diagram. Let us say I have a scenario I have let us say one point here and one point here. Let us say this is your S and this is your T. So what this method says is that, you start the wave front propagation not only from S, but also from T. So wave fronts touch somewhere, so you get a path. So here the initiative idea is that if you start only for one position the wave fronts are expected to spread in all directions, but if you start from both directions then you are not allowing the wave front to expand in all directions, before that they are touching each other.

So, the wave fronts are slowly expanding in their sizes and they are coming closer and closer together. So before the expand to their full size and start filling up cells much more rapidly, they touch in each other which means you are getting a path much before that. So again here the average number of cells you will be leveling will is expected to be matchless as compared to the case where you start the wave front propagation from one point, alright. Now the third strategy is something called framing. Here what you do you imagine an artificial boundary or an artificial rectangle, that is outside the terminal pairs which is typically 10 to 20 percent larger than the smallest bounding box.

Let me again illustrate this. Let us say I have a point here and I have a point here which I want to connect. So what it says is that you imaging a rectangular bounding box which encloses the 2 points that you want to connect may be like this, which is say approximately 10 to 20 percent larger than this smallest bounding rectangle. So if you take this prints the rectangle would have been this you take a little more because the reason is that, here is the idea that when you generate the wave fronts from one point to the other, you do not allow the wave front to cross this boundary. Which means your wave fronts will only be limited to this rectangle; that means, cells outside this rectangle will not be labeled. So you are directly restricting the number of cells you are labeling right.

Now, the reason you are taking this 10 to 20 percent extra is that see within this red rectangle there might be some obstacles already existing, like there may be an obstacle like this. So you may have to find out a path around the obstacle like this. So you are taking a little x cover 10 to 20 percent so that such obstacles can be avoided. Now these approaches, these are you can some kind of heuristics or some kind of means intuitive approaches, which try to restrict the number of cells that we are labeling. In addition, you can also have another scheme you define or choose some maximum number m you do not do not allow your label to go beyond them because you know that your maximum length of the path is expected to be within m only. So if you have a idea regarding that m you can also take that m as a parameter the wave front will not go beyond m .

(Refer Slide Time: 25:47)



Connecting Multi-point Nets

- A multi-pin net consists of three or more terminal points to be connected.
- Extension of Lee's algorithm:
 - One of the terminals of the net is treated as source, and the rest as targets.
 - A wave is propagated from the source until one of the targets is reached.
 - All the cells in the determined path are next labeled as source cells, and the remaining unconnected terminals as targets.
 - Process continues.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

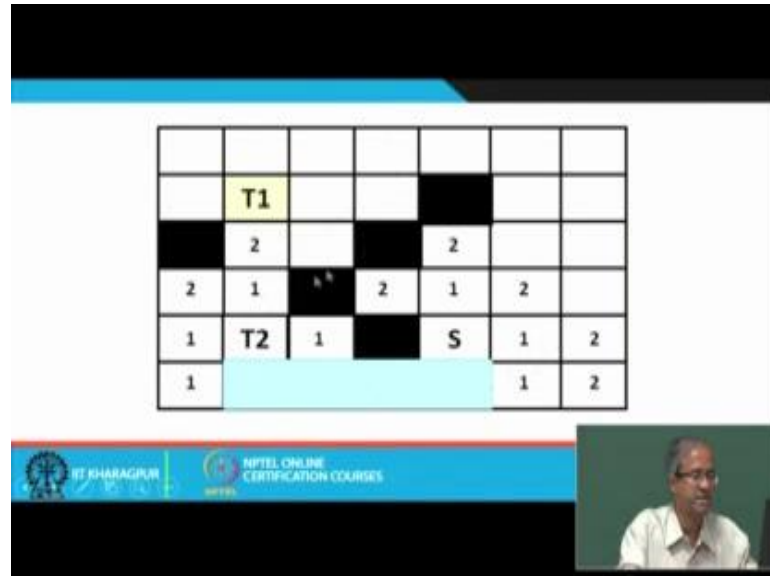
NPTEL

Video inset showing a speaker.

So, these are some approaches, you can use for reducing the running time. And lastly we consider method run approach in which you can extend Lee's algorithm to handle multi point nets. Multi point net means you have not 2, but 3 or more terminal points which have to be connected together. Now there can be several extensions you can think of, here I am just talking about one. It says you select one of the terminals of the net as a source and the remaining as targets. Let a wave start propagating from the source till it hits one of the targets. So you follow Lee's algorithm you determine the path from the source to the target you have hit, and in the next step all the cells in the determined path are identified as source cells. And wave fronts are emanating from all the sources

together. So I will show you an example and this process continues till all the targets are covered

(Refer Slide Time: 27:01)



Let us take an example like this where I have the source here and the 2 targets. Well I am not showing all the steps initially. So I am assuming that these labeling starts with S 1, 1, 1, 2, 2, 2, 2. As you can see T 2 will be closer as compared to T 1. So S will first detect this T 2 as it is nearest neighbor and may be a path like this will be determined from S up to T 2. This blue box indicates the path right. Now once you have identified this path what this extinction says is that all these like here all these cells not only S also T 2 and also these cells which have been marked as the path they are all identified as the source cell and we start these algorithms by numbering starting from there like 1, 1, neighbor of this cell will also be a 1, neighbor of this cell will be 1, this will be 1, this will be 1, this will also be a 1. So we are just starting this wave front emanation from all grid points along the path not only a single point s like earlier. In this way it continues it will be 2 like this again. So you get this connection so this will be connected after that.

So, this process will continue after you hit T 1 this will continue if there is a T 3, that T 3 will come next. In the next step all these points will start emanating wave points and T 3 will be touched right. So this will be another path which will be identified like that it will go on right. So I think in this lecture we have discussed some of the variations of Lee's algorithm which can help it to save space in terms of storage to run faster, and also some

simple way in which you can extend the basic algorithm to handle multi pin nets on the next lecture we shall be looking at some other approaches which are more efficient in terms of the running time.

Thank you.